






Universitat Autònoma de Barcelona

ADVERTIMENT. L'accés als continguts d'aquesta tesi queda condicionat a l'acceptació de les condicions d'ús establertes per la següent llicència Creative Commons:  http://cat.creativecommons.org/?page_id=184

ADVERTENCIA. El acceso a los contenidos de esta tesis queda condicionado a la aceptación de las condiciones de uso establecidas por la siguiente licencia Creative Commons:  <http://es.creativecommons.org/blog/licencias/>

WARNING. The access to the contents of this doctoral thesis it is limited to the acceptance of the use conditions set by the following Creative Commons license:  <https://creativecommons.org/licenses/?lang=en>

UNIVERSITAT AUTÒNOMA DE BARCELONA

Departament de Genètica, Facultat de Biociències

CENTRE DE RECERCA EN AGRIGENÒMICA

Departament de Genètica Animal

**KERNEL APPROACHES FOR
COMPLEX PHENOTYPE PREDICTION**

Elies Ramon Gurrea

Doctoral thesis to obtain the PhD degree in Genetics by the Universitat Autònoma de
Barcelona, October 2020

Supervisors

Dr. Miguel Pérez Enciso

Dr. Lluís Antoni Belanche Muñoz

Tutor

Dr. Antonio Barbadilla Prados

El **Dr. Miguel Pérez Enciso**, investigador ICREA del Departament de Ciència Animal i dels Aliments de la Universitat Autònoma de Barcelona (UAB), i el **Dr. Lluís Antoni Belanche Muñoz**, professor titular de la Universitat Politècnica de Catalunya (UPC),

fan constar,

que **Elies Ramon Gurrea** ha realitzat sota la seva direcció el treball de recerca

Kernel approaches for complex phenotype prediction

i certifiquen

que aquest treball s'ha dut a terme al departament de Genètica Animal del Centre de Recerca en Agrigenòmica (CRAG)

considerant

que la memòria resultant és apta per optar al grau de Doctor en Genètica per la Universitat Autònoma de Barcelona,

i perquè quedi constància, signen aquest document a

Bellaterra, a 17 de setembre de 2020.

Dr. Miguel Pérez Enciso

Elies Ramon Gurrea

Dr. Lluís Antoni Belanche Muñoz



Generalitat de Catalunya
Departament d'Empresa i Coneixement
Secretaria d'Universitats i Recerca



Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Elies Ramon Gurrea ha rebut finançació del Programa d'Ajuts per a la Contractació de Personal Investigador Novell (FI-DGR) atorgat per l'Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR). Aquesta tesi ha sét realitzada amb els suports de la Secretaria d'Universitats i Recerca de la Generalitat de Catalunya i del Fons Social Europeu.

Front cover: HIV in blood serum. Back cover: DNA being replicated in the nucleus.
Illustrations by David S. Goodsell, the Scripps Research Institute.

*“...També s’explica que, una vegada,
Ptolemeu I va preguntar Euclides si no hi
havia cap via més curta a la geometria que a
través dels Elements, i que Euclides va
replicar: No existeix un camí ral cap a la
geometria.*

Procle Diàdoc — *Comentari al 1r Llibre dels
Elements d’Euclides*

CONTENTS

SUMMARY	17
RESUM	19
LIST OF FIGURES.....	21
LIST OF TABLES	25
LIST OF PUBLICATIONS.....	27
ABBREVIATIONS	29
SYMBOLS	33
1. GENERAL INTRODUCTION.....	37
1.1 Phenotype-genotype relationship	37
1.2 Data types in genetics and genomics.....	39
1.2.1 Sequences	39
1.2.2 Counts	42
1.2.3 Graphs	43
1.3 Spatial and temporal structured datasets.....	43
1.3.1 Spatial dimension.....	44
1.3.2 Temporal dimension.....	45
1.4 The <i>big data</i> paradigm in biology	47
1.5 Machine Learning primer	47
1.5.1 Supervised learning	48
1.5.2 Unsupervised learning.....	51

1.5.3 Other types of learning.....	51
1.6 ML methods applied to genetics and genomics	52
1.6.1 Decision Trees.....	52
1.6.2 Random Forests	54
1.6.3 Artificial Neural Networks	54
1.6.4 Challenges of ML applied to genomic data.....	57
2. OBJECTIVES	59
3. GENERAL METHODS	61
3.1 Introduction to the kernel framework	61
3.1.1 Kernel functions.....	61
3.1.2 Kernel matrices	63
3.2 Kernel methods.....	65
3.2.1 Support Vector Machine (SVM)	65
3.2.2 Kernel Principal Components Analysis (kPCA).....	68
3.3 Kernel functions used throughout this thesis.....	70
3.3.1 Kernels for real vectors	70
3.3.2 Kernels for categorical data & categorical sets	72
3.3.3 Kernels for graphs.....	74
3.3.4 Kernels for count data.....	75
3.3.5 Kernels for temporal-structured data.....	78
3.4 Recovering variable importance from SVM.....	80
4. HIV DRUG RESISTANCE PREDICTION WITH WEIGHTED CATEGORICAL KERNEL FUNCTIONS	83
Abstract	85
4.1 Background	87
4.2 Methods.....	89
4.2.1 Datasets and data pre-processing	89

4.2.2 Methods	91
4.2.3 Experimental setup and model tuning	93
4.2.4 Visualization	94
4.2.5 Performance comparison to other approaches	94
4.3 Results.....	95
4.3.1 Performance overview	95
4.3.2 Factors affecting prediction error	99
4.3.3 Kernel PCA	101
4.3.4 Stacked models	101
4.3.5 Performance comparison to other approaches	101
4.4 Discussion	102
4.5 Conclusions.....	106
Declarations.....	106
5. STRUCTURAL AND FUNCTIONAL ASSOCIATIONS IN HIV DRUG RESISTANCE PREDICTION WITH KERNEL METHODS	
Abstract.....	111
5.1 Introduction	111
5.2 Material and Methods.....	113
5.2.1 Protein structures, sequence datasets and data pre-processing	113
5.2.2 Random walk kernel.....	113
5.2.3 Kernel function implementation.....	115
5.2.4 Experimental set-up and kernel visualization.....	116
5.2.5 LD measures	117
5.3. Results.....	118
5.4 Comparison with other works.....	120
5.5. Discussion	122

6. <i>kernInt</i> : A KERNEL FRAMEWORK FOR INTEGRATING SUPERVISED AND UNSUPERVISED ANALYSES IN SPATIO-TEMPORAL METAGENOMIC DATASETS	125
Abstract	127
6.1 Introduction	127
6.2 Material and methods	130
6.2.1 Kernels for microbiome data	130
6.2.2 Kernel methods and framework	131
6.2.3 Spatial data	131
6.2.4 Temporal data	132
6.2.5 Microbial signature	133
6.2.6 Case studies and data preprocessing	133
6.2.7 Experimental set-up	134
6.3 Results	136
6.3.1 Soil data	136
6.3.2 Smokers data	137
6.3.3 Pig data	139
6.4 Discussion	141
Declarations	144
DISCUSSION AND FUTURE RESEARCH	145
7.1 General remarks	146
7.2 The approach to HIV data particularities	149
7.3 Synthesis of current microbiome analysis	154
CONCLUSIONS	159
REFERENCES	161
ANNEXES	171
Supplementary material Chapter 4	171
Supplementary material Chapter 5	196

Supplementary material Chapter 6	209
Kernel normalization	215
Hyperparameters	216
'catkern' vignette.....	217
'kernInt' vignette.....	225
ACKNOWLEDGEMENTS.....	237

SUMMARY

The relationship between phenotype and genotypic information is considerably intricate and complex. Machine Learning (ML) methods have been successfully used for phenotype prediction in a great range of problems within genetics and genomics. However, biological data is usually structured and belongs to 'nonstandard' data types, which can pose a challenge to most ML methods. Among them, kernel methods bring along a very versatile approach to handle different types of data and problems through a family of functions called kernels.

The main goal of this PhD thesis is the development and evaluation of specific kernel approaches for complex phenotype prediction, focusing on biological problems with structured data types or study designs.

In the first study, we predict drug resistance from HIV mutated protein sequences (protease, reverse transcriptase and integrase). We propose two categorical kernel functions (Overlap and Jaccard) that take into account HIV data particularities, such as allele mixtures. The proposed kernels are coupled with Support Vector Machines (SVM) and compared against two well-known standard kernel functions (Linear and RBF) and two nonkernel methods: Random Forests (RF) and the Multilayer Perceptron neural network. We also include a relative weight into the aforementioned kernels, representing the importance of each protein position regarding drug resistance. Our results show that taking into account both the categorical nature of data and the presence of mixtures consistently delivers better predictions. The weighting effect is higher in reverse transcriptase and integrase inhibitors, which may be related to the different mutational patterns in the viral enzymes regarding drug resistance.

Next, we extend the previous study to consider that protein positions may be not independent. Mutated sequences are modeled as graphs, with edges weighted by the Euclidean distance between residues, as obtained from crystal three-dimensional structures. A kernel for graphs (the exponential random walk kernel) that integrates the previous Overlap and Jaccard kernels is then computed. Despite the potential advantages of this kernel

for graphs, an improvement on predictive ability as compared to the kernels of the first study is not observed.

In the third study, we propose a kernel framework to unify unsupervised and supervised microbiome analysis. To do so, we use the same kernel matrix to perform phenotype prediction via SVM and visualization via kernel Principal Components Analysis (kPCA). We define two kernels for compositional data (Aitchison-RBF and compositional linear) and discuss the transformation of beta-diversity measures into kernels. The compositional linear kernel also allows the retrieval of taxa importances (microbial signatures) from the SVM model. Spatial and time-structured datasets are handled with Multiple Kernel Learning and kernels for time series, respectively. We illustrate the kernel framework with three datasets: a single point soil dataset, a human dataset with a spatial component, and a previously unpublished longitudinal dataset concerning pig production. Analyses across the three case studies include a comparison with the original reports (for the two former datasets), as well as contrast with results from RF. Our kernel framework does not only allow a holistic view of data but also gives good results in each learning area. In unsupervised analyses, the visual main patterns detected in the original reports are conserved in kPCA. In supervised analyses SVM has better (or, in some cases, equivalent) performance than RF, while microbial signatures are consistent with the original studies and previous literature.

RESUM

La relació entre fenotip i informació genotípica és considerablement intricada i complexa. Els mètodes d'aprenentatge automàtic s'han utilitzat amb èxit per a la predicció de fenotips en un gran ventall de problemes dins de la genètica i la genòmica. Tanmateix, les dades biològiques sovint estan estructurades i són de tipus "no estàndard", el que pot suposar un repte per a la majoria de mètodes d'aprenentatge automàtic. Entre aquests, els mètodes kernel proporcionen un enfocament molt versàtil per manejar diferents tipus de dades i problemes mitjançant la utilització d'una família de funcions anomenades de kernel.

L'objectiu principal d'aquesta tesi doctoral és el desenvolupament i l'avaluació d'estratègies de kernel específiques per a la predicció de fenotips complexos, especialment en problemes biològics amb dades o dissenys experimentals de tipus estructurat.

En el primer estudi, utilitzam seqüències de proteasa, transcriptasa inversa i integrasa per predir la resistència del VIH a fàrmacs antiretrovirals. Proposam dos kernels categòrics (Overlap i Jaccard) que tenen en compte les particularitats de les dades de VIH, com per exemple les barreges d'al·lels. Els kernels proposats es combinen amb Support Vector Machines (SVM) i es comparen amb dos kernels estàndard (Linear i RBF) i dos mètodes que no són de kernel: els boscos aleatoris (RF) i un tipus de xarxa neuronal (el perceptró multicapa). També incloem en els kernels la importància relativa de cada posició de la proteïna pel que fa a la resistència. Els resultats mostren que tenir en compte la naturalesa categòrica de les dades i la presència de barreges millora sistemàticament la predicció. L'efecte de ponderar les posicions per la seua importància és més gran en la transcriptasa inversa i en la integrasa, el que podria estar relacionat amb les diferències que hi ha entre els tres enzims pel que fa als patrons de mutació per adquirir resistència a fàrmacs antiretrovirals.

A continuació, ampliam l'estudi anterior per considerar la possibilitat de no-independència entre les posicions de la proteïna. Representam les proteïnes com grafs i ponderam cada aresta entre dos residus per la seua distància euclidiana, calculada a partir de dades de cristal·lografia de rajos X. A continuació, hi aplicam un kernel per a grafs (el random walk

exponential kernel) que integra els kernels Overlap i Jaccard. A pesar dels avantatges potencials d'aquesta estratègia, no aconseguim millorar els resultats obtinguts en la primera part.

En el tercer estudi, proposam un kernel framework per unificar els anàlisis supervisats i no supervisats en el camp del microbioma. Aprofitam la mateixa matriu de kernel per predicció mitjançant SVM i visualització mitjançant anàlisi de components principals (kPCA). Discutim com transformar mesures de beta-diversitat en kernels, i definim dos kernels per a dades composicionals (Aitchison-RBF i compositional lineal). Aquest darrer kernel també permet obtenir les importàncies dels tàxons respecte del fenotip predit (signatures microbianes). Per a les dades amb estructuració espacial i temporal utilitzam Multiple Kernel Learning i kernels per a sèries temporals, respectivament. El framework s'il·lustra amb tres bases de dades: la primera conté mostres de sòl, la segona mostres humanes amb una component espacial i la tercera, no publicada fins ara, dades longitudinals de porcs. Totes les anàlisis es contrasten amb els estudis originals (en els dos primers casos) i també amb resultats de RF. El nostre kernel framework no només permet una visió global de les dades, sinó que també dona bons resultats a cada àrea d'aprenentatge. En les anàlisis no supervisades, els patrons visuals detectats en estudis previs es conserven a la kPCA. En anàlisis supervisades, el SVM té un rendiment superior (o, al menys, equivalent) al dels RF, mentre que les signatures microbianes són coherents amb els estudis originals i la literatura prèvia.

LIST OF FIGURES

Chapter 1

Figure 1.1. DNA, RNA and protein sequences. Figure taken from Cao and Xiong (2014). License: CC BY 3.0	41
Figure 1.2. Sanger sequencing (left) versus NGS sequencing (right). Taken from Price <i>et al.</i> (2018) . License: CC BY-NC 4.0	42
Figure 1.3. Graphs represented as diagrams (above) or as adjacency matrices (below). Taken from EMBL-EBI webpage (https://www.ebi.ac.uk/training/online/course/network-analysis-protein-interaction-data-introduction/). License: CC BY-SA 4.0	44
Figure 1.4. Left: Height versus age of 54 girls and 39 boys (Berkeley Growth Dataset). Right: height distribution for girls and boys at age 18 (above) and age 11 (below).....	46
Figure 1.5. Growth profile during puberty for the boys (left) and girls (right) of the Berkeley Dataset.....	46
Figure 1.6. K Cross-validation. K itself is an hyperparameter whose value has to be decided.....	50
Figure 1.7. Visual example of classification (left) and regression (right) models.....	50
Figure 1.8. Decision Tree for a classification problem. Adapted from Bishop (2006). Copyright (2006) by Christopher M. Bishop.....	53
Figure 1.9. Graphical representation of a MLP. This specific architecture is denoted as 3-3-2. Taken from: https://commons.wikimedia.org/wiki/File:MultiLayerNeuralNetworkBigger_english.png . Author: Christoph Burgmer. License: CC BY-SA 3.0	55

Chapter 3

Figure 3.1. A classification example of how the kernel trick allows solving nonlinear problems in a linear manner.....	63
Figure 3.2. SVM graphical representation. Left: The black straight line separates the two classes. This line is the intersection of the feature space where lie the objects \mathbf{x} with the separating (hyper)plane. Support vectors are highlighted. Right: The same with examples of points violating the margin, C and α	66

Figure 3.3. SVM for regression. In blue: points within the ε -tube, so the difference between actual and predicted value is considered 0. In black: points outside the ε -tube, so $L_\varepsilon = |y - F(x)| - \varepsilon$ applies instead. Support vectors are highlighted.....68

Figure 3.4. Effect of σ in defining the RBF-SVM boundary shape around the support vectors. Left: $\sigma = 0.5$, right: $\sigma = 5$. Created with <https://cs.stanford.edu/people/karpathy/svmjs/demo/>.....71

Figure 3.5. The S^2 simplex imbedded in \mathbf{R}^3 . 3-part compositions \mathbf{x}_i , \mathbf{x}_j , \mathbf{x}_a and \mathbf{x}_b are all different in terms of absolute counts, but \mathbf{x}_i is equivalent to \mathbf{x}_j in terms of proportions (*i.e.* compositionally equivalent), and the same is true for \mathbf{x}_a and \mathbf{x}_b77

Chapter 4

Figure 4.1. NMSE distributions for a PI (FPV), an NRTI (DDI), an NNRTI (NVP) and an INI (EVG). Note that NMSE scale varies between panels.....96

Figure 4.2. RF relative importance of each protein position for three drugs: a protease inhibitor (A), a reverse transcriptase inhibitor (B) and an integrase inhibitor (C). Standard error across the 40 replicates is marked with error bars. Asterisks highlight the major drug related positions reported in the literature (Iyidogan & Anderson, 2014).....97

Figure 4.3. A: Wild type protease (in yellow and blue) with an inhibitor (NFV, in green) (PDB code: 3EKX). We highlight the ten most important positions according to RF: 10, 90, 54, 46, 71, 88, 84, 30, 20 and 83. These positions are scattered throughout the protein and only a few belong to the drug binding site (*e.g.* 30, 82 and 84). Mutations at the binding site reduce the affinity for the inhibitor, but can impair the protease catalytic activity as a collateral damage. Mutations in distant residues are typically concurrent with these binding site mutations and often have a compensatory role (*e.g.* stabilizing the protease structure or restoring the catalytic activity). Position 30 appears to be important only in the case of the NFV drug, while the other positions are found in all (or almost all) protease inhibitors. This agrees with the literature (Iyidogan & Anderson, 2014). B: Binding pocket of the reverse transcriptase (in yellow) with an NNRTI (NVP, in pink) (PDB code: 3V81). We highlight the five most important positions for NVP according to RF: 103, 181, 190, 188 and 101. All these positions reside in the NNRTI binding pocket of the enzyme, and also appear in the other NNRTIs analyzed. Thus, in EFV, we find 100 (but not 181) in the top 5; and in ETR, we have 179 instead of 188 (also highlighted). Positions 103 and 101 are located near the entry of the inhibitor binding pocket and, when mutated, interfere with the entrance of the inhibitor to the binding site. Y181 and Y188 have a crucial contribution the NVP binding via stacking interactions between its side chains and the inhibitor aromatic groups. G190 mutations lead to resistance through steric hindrance, because of the substitution by a more voluminous side chain. L100 effect is also related to steric hindrance (Iyidogan & Anderson, 2014).....98

Figure 4.4. A, B and C: NMSE residuals (*observed* – *fitted* values) of the linear model containing only data size (N) and kernel (K) vs. Gini index. Each color represents a different drug. Note different scale for the Gini index between panels. D, E and F: Residuals (*observed* – *fitted* values) of the linear model containing K , W and $GINI$ vs. data size (N). Each color represents a different drug.....100

Figure 4.5. The Jaccard kPCA in a protease inhibitor (FPV, panels A and B) and a reverse transcriptase inhibitor (NVP, panels C and D). Panels A and C correspond to unweighted Jaccard, and B and D to weighted Jaccard. Dot color represents the actual log-resistance value for each specific drug; in red the

more resistant, and in green the least resistant. Sequences with missing resistance value are in gray.
102

Figure 4.6. Mean NMSE values and their corresponding standard errors for the SVM + weighted Jaccard kernels (red), ANN1 (light gray) and ANN2 (dark gray). PIs are shown in panel A, NRTIs in panel C, NNRTIs in panel B and INIs in panel D.....103

Chapter 5

Figure 5.1. NMSE distributions for two PIs (FPV and ATV, panel A and B), an NRTI (DDI, panel C) and an NNRTI (NVP, panel D). Plain colors correspond to the unweighted kernels over sequences from Chapter 4, while the striped lines correspond to the exponential random walk kernels. RF results are also included as an additional benchmark. Note that NMSE scale varies between panels.....118

Figure 5.2. r^2 values (averaged over the 40 replicates) of protease (A) and reverse transcriptase (B).....119-120

Chapter 6

Figure 6.1. A) Compositional linear kPCA over the 88 soils. Color represents pH, while point size stands for the number of different observed taxa. B) pH prediction error distribution over the 40 replicates. C) and D) Top relevant taxa for pH prediction according to RF and cLin-SVM. Standard error across the 40 replicates is marked with error bars.....136

Figure 6.2. Analysis using the Jaccard kernel. A) Nonsmoker/smoker accuracies from taxonomic data: NoseL, NoseR, OroL and OroR models are obtained from single datasets data, Conc from the concatenation of the datasets, and Oro, Nose and All models from MKL. B) Top ten of the global cLin-SVM importances across all body sites. C) Similarity across the kernel matrices of the four sites (Nasopharynx Right and Left and Oropharynx Right and Left). D) Jaccard kPCA of the taxonomic abundances. Color code represent airway site, whereas shape indicates the laterality of the samples.....138

Figure 6.3. A) Accuracy for RF, non-longitudinal kernels (cRBF, cLin) and longitudinal kernels (fLin, fRBF) in the prediction of neonatal diarrhea from all available days. B) Accuracy for RF and non-longitudinal cLin and cRBF kernels from metagenomic data of days 0, 3 and 7 post birth separately. In both panels, the red dashed line marks the accuracy of the random model.....139

Figure 6.4. Above: kPCA of fLin in panel A and of cLin (day 7) in panel B. Below: Microbial signatures at the Genera level. Global importance for the first week is in panel C. Importances for the day 7 according to the cLin kernel and RF are in panels D and E. Standard error across the 40 replicates is marked with error bars.....140

Chapter 7

Figure 7.1. A possible classification of the canonic amino acids by their physicochemical properties. Taken from Esquivel *et al.* (2013). License: [CC BY 3.0](https://creativecommons.org/licenses/by/3.0/).....153

Figure 7.2. Metagenomic analysis workflow when using the kernel framework. The pivotal position of the kernel matrix is clearly observed. In grey, several tasks not performed during the present work but that merit future research.....155

Annexes

Figures S1-S9. NMSE distribution for 17 antiretroviral drugs. Same legend as that of Figure 4.1.171-175

Figures S10-S27. RF relative importance of each protein position, averaged over 40 replicates, for 18 drugs. Asterisks mark the major drug-related positions reported in the literature.....176-184

Figures S28-S46. The unweighted (A) and weighted (B) Jaccard kPCA for 19 drugs. Gray dots represent sequences with missing resistance value.....185-194

Figures S47-S53. NMSE distribution for 14 drugs. Same legend as that of Figure 5.1.....196-199

Figures S54-S71. The expJac kPCA for 18 drugs. Gray dots represent sequences with missing resistance value..199-208

Figure S72. Soil kPCAs for the cRBF kernel, JSK and the Jaccard kernel. Legend as in Figure 6.1A.....209

Figure S73. Morton *et al.* (2017) revisit Lauber *et al.* (2009) data. The Correspondence Analysis in panel a shows a clear U-shaped effect. Panel c demonstrates the band nature of the Soil dataset.....210

Figure S74. Original results by Lauber *et al.* (2009). Left: MDS plot derived from Unifrac distances with shape indicating soil pH. The arch pattern is visible but less steep than in figures 6.1A, S1 and S2 because of the coarser taxonomic resolution in the original study (Morton *et al.*, 2017). Right: soil pH correlation to number of phylotypes.....210

Figure S75. Original results of Charlson *et al.* (2010). Above: PCoA comparison of the taxonomic abundances. Colors denote body sites: oropharynx (red), nasopharynx (pink) and fecal (blue). Below: RF missclassification (1-Accuracy) versus missclassification of the random model (Guess).....211

Figure S76. Smoker/nonsmoker prediction accuracy of cLin, cRBF and JSK+SVM. Legend as in Figure 6.2A.....212

Figure S77. Top ten relevant taxa for the nasopharynx (A) and oropharynx (D) MKL models, and for the four body sampling sites separately (B, C, E, F)....212

Figure S78. cLin, cRBF and Jensen-Shannon kPCA. Legend as in Figure 6.2D.....213

Figure S79. fRBF and cRBF kPCA for the ASV data. Legend as in Figure 6.4 A and B.....213

Figure S80. Pig dataset: importance distribution at the Phyla, Family, Genera and Species level of the top 5% for cLin and RF (day 7) and fLin (global).....214

LIST OF TABLES

Chapter 1

Table 1.1 Common data types in genomics and genetics. Adapted from Schölkopf *et al.* (2004).....40

Table 1.2. An example of possible activation functions, loss functions and number of neurons at the output layer in MLP architectures for regression, binary and multi-class classification.....56

Chapter 4

Table 4.1 Final number of HIV isolates per drug.....90

Table 4.2. Linear model coefficient estimates and p-values.....99

Annexes

Table S1. Mean NMSE for all 21 analyzed drugs. wLIN, wRBF, wOV and wJAC stand for (individual) weighted Linear, RBF, Overlap and Jaccard models. sLIN, sRBF, sOV and sJAC correspond to their stacked counterparts.....195

Table S2. NMSE standard error for all 21 analyzed drugs. Abbreviations as in Table S1.....195

Table S3. SVM candidate hyperparameters' values.....216

LIST OF PUBLICATIONS

The present PhD thesis is based on the articles listed below:

- **Ramon, E.**, Belanche-Muñoz, L., & Pérez-Enciso, M. (2019). HIV drug resistance prediction with weighted categorical kernel functions. *BMC bioinformatics*, 20(1), 410.
- **Ramon, E.**, & Belanche-Muñoz, L. (2020). Structural and functional associations in HIV drug resistance prediction with kernel methods (*in preparation*).
- **Ramon, E.**, Belanche-Muñoz, L., Molist, F., Quintanilla, R., Pérez-Enciso, M., & Ramayo-Caldas, Y. (2020). kernInt: A kernel framework for integrating supervised and unsupervised analyses in spatio-temporal metagenomic datasets (*submitted*).

As well as the following software:

- **Ramon, E.** (2019) 'catkern': kernel functions for categorical data (version 0.1.0) https://bitbucket.org/elies_ramon/catkern/src [Computer software].
- **Ramon, E.** (2020) 'kernInt': kernel Integration of microbiome analysis methods & data (version 0.1.1) <https://github.com/elies-ramon/kernInt> [Computer software].

Not included works from the same author are:

- **Ramon, E.**, Pérez-Enciso, M., & Belanche-Muñoz, L. (2019, May). HIV Drug Resistance Prediction with Categorical Kernel Functions. In *International Work-Conference on Bioinformatics and Biomedical Engineering* (pp. 233-244). Springer, Cham [Conference paper].
- Santamaría, J., Álvarez-Álvarez, M. M., Esteban, M. E., **Ramon-Gurrea, E.**, & Moral, P. (2018). Dinucleotide (CA) n tandem repeats on the human X-chromosome and the history of the Mediterranean populations. *Annals of human biology*, 45(1), 72-76.

ABBREVIATIONS

3TC	Lamivudine
ABC	Abacavir
AIDS	Acquired immunodeficiency syndrome
ANN	Artificial Neural Networks
ASV	Amplicon Sequence Variant
ATV	Atazanavir
AZT	Zidovudine
BCD	Bray-Curtis Dissimilarity
BIC	Bictegravir
CAB	Cabotegravir
clr	Center log-ratio
CSS	Cumulative Sum Scaling
D4T	Stavudine
DDI	Didanosine
DRV	Darunavir
DT	Decision Trees
DTG	Dolutegravir
EFV	Efavirenz
ETR	Etravirine
EVG	Elvitegravir
FPV	Fosamprenavir
HIV	Human Immunodeficiency Virus
IC50	Half maximal inhibitory concentration

IDV	Indinavir
ilr	Isometric log-ratio
INI	Integrase Inhibitor
JSD	Jensen-Shannon Divergence
JSK	Jensen-Shannon Kernel
kPCA	Kernel Principal Components Analysis
LD	Linkage Disequilibrium
LPV	Lopinavir
MDS	Multidimensional scaling.
MKL	Multiple Kernel Learning
ML	Machine Learning
MLP	Multilayer perceptron
MSE	Mean Squared Error
NFV	Nelfinavir
NGS	Next Generation Sequencing
NMSE	Normalized Mean Squared Error
NNRTI	Non-nucleoside Reverse Transcriptase Inhibitor
NRTI	Nucleoside Reverse Transcriptase Inhibitor
NVP	Nevirapine
OTU	Operational Taxonomic Unit
PC	Principal Component
PCA	Principal Components Analysis
PCoA	Principal Coordinates Analysis
PhILR	Phylogenetic isometric log-ratio
PI	Protease Inhibitor
PSD	Positive semi-definite
RAL	Raltegravir
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest

RNA-seq	RNA sequencing
RPV	Rilpivirine
RSS	Residual Sum of Squares
SQV	Saquinavir
SVM	Support Vector Machine
SVM-RFE	Support Vector Machine – Recursive Feature Elimination
TDF	Tenofovir
TPV	Tipranavir
WHO	World Health Organization

SYMBOLS

General

\mathcal{X}	Data space
S	A given dataset
N	Number of objects of a given dataset
D	Object dimension (number of variables)
T	Time dimension (number of time points)
x	Features (predictor variables)
y	Target variable of a prediction model
t	Time variable
F	Model obtained by using a supervised ML method
K	Number of cross-validation partitions
N_{tr}	Number of training instances
M	Number of different data sources
m	Number of categories (or modalities) of a categorical variable
\mathcal{A}	Set of canonical amino acids
\emptyset	Empty set
S^D	Simplex
c	Constraining positive constant (compositional data)
d	A distance
s	A similarity
d_A	Aitchison distance
d_E	Euclidean distance
δ	Eigenvalue
\mathbf{v}	Eigenvector

Λ	Diagonal matrix containing eigenvalues
U	Matrix containing eigenvectors
I	Identity matrix
E	Covariance
χ^2	Chi-square distribution
O	Asymptotic upper bound of an algorithm
<i>geo</i>	Geometric mean
p	Frequency or proportion
$++$	Vector concatenation
\otimes	Kronecker product

Graphs

G	Graph
V	Set of nodes (vertices) of a graph
E	Set of edges (arcs) of a graph
v	Node
ω	Weight of an edge
A	Adjacency matrix of a graph
G_{\times}	Product graph
V_{\times}	Set of nodes (vertices) of the direct product graph
E_{\times}	Set of edges (vertices) of the direct product graph
A_{\times}	Adjacency matrix of the direct product graph

Linkage Disequilibrium

D	Matrix containing linkage disequilibrium coefficients
r^2	Matrix containing correlation coefficients of linkage disequilibrium

Decision Trees

f_n	Decision (splitting) function of the n -th node
\hat{y}	Average target value in each branch after splitting.

Artificial Neural Networks

φ	Combination function
ϕ	Activation function
ϕ_o	Activation function at the output layer
λ	L ² regularization parameter

Kernel methods

κ	Kernel (<i>i.e.</i> kernel function)
\mathbf{K}	Kernel matrix
\mathcal{X}	Original (input) space
H	Feature space
ϕ	Implicit mapping used by the kernel
\mathbf{w}	Normal vector of the optimal separating hyperplane (SVM)
b	Model intercept (SVM)
HL	Hinge loss function
L_ε	ε -insensitivity loss function
ξ	Vector of slack variables (SVM)
α, η	Lagrange multipliers (SVM)
C	Cost hyperparameter (SVM)
ε	Epsilon hyperparameter (SVM for regression)
σ	Sigma hyperparameter (RBF)
γ	Gamma hyperparameter (RBF, Overlap, Jaccard, exponential random walk, cRBF)
DJ	Difference in cost function
β	MKL coefficients
\mathbf{w}	Vector of weights (weighted kernels)

CHAPTER 1

GENERAL INTRODUCTION

The inference of phenotypic traits from genotypic data has the utmost importance in biology. The present chapter introduces several themes that are at the core of this thesis: the complexity of the genotype-phenotype map (section 1.1), how this has spurred the generation of huge amounts (both in diversity and in volume) of data, and the current application of machine learning as a way to analyze them (sections 1.2 and 1.4). We overview some typical kinds of data and structured study designs (sections 1.2 and 1.3) in the field, and summarize the machine learning basics that underlie this thesis (section 1.5), including widely used methods like Decision Trees, Random Forests and Artificial Neural Networks (section 1.6). Finally, we outline some of the challenges faced by most machine learning methods when applied to current genetic and genomic data.

1.1 Phenotype-genotype relationship

One of the most striking traits of the living beings is their great variability. Variation is not limited to the inter-individual level: individuals themselves also undergo dramatic physical changes with time, from embryo to adult and old age form.

With the term *phenotype* we refer to the observable traits of an organism (Orgogozo *et al.*, 2015). This covers not only its morphological traits, but also its physiology, developmental processes and even behavior. Thus, the scope of phenotype concept is not circumscribed strictly to the “straightforward” and “external” appearance of the organism: it also includes

internal structures or molecules (*e.g.* blood groups), which are not immediately visible to the human eye but which can be observed using different techniques.

The study of the phenotype variation has not been limited to cataloging living organisms according to their traits (as is done, for instance, in taxonomy). The objective is also looking for the underlying causes that make these differences to arise in the first place (Ahnert, 2017). In the 19th century, Mendel's classical studies in pea plants pointed towards the existence of a hereditary information (or *genetic information*) that is passed from progenitors to their offspring, and which is expressed in the form of phenotypes. Three basic rules that explain both inheritance mechanisms and trait expression were summarized in the Mendel's laws. Thus, the concept of phenotype was opposed to that of *genotype*, which corresponds to the *genetic* makeup inherited by the organisms, and whose changes may be mirrored by phenotypic changes (Orgogozo *et al.*, 2015). The basic unit of genetic information was called *gene*, while the different variants of the same gene were referred as *alleles*.

The link from genotype to phenotype has the utmost importance in biology. For instance, the identification of the genetic basis of a disease may allow its diagnosis, a better understanding of its molecular mechanisms, the design of specific drugs to target it and/or personalized therapies based on the patient genetic background. In agriculture and livestock production, it is useful to improve traits with economic interest, *e.g.* using breeding programs. However, in most cases the map from genotype to phenotype is considerably intricate and complex. One-to-one deterministic relationships, in which a single genetic change is both necessary and sufficient to change a phenotypic trait (as in certain rare pathologies like sickle cell disease) constitute a minority of cases (Chial, 2008). First, because the majority of phenotypic traits are polygenic –*i.e.* they are influenced by multiple genes, which may interact. Second, because the genetic information is not the only factor determining phenotype: so is the environment. Therefore, complex phenotypes are produced as the interplay of multiple genes and environmental variables (Orgogozo *et al.*, 2015). Last, because genes need being placed in a cell, and surrounded with certain molecular and physicochemical environments, to be expressed. Nongenetic but heritable cell material, which may control genic expression (*epigenetics*), and/or stochastic events during early stages of the development of an organism can determine some phenotypic aspects (West-Eberhard, 2003).

Large-scale DNA molecular techniques have caused a progressive drift from classical *genetics* (focused on studying individual genes and their effect in a given phenotype) to current *genomics* (the study of all genes of a given organism as a whole, that is, its *genome*). In addition, the global study of genes is not limited to a single individual, but it can be extended

to an environment; this is the case of *metagenomics*, where all microbial genomes of a given sample (*e.g.* a soil or water sample) are analyzed together.

1.2 Data types in genetics and genomics

In all living organisms, as well as some viruses, genetic information is physically stored in DNA molecules. In the rest of viruses this role is played by RNA, although in some of them (*e.g.* the Human Immunodeficiency Virus or HIV) DNA and RNA are sequentially used at different points of their replication cycle. The central dogma of molecular biology states that the information flow within an organism is from DNA to RNA (*transcription*) and from a specific class of RNA (mRNA, shortened form of *messenger RNA*) to proteins (*translation*). In the particular case of group VI and VII viruses (*e.g.* HIV), *reverse transcription* from RNA to DNA is also possible. Reproduction implies the copy of the whole genome (be it DNA or RNA) by the cell machinery.

Although DNA (and, secondarily, RNA) is the basic object of study in genetics and genomics, the aforementioned complexity of the genotype-phenotype map has widened the lens to include the analysis of higher-level molecular data and processes. For instance, transcriptomic technologies allow studying the ensemble of all RNA transcripts of a cell or tissue, proteomics the same with the proteome, and metabolomics with the metabolites. Overall, this has allowed focusing on different aspects of cells, tissues and organisms (Gligorijević & Pržulj, 2015). When dealing with a complex phenotype, including these intermediate layers decreases the gap between the trait and the basic genotype information. This hierarchical and multi-level analysis relies on the production of vast amounts of different types of biological data, which are not limited to ‘omics’. Without intending to present a complete list, some examples are given in Table 1.1.

In order to be stored, curated and analyzed, all this biological data generated by different techniques need first be properly represented. Some very general *data representations* typical of the genetics and genomics field are discussed in the following subsections.

1.2.1 Sequences

DNA, RNA and proteins are long linear biopolymers. Their basic units (or *monomers*) are nucleotides in the case of DNA and RNA, and amino acids in the case of proteins. Thus, a straightforward way of representing DNA, RNA or protein chains is as sequences, *i.e.* vectors of characters wherein each monomer is represented as a different letter (see Figure 1.1).

Thus, there is a set of four letters for DNA bases (A, C, G and T), four for RNA bases (A, C, G and U) and twenty for the canonical amino acids.

Table 1.1 Common data types in genomics and genetics. Adapted from Schölkopf *et al.* (2004).

Data type	Details	Representation
Sequences		
DNA	Genes and genomes	String over {A, C, G, T}
Expressed sequence tags, full-length mRNAs and other classes of RNAs	Gene transcriptions	String over {A, C, G, U}
Proteins		String over amino acids
Structures		
Metabolites	Atomic positions and bonds	Labeled graph
Macromolecules	<i>e.g.</i> DNA, RNAs, proteins	Labeled graph
Interactions		
Proteins x Metabolites		Graphs or real vectors (binding energies)
Proteins x DNA		Graphs
Proteins x Proteins		Graphs
Expression / Localization data		
Gene expression	Abundance of mRNAs	Count vector or matrix
Protein expression	Protein abundance	Count vector or matrix
Metabolite expression	Concentration of metabolites	Count vector or matrix
Protein localization	Compartment	Categorical Presence/absence
Cell / Organism data		
Genotype	Single Nucleotide Polymorphisms	Vector of {A, C, G, T}
Phenotype	Observable traits	Vector of real and/or categorical attributes
Clinical data	Disease presence, blood analysis, etc.	Vector of real and/or categorical attributes
Environment	Temperature, pH, etc.	Vector of real and/or categorical attributes
Population data		
Linkage disequilibrium	LOD scores	Real numbers
Pedigrees		Tree-like graphs
Phylogenies		Tree-like graphs

Sequence data is obtained through a highly automated process called *sequencing*. The first DNA sequencers were based on the Sanger sequencing developed in the 70's. This technique allows sequence reads of >500 nucleotides and is adequate for studying single genes or small genomic regions. However, it is very slow when dealing with whole genomes. During the first sequencing of the human genome (which started in 1990 and ended in 2003), it was evident the necessity of new large-scale, high-throughput and low cost (Collins *et al.*, 2003) technologies. The advent of next-generation sequencing (NGS) platforms in 2005 allowed massive parallel sequencing of whole genomes, while the costs dropped dramatically

(National Human Genome Research Institute [NHGRI], n.d.). Figure 1.2 summarizes the differences between the two approaches.

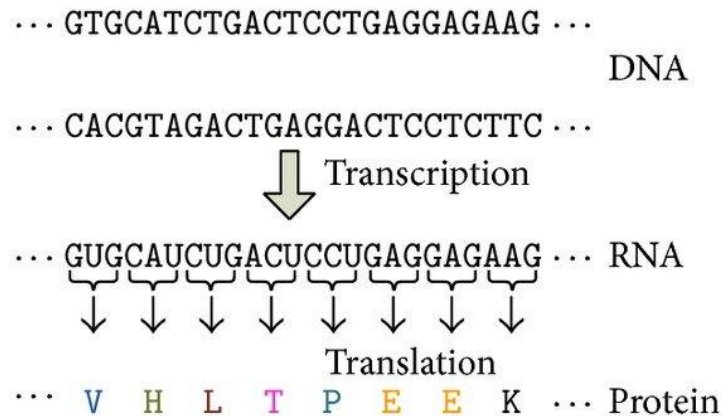


Figure 1.1. DNA, RNA and protein sequences. Figure taken from Cao and Xiong (2014). License: [CC BY 3.0](https://creativecommons.org/licenses/by/3.0/).

RNA and proteins can also be sequenced. However, the process typically relies at some point on DNA sequencing. In the case of RNA, by using reverse transcription that gives complementary DNA (cDNA) strands. Protein sequences can be determined with Edman degradation or mass spectroscopy (Gundry *et al.*, 2010). However, in most cases the knowledge of both the genetic code that governs nucleic acids → protein translation and the posttranscriptional edition processes endured by mRNA (*e.g.* splicing) are exploited to save costs. For instance, >95% of the protein sequences stored in UniProtKB database come from automatic translations of coding regions obtained by DNA sequencing (UniProt, n.d.).

Sequences are probably the most classic data type in genetics and genomics. Detection of differences among sequences (called *mutations*, *polymorphisms* or *alleles*, depending on the context) is fundamental to trace the genetic basis of phenotypes. However, analysis of sequences present some challenges. First, their nature as categorical (not numeric) vectors. Second, because mutations may affect single monomers, as in the case of single nucleotide polymorphisms, but also whole regions of the sequence (*e.g.* insertions, deletions, inversions, copy number variants...). Furthermore, polymorphic positions within the same sequence may be in *linkage disequilibrium* (LD) (*i.e.* their alleles are non-randomly associated) by the mere fact of being physically placed in the same DNA biomolecule. The recombination of homologous DNA sequences that occurs naturally in cells breaks down LD and prevents the association to be absolute. Thus, linkage varies with distance: the closer two positions are in a sequence, the higher the disequilibrium between them, while very distant positions tend to not be in LD at all.

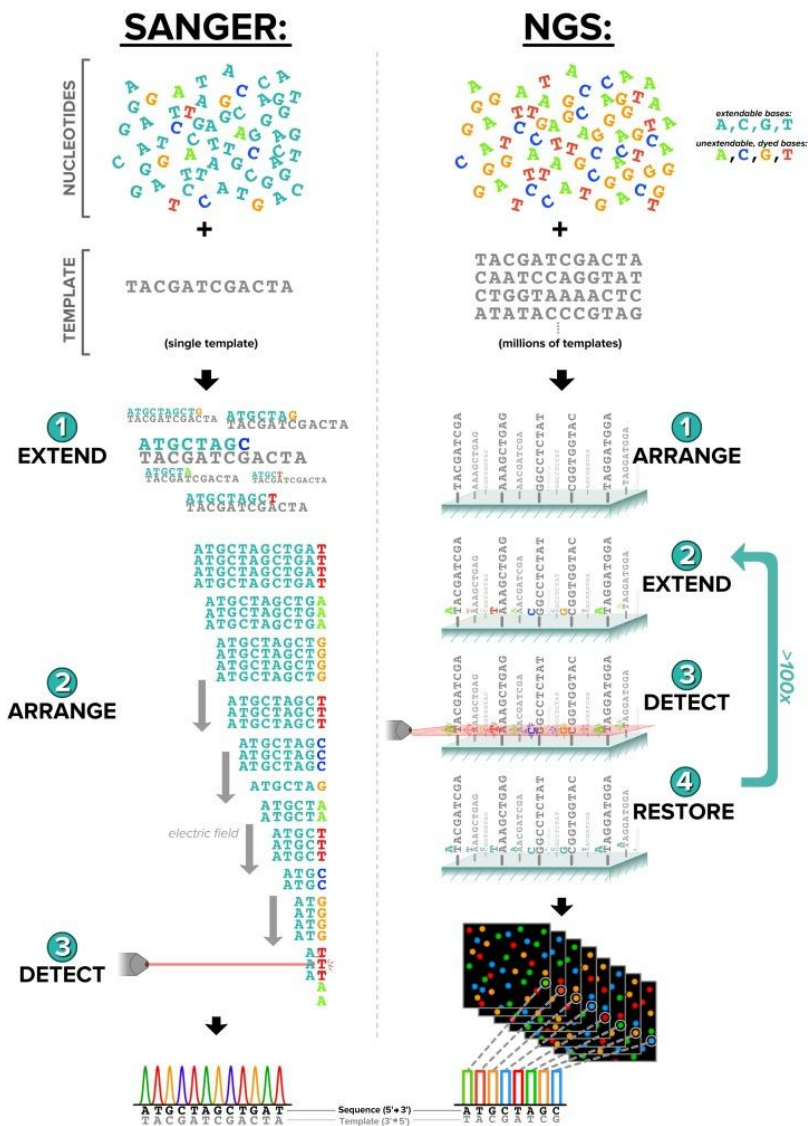


Figure 1.2. Sanger sequencing (left) versus NGS sequencing (right). Taken from Price *et al.* (2018). License: [CC BY-NC 4.0](https://creativecommons.org/licenses/by-nc/4.0/).

1.2.2 Counts

Count data is aimed to quantify the abundance of some trait or analyzed variable. Formally, it consists of non-negative integer values. This is a widespread data type in biological science. For instance, it is usually encountered in ecology, *e.g.*: number of species in a given area, number of individuals in each species, number of offspring, etc. (St-Pierre *et al.*, 2018). The development of NGS technologies also brought this type of quantification analyses to ‘omics’ data. As stated in subsection 1.2.1, NGS were primarily aimed to sequencing applications as genome assembly and variant discovery. However, NGS can be used as well to quantify the abundance of previously known (and well-assembled, annotated, etc.) sequences in a given

cell, tissue or sample (Holmes & Huber, 2018; Quinn *et al.*, 2018). Examples are RNA-seq or microarrays for gene expression profiling (Nguyen *et al.*, 2016). Metagenomics (which rely heavily on NGS) are not focused only on detecting and solving the taxonomic classification of the microorganisms present in a given ecosystem from their DNA sequence, but also on the obtention of their abundances.

Count data present several challenges and is not always easy to handle with traditional statistical analyses (St-Pierre *et al.*, 2018). It has a nonsymmetric distribution, a potentially large dynamic range (from zero up to millions) (Holmes & Huber, 2018), and there exist upper and lower bounds on detection caused, for instance, by the instrument resolution. In some cases, like most metagenomic studies, this data is notably prone to zeros (sparsity) within the “count matrix” (Quinn *et al.*, 2018), which may not reflect an actual absence, but a quantity below the detection limit.

1.2.3 Graphs

The chemical and biological functions of (bio)molecules are strongly dependent on their three-dimensional structure. That is the case of proteins, which perform essential catalytic and structural functions in all living organisms. For instance, *enzymes* (proteins specialized in catalyzing chemical reactions) need to be properly folded to perform their activity in optimal conditions. The structural arrangement of proteins and other biomolecules can be represented using graphs. Apart of three-dimensional relationships, graphs can model other relationships like pathways or networks (*e.g.* metabolic and signaling pathways, gene regulatory networks) or phylogenetic trees (Gligorijević & Pržulj, 2015; Wooley *et al.*, 2005).

Graphs consist of a set of objects, called *vertices* or *nodes*, such that some of these nodes are pairwise related, and that relationship is represented by an *edge*. Graphs can be represented in a diagram form or as an adjacency matrix (Figure 1.3). Each row (and column) of the adjacency matrix correspond to a node, so the elements of the matrix indicate the existence (and in some cases, additional information like the weight) of edges linking each pair of nodes.

1.3 Spatial and temporal structured datasets

Living organisms are very dynamic entities, so part of the phenotypic variation occurs over time and space. For this reason some studies are not limited to single-point data; this is especially true if the objects of analysis are systems or processes, which are not well grasped

with a partial snapshot but with a complete follow-up in its spatial and/or temporal dimension. A typical challenge of this kind of data is autocorrelation, *i.e.* the presence of related samples that structure the dataset in a particular way. For instance, repeated measures of the same subjects over time are dependent on previous measures (Coenen *et al.*, 2020), while in the spatial case there may be more similarities between closer samples than between distant ones.

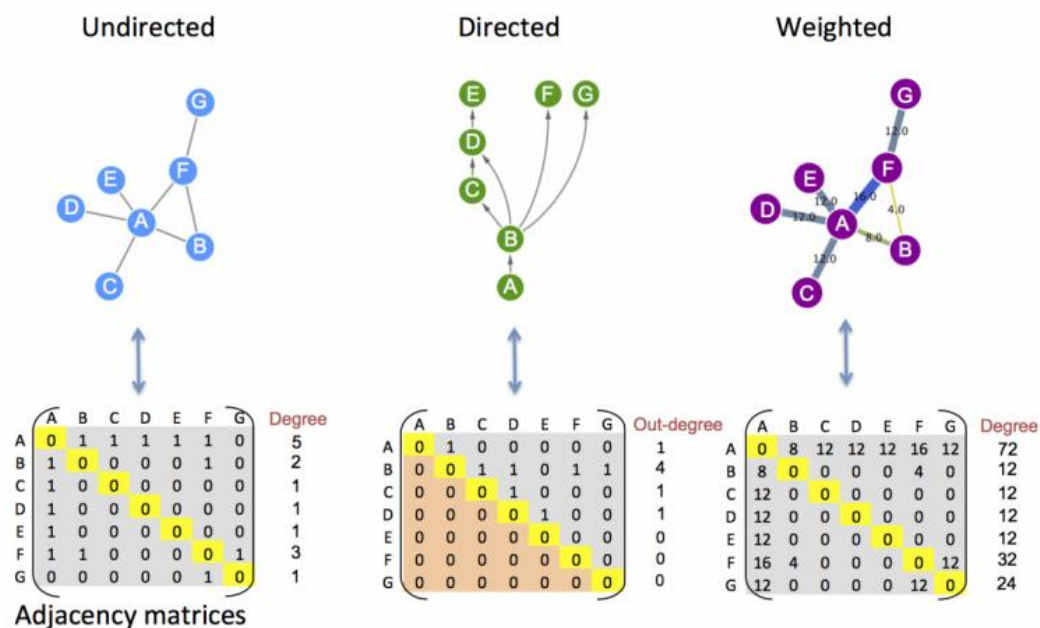


Figure 1.3. Graphs represented as diagrams (above) or as adjacency matrices (below). Taken from EMBL-EBI webpage (<https://www.ebi.ac.uk/training/online/course/network-analysis-protein-interaction-data-introduction/>). License: [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/).

1.3.1 Spatial dimension

The relevance of spatial axis as a dimension that structure life is present in every biological level and resolution. The clearest example is in ecological studies, with the variation of biological populations along the geographic space. For instance, the composition of microbial communities varies across soils in a continental scale (Lauber *et al.*, 2009). There also exist internal spatial structures within a given ecosystem, with different species occupying unique niches (Berg *et al.*, 2020). At the intra-individual level, living organisms can be viewed as multi-compartment systems able to decrease their local entropy (Davies *et al.*, 2013). For instance, cells (especially eukaryotic cells) are strongly structured entities, in which different parts have distinct functions. Returning to the general data types of section 1.2, genomic data in pluricellular organisms is expected to be constant across tissues, but this is not the case of proteomic, transcriptomic or metabolomic data. In fact, a lot of effort has been devoted to search for differential expression profiles in different tissues (Li *et al.*, 2018). In metagenomic

studies, it has also been observed that microbial communities vary across different parts of the body within the same individual (Charlson *et al.*, 2010; Costello *et al.*, 2009). Then, analogously to the case of the multi-level, vertical integration of different 'omic' data, the combination of spatial-structured data allows a more unitary and holistic view of living organisms as complex systems. Formally, data containing different spatial samples of the same individuals can be represented as a collection or list of datasets sharing the same data type.

1.3.2 Temporal dimension

A time series consist in an ordered set of repeated samples indexed by time. When the research design involves the follow-up of several individuals, the resulting data is referred as *longitudinal*. A natural way to summarize time series is through mathematical functions.

Longitudinal analyses have a great relevance in health research. For instance, the purpose of this kind of studies may be inferring a given trait (*e.g.* the outcome of a certain disease) from the evolution over time of the subjects. This study design is expected to be more informative than focusing in a single time point.

The Berkeley Growth Dataset (Tuddenham & Snyder, 1954) gives a good illustration of this approach. This dataset follows the physical growth of 54 girls and 39 boys from the ages 1 to 18 (Figure 1.4, left). It can be observed that, for most of the follow-up, the height distribution in boys and girls is similar. Age 18 is the time point with most marked difference between the two groups (mean heights: 180 vs. 166 cm). However, even in the 18-years-old, both distributions are not completely set apart, and have an overlap area of 0.29 (Figure 1.4, right). This is to be expected, as height is a multifactorial trait that is not only polygenic –it involves more than 400 *loci* according to Wood *et al.* (2014)– but, also, is affected by environmental factors as poor nutrition (Roser *et al.*, 2013). This illustrates that, even at the time point of the dataset when the two groups have more distant distributions, height alone is an imperfect predictor of sex.

Let's then focus not on height, but on *height evolution over time* in girls and boys. Specifically, we will focus on the puberty growth spurt (see Figure 1.5), as is the time when more striking differences between the two groups arise. Typical puberty in girls is dominated by the sex hormone estradiol, which accelerates both the growth spurt and the closure of epiphyseal plates, thus stopping the growth (Weise *et al.*, 2001). Estradiol is produced by ovaries but also in low levels as a part of testosterone metabolism, and in this case the physical growth is slowed and prolonged. This explains why epiphyseal closure is reported to happen between

ages 12-16 for girls and 14-19 for boys (Crowder & Austin, 2005). In this case, as illustrated in Figure 1.5 and Usage – Longitudinal data ('kernInt' vignette, Annexes), taking into account the whole time series is more informative than taking a single time point, even if it is that of maximum separation between the groups.

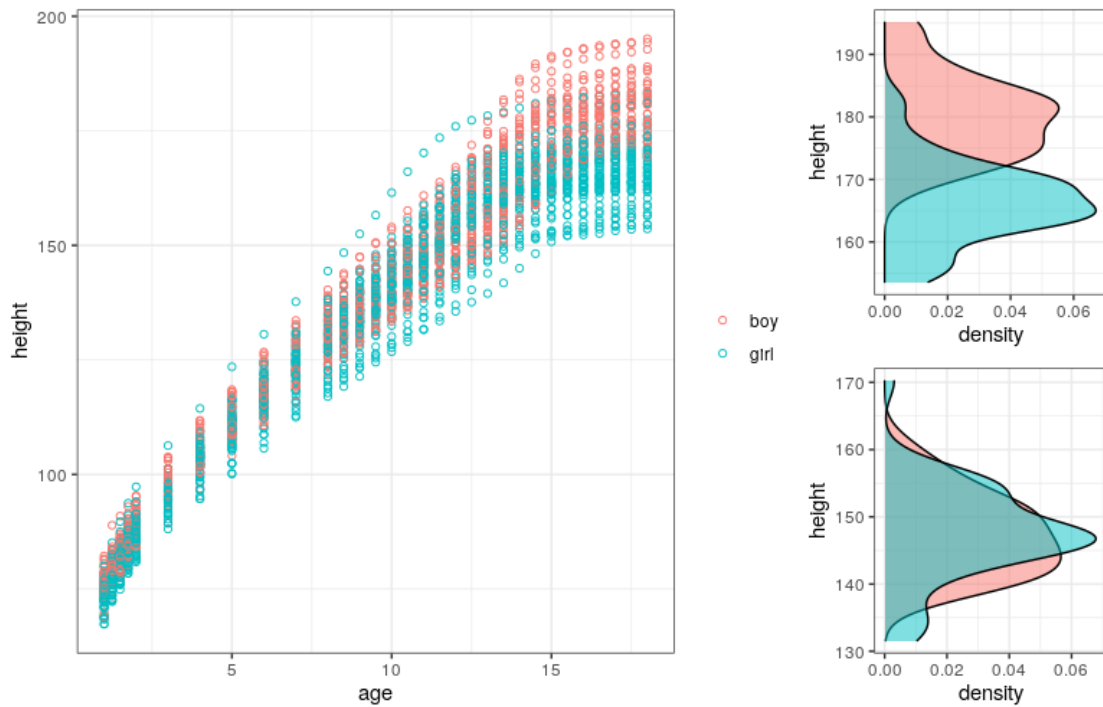


Figure 1.4. Left: height versus age of 54 girls and 39 boys (Berkeley Growth Dataset). Right: height distribution for girls and boys at age 18 (above) and age 11 (below).

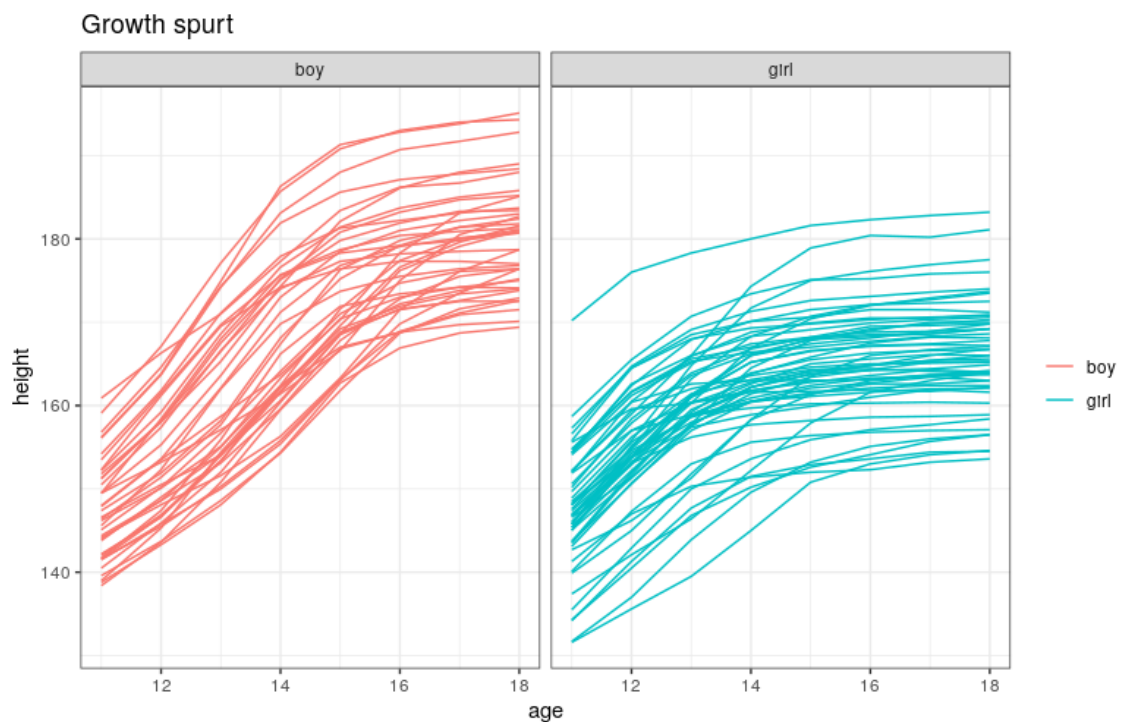


Figure 1.5. Growth profile during puberty for the boys (left) and girls (right) of the Berkeley Dataset.

1.4 The *big data* paradigm in biology

The advances that led to NGS and ‘omics’ technologies have had a profound impact in modern molecular biology. The classical genetics approach was insufficient to tackle the inherent complexity of the genotype-phenotype relationship, thus justifying genomic studies that could get a better grasp of complex diseases and phenotypes. Nowadays, large volumes of big and heterogeneous datasets, with a high variety of data types and hierarchical layers (*i.e. big data*), are produced routinely. However, these datasets may be difficult to analyze with traditional, one-variable-at-a-time methods (Yip *et al.*, 2013). Thus, big data revolution not only has turned biology into a data-intensive enterprise: it also has affected deeply the approaches to research itself.

Traditional biology research is primarily directed by the hypothesis-driven scientific method. This implies adopting a deductive approach: first formulate a hypothesis, then test it with an experiment, analyze the results and finally accept or reject the hypothesis (Mazzocchi, 2015). In contrast, data-driven research is inductive. The aim is finding patterns or hidden structures in data examples that may lead to generalize on other samples drawn from the same source (Shawe-Taylor & Cristianini, 2004). In biology, the Human Genome Project was a historical turning point that spurred a more data-driven way of thinking research (Collins *et al.*, 2003). Although the over-reliance in this paradigm has been criticized (Mazzocchi, 2015), the fact is that data-driven methodologies are becoming increasingly attractive to deal with the massive amount of NGS and ‘omics’ datasets. An example of data-driven approach is *machine learning* (ML). ML allows to automatically identify patterns in data, which is highly useful when the problem at hand is novel or a very complex one, when the expert knowledge is incomplete or scarce, or when the amount of available data is simply too large to be handled. As in this situation ML can be the best option or even the only one feasible, it is not surprising that its popularity had risen in the last years. The following sections provide a primer about ML and its application to biological sciences.

1.5 Machine Learning primer

In a broad sense, an *algorithm* can be defined as the steps that should be followed to solve a problem or performing a task. More precisely, an algorithm transforms a given *input* to the desired *output* through a finite sequence of unambiguous instructions (Alpaydın, 2010). An essential part of computer science involves designing and implementing algorithms. Design

and implementation tasks are usually carried out by a programmer, while the computer role is running the program, thus obtaining the output from the given input in an automatic way.

However, there exists a subset of problems for which designing an algorithm is a very complex or even impossible task. An example is face recognition. Most people recognize faces effortlessly but, at the same time, they cannot define or instruct others how to do so. In other cases, an algorithm may be effectively defined, but is too simple to grasp the intricacies of the problem. *Machine learning* field is focused in providing tools to solve this kind of challenges. When a human-defined algorithm is either difficult, insufficient or impossible to achieve, we can let the computer to automatically generate it from available data examples. The key idea is that the computer *learns* underlying patterns from data that are useful for solving the problem at hand. Human intervention is no longer dictating explicit instructions to do so, but to assist the learning process. In this thesis, the human-selected and designed algorithm that guides learning is called a ML *method* (and sometimes simply *method* or *algorithm* or even *machine*). Instead, a *model* always refers to the automatically learned algorithm whose aim is solving a particular problem.

Depending on the final goal and how the learning is performed, ML field is divided in several areas:

1.5.1 Supervised learning

In supervised learning, we have a set of N data objects $\{x_1, \dots, x_N\}$ such that each object is associated or paired to a label $\{y_1, \dots, y_N\}$. The objects consist of *features* or *predictor variables*, while the label is also called the *target variable*. The main objective is building a predictive model, *i.e.* a model that from the feature variables (the input) is able to return the right label as output. For instance, one might want to use genomic data x for predicting a certain phenotype y . To do so, it is necessary to find a map $F: x \rightarrow y$. In turn, obtaining this map (*i.e.* the model) from example data is the desired result of using the ML method (Bishop, 2006).

In supervised learning, the most important thing about a model is its generalization power. A model is said to *generalize* when is able to correctly label new data examples extracted from the same source (Alpaydm, 2010). Then, not only is it important to build a model, but also evaluating how good it is at predicting previously unseen data. To do so, the learning process typically consists in splitting the original dataset in two parts: the *training set* and the *test set*. The training set is the fraction of data used to build the model, while the test set is the fraction of data reserved to evaluate its prediction performance. In a first step, a model is *learned* using the knowledge extracted from the features-target pairs of the training set.

Methods typically do so by minimizing the misfit, measured by a loss (error) function, between the original labels of the training data and the models' labels. On a second step, the feature variables of test data are used as input to the model, while the real labels are withheld. The generalization ability is assessed afterwards by comparing the model's predicted labels with the observed (real) labels, and computing the prediction performance. However, for the results to be reliable, the training and the test set should be completely independent.

For a model, there exist two ways to “wrongly” grasp the underlying nature of a problem. If the model is not complex enough to capture the relationship between features and the target variable (for instance, a linear regression is used to model a nonlinear relationship), we have a poor performance caused by *underfitting*. Instead, *overfitting* appears when the model is too complex and takes the noise of the training set for meaningful variation. Overfitted models do not generalize well: they perform much better in the training set (in some way, they memorize it) than in the test set. Both concepts are related to the *bias/variance trade-off*. There is bias if we observe a systematic error when training the method with different example data. That means that the model is missing relevant relationships between the features and the target, which is related to underfitting. Instead, there is variance if prediction is highly changing in the different training sets used to generate the model. Variance is akin to overfitting, as it indicates too much dependence on the specific training data. The optimal solution is the one with best balance between bias and variance, and that does not fall in underfitting, nor in overfitting.

Supervised methods typically provide a way to assist the learning process and avoid under/overfitting: an example is the tuning of *hyperparameters*. In ML, hyperparameters are parameters whose value is not inferred during training (like normal parameters) but that should be provided externally. Hyperparameters introduce flexibility in the modeling process. *Regularization* is another technique that is often used in supervised methods to control overfitting. Regularization consists in adding a penalty term to the error function, which hinders the model's coefficients to become excessively large and controls its effective complexity. Within the new error function, the weight of this regularization term (with respect to the error term) is governed by a regularization hyperparameter. Either way, the values of the hyperparameters may be fixed *a priori* or by selecting the best one from a set of candidate values. In the latter case, a *validation* step should be added between training and test. During this stage, different models (say, with different hyperparameter values) may be compared, so the best one is selected. The original dataset may be split in training, validation and test sets or, if data is scarce, a related technique called cross-validation can be used

instead over the training set (see Figure 1.6). The (cross)validation results assess how the models will generalize when faced to an independent test set.

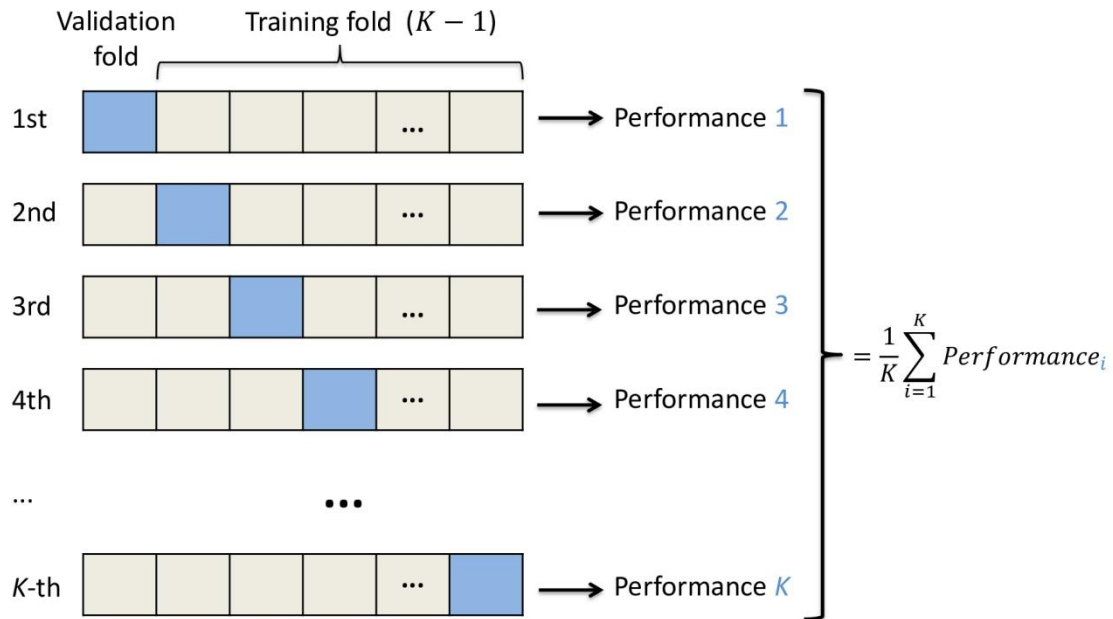


Figure 1.6. K cross-validation. K itself is a hyperparameter whose value has to be decided.

There exist different types of supervised problems. Depending on the nature of the target (or targets) y , the two main types are:

- *Classification*: In this case, the target is categorical – *i.e.* it can take as value a finite number of discrete categories (Figure 1.7, left).
- *Regression*: The target is continuous (Figure 1.7, right).

Some supervised methods are specialized either on classification or on regression, while others may be used for both (Bishop, 2006).

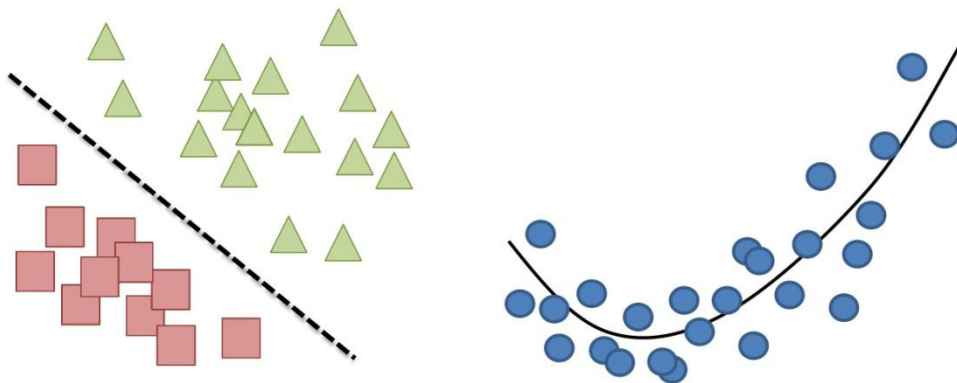


Figure 1.7. Visual example of classification (left) and regression (right) models.

1.5.2 Unsupervised learning

Unlike supervised learning, data objects in unsupervised learning are unlabeled. Therefore, there is not a ‘correct’ solution known beforehand. The objective of unsupervised learning is instead finding patterns and hidden associations that structure the dataset.

There exist different types and applications of unsupervised learning. Some of the most widely used are summarized next:

- *Clustering*: The aim is finding groups (or clusters) in data (Alpayđın, 2010). Thus, objects belonging to the same cluster are expected to be more similar to each other than to the remaining objects.
- *Outlier / novelty detection*: The approach is in some sense opposite to clustering. Here, the main goal is the identification of “rare” objects, *i.e.* objects that differ from the rest and cannot be easily placed in any group.
- *Ordination*: Ordination techniques intent to reduce the complexity of a dataset. If some degree of redundancy across the original variables exists, they can be rearranged and combined to derive new variables (*feature extraction*). This is especially useful to “compress” datasets with a large number of variables (*i.e.* high-dimensional). Another utility is visualization: the compression of the original features to two or three variables enables data projection to \mathbb{R}^2 or \mathbb{R}^3 and its graphical visualization, as in Principal Component Analysis (PCA).

1.5.3 Other types of learning

There exist other types of learning that cannot be classified as unsupervised or supervised. For instance, *semi-supervised learning* combines the two previous approaches. Here, the dataset consist of some labeled data instances together with a large amount of unlabeled data. The basic idea is to apply an unsupervised approach to detect which objects are similar, and then using the labeled examples to assign labels to the rest of data. Semi-supervised learning is especially useful when acquiring large volumes of unlabeled data is easy and/or cheap, whilst the process of labeling is comparatively expensive. Semi-supervised learning, for instance, has been applied to drug-protein interaction prediction (Xia *et al.*, 2010).

In the case of reinforcement learning, the learning process is aimed to find an optimal sequence of actions for reaching a goal. Thus, the method is not given output examples, but has to discover the best sequence by trial and error (Bishop, 2006). Typical application examples are games like backgammon or chess. Beyond playing games, reinforcement learning can be applied to complex interactions between a living organism and its

environment, for instance predator-prey relationships or cases of habitat destruction (Neftci & Averbeck, 2019).

1.6 ML methods applied to genetics and genomics

Currently, ML is applied to a broad range of areas within genetics and genomics. ML methods have been used to find patterns in a wide variety of data (*e.g.*: DNA sequences, microarray or RNA-seq expression data, DNAase-seq, protein sequences, and more) and to solve problems like prediction of transcription start sites, promoters or enhancers, to identify potentially valuable disease biomarkers or to assign functional annotations to genes (Libbrecht & Noble, 2015). In phenotype prediction problems, a great range of different supervised methods have been tested. Without intending to perform an exhaustive review, popular supervised methods in current biology like Decision Trees, Random Forests and Neural Networks are presented in subsections 1.6.1-1.6.3, while an introduction to kernel methods can be found in section 3.1.

1.6.1 Decision Trees

Decision Trees (DT) are a type of supervised learning models. Their structure is reminiscent to that of a tree (Figure 1.8): they consist in directed, hierarchical graphs with a root, internal decision nodes, branches and terminal leaves. Starting at the root, each node n implements a decision function $f_n(x)$. These decision functions interrogate each data object, so the outcome will redirect it to a particular node on the next layer. This process is repeated recursively until the object arrives to a terminal leaf, at which point the particular label on the leaf constitutes the output (Alpaydın, 2010). DT can be used both for classification and regression. In the former case, each terminal leaf represent a different class of the target variable, while in the latter case the leaves contain intervals of numeric values.

There exist different DT methods for producing the DT model. In all cases, the final goal is the generation of a map $F: x \rightarrow y$ (*i.e.* the DT) in which F is broken down to decision functions $f_n(x)$ (the nodes) ordered with a certain hierarchy. $f_n(x)$ are simple functions, which often split the input space according to if a variable is higher or lower than a threshold (with continuous feature variables) or by specific classes (with categorical variables). There exist different DT subtypes that differ in how the $f_n(x)$ are generated. However, in all approaches, and even from a fixed number of nodes, the determination of the tree's optimal structure (and this includes which features are chosen for each split) by testing all possible

combinations is computationally infeasible (Bishop, 2006). Instead, all DT methods carry out a greedy optimization process: that means that, instead of searching for the global optimum, they try to find a local optimum choice at each step. Thus, the root node first splits the whole training set, and then the tree is grown “organically” by sequentially adding nodes that further split the space. This strategy might not lead to the optimal tree arrangement, but it may approximate it with an inferior computational cost. At each step, the set of variables that is used to split (as well as the threshold for each one of them) is chosen. Fortunately, this joint optimization can be efficiently implemented via exhaustive search.

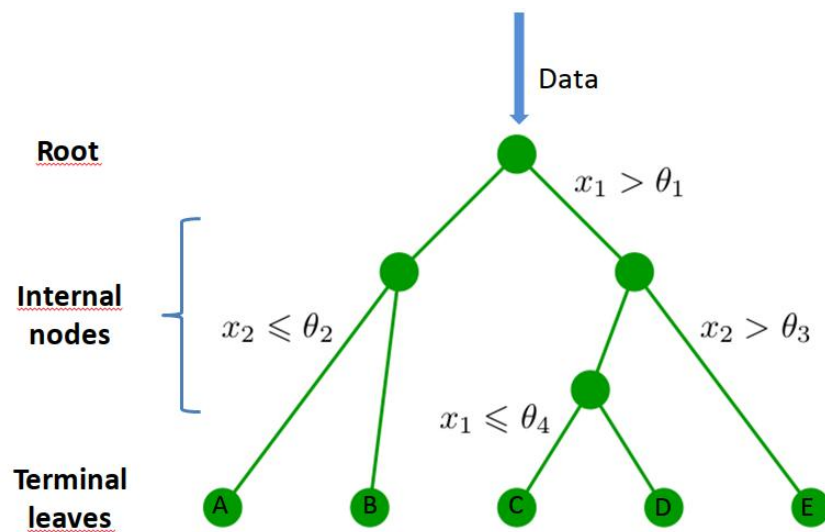


Figure 1.8. Decision Tree for a classification problem. Adapted from Bishop (2006). Copyright (2006) by Christopher M. Bishop.

In classification trees, the “goodness” of a node is quantified by its *impurity*: a split is pure if (for all its output branches) all the objects on a branch belong to the same class. In that case, it is not necessary to split anymore, as all objects have been successfully separated; there only rest to add the terminal leaves. Instead, if the node is not pure, more splits are required to decrease impurity. Among all possibilities, we look for the split that minimizes node impurity, which is the difference between the impurity of branches reaching node n and the impurity in its output branches after the split (Alpaydın, 2010). In classification, the impurity may be quantified in different ways. One of them is the Gini index:

$$Gini = 1 - \sum_c p(c)^2 \quad (1.1)$$

where $p(c)$ is the proportion of training data in a given branch that belong to class c .

For regression, the impurity is measured by residual sum of squares within that node:

$$RSS = \sum_{i=1}^{Ntr} (y_i - \hat{y}_i)^2 \quad (1.2)$$

where Ntr is the number of training instances arriving to the node, y their actual target values, and \hat{y} their average value in each branch after splitting.

DT present several advantages over other methods. They are computationally fast, accept numeric and categorical data in the target variable and also in the feature variables, have an inherent in-built variable importance metric (the total decrease in node impurity) and, therefore, perform their own internal *feature selection* by not using irrelevant variables. Most of all, they are highly interpretable: DT models consist in a set of if/then rules that mirrors human decision making, and that can be graphically displayed. Their main weaknesses are: overfitting tendency, great dependence on the particular composition of the training set, and lower prediction performance than other ML methods (Alpaydın, 2010).

1.6.2 Random Forests

Random Forests (RF) consist of an ensemble of DT. The underlying idea is that setting a group of DT (the “forest”) to predict a given target and then combining their predictions is a better approach than relying on a single DT. RF belong to a specific type of ensemble methods called *bagging*. Bagging means “bootstrap aggregating” and is a learning technique focused on decreasing the model variance. Thus, RF was proposed as a way to correct the instability and propensity to overfit of DT. The general steps followed by the method are:

1. The original training set is sampled with replacement L times (bootstrap), so L slightly different versions of the training set are obtained.
2. Each one of the training sets is used to grown a different DT. However, to avoid correlation across the trees, a different random subset of the D original features is used at each candidate split. A classic heuristic consist in using \sqrt{D} features in classification and $D/3$ features in regression.
3. In the test stage, the L trees independently predict the label assignation on each object, and finally:
4. the definitive prediction is computed as the most voted class (in classification) or the mean value (in regression).

1.6.3 Artificial Neural Networks

The term “Artificial Neural Network” (ANN) covers a wide set of different types of models, so broad that there exist specific variants for supervised, reinforcement and unsupervised

learning, which have in common that their structure loosely remind that of a biological neural network. The general structure of an ANN consists in a directed graph, where each *neuron* (*i.e.* node) receives an input, transforms it with a *combination function* $\varphi: \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$, and produces an output that (analogously to the activation threshold of biological neurons) may be smoothed by an *activation function* $\phi: \mathbb{R} \rightarrow (a, b)$. There exist different functions that can fill the latter role, as the logistic sigmoid, the hyperbolic tangent, or the ReLU (Rectified Linear Unit). Activation functions introduce nonlinearity, while the combination function is often a linear combination over input data using certain coefficients. Neurons are interconnected and have multiple possible inputs and outputs. Each connection is assigned a weight w that controls their importance. If the resulting graph is acyclic (*i.e.* the information moves in only one direction) the resulting model is a *feedforward neural network*. That was the first type of ANN devised. Instead, the *recurrent neural networks* include loops in their structure.

The most known ANN architecture is probably the Multilayer Perceptron or MLP. This is a type of feedforward neural network wherein neurons are organized in layers. Neurons of a given layer are directly connected only to the precedent and following layers (Figure 1.9). The *output layer* produces the final outcome, *e.g.* in supervised learning, the labeling of the object. The activation function ϕ_o and number of neurons of this layer depend on the nature of the output: for example, in supervised learning, if the target is continuous or categorical – *i.e.* if the network is intended to perform regression or classification (Table 1.2). The preceding layers are called *hidden layers*, as their output is used as the feeding of the subsequent layers and, therefore, is “hidden”. *Deep learning* architectures are a subtype of ANN characterized for having multiple hidden layers.

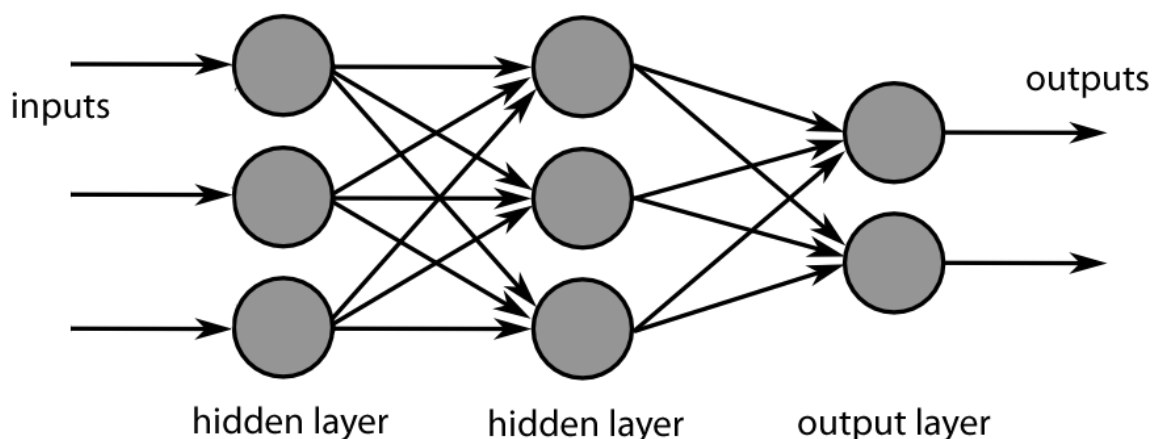


Figure 1.9. Graphical representation of a MLP. This specific architecture is denoted as 3-3-2. Taken from: https://commons.wikimedia.org/wiki/File:MultiLayerNeuralNetworkBigger_english.png. Author: Christoph Burgmer. License: [CC BY-SA 3.0](https://creativecommons.org/licenses/by-sa/3.0/).

Neural networks are a very general class of parametric nonlinear functions. Thus, they are very complex maps F that have a vector x as input and give in return a certain output y (Bishop, 2006). The learning process of a MLP implies determining a great quantity of parameters (the weights). Instead, the number of layers and neurons per layer are hyperparameters, which may be fixed *a priori* or chosen by cross-validation. The determination of network parameters can be done by maximum likelihood, which results on a nonlinear optimization problem that involves the derivatives of the log-likelihood function with respect to the MLP parameters. This is carried out with relative efficiency through *error backpropagation* (Bishop, 2006). Thus, we turn maximization of the log-likelihood to a minimization of a loss function, which depends on the activation function at the output (Table 1.2). The MLP is feed forward to fit the objects on the training set and the difference between true and fitted values is computed. This is followed by a backward process of computing on each layer the gradient of the loss function with respect to each weight, iterating until the first layer is reached. This way, the weights can be updated. The whole process should be repeated (each one of this iterations is called an *epoch*) until convergence.

Table 1.2. An example of possible activation functions, loss functions and number of neurons at the output layer in MLP architectures for regression, binary and multi-class classification.

	Regression	Binary Classification	Multi-class classification
ϕ_o	Identity	Logistic sigmoid	Softmax
Loss function	Sum-of-squares	Cross-entropy	Multiclass cross-entropy
Number of neurons at the output layer	As many as target variables	One	As many as classes

Using ANN presents important advantages. First, they can approximate any continuous function to an arbitrary accuracy (Zhou, 2020). Thus, ANN are able to model complex nonlinear relationships between dependent and independent variables, and also detect all possible interactions between predictor variables (Tu, 1996). There exist subtypes of ANN that can handle structured data, *e.g.* Convolutional Neural Networks for time series (Binkowski & Donnat, 2018). In general, they have been successfully adapted to all classes of supervised, unsupervised and reinforcement learning problems.

Some of the ANN drawbacks are related to the many architectures available and the large number of parameters they need. ANN (especially if large and complex) require considerable time and computing resources during the training phase. They are prone to overfitting (Tu, 1996) and may have problems when the number of dimensions is much higher than the number of data objects. Also, unlike RF, they function like “black boxes”: a network can be very successful (*e.g.* at predicting) in the context of a particular problem, but that does not

mean that it is easy to determine why. For instance, it is not trivial to retrieve which variables are the most important contributors to a particular output; also, the model may rely in a certain number of unimportant predictor variables.

1.6.4 Challenges of ML applied to genomic data

Although ML methods have been successfully applied to a great variety of problems, they face several general challenges. As seen in section 1.2, typical data in genetics and genomics does not consist in “simple” real vectors, but in more complex data types with some kind of structure. Also, in section 1.3 it is shown that certain study designs produce related samples that structure the dataset in a particular way. However, the standard input data type in most ML methods is real data (Schölkopf *et al.*, 2004). That is the case of ANN, while RF can additionally accept categorical data. In general, this forces the recoding of complex data types into numeric vectors. For example, categorical data may be recoded by *one-hot encoding*: for each categorical variable, m or $m - 1$ binary (also called “dummy”) variables are generated, where m is the number of different categories.

Another challenge in current ‘omics’ is the high dimensionality of microarray and NGS data. Although the cost of sequencing continues to decrease, the dimension (number of variables) of whole genomes (and transcriptomes, etc.) is typically much larger than the number of analyzed individuals. For most ML methods, as for example ANN, a prohibitive quantity of data objects is needed to work with these high-dimensional datasets. Otherwise, the method may fail to achieve good estimations of the parameters and overfitting models are produced instead. This phenomenon is known as *the curse of dimensionality* (Libbrecht & Noble, 2015; Bishop, 2006).

Finally, one of the strengths of traditional hypothesis-driven research was that correctly understanding a problem provided the solution or, at least, the way to reach it. Instead, in machine learning we often have a trade-off between solving the problem and understanding it (Libbrecht & Noble, 2015). Very sophisticated ANN may achieve highly accurate predictions, and while this has very useful applications (recall, for instance, the face recognition problem), knowledge about an underlying biological mechanism is not necessarily acquired. To counter this, a strategy is using methods that are “clear boxes” (in the sense that we can see how the model approached the problem, *e.g.* which predictor variables it considered important) or developing strategies to retrieve this information out of the models.

Apart from gaining some kind of new knowledge (in addition to the outcome of the learning process), in some cases it is interesting to improve prediction by including additional information at the input of the ML method. This beforehand-known information is frequently

supplied in indirect ways, *e.g.* pre-processing decisions about data representation and curation, feature extraction, etc. Direct encodings of additional knowledge (probabilistic priors, for instance) into the model tend to be more difficult to perform, at least in ANN and RF (Libbrecht & Noble, 2015).

CHAPTER 2

OBJECTIVES

The main objective of this PhD thesis is the evaluation and development of specific kernel approaches to phenotypic prediction and pattern inference from biological data, focusing on problems with structured data types or study designs.

The specific objectives are:

- Development and evaluation of the performance in HIV drug resistance prediction of novel categorical kernels adapted to HIV sequence data intricacies.
- Inclusion into the aforementioned kernels of prior knowledge about the proteins targeted by the antiretroviral drugs. That includes: (i) weighting each protein position by its importance and (ii) protein three-dimensional structure.
- Proposal of specific kernels for compositional metagenomic data, along with kernels derived from well-known beta-diversity measures.
- Integration of unsupervised, supervised and taxa importance analyses in the microbiome area using the kernel framework.
- Meaningful integration of spatial and temporal-structured samples exploiting the advantages of the kernel approach.

CHAPTER 3

GENERAL METHODS

The present chapter is entirely devoted to kernels and kernel methods. It is organized as follows: in section 3.1 we present the basics about kernel functions, kernel matrices and the kernel methods used in this thesis (Support Vector Machines and kernel Principal Component Analysis). The advantages of using kernels to tackle typical genetic and genomic problems are highlighted. In section 3.2, a general view of all kernels used throughout this thesis is given. We discuss the most important traits of each one of them and explain for which problems we used them and why. Finally, in section 3.3, we show how the importance given to each variable in a Support Vector Machine model can be recovered.

3.1 Introduction to the kernel framework

Kernel methods are a subset of ML methods that share the use of kernel functions. Let $S = \{x_1, \dots, x_N\}$ be a dataset containing N objects of any type. The kernel function evaluates all possible pairs of objects, and these evaluations are stored in a kernel matrix: $[\mathbf{K}]_{ij} = \kappa(x_i, x_j)$. Thus, the kernel matrix (and not the original data) is the input for the kernel method. In the sequel we explain the details of this approach and the advantages it presents when dealing with biological data.

3.1.1 Kernel functions

Kernel functions (also called simply *kernels*) have a crucial role in kernel methods. Intuitively, $\kappa(x_i, x_j)$ can be understood as a function that measure the similarity between x_i and x_j . Generally speaking, all kernels are similarities; however, not every similarity measure is a

kernel: only those that generate squared, symmetric and positive semi-definite (PSD) matrices (*i.e.* kernel matrices). A more formal definition of kernel is:

Definition 1: A mathematical function $\kappa: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a kernel iff:

- (i) $\kappa(x_i, x_j) = \kappa(x_j, x_i)$ for any $x_i, x_j \in \mathcal{X}$
- (ii) $\sum_{i=1}^N \sum_{j=1}^N a_i a_j \kappa(x_i, x_j) \geq 0$

For any $N > 0$, any finite set of objects $x_1, \dots, x_N \in \mathcal{X}$ and any choice of numbers $a_1, \dots, a_N \in \mathbb{R}$ (Schölkopf *et al.*, 2004).

As the objects x_1, \dots, x_N are never represented explicitly, only via their pairwise similarities, kernels can handle directly data types that are not standard. Thus, it is no longer necessary to find a way of recoding complex data types to real vectors so they can be used as input of a ML method. Instead, the challenge is designing kernels that extract valuable information from specific data types or problems. This requires a notion of what is considered “similar” in that given context. Prior knowledge about the problem at hand can also be encoded explicitly into the kernel (Libbrecht & Noble, 2015). Furthermore, kernel design is modular, allowing the creation of complex functions from simpler ones, each one capturing one aspect of the data (Schölkopf *et al.*, 2004). An inadequate choice of the kernel at this stage will cause a suboptimal performance of the method, as the only contact it has with the original data is through the kernel: once the kernel matrix is generated, all information left behind is lost.

The first kernels described were intended for standard continuous data. The most basic kernel is called the *linear kernel* and is defined as the inner product of two real vectors:

$$Lin(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \quad (3.1)$$

Where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^D$, being D the vector dimension. It is related to the cosine of the angle between two vectors, which is maximum when they share direction and 0 if they are perpendicular. This kernel has a clear linear nature, as it can be rewritten as:

$$Lin(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^D x_{ik} x_{jk} \quad (3.2)$$

The definition of the linear kernel holds a great importance because all kernels can be expressed as inner products in some space:

$$\kappa(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad (3.3)$$

$\phi: \mathcal{X} \rightarrow H$ maps an object from the original (input) space \mathcal{X} to the so-called *feature space* H . This feature space is endowed with an inner product and, typically, has a higher dimension than the original space. However, all the process happens behind the scenes: when using a kernel, one does not have to compute or even know the mapping ϕ that it used. This is because the object representations are never explicitly used. Instead, what the kernel does is to directly compute their similarities (*i.e.* inner products) in feature space. This is known as the *kernel trick*. The advantages of this approach are essentially two:

- (i) It allows solving nonlinear problems using linear methods. An visual example can be seen in Figure 3.1: by using ϕ , the kernel takes the original nonlinear data to a higher dimensional space where it can be operated in a linear way with inner products.
- (ii) It avoids the computational cost of explicitly computing the representation of the input data in feature space + the inner product. Not only this cost can be quite heavy (depending on the dimensionality of the feature space), but also in some cases the explicit computation is impossible because H has infinite dimension.

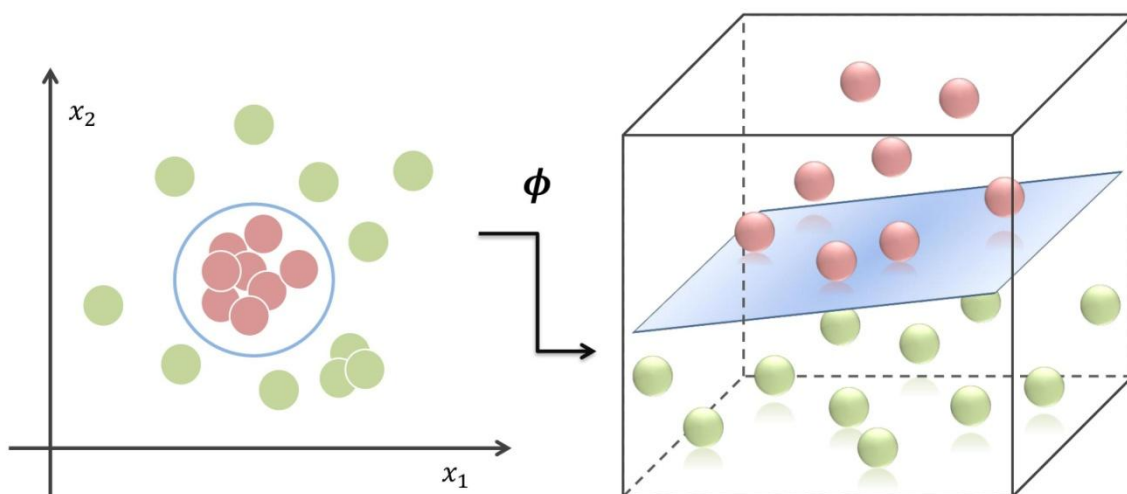


Figure 3.1. A classification example of how the kernel trick allows solving nonlinear problems in a linear manner.

3.1.2 Kernel matrices

Kernel matrices store the evaluations of the kernel function and serve as input for the kernel method. As stated before, only squared, symmetric and PSD matrices are considered kernel matrices. This limits the spectrum of functions that one can use, but also guarantees that every matrix generated can be processed by the kernel method. There exist a complete

independence between the choice of $\kappa(x_i, x_j)$ and the choice of the method, so any kernel function can be combined with any kernel method.

Kernel matrices have dimension $N \times N$, which has important consequences in computational complexity. For instance, consider that the dataset objects are real vectors of length D . The original dataset is transformed from a $\mathbf{S}_{N \times D}$ table to a $\mathbf{K}_{N \times N}$ kernel matrix. Unlike RF, ANN or ridge regression, which extract the information from the variables (dataset columns), kernel methods “look” at data focusing on the objects (rows). A dataset of N objects is always represented by a $N \times N$ matrix, irrespectively of the dimension of the objects, which is very attractive in cases where $N \gg D$ (see subsection 1.6.4). This also sets the minimum asymptotic complexity of computing a kernel matrix in $O(N^2)$.

As stated in section 1.2, after the advent of the ‘omics’ era more interest is being paid to the problem of data fusion. It is increasingly common to have data, for the same N individuals, from different M sources, thus raising the question of how to integrate them. In the kernel framework, we have a straightforward approach: directly combining the kernel matrices. The easiest way to do so is:

$$\mathbf{K}^* = \sum_{z=1}^M \beta_z \mathbf{K}_z \quad (3.4)$$

\mathbf{K}^* is a lineal combination of valid kernel matrices and thus a kernel matrix itself, as long as the β_z are nonnegative. The goal is to choose the best β_z coefficients of the combination: that is called *Multiple Kernel Learning* or MKL. A lot of research has been done in this area, mostly in supervised learning (Schölkopf *et al.*, 2004), but also in unsupervised scenarios. In the latter case, a consensus matrix can be obtained choosing the β that maximize \mathbf{K}^* average similarity with all \mathbf{K}_z matrices (Mariette & Villa-Vialaneix, 2018). MKL allows the fusion of data coming from the same individuals, irrespectively if this data have different number of variables, or even if it is not of the same type. Thus, for instance, data as different as genomic, metabolomic and blood analyses from the same patients can be integrated at the kernel matrix level if suitable kernel matrices are chosen for the three kinds of data. Also, the kernel approach is useful when we have missing individuals in some sources of data but not in others. This is typical, especially, of clinical data. The missing entries can be by-passed using the kernel matrix or matrices from alternative data sources (Schölkopf *et al.*, 2004), or even a mutual matrix completion can be performed. Although this is beyond the scope of this work, some techniques to do so can be found in Bhadra *et al.* (2017) and Rivero *et al.* (2017).

3.2 Kernel methods

The first kernel method as such was the Support Vector Machine (SVM) (Boser *et al.*, 1992). Since then, new kernel methods have been proposed –*e.g.* Relevance Vector Machine (Tipping, 2000)– to perform a great range of tasks, including clustering, classification, regression or visualization. Also, classical ML methods that can be reformulated in terms of inner products have been successfully “kernelized”, as is the case of kernel Principal Components Analysis (kPCA). SVM and kPCA will be used extensively throughout this thesis and are reviewed next:

3.2.1 Support Vector Machine (SVM)

SVM are mostly used in supervised learning problems. From a dataset containing N data objects $\{x_1, \dots, x_N\} \in \mathcal{X}$, the purpose of SVM is to predict a target variable y (which can be categorical or continuous) by finding a mapping $F: x \rightarrow y$.

In the most basic case, $\mathcal{X} = \mathbb{R}^D$ and the target variable is binary: $y \in \{-1, +1\}$. Classification is achieved by constructing a hyperplane that separates the two classes. The map is defined as:

$$F(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \quad (3.5)$$

where a label $+1$ is assigned if $F(\mathbf{x}) \geq 0$, and a label -1 is assigned if $F(\mathbf{x}) < 0$. $b \in \mathbb{R}$ is the intercept, and $\mathbf{w} \in \mathbb{R}^D$ is the normal vector of the separating hyperplane. The distance between the two resulting half-spaces is called *margin* and is exactly $2/\|\mathbf{w}\|$. The *optimal separating hyperplane* is that of maximum margin. However, infinite hyperplanes fulfill (3.5). Thus, an additional constraint to find a unique solution is required:

$$|\mathbf{w}^T \mathbf{x} + b| \geq 1 \quad (3.6)$$

The objects for which (3.6) is an equality are called *support vectors*. The hyperplane found by the SVM does not change when non-support vectors are removed from the training set. A visual example is given in Figure 3.2, left.

Leaning excessively onto the training data increases the complexity of the model and causes overfitting. Thus, some data points are allowed to dwell within the margin (Figure 3.2, right) by introducing a regularization hyperparameter: the cost (C). C can be understood as the cost of misclassification. When C takes larger values, the margin narrows and more training points are correctly classified with strong confidence. However, the complexity increases. On the other hand, the closer C is to 0 the wider is the margin, causing a progressive decrease of the

complexity so underfitting can be an issue. The best separating hyperplane is found solving the optimization problem:

$$\underset{w,b}{\text{minimize}} \frac{\|w\|^2}{2} + C \sum_{i=1}^{Ntr} HL(F, \mathbf{x}, y) \quad (3.7)$$

subject to constraints (3.6).

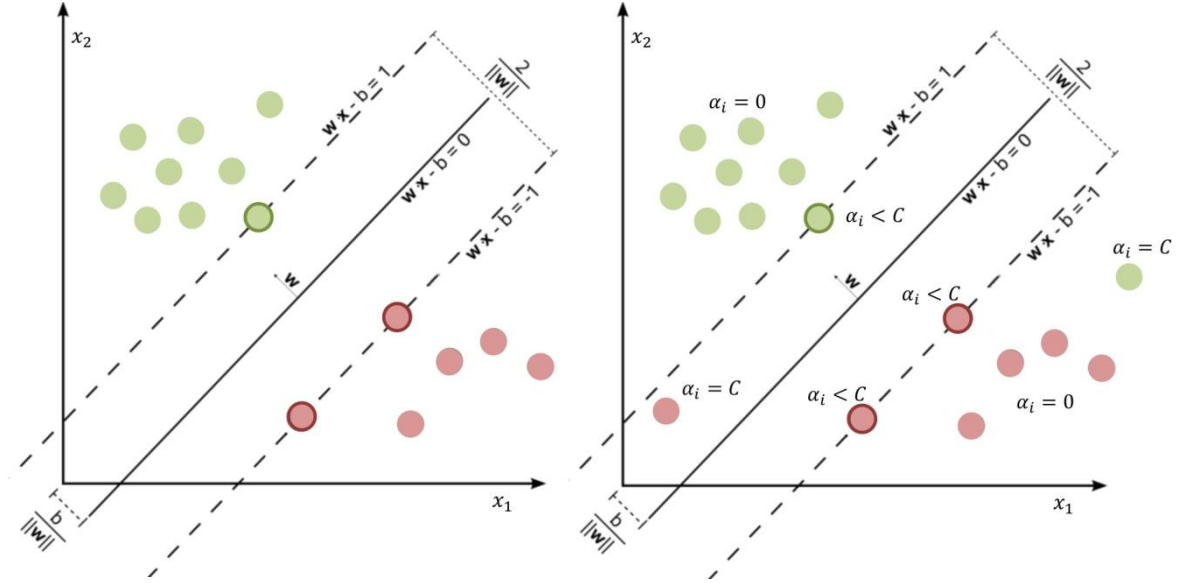


Figure 3.2. SVM graphical representation. Left: the black straight line separates the two classes. This line is the intersection of the feature space where lie the objects \mathbf{x} with the separating (hyper)plane. Support vectors are highlighted. Right: the same with examples of points violating the margin, C and α .

$HL(F, \mathbf{x}, y) = \max(0, 1 - yF(\mathbf{x}))$ is the hinge loss error function. It is not differentiable, so the minimization is done numerically. Let us introduce a vector of nonnegative slack variables ξ_1, \dots, ξ_{Ntr} that measure the amount of violation of the constraint. Thus we rewrite (3.7) as:

$$\underset{w,b,\xi_1,\dots,\xi_{Ntr}}{\text{minimize}} \frac{\|w\|^2}{2} + C \sum_{i=1}^{Ntr} \xi_i \quad (3.8)$$

subject to constraints: $\text{for } i = 1, \dots, Ntr \begin{cases} \xi_i \geq 0 \\ |\mathbf{w}^T \mathbf{x} + b| \geq 1 - \xi_i \end{cases}$

This can be expressed and solved using the Karush-Kuhn-Tucker approach. We introduce nonnegative Lagrange multipliers: $\alpha_1, \dots, \alpha_{Ntr}$ for each of the constraints $|\mathbf{w}^T \mathbf{x} + b| \geq 1 - \xi_i$, and $\eta_1, \dots, \eta_{Ntr}$ for each of the constraints $\xi_i \geq 0$:

$$L_P(\mathbf{w}, b, \xi, \alpha, \eta) = \frac{\|w\|^2}{2} + C \sum_{i=1}^{Ntr} \xi_i - \sum_{i=1}^{Ntr} \alpha_i (\xi_i - 1 + |\mathbf{w}^T \mathbf{x} + b|) - \sum_{i=1}^{Ntr} \eta_i \quad (3.9)$$

The Lagrangian has to be minimized with respect to \mathbf{w}, b, ξ and maximized with respect to $\alpha, \eta \geq 0$. The former step, the “primal” problem, gives us:

$$\frac{\partial L_P}{\partial b} = 0 \rightarrow \sum_{i=1}^{Ntr} \alpha_i y_i = 0 \quad (3.10)$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \rightarrow C = \alpha_i - \eta_i \quad (3.11)$$

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{Ntr} \alpha_i y_i \mathbf{x}_i \quad (3.12)$$

Then we plug (3.12), which gives us the optimal separating hyperplane, into (3.9), and proceed with the maximization step (the “dual” problem). At this point, η disappears of the expression. It can be recovered with (3.11), and so we have to check that $0 \leq \alpha_i \leq C$ holds. The other constraint is (3.10). Then, the dual problem is finally set as:

$$L_D(\alpha) = \sum_{i=1}^{Ntr} \alpha_i - \frac{1}{2} \sum_{i=1}^{Ntr} \sum_{j=1}^{Ntr} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (3.13)$$

In (3.13), $\alpha_i = 0$ for nonsupport vectors. Support vectors satisfy $\alpha_i < C$ if they are on the boundary margin and $\alpha_i = C$ if they lie on the wrong side of the boundary. The linear kernel term (*i.e.* $\mathbf{x}_i^T \mathbf{x}_j$) can be replaced by any other kernel $\kappa(x_i, x_j)$ (3.3). In this case, the SVM separation no longer happens on the input space but on feature space, where data is implicitly mapped. Then, all advantages of using kernels can be achieved: *e.g.* using nonstandard data and/or achieving linear separations in the feature space that are nonlinear with respect to the original space.

Once the model is solved, for predicting the label of a (previously unseen) object x_j , we use:

$$F(x_j) = \text{sign}\left(\sum_{i=1}^{Ntr} \alpha_i y_i k(x_i, x_j) + b\right) \quad (3.14)$$

The SVM is not only used for classification purposes. It can be applied to regression, as exemplified in Figure 3.3. In this case, the intersection of the feature space with the hyperplane defines a regression line, instead of separating the classes. With the purpose of controlling the sparsity in the support vectors, we use ε -insensitivity as the loss function:

$$L_\varepsilon = \begin{cases} 0, & |y - F(x)| \leq \varepsilon \\ |y - F(x)| - \varepsilon, & \text{otherwise} \end{cases} \quad (3.15)$$

Thus, the ε -insensitive loss considers equal to zero all errors that are within ε distance of the actual value of the target. This sets the width of the margin: small values of ε increase the number of support vectors and thus the complexity of the model, and vice versa. Meanwhile, C value controls the penalty imposed on the objects that lie outside the ε -limited tube.

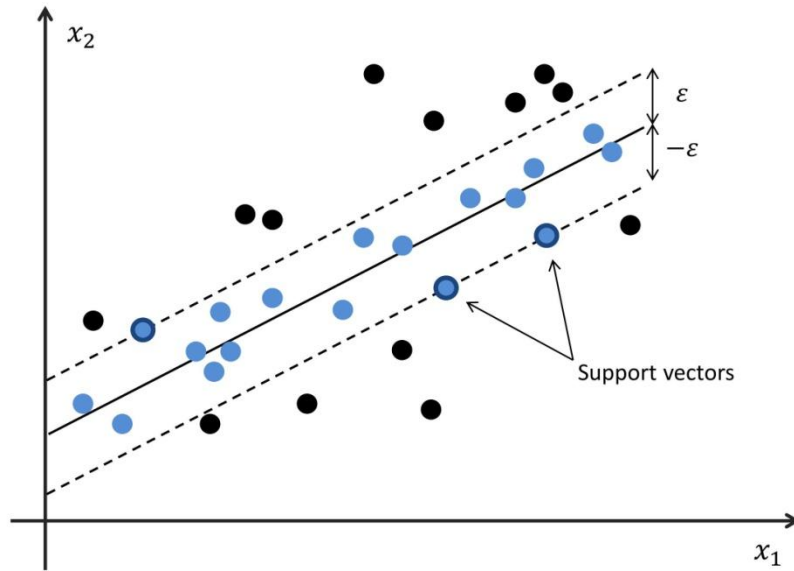


Figure 3.3. SVM for regression. In blue: points within the ε -tube, so the difference between actual and predicted value is considered 0. In black: points outside the ε -tube, so $L_\varepsilon = |y - F(x)| - \varepsilon$ applies instead. Support vectors are highlighted.

3.2.2 Kernel Principal Components Analysis (kPCA)

PCA (Principal Components Analysis) is an ordination technique that is routinely used for feature extraction and data visualization. The aim is to project the original data in \mathbb{R}^D to a lower dimensional space (usually \mathbb{R}^2 or \mathbb{R}^3) such that the maximum variance is preserved on the projections. The new axes are called Principal Components (PC) and are a linear combination of the original axes. The first PC is the direction with maximum variance, and next PCs are defined as the orthogonal projection directions sorted by decreasing variance.

Let $\mathbf{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in \mathbb{R}^D$ be a set of N centered vectors of dimension D . In that case, the $D \times D$ sample covariance is given by:

$$\mathbf{E} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i \mathbf{x}_i^T) = \mathbf{S} \mathbf{S}^T \quad (3.16)$$

The direction $\mathbf{v}_{(1)}$ of maximum variance (*i.e.* the first PC) is the eigenvector \mathbf{v} of \mathbf{E} with largest eigenvalue:

$$\mathbf{E}\mathbf{v}_{(1)} = \delta_{(1)}\mathbf{v}_{(1)} \quad (3.17)$$

The eigenvalues δ correspond to the variance of the projections. Thus, the rest of PCs are the D (orthogonal) eigenvectors of \mathbf{E} such as: $\delta_{(1)} \geq \delta_{(2)} \geq \dots \geq \delta_{(D)} \geq 0$. There are as many PCs as original dimensions, but for visualization purposes usually only the first two or three are kept.

(3.16) introduces the possibility of kernelizing the standard PCA (Bishop, 2006). In kPCA, the data is implicitly mapped (by the kernel trick) onto some feature space, where a standard PCA is performed. The new sample covariance is:

$$\mathbf{E} = \frac{1}{N} \sum_{i=1}^N (\phi(x_i)\phi(x_i)^T) \quad (3.18)$$

The PCs are obtained by substituting (3.18) in (3.17). All solutions \mathbf{v}_k lie in the span of $\phi(x_i)$, and then a given eigenvector is the linear combination of the mapped data: $\mathbf{v}_k = \sum_{i=1}^N \alpha_{ik}\phi(x_i)$. Thus, we have:

$$\frac{1}{N} \sum_{i=1}^N \phi(x_i)\phi(x_i)^T \sum_{i=1}^N \alpha_{ik}\phi(x_i) = \delta_k \sum_{i=1}^N \alpha_{ik}\phi(x_i) \quad (3.19)$$

As a kernel matrix is $[\mathbf{K}]_{ij} = \phi(x_i)^T\phi(x_j)$, (3.19) finally gives:

$$\delta_k \mathbf{K}\boldsymbol{\alpha}_k = \frac{1}{N} \mathbf{K}^2 \boldsymbol{\alpha}_k \quad (3.20)$$

The kernel matrix \mathbf{K} is precisely the covariance matrix in feature space. To center the projected data, we do:

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N \mathbf{K} - \mathbf{K} \mathbf{1}_N - \mathbf{1}_N \mathbf{K} \mathbf{1}_N \quad (3.21)$$

$\mathbf{1}_N$ denotes the $N \times N$ matrix filled with $1/N$. Now, we rewrite (3.20) as an eigenvector/eigenvalue problem:

$$\delta_k \boldsymbol{\alpha}_k = \frac{1}{N} \tilde{\mathbf{K}} \boldsymbol{\alpha}_k \quad (3.22)$$

The solutions of (3.22) satisfy (3.20). If we require that eigenvectors in feature space are normalized:

$$1 = \mathbf{v}_k^T \mathbf{v}_k = \sum_{i=1}^N \sum_{j=1}^N \alpha_{ik} \alpha_{jk} \tilde{\phi}(x_i)^T \tilde{\phi}(x_j) = \delta_k \boldsymbol{\alpha}_k^T \boldsymbol{\alpha}_k = \frac{1}{N} \boldsymbol{\alpha}_k^T \tilde{\mathbf{K}} \boldsymbol{\alpha}_k \quad (3.23)$$

The eigenvector problem is solved, and then the projection of a data object x onto the eigenvector k is given by:

$$x' = \tilde{\phi}(x)^T \mathbf{v}_k = \sum_{i=1}^N \alpha_{ik} \tilde{\phi}(x_i)^T \tilde{\phi}(x) = \sum_{i=1}^N \alpha_{ik} \tilde{k}(x, x_i) \quad (3.24)$$

kPCA presents several advantages when compared to standard PCA. It enables nonlinear projections and using nonstandard data as graphs or sequences. Kernel matrix \mathbf{K} grows with the number of objects N , while matrix $\mathbf{S}\mathbf{S}^T$ grows with dimension D . However, there is also an important drawback: the PCs no longer are computed explicitly (since they reside in feature space), and only the projections of our data onto them are known.

3.3 Kernel functions used throughout this thesis

In this section we present all kernels used over the course of this thesis. Among them, the Linear and RBF kernels are both well known and widely used. The Overlap kernel, the Jaccard kernel (both for presence/absence and quantitative data), the exponential random walk kernel, the Jensen-Shannon kernel and the RBF for time series are already described elsewhere, but have been mostly used in restricted fields of research. The “RBF-like” and “structural” versions (via the exponential random walk kernel) of the Overlap and Jaccard kernels are original contributions of this thesis, and the same applies to the Aitchison-RBF and compositional linear kernels.

3.3.1 Kernels for real vectors

The standard input in most ML methods is continuous data in the form of real vectors with length D , so each object $\chi \in \mathbb{R}^D$. Two well-known kernels for this kind of data, used in Chapter 4, are presented next:

Linear kernel

The linear kernel is the most simple kernel function. (3.1) shows that it is defined as the inner product of two vectors, and (3.2) its linear nature. When used in a kernelized method the result is the same that of the original method (*e.g.* linear kPCA is equivalent to standard PCA). It has no hyperparameters to optimize, which may speed up the model calculation compared to other kernels.

Gaussian Radial Basis Function (RBF) kernel

The Gaussian Radial Basis Function (RBF) kernel is extensively used on real-life applications and is considered the gold standard among kernels. Its most common formulation is:

$$RBF(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \quad (3.25)$$

This fulfills (3.3) but, as data is mapped onto an infinite-dimensional feature space, ϕ is not obvious (Schölkopf *et al.*, 2004; Shawe-Taylor & Cristianini, 2004). The $\|\mathbf{x}_i - \mathbf{x}_j\|$ term can be easily identified as the Euclidean distance (d_E) between two real vectors, while $\sigma > 0$ is an hyperparameter. Similarities and distances are opposite: the closer two points, the smaller is the distance between them and the higher the value given by the RBF kernel. The maximum value is 1, and is given by $\mathbf{x}_i = \mathbf{x}_j$ and $d_E = 0$.

The RBF kernel is related to the linear kernel, as the inner product induces a L^2 norm that, in real vectors, is equivalent to the Euclidean distance. However, the RBF kernel is nonlinear and can be used to model any decision boundary. The hyperparameter σ has great importance on how this kernel adapts to data. For instance, when it is coupled to a SVM, F is a sum of Gaussians with width controlled by σ and centered on the support vectors (see Figure 3.4). The smaller the σ value, the more sharp the distributions around support vectors, with the subsequent rise in model complexity and greater risk of overfitting. When σ increases, the Gaussians become smoother and the RBF kernel loses its nonlinear power, to the point that (with very large values) it behaves similarly to the linear kernel (Shawe-Taylor & Cristianini, 2004). For simplicity, in this thesis the hyperparameter $\gamma = \frac{1}{2\sigma^2}$ is used in (3.25) instead of σ .

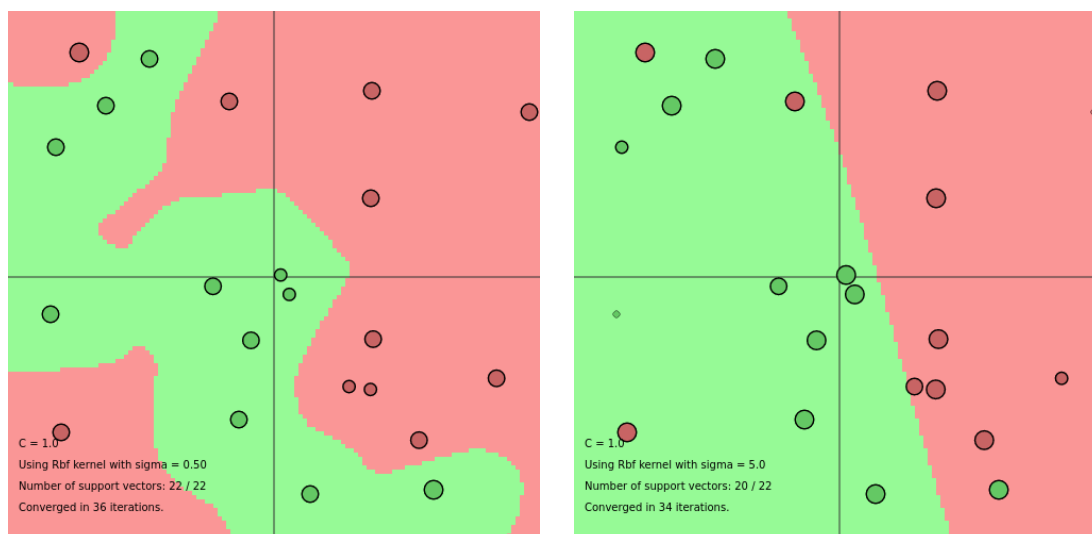


Figure 3.4. Effect of σ in defining the RBF-SVM boundary shape around the support vectors. Left: $\sigma = 0.5$, right: $\sigma = 5$. Created with <https://cs.stanford.edu/people/karpathy/svmjs/demo/>.

3.3.2 Kernels for categorical data & categorical sets

In Chapter 4, HIV protein sequences are analyzed as vectors of categorical data. That is: if \mathcal{A} is the set of the 20 canonical amino acids, a sequence of length D is $\chi \in \mathcal{A}^D$. The aligned sequences were compared position by position. In this view, the independence of the different polymorphic positions (*i.e.* absence of LD) is assumed. The order of the sequence is also not considered. Instead, the main focus was set on the phenomenon of amino acid mixtures, which is very widespread in HIV sequence data. Thus, each position of the sequence (understood as a categorical variable) may contain a single amino acid (a single category) or a mixture of amino acids (a categorical set). This justifies the use of two kernels: a kernel for categorical variables, derived from the Overlap kernel, and another for sets, derived from the Jaccard kernel.

The Overlap kernel

The Overlap kernel is the most basic categorical kernel. It assigns a similarity of 1 if the two compared objects are identical and 0 if they are different. If only one categorical variable is assessed, then:

$$Ov(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 0 & \text{if } x_i \neq x_j \end{cases} \quad (3.26)$$

This definition fulfills (3.3), as the Overlap kernel is equivalent to apply (3.1) after performing one-hot encoding (which is precisely the ϕ map) over the dataset. Here the kernel trick saves us the costs associated to an explicit computing: the one-hot encoding transforms each categorical variable with m categories to m variables in feature space.

The Overlap kernel is also named the Dirac kernel (Belanche & Villegas, 2013) when applied to objects with more than one categorical variable (*i.e.* the multivariate case):

$$Dir(x_i, x_j) = \frac{1}{D} \sum_{k=1}^D Ov(x_{ik}, x_{jk}) \quad (3.27)$$

where D stands for the number of variables. The sum of kernels is guaranteed to give another kernel (Shawe-Taylor & Cristianini, 2004).

In Chapter 4, a new categorical kernel that combines the multivariate Overlap with the RBF nonlinearity is used:

$$wOv(x_i, x_j) = \exp(\gamma \sum_{k=1}^D w_k \cdot Ov(x_{ik}, x_{jk})) \quad (3.28)$$

It is also well known that, for any $\kappa(x_i, x_j)$, $\exp(\kappa(x_i, x_j))$ gives a valid kernel (Shawe-Taylor & Cristianini, 2004). Prior knowledge about the importance of each categorical variable can be introduced in a vector of weights \mathbf{w} . That results in a kernel as long as that nonnegative weights are used (Schölkopf *et al.*, 2004).

The Jaccard kernel

The Jaccard index, extensively used in some fields such as community ecology, measures the similarity between two finite sets. It is proven elsewhere (Bouchard *et al.*, 2013) that it is positive definite, and thus a kernel. Let X_i and X_j be two sets. Then, a possible definition of the Jaccard index is:

$$Jac(X_i, X_j) = \begin{cases} 1 & \text{if } X_i = X_j = \emptyset \\ \frac{|X_i \cap X_j|}{|X_i \cup X_j|} & \text{otherwise} \end{cases} \quad (3.29)$$

where $|\cdot|$ denotes the cardinality of the set. An alternative formulation is given if we represent both sets as bitstrings $\mathbf{x}_i, \mathbf{x}_j \in \{0,1\}^m$ by one-hot encoding. For instance, we can consider that 0 represents the absence and 1 the presence of a given element. m is the number of different elements observed on the sets (*i.e.* the cardinality) and now the Jaccard kernel is defined by comparing the number of matches in the two bitstrings:

$$Jac(\mathbf{x}_i, \mathbf{x}_j) = \frac{|1-1 \text{ matches}|}{|0-1 \text{ matches}| + |1-0 \text{ matches}| + |1-1 \text{ matches}|} \quad (3.30)$$

It can be seen that categories absent in both objects (0-0 matches) are irrelevant for computing the result. This is not the case of other kernels on sets, for instance, the Simple Matching Coefficient (Pekalska *et al.*, 2001).

As in the case of the Overlap kernel, a new nonlinear and multivariate variant of the Jaccard kernel is used in Chapter 4:

$$wJac(X_i, X_j) = \exp\left(\gamma \sum_{k=1}^D w_k \cdot Jac(X_{ik}, X_{jk})\right) \quad (3.31)$$

As 0-0 matches have no relevance in (3.30), when $|X_i| = |X_j| = 1$ (*i.e.* no mixtures) the two versions provided of the Jaccard kernel reduce to the analogous Overlap kernel – (3.29) and (3.30) to (3.26), and (3.31) to (3.28).

3.3.3 Kernels for graphs

A graph $G = (V, E)$ consists of a set of nodes V and a set of edges E . A labeled graph has labels at the nodes. Conversely, a weighted graph is a graph with weights at the edges. The adjacency matrix of G is represented by:

$$[\mathbf{A}]_{ij} = \begin{cases} \omega & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (3.32)$$

In an unweighted graph, $\omega = 1$ for all connected pairs of nodes.

In Chapter 5 we reanalyzed the protein sequences of Chapter 4, but considering potential associations between the protein positions. The three-dimensional structures of the proteins were represented as graphs, wherein each node represented a residue, and the weights on the edges were the Euclidean distance (in Å) among residues.

Exponential Random Walk kernel

A random walk of length $k - 1$ over a graph is the sequence of nodes v_1, v_2, \dots, v_k connected by edges, so that at each step: $(v_{i-1}, v_i) \in E$ for $1 < i \leq k$. Two objects modeled as graphs (e.g. two proteins, or two mutated variations of a protein) can be compared by performing all possible random walks in both graphs separately, and then assessing the similarity between these random walks with a kernel. The computation can be simplified using the direct product of two graphs (Vishwanathan *et al.*, 2010):

Definition 2: The direct product of two graphs $G = (V, E)$ and $G' = (V', E')$ generates a graph $G_{\times} = (V_{\times}, E_{\times})$ such that:

$$V_{\times} = \{(v_i, v'_i) \in V \times V'\}$$

$$E_{\times} = \{((v_i, v'_i), (v_j, v'_j)) \in V_{\times} : (v_i, v_j) \in E \wedge (v'_i, v'_j) \in E'\}$$

It is well known that performing random walks on G_{\times} is equivalent to doing so on the two graphs G and G' and then comparing the walks. Then, the exponential random walk kernel over two graphs is defined as in Gärtner *et al.* (2003):

$$\text{Exp}(G, G') = \text{Exp}(G_{\times}) = \sum_{i,j=1}^{|V_{\times}|} \sum_{k=0}^{\infty} \frac{1}{k!} \gamma^k [\mathbf{A}_{\times}^k]_{ij} = \sum_{i,j=1}^{|V_{\times}|} e^{\gamma [\mathbf{A}_{\times}]_{ij}} \quad (3.33)$$

Where the adjacency matrix of G_{\times} is denoted \mathbf{A}_{\times} , and $\gamma \in (0,1)$ is a hyperparameter that controls the “decay”. Thus, the more edges separate two nodes, the weaker is their interaction, so each edge crossed in the random walk is penalized. The penalty grows as γ takes values closer to 0, thus causing a faster decay. Unlike previous kernels, which have

straightforward $O(N^2D)$ time complexities, a naive implementation of (3.33) is computationally infeasible. Some possible optimizations are discussed in Chapter 5.

3.3.4 Kernels for count data

Count data consists in nonnegative integer vectors of dimension D ($\mathbf{x} \in \mathbb{N}_0^D$). In Chapter 6 we analyzed the microbial abundance of several soil, human and piglet samples, where D represented the number of taxa. Two different classes of kernels were used: on one hand, kernels derived from traditional ecology *beta-diversities*; on the other hand, kernels that take into account some particularities of count data obtained with NGS technologies, which we called *compositional kernels*.

Kernels derived from ecological beta-diversities

In community ecology, the *diversity indexes* measure how many different species (or, more generally, how many taxa of interest) are present in a given habitat or habitats. The *alpha diversity* is the diversity in one habitat, whereas the *beta diversity* represents the difference between the communities of two habitats (Gardener, 2014). There exists a large catalogue of beta-diversity measures, which may be distances (metric) or dissimilarities (semimetric). A dissimilarity or distance, d , can be transformed into a similarity s in several ways. An example is: $s = 1 - d$, considering that d is normalized to have a maximum value of 1. However, not all similarities that one can obtain fulfill Definition 1 and are, therefore, kernels.

Beta-diversity indices can be computed from abundance tables only, as Bray-Curtis and Jensen-Shannon (Gloor *et al.*, 2017; Gardener, 2014), or may also need a phylogenetic tree, as Unifrac. The present work is only concerned with the kernelization of the former kind of measures. The definition of the Jensen-Shannon divergence is:

$$JSD(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \left[\sum_{k=1}^D x_{ik} \ln \left(\frac{2x_{ik}}{x_{ik} + x_{jk}} \right) + \sum_{k=1}^D x_{jk} \ln \left(\frac{2x_{jk}}{x_{ik} + x_{jk}} \right) \right] \quad (3.34)$$

It can be easily converted into a kernel, which is already described in Bai and Hancock (2011) as the Jensen-Shannon kernel:

$$JSK(\mathbf{x}_i, \mathbf{x}_j) = 1 - JSD(\mathbf{x}_i, \mathbf{x}_j) \quad (3.35)$$

Considering that the abundance counts are converted from absolute to relative frequencies.

There exist several definitions of the Bray-Curtis dissimilarity. For instance, it can be defined as in Coenen *et al.* (2020):

$$BCD(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{2 \sum_{k=1}^D \min(x_{ik}, x_{jk})}{\sum_{k=1}^D (x_{ik} + x_{jk})} \quad (3.36)$$

BCD is semimetric and, therefore, not a distance in the strict sense. The Jaccard distance, also used in community ecology, is rank-order similar to BCD and fulfills the conditions to be a metric (Gardener, 2014). It is paired with a kernel that was already presented in (3.30) for presence/absence data. For abundance data, the following quantitative version exists:

$$qJac(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^D \frac{\min(x_{ik}, x_{jk})}{\max(x_{ik}, x_{jk})} \quad (3.37)$$

Some alternate names are the Ružička index or the min-max kernel (Li, 2015).

Kernels for compositional data

Taxonomic counts obtained by NGS technologies are a particular case within abundance data. They are not equivalent to counts obtained by direct observation, because the number of reads delivered is constrained by the instrument capacity (see also subsection 1.2.2). Thus, after sequencing, information about actual absolute frequencies is lost and only relative frequencies are of use. Data consisting in proportions with uninformative sum is called *compositional* and deserves a specific mathematical treatment. This data does not live in the real space (*i.e.* \mathbb{R}^D) but in the simplex, a $D - 1$ subspace of it (Quinn *et al.*, 2018):

Definition 3: A D -part composition is represented as a vector $\mathbf{x} = (x_1, x_2, \dots, x_D)$ with sample space the simplex: $\mathcal{S}^{D-1} = \{\mathbf{x} = (x_1, \dots, x_k, \dots, x_D) : x_k > 0 (k = 1, 2, \dots, D), \sum_{k=1}^D x_k = c\}$, where c is a given positive constant (Mateu-Figueras *et al.*, 2011).

A visual example of count vectors that are different in real space but equivalent in the simplex is shown in Figure 3.5. As D -parts compositions are usually expressed as numeric vectors in \mathbb{R}^D , it is very tempting to compute standard distances, correlation measures and multivariate statistical methods typical of real data when dealing compositional samples. In fact, this can lead to spurious results (Quinn *et al.*, 2018; Mateu-Figueras *et al.*, 2011). This is because of their nature as proportions, which leads to a mutual dependence among features: *i.e.* increasing the abundance of one decreases the abundance of the rest.

There exist at least two strategies for facing the treatment of compositional data. The first one is to think of \mathcal{S}^{D-1} as equipped with its own geometry (known as the *Aitchison geometry*) and use suitable metrics and statistical methods for working “within” the simplex (Mateu-Figueras *et al.*, 2011). The most widespread strategy, however, consist in using

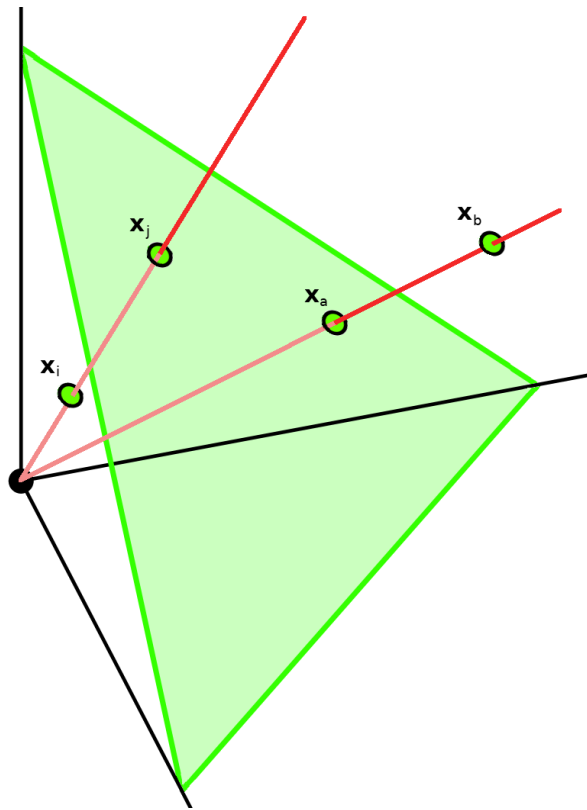


Figure 3.5. The S^2 simplex embedded in \mathbb{R}^3 . 3-part compositions \mathbf{x}_i , \mathbf{x}_j , \mathbf{x}_a and \mathbf{x}_b are all different in terms of absolute counts, but \mathbf{x}_i is equivalent to \mathbf{x}_j in terms of proportions (*i.e.* compositionally equivalent), and the same is true for \mathbf{x}_a and \mathbf{x}_b .

transformations that map compositional data into the real space. One the most used (Gloor *et al.*, 2017) is the center log-ratio (clr) transformation:

$$clr(\mathbf{x}) = \left[\log \left(\frac{x_1}{geo(\mathbf{x})} \right), \log \left(\frac{x_2}{geo(\mathbf{x})} \right), \dots, \log \left(\frac{x_D}{geo(\mathbf{x})} \right) \right] \quad (3.38)$$

where \mathbf{x} here is a count vector of dimension D , and $geo(\mathbf{x}) = (\prod_{k=1}^D x_k)^{1/D}$ is the geometric mean. Recently, it has also been suggested the PhILR mapping (based on the isometric log ratio or ilr transformation), which take into account phylogenetic information (Silverman *et al.*, 2017) as an alternative to Unifrac.

In Chapter 6, two compositional kernels (*i.e.* kernels for compositional data) analogous to the linear (3.2) and RBF (3.25) kernels are proposed. From now on we define the compositional linear kernel as:

$$cLin(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^D \log \left(\frac{x_{ik}}{geo(\mathbf{x}_i)} \right) \log \left(\frac{x_{jk}}{geo(\mathbf{x}_j)} \right) \quad (3.39)$$

Returning to (3.3), it is immediately clear that here ϕ is the clr-transformation. Changing it for other transformations (e.g. the aforementioned PhilR) would generate other variants that could, for instance, encode phylogenetic information into the kernel.

Currently, an extensive research is being carried out in the microbiome area for generating new tools that take into account the compositionality of data (Gloor *et al.*, 2017). Apart from clr and other related mappings, the Aitchison distance has been proposed as an alternative to beta-diversity measures like BCD in ordination and clustering (Quinn *et al.*, 2018; Gloor *et al.*, 2017). It is defined as:

$$d_A = \sqrt{\sum_{k=1}^D \left(\log \left(\frac{x_{ik}}{\text{geo}(\mathbf{x}_i)} \right) - \log \left(\frac{x_{jk}}{\text{geo}(\mathbf{x}_j)} \right) \right)^2} \quad (3.40)$$

The Aitchison distance over compositional data has the properties of scale invariance, permutation invariance (*i.e.* is not affected if the order of the features is changed), perturbation invariance, and subcompositional dominance (Quinn *et al.*, 2018; Aitchison *et al.*, 2000). A “perturbation” is analogous to a “translation” in real space, while subcompositional dominance means that the distance between two full compositions is greater or equal to the distance between them when considering any subcomposition (Pawlowsky-Glahn *et al.*, 2007). Besides, d_A gives the same result than computing the L^2 norm of two compositions “within” the simplex using the Aitchison geometry (Mateu-Figueras *et al.*, 2011; Aitchison *et al.*, 2000).

It can be seen in (3.40) that the Aitchison distance is equivalent to the Euclidean distance after performing a clr-transformation over the compositions. By changing d_E for d_A in (3.25) we define the Aitchison-RBF kernel:

$$cRBF(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \sum_{k=1}^D \left(\log \left(\frac{x_{ik}}{\text{geo}(\mathbf{x}_i)} \right) - \log \left(\frac{x_{jk}}{\text{geo}(\mathbf{x}_j)} \right) \right)^2\right) \quad (3.41)$$

It is straightforward to prove that this is a valid kernel. Let $\kappa_{real}(x_i, x_j)$ be any kernel over $\mathbb{R}^D \times \mathbb{R}^D$, and ϕ a map $\chi \rightarrow \mathbb{R}^D$ (as it is the case of the clr-transformation). In that case, it is well known that $\kappa_{real}(\phi(x_i), \phi(x_j))$ is PSD. See Shawe-Taylor and Cristianini (2004) for a complete demonstration.

3.3.5 Kernels for temporal-structured data

In Chapter 6, a dataset concerning the evolution of the microbial composition of the gut during the first week of life of several piglets is presented. As stated in subsection 1.3.2, the

time series from an individual is typically obtained as a set of samples indexed by time $\{x_i, t_i\}$, which is usually summarized through a function. This function modeling the data can be obtained in several ways, *e.g.* polynomial interpolation, splines, etc. Functions can be used as input to specific kernels; thus, the evolution over time among individuals is compared and used afterwards for phenotype prediction or unsupervised tasks.

The inner product defined for vectors can be generalized to continuous functions (Lipshutz & Lipson, 2009). Thus, a suitable kernel, analogous to the standard linear kernel for real data, can be defined. Let $f(t)$ and $g(t)$ be continuous and real valued functions that model the time series of two different individuals. In a given interval $[t_a, t_b]$, the functional linear kernel can be defined as:

$$fLin(f, g) = \int_{t_a}^{t_b} f(t) g(t) dt \quad (3.42)$$

As an inner product induces an L^2 norm, a RBF kernel for functions, analogous to the standard RBF kernel (3.25), can also be obtained. The functional RBF kernel, described in Chen *et al.* (2013), derives from the L^2 norm for functions in the interval $[t_a, t_b]$:

$$fRBF(f, g) = \exp(-\gamma \int_{t_a}^{t_b} |f(t) - g(t)|^2 dt) \quad (3.43)$$

$fLin$ and $fRBF$ can also be defined for discrete functions. In this case, $f(t)$ and $g(t)$ may directly denote the original objects $\{x_i, t_i\}$, so each time value directly maps to a certain value of the feature variable x . If T is the total number of time points and Δt the time increment, then:

$$fRBF(f, g) = \exp(-\gamma \sum_{i=1}^T (f(t_i) - g(t_i))^2) \quad (3.44)$$

$$fLin(f, g) = \Delta t \sum_{i=1}^T f(t_i) g(t_i) \quad (3.45)$$

This way the calculation of the integral can be avoided. However, unlike (3.42)-(3.43), the time points should coincide across the individuals assessed and no missing values are allowed. Here Δt is considered constant across all assessed points.

If multiple variables are followed over time, the aforementioned kernels can be combined as in:

$$fLin'(f, g) = \sum_{k=1}^D fLin(f_k, g_k) \quad (3.46)$$

$$fRBF'(f, g) = \prod_{k=1}^D fRBF(f_k, g_k) \quad (3.47)$$

where f_k and g_k are continuous or discrete functions that model variable k in two individuals, and D is the total number of variables. The computational complexity is $O(N^2TD)$ if fLin and fRBF are computed using (3.44) or (3.45).

3.4 Recovering variable importance from SVM

As shown in subsections 1.6.1 and 1.6.2, ML methods like DT or RF focus on extracting information from the predictor variables. Instead, kernel methods are more interested on the similarity between data objects. The latter approach presents some major advantages (as exposed in sections 3.1 and 3.2) but also has a drawback: unlike DT, RF or ridge regression, computing the variable importances is not immediately obvious in SVM. These importances can be used in a descriptive way, to gain knowledge about the relationship between predictor variables and the target (see subsection 1.6.4), or to perform feature selection. In the case of SVM, the most famous feature selection technique is called SVM-RFE, where RFE means Recursive Feature Elimination (Guyon *et al.*, 2002). The steps taken by SVM-RFE are:

1. Learn a SVM model from the training set.
2. Rank the variables according to a certain criterion.
3. If dimension is still higher than intended, remove the lowest-ranked variable (or, to speed-up the algorithm, half of the remaining variables) and go to step 1.

A key part of the algorithm lies in the ranking criterion chosen. In the original work it was proposed that the importance of the k -th variable can be computed as $(w_k)^2$. \mathbf{w} is the normal vector of the separating hyperplane in the SVM, which is computed with (3.12). The main idea is that the orientation of the hyperplane indicates how relevant the SVM model considers the variables with respect to predicting the target: the closer to orthogonal the hyperplane is to a particular dimension, the more important it is, and vice versa. This criterion acknowledges mutual information across features and handles well the cases of multicollinearity. However, as stated in the original paper, it is necessary to use the linear kernel for retrieving \mathbf{w} . This is because the separating hyperplane does not lie in input space, but in feature space, and is computed using the mapped objects. Only in the case of the linear kernel (3.1) the input objects are unmapped and, thus, this issue does not exist. In order to perform SVM-RFE with other kernels, several general strategies have been proposed. These strategies include obtaining $(w_k)^2$ using the linear kernel and then performing SVM-RFE with another kernel, or

changing the ranking criterion, so the importance of variable k is the difference in the cost function (3.13) when k is removed of the dataset (Guyon *et al.*, 2002):

$$DJ(k) = \frac{1}{2} \left[\sum_{i=1}^{Ntr} \sum_{j=1}^{Ntr} \alpha_i \alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{Ntr} \sum_{j=1}^{Ntr} \alpha_i \alpha_j y_i y_j k(x_i(-k), x_j(-k)) \right] \quad (3.48)$$

Removing or modifying a feature (or a group of features) at a time to assess its impact on prediction is a very general feature extraction technique, valid for virtually all ML methods, and in the case of SVM it has been extensively used for nonlinear kernels (Sanz *et al.*, 2018; Alonso-Atienza *et al.*, 2012; Maldonado & Weber, 2009). However, it can be quite heavy computationally.

Indeed, in some cases $(w_k)^2$ can be computed in an exact form even if the linear kernel is not used. The condition is that the specific map ϕ used by the kernel should be known. If that is the case, we can map the original data onto feature space explicitly, thus doing $\phi(x) \rightarrow \mathbf{x}$. Then, instead of (3.12), we can use:

$$\mathbf{w} = \sum_{i=1}^{Ntr} \alpha_i y_i \phi(x_i) \quad (3.49)$$

Although obtaining ϕ is not feasible for all kernels (not for the RBF and RBF-like kernels, for example) it allows extending variable ranking and SVM-RFE to a subset of kernels beyond the linear kernel. Of kernels reviewed in section 3.3, importances can be computed explicitly for:

- (i) The linear kernel Lin (3.1)
- (ii) The compositional linear kernel cLin. In this case, ϕ is the clr-transformation (3.39).
- (iii) The functional linear kernel fLin for discrete functions (3.45).
- (iv) The univariate (3.26) and multivariate (3.27) Overlap kernels, as ϕ is the one-hot encoding.

(They can also be computed explicitly for some kernels, *e.g.* polynomial kernels, not treated in this thesis.)

In some cases, like (iii) and (iv), the number of variables in feature space does not coincide with the number of original variables. For instance, for the Overlap kernel, the importances of each one of the modalities are obtained instead. As operations in feature space are of linear nature, to recover the global importance of each input variable we can sum the partial

importances of its modalities. A similar approach can be used with the functional linear kernel by adding the importances of all time points of each feature.

In some cases, importances can be also retrieved when dealing with data from different sources. Let's return to the MKL definition (3.4), but now expressed as a convex combination of kernels instead of kernel matrices:

$$\kappa^*(x_i, x_j) = \sum_{z=1}^M \beta_z \kappa_z(x_{iz}, x_{jz}) = \sum_{z=1}^M \beta_z \phi_z(x_{iz})^T \phi_z(x_{jz}) \quad (3.50)$$

Data is operated as inner products (*i.e.* the linear kernel) in feature space, as it can be seen in (3.3) and (3.13). If all $\phi_z: x_z \rightarrow \mathbf{x}_z$ are known, we can map the original objects onto feature space, and then do:

$$\sum_{z=1}^M \beta_z \text{Lin}(\mathbf{x}_{iz}, \mathbf{x}_{jz}) = \sum_{z=1}^M \beta_z (\mathbf{x}_{iz}^T \mathbf{x}_{jz}) = \sum_{z=1}^M (\sqrt{\beta_z} \mathbf{x}_{iz}^T \sqrt{\beta_z} \mathbf{x}_{jz}) \quad (3.51)$$

For notation simplicity, only the case with $M = 2$ different sources is illustrated. The ++ operator denotes concatenation of two vectors:

$$\sqrt{\beta_1} \mathbf{x}_{i1} \sqrt{\beta_1} \mathbf{x}_{j1} + \sqrt{\beta_2} \mathbf{x}_{i2} \sqrt{\beta_2} \mathbf{x}_{j2} = [(\sqrt{\beta_1} \mathbf{x}_{i1}) ++ (\sqrt{\beta_2} \mathbf{x}_{i2})] [(\sqrt{\beta_1} \mathbf{x}_{j1}) ++ (\sqrt{\beta_2} \mathbf{x}_{j2})]$$

Then, performing MKL with the linear kernel over M real data tables is equivalent to applying the linear kernel over the concatenation of the same M tables, each one weighted by its respective β_z . Changing $\phi(x_i)$ for $[\sqrt{\beta_1} \phi_1(x_{i1}) ++ \sqrt{\beta_2} \phi_2(x_{i2})]$ in (3.49) gives the importance of all variables in feature space. When the variables of different data sources coincide (as in Chapter 6, wherein the same taxa over four airway sites are used to predict smoking status) they can be added to obtain the overall importance across all sources.

CHAPTER 4

HIV DRUG RESISTANCE PREDICTION WITH WEIGHTED CATEGORICAL KERNEL FUNCTIONS

Elies Ramon¹, Lluís Belanche-Muñoz² and Miguel Pérez-Enciso^{1,3}

¹ Centre for Research in Agricultural Genomics (CRAG), CSIC-IRTA-UAB-UB, Campus UAB, 08193, Bellaterra, Barcelona, Spain.

² Computer Science Department, Technical University of Catalonia, Carrer de Jordi Girona 1-3, 08034, Barcelona, Spain.

³ Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig de Lluís Companys 23, 08010, Barcelona, Spain.

E-mail addresses:

elies.ramon@cragenomica.es (ER)

belanche@cs.upc.edu (LBM)

miguel.perez@uab.es (MPE)

Correspondence: elies.ramon@cragenomica.es

BMC Bioinformatics. 2019;20(1):410. doi:10.1186/s12859-019-2991-2

Published 2019 Jul 30.

Abstract

Background: Antiretroviral drugs are a very effective therapy against HIV infection. However, the high mutation rate of HIV permits the emergence of variants that can be resistant to the drug treatment. Predicting drug resistance to previously unobserved variants is therefore very important for an optimum medical treatment. In this paper, we propose the use of weighted categorical kernel functions to predict drug resistance from virus sequence data. These kernel functions are very simple to implement and are able to take into account HIV data particularities, such as allele mixtures, and to weigh the different importance of each protein residue, as it is known that not all positions contribute equally to the resistance.

Results: We analyzed 21 drugs of four classes: protease inhibitors (PI), integrase inhibitors (INI), nucleoside reverse transcriptase inhibitors (NRTI) and non-nucleoside reverse transcriptase inhibitors (NNRTI). We compared two categorical kernel functions, Overlap and Jaccard, against two well-known noncategorical kernel functions (Linear and RBF) and Random Forest (RF). Weighted versions of these kernels were also considered, where the weights were obtained from the RF decrease in node impurity. The Jaccard kernel was the best method, either in its weighted or unweighted form, for 20 out of the 21 drugs.

Conclusions: Results show that kernels that take into account both the categorical nature of the data and the presence of mixtures consistently result in the best prediction model. The advantage of including weights depended on the protein targeted by the drug. In the case of reverse transcriptase, weights based in the relative importance of each position clearly increased the prediction performance, while the improvement in the protease was much smaller. This seems to be related to the distribution of weights, as measured by the Gini index. All methods described, together with documentation and examples, are freely available at https://bitbucket.org/elies_ramon/catkern.

Keywords: HIV, Drug Resistance Prediction, Categorical Kernel, Weighted Kernel, PI, NRTI, NNRTI, INI, Machine Learning, Support Vector Machine, Random Forest, kernel PCA

4.1 Background

HIV is a retrovirus that infects human immune cells, causing a progressive weakening of the immune system. When untreated, the affected person develops acquired immunodeficiency syndrome (AIDS), which leads to a rise of opportunistic infections and, finally, death. HIV has infected more than 35 million people worldwide and is considered a global pandemic (The Joint United Nations Programme on HIV/AIDS [UNAIDS], 2019). Despite the efforts, to date there is no definitive cure that eradicates the virus from the organism. However, the lifespan and quality of life of many people that live with HIV have expanded greatly thanks to antiretroviral therapy. Antiretroviral drugs lower the virus level in blood by targeting different stages of the virus life cycle. The most important classes of antiretroviral drugs are protease inhibitors (PIs), which target the protease, and nucleoside and non-nucleoside reverse transcriptase inhibitors (NRTIs and NNRTIs, respectively) which target the reverse transcriptase. Other classes of antiretroviral drugs are the integrase inhibitors (INIs) and the fusion inhibitors.

Some of the main reasons why HIV is so difficult to fight are its short life cycle (1-2 days), high replication rate (10^8 - 10^9 new virions each day), and high mutation rate (10^{-4} - 10^{-5} mutations per nucleotide site per replication cycle) caused because reverse transcriptase lacks proofreading activity. This permits the fast emergence of new HIV variants, some of which may be resistant to the drug treatment (Iyidogan & Anderson, 2014). These variants can be transmitted, and some studies show that ~10% of patients who had never been on antiretroviral therapy carry at least one resistant HIV (German Advisory Committee Blood Subgroup 'Assessment of Pathogens Transmissible by Blood' [Blood G. A. C], 2016). Cross-resistance (simultaneous resistance to two or more drugs, often of the same class) is also a common phenomenon. It is therefore advisable to do a resistance test before the treatment to find the best drug choice (Iyidogan & Anderson, 2014; Shafer *et al.*, 2001), especially in developing countries, as recommended by the WHO and the International AIDS Society-USA Panel (Blood G. A. C, 2016). A resistance test can be performed *in vitro*, obtaining HIV samples from the patient and using them to infect host cells cultured in presence of increasing levels of drug concentration. The virus susceptibility is then obtained empirically as the IC50 (Shafer *et al.*, 2001) and usually delivered as the relative IC50 (resistance of the virus variant compared to the wild type). Another strategy is to infer the HIV variant resistance from its sequence. This can be either gene sequence or the translated protein sequence; this latter approach eliminates the noise of synonymous mutations. In any case, as genome sequencing is cheaper, faster and more widely available than performing an *in vitro* drug susceptibility

test, much effort has been invested in developing algorithms that predict the drug resistance from the virus sequence (Bonet, 2015).

The first attempts of automatic prediction can be traced back, at least, to the early 2000s (Schmidt, 2000). These approaches were rule-based: study the mutational profile of the HIV variant to look for known major drug-associated resistance mutations –lists of these mutations are periodically updated and can be found in reviews, *e.g.*, Wensing *et al.* (2017)–. The rule-based algorithms continue to be used to this day because of their interpretability. Some publicly available examples are the Stanford HIVdb, Rega or ANRS softwares (Bonet, 2015). However, the aforementioned high mutation rate of the HIV, which favors the emergence of large numbers of new resistance mutations and complex mutational patterns, makes the rule-based approach suboptimal. In this scenario machine learning methods can be extremely helpful, especially in recent years with the increasing size of available data. This second approach is also very popular and there exists machine learning software to predict resistance online (Riemenschneider, Hummel, *et al.*, 2016; Beerenwinkel *et al.*, 2003). Different methods have been proposed, the most common ones being Linear Regression (Yu *et al.*, 2014; Rhee *et al.*, 2006), Artificial Neural Networks (ANN) (Sheik Amamuddy *et al.*, 2017; Pasomsub *et al.*, 2010; Rhee *et al.*, 2006; Wang & Larder, 2003), Support Vector Machines (SVM) (Khalid & Sezerman, 2016; Masso & Vaisman, 2013; Rhee *et al.*, 2006), Decision Trees (DT) (Rhee *et al.*, 2006; Beerenwinkel *et al.*, 2002) and their ensemble counterpart, Random Forests (RF) (Tarasova *et al.*, 2018; Shen *et al.*, 2016; Khalid & Sezerman, 2016; Yu *et al.*, 2014). Some machine learning studies have complemented the sequence data with structural information (Khalid & Sezerman, 2016; Shen *et al.*, 2016; Yu *et al.*, 2014; Masso & Vaisman, 2013), or have benefited from the knowledge about major drug associated mutations to perform feature selection. The inclusion of cross-resistance information in the form of ensemble methods has also been reported to improve resistance prediction (Xing *et al.*, 2019; Riemenschneider, Senge, *et al.*, 2016; Heider *et al.*, 2013).

Nevertheless, HIV sequence data specificities pose significant challenges to resistance prediction. First, sequence data is categorical in nature. However, most machine learning algorithms are designed to cope with numeric data (DT and RF being exceptions), thus obliging to perform some kind of pre-processing. A typical approach is to recode each position into m or $m - 1$ "dummy variables", which can take the values 0 or 1 (Bonet, 2015). Usually, m is the number of all possible alleles that can be potentially found in a position (*i.e.*, $m = 20$ in protein sequences). However, some authors restrict the dummy variables to the drug associated mutations already appearing in the literature (Schmidt, 2000; Rhee *et al.*, 2006; Wang & Larder, 2003). A very different approach is found in Sheik Amamuddy *et al.*

(2017), where each amino acid was codified as an integer ranging 1-22 (the 20 canonical amino acids plus two extra characters B and Z). Other encodings have been used with HIV sequence data, like amino acid composition frequencies, reduced amino acid alphabets or physicochemical properties (Khalid & Sezerman, 2016; Bonet, 2015; Heider *et al.*, 2013).

Another challenge is the presence of mixtures of alleles (normally two, rarely three or four) in at least one position of the viral sequence for most clinical samples. In the case of HIV, this event indicates that the patient carries two or more virus variants (Shafer *et al.*, 2001). It is well established that HIV tends to generate viral swarms of closely related viruses (*quasispecies*), as a consequence of its high mutation rate (Iyidogan & Anderson, 2014). Mixtures introduce ambiguity in the genotype-phenotype correlation (Schmidt, 2000) and a problem of technical nature: the vast majority of machine learning methods are not able to deal directly with these “multiallelic” codes. To our knowledge, algorithms so far have handled allele mixtures with some sort of previous pre-processing of the data, *e.g.* keeping only the most frequent amino acid of the mixture (Tarasova *et al.*, 2018), replacing the positions by a missing value (Beerenwinkel *et al.*, 2002), excluding the affected sequences (Masso & Vaisman, 2013) or expanding the data to obtain all the possible sequences that could be generated with the observed mixtures (Sheik Amamuddy *et al.*, 2017; Shen *et al.*, 2016; Yu *et al.*, 2014).

In this paper, we propose the use of kernel functions specifically adapted to the aforementioned HIV data intricacies, and able to integrate the relevance of the major resistance-associated protein residues. Kernels are mathematical functions with interesting properties. They can be coupled to numerous machine learning algorithms, the so-called kernel methods, and provide a framework to deal with data of virtually any type (*e.g.* vectors, strings, graphs). They can also encode complementary knowledge about a problem, as long as some mathematical conditions are satisfied (Schölkopf *et al.*, 2004). Our aim using kernel functions that address the aforementioned HIV data particularities was not only to improve prediction, but also reduce pre-processing, thus preserving the data integrity and lowering the risk of inserting spurious patterns.

4.2 Methods

4.2.1 Datasets and data pre-processing

The Genotype-Phenotype Stanford HIV Drug Resistance Database (n.d) is a public dataset with sequences from HIV isolates and its relative susceptibility to several antiretroviral drugs.

We retrieved the PhenoSense dataset from Stanford webpage (version date: 2019-2-20). The data is split in four databases (PI, NRTI, NNRTI and INI), which contain between 1,000-3,500 HIV isolates. INI is a new addition to the Stanford database and includes some of the drugs most recently approved for therapeutic use. The complete dataset contains eight protease inhibitors: atazanavir (ATV), darunavir (DRV), fosamprenavir (FPV), indinavir (IDV), lopinavir (LPV), nelfinavir (NFV), saquinavir (SQV) and tipranavir (TPV); five integrase inhibitors: bictegravir (BIC), cabotegravir (CAB), dolutegravir (DTG), elvitegravir (EVG) and raltegravir (RAL); and two classes of reverse transcriptase inhibitors: six NRTIs, lamivudine (3TC), abacavir (ABC), zidovudine (AZT), stavudine (D4T), didanosine (DDI) and tenofovir (TDF); and four NNRTIs, efavirenz (EFV), etravirine (ETR), nevirapine (NVP) and rilpivirine (RPV). Sequence length is 99 amino acids in the case of PI database, 288 in the case of INI database and 240 in the case of NRTI and NNRTI databases. The dataset contains the strain virus resistance (relative IC50) to each drug, and the sequence of the protein targeted by this drug. We built the regression models for each drug separately, taking each polymorphic protein position as a predictor variable and the drug resistance value as the target variable. Since the distributions of resistances are highly skewed we used the log-transformed values, as recommended in (Bonet, 2015). Redundant viruses obtained from the same patient were removed to minimize bias. We deleted all sequences affected by events that changed protein length (protein truncations, insertions and deletions). These events were uncommon in the dataset and affected less than 5% of HIV sequences. Also, we removed all isolates with one or more missing values. Missing values are present in the target variables as well as in the sequences, because not all HIV isolates have been tested for all drugs. The final number of data instances for each drug is shown in Table 4.1. To ensure a minimum of data rows for training/test partitions and cross-validation, we did not consider drugs with a sample size lower than 100.

Table 4.1 Final number of HIV isolates per drug.

Drug	Data size	Drug	Data size	Drug	Data size	Drug	Data size
ATV	1058	SQV	1603	DDI	1511	BIC	84
DRV	665	TPV	766	TDF	1236	CAB	0
FPV	1559	3TC	1482	EFV	1511	DTG	209
IDV	1607	ABC	1513	ETR	502	EVG	577
LPV	1372	AZT	1502	NVP	1517	RAL	636
NFV	1654	D4T	1510	RPV	181		

4.2.2 Methods

We compared the performance of a nonlinear, nonkernel method (RF) to a kernel method: SVM. SVM can be either linear or nonlinear, depending on the kernel used. The linear kernel is the simplest of all kernel functions, given by the inner product of two vectors in input space, \mathbf{x}_i and \mathbf{x}_j :

$$Lin(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \quad (4.1)$$

In our case, \mathbf{x}_i and \mathbf{x}_j represent the protein sequence of two HIV isolates, i and j , recoded as dummy variables (Belanche & Villegas, 2013). We used this kernel as the linear method of reference. An alternative expression is:

$$wLin(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^D \omega_k x_{ik} x_{jk} \quad (4.2)$$

where D is the length of the sequence. This expression stresses the possibility of assigning a weight ω_k to each protein position, as it is known that not all positions contribute equally to the virus resistance (Iyidogan & Anderson, 2014). Weights are nonnegative and sum to one. We considered two options: the simplest one was to consider that all positions have the same importance, *i.e.*, assigning equal weight $1/D$ to all variables. The second one was including additional information into the kernels, using RF mean decrease in node impurity as a metric for position importance.

RBF kernel

It is a nonlinear kernel, usually defined as:

$$RBF(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad (4.3)$$

Where $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ is the squared Euclidean distance between two vectors, and $\gamma > 0$ is a hyperparameter. As in the case of the linear kernel, the original data was recoded. We also introduced the possibility of weighting the positions:

$$wRBF(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \sum_{k=1}^D \omega_k (x_{ik} - x_{jk})^2) \quad (4.4)$$

The RBF kernel is a widely accepted default method (Belanche & Villegas, 2013; Schölkopf *et al.*, 2004), so we used it as a benchmark to compare with the categorical kernels.

Overlap kernel

This is the most basic categorical kernel. This kernel assigns 1 if the two instances compared are equal and 0 otherwise.

$$Ov(x_{ik}, x_{jk}) = \begin{cases} 1 & \text{if } x_{ik} = x_{jk} \\ 0 & \text{if } x_{ik} \neq x_{jk} \end{cases} \quad (4.5)$$

where x_{ik} and x_{jk} represent the alleles of a given protein position k in two HIV sequences, x_i and x_j .

Jaccard kernel

The Jaccard index measures the similarity between two finite sets and is a valid kernel function (Bouchard *et al.*, 2013). We used it to handle allele mixtures, while in the rest of methods we randomly sampled one allele of the mixture. Letting again k denote a given protein position (so that X_{ik} and X_{jk} are non-empty sets of alleles in the k -th position for isolates i and j) then:

$$Jac(X_{ik}, X_{jk}) = \frac{|X_{ik} \cap X_{jk}|}{|X_{ik} \cup X_{jk}|} \quad (4.6)$$

When $|X_{ik}| = |Y_{ik}| = 1$, *i.e.*, none of the individuals have an allele mixture at that k -th position, Jaccard reduces to the Overlap kernel. Unlike Overlap, the Jaccard kernel can deal simultaneously with allele mixtures and categorical data.

“RBF-like” categorical kernels

For the whole protein sequences, we can aggregate all single position Overlap and Jaccard evaluations as the convex combination of kernels evaluations –(4.5) or (4.6)– and position weights. This results in a valid kernel function, since the product of a positive scalar and a kernel is a kernel, and the sum of kernels is also a kernel. To ensure that the only difference between categorical kernels and RBF was the categorical part, we introduced an exponential factor and the hyperparameter γ , in a way analogous to (4.3) and (4.4):

$$\kappa_{cat}(x_i, x_j) = \exp(-\gamma) \exp\left(\gamma \sum_{k=1}^D \omega_k \cdot \kappa(x_{ik}, x_{jk})\right) \quad (4.7)$$

This is also a valid kernel function, since the exponential of a kernel gives another kernel, and where $e^{-\gamma}$ normalizes the kernel matrix, keeping the evaluations between 0 and 1. The final versions of the Overlap and the Jaccard kernels are obtained replacing the $\kappa(x_{ik}, x_{jk})$ term by (4.5) or (4.6), respectively. In our analyses, we compared weighted and unweighted versions

for all linear, RBF, Overlap and Jaccard kernels. Thus we can ensure a fair comparison between the categorical and the noncategorical kernels.

Stacked models

So far, we have built prediction models for each inhibitor separately. As mentioned in the Introduction (section 4.1), it is reported that there exists some degree of relationship between the resistance of different drugs (*e.g.* in case of cross-resistance). To check whether the use of this information can improve prediction, we implemented the stacking algorithm described in Xing *et al.* (2019) for continuous outcomes. This meta-learner approach consists of two principal steps. In the first step, single drug models are built from the training data as usual. In the second step, the fitted values (*i.e.* predictions of the training data) of all drugs obtained in step 1 are used as input to a new (stacked) model, being each drug a different predictor. The method that integrates the single drug models in step 2 and delivers the definitive predictions is called a combiner algorithm. Data size largely varied between drugs (see Table 4.1), even within the same drug class, so we chose Decision Trees (DT) as our combiner algorithm, as they can easily handle missing data. We combined the drugs within the same database (PI, NRTI, NNRTI and INI) and applied this stacking methodology to our previously proposed weighted kernels (Linear, RBF, Overlap and Jaccard).

4.2.3 Experimental setup and model tuning

To assess the performance of the methods used, each database was split at random in two partitions: training set (60% of the database) and test set (40%). Hyperparameter optimization was done by a 10x10 cross-validation on the training set. Once the optimum hyperparameter was found, the final model was built using the whole training set. To assess the model performance, the NMSE (Normalized Mean Square Error) between the actual and the predicted drug resistances of the test set was computed:

$$NMSE(observed, predicted) = \frac{\sum(observed - predicted)^2}{(N - 1) \cdot var(observed)} \quad (4.8)$$

NMSE can be understood as the fraction of target variance not explained by the model.

We repeated the whole process 40 times, each time with different 60/40 randomly split training/test partitions, to obtain an error distribution. Kernel position weights were calculated using the training set only. Note that only the Jaccard kernel can directly handle allele mixtures; for the rest of kernels and the RF, we generated 40 versions of the database randomly sampling one allele at a time. Then, the 40 replicates were used to compute all the

models except Jaccard, which could deal directly with the database without further preprocessing. This way we can ensure an honest comparison between Jaccard and the rest of kernels and methods.

All analyses were implemented in the R statistical computing language. A documented package implementing these methods is available at https://bitbucket.org/elies_ramon/catkern/.

4.2.4 Visualization

Kernel PCA (kPCA) is a kernel method obtained by coupling kernel functions to a Principal Components Analysis. We used the Jaccard kPCA to visually check whether sequences that are considered more similar by the kernel function are also similar in their drug resistance. As this method is for visualization purposes only, we did not separate training and test sequences. Thus, we used the mean kernel weights of the 40 training sets to compute the weighted Jaccard.

To check whether the important protein positions (*i.e.* kernel weights) detected by RF could have an structural relevance, we highlighted our top ranking positions on the three-dimensional structure of the protein. Pictures of protein-drug complexes were generated with Molsoft ICM-Browser v.3.7-2 using structural data obtained from RCSB Protein Data Bank.

4.2.5 Performance comparison to other approaches

We compared our SVM plus weighted Jaccard with the ANN approach described in Sheik Amamuddy *et al.* (2017), which to our knowledge achieves the best performance so far in this dataset. We used the R interface to keras to implement the ANN. First, we followed the specifications described in Sheik Amamuddy *et al.* (2017) about the range of candidate architectures (1-3 hidden layers, with 2-10 nodes per layer, for all drugs), number of epochs and early stopping. As our dataset version and data pre-processing differ from Sheik Amamuddy *et al.* (2017), we also evaluated a different range of hyperparameters: three fixed ANN architectures (one hidden layer with 30 nodes, two hidden layers with 20 and 10 nodes respectively, and three hidden layers with 30, 20 and 10 nodes) with the L^2 regularization parameter λ . Both approaches (from now on referred to as ANN1 and ANN2) were trained and tested as for the rest of methods (see: Data and dataset pre-processing), with the previously described 40 replicates, allele mixture treatment, training/test ratio and 10x10 cross-validation to choose the best number of layers and nodes per layer (in the case of ANN1) or λ (in the case of ANN2). We chose the best architecture obtained in training within ANN1 and ANN2 options for each drug.

4.3 Results

As expected, HIV protein sequences showed a large variability. As many as 93% of the protease positions were polymorphic and, among these, the number of different observed alleles varied between 2 and 16. In the case of reverse transcriptase, 89% of the positions were polymorphic and the number of alleles per polymorphic position ranged between 2 and 14. Integrase was the least variable protein: 75% of the positions were polymorphic and, in these positions, the number of alleles ranged between 2 and 8. Almost 60% of the sequences had at least one allele mixture.

Figure 4.1 shows the NMSE distribution boxplot for four representative drugs: FPV (PI database, panel A), DDI (NRTI database, panel B), NVP (NNRTI database, panel C) and EVG (INI database, panel D). The remaining 17 boxplots can be found in Supplementary material Chapter 4, figures S1-S9 (Annexes).

4.3.1 Performance overview

NMSE varied greatly across drugs and methods. The best prediction was achieved for 3TC, with an average NMSE ranging 0.07-0.16 depending on the method used (Figure S4, right). The drug with worst prediction error was DTG, with an average NMSE ranging 0.65-0.75 (Figure S8, right). This was also the second drug with lowest data size (Table 4.1). Not unexpectedly, methods applied to drugs with low N had considerably worse performance overall (especially DTG, RPV, ETR and TPV, but also TDF and to some extent DRV). In the PI database, errors were fairly similar across all drugs and around 0.12-0.20 on average (*e.g.* Figure 4.1A), with the sole exception of TPV, with an average NMSE ranging 0.30-0.45. In turn, predictive performances for the integrase and reverse transcriptase inhibitors were far more variable across drugs. Overall, the best method was the SVM with the Jaccard kernel (either in its weighted or in its unweighted version), which achieved the best performance in 20 out of 21 drugs.

Unweighted case

Nonlinear kernels performed much better than the linear kernel in almost all drugs, with the only exception of ETR and D4T. Categorical kernels outperformed RBF, although RBF was close to Overlap (or even marginally better) in some cases. Among categorical kernels, the Jaccard kernel performed better than Overlap in all inhibitors, sometimes by a large margin, as in the cases of SQV, 3TC, AZT, EFV, NVP, RAL or EVG (Figure 4.1 panels C and D). Predictive performances of unweighted kernels and of RF were markedly different in protease with respect to integrase and transcriptase inhibitors. RF was consistently worse than kernel

methods for the PI database (e.g. Figure 4.1A), whereas RF performance was comparable or better than those of kernel methods in both reverse transcriptase and integrase inhibitors (e.g. Figure 4.1B, C and D).

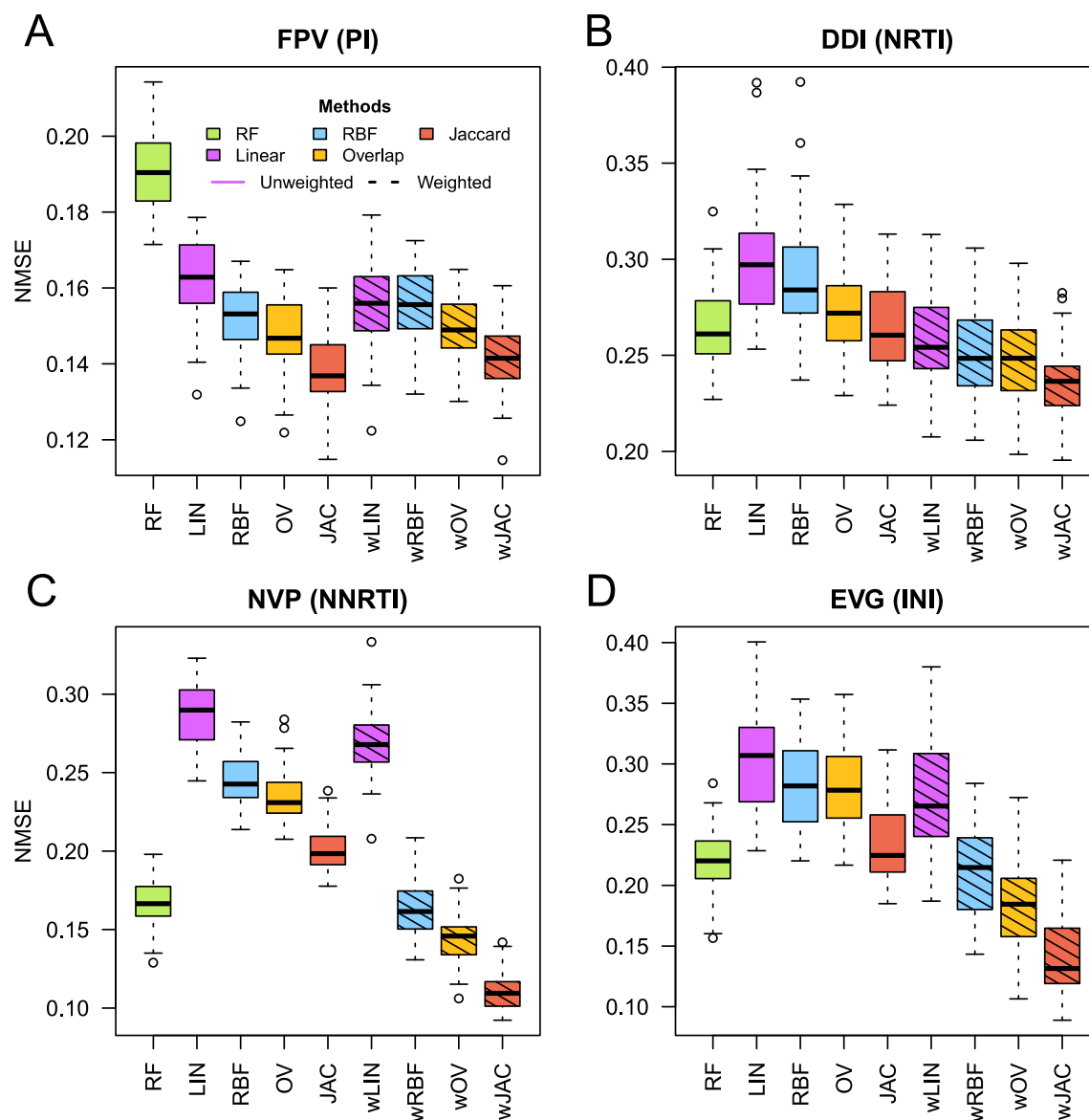


Figure 4.1. NMSE distributions for a PI (FPV), an NRTI (DDI), an NNRTI (NVP) and an INI (EVG). Note that NMSE scale varies between panels.

Weighted case

Figure 4.2 shows three representative examples of the weights obtained from RF. The remaining plots are shown in Suppl mat, figures S10-S27. We ascertained that RF detected most of the major resistance-associated positions described in the literature –e.g. review in Iyidogan and Anderson (2014)–. Overall, a higher percentage of relevant positions were identified in protease inhibitors than in both reverse transcriptase and integrase inhibitors. To

evaluate this numerically, we computed the Gini index of the RF importance distributions for each of the drugs. This index is shown in Figure 4.2 and Suppl mat, figures S10-S27. We also noticed differences regarding the location of the important positions in the three-dimensional structures of protease (Figure 4.3A) and reverse transcriptase (Figure 4.3B). The most important protease positions according to RF are distributed over the whole structure, whereas in the case of the reverse transcriptase they are located at the drug binding site.

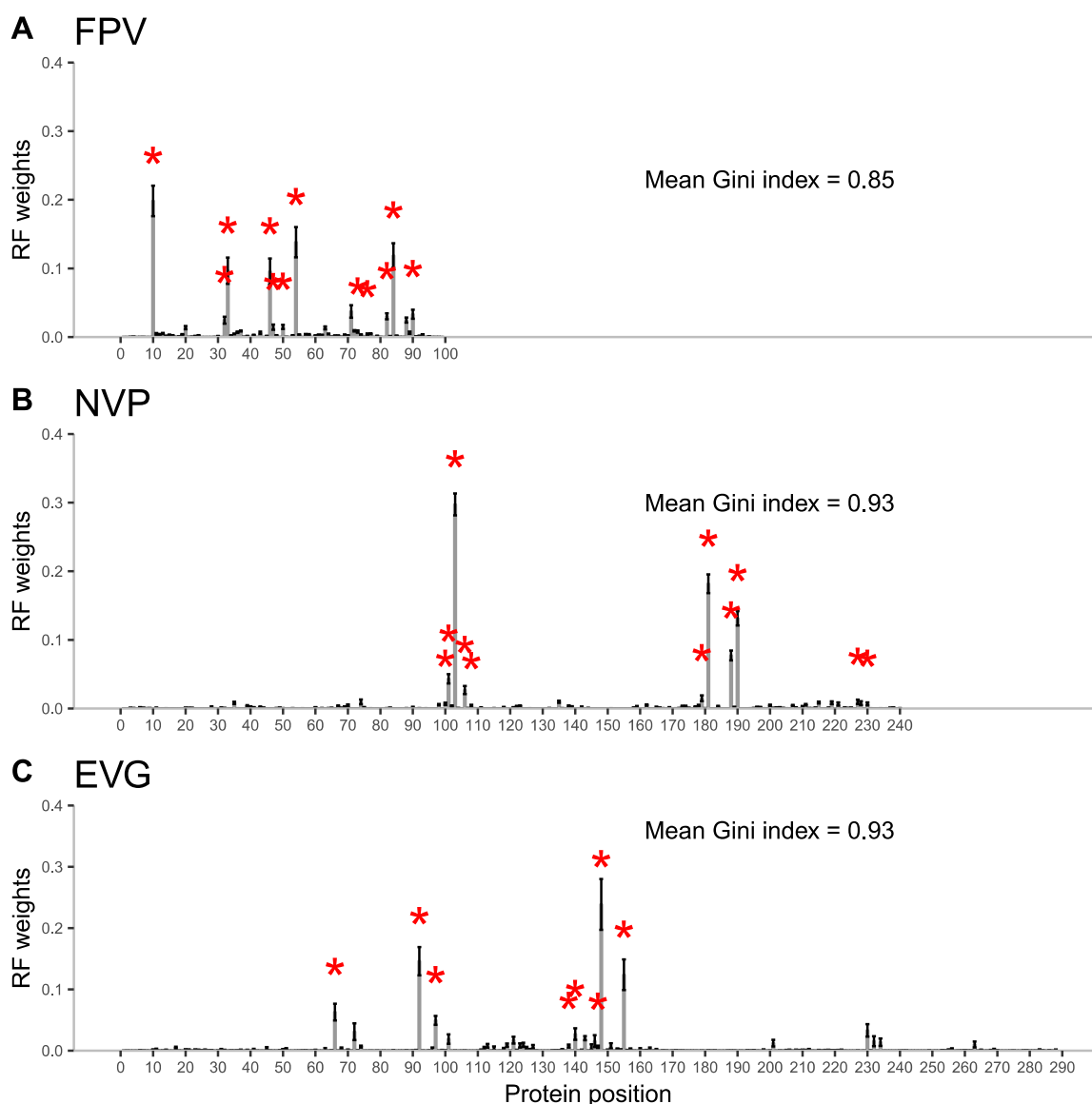


Figure 4.2. RF relative importance of each protein position for three drugs: a protease inhibitor (A), a reverse transcriptase inhibitor (B) and an integrase inhibitor (C). Standard error across the 40 replicates is marked with error bars. Asterisks highlight the major drug related positions reported in the literature (Iyidogan & Anderson, 2014).

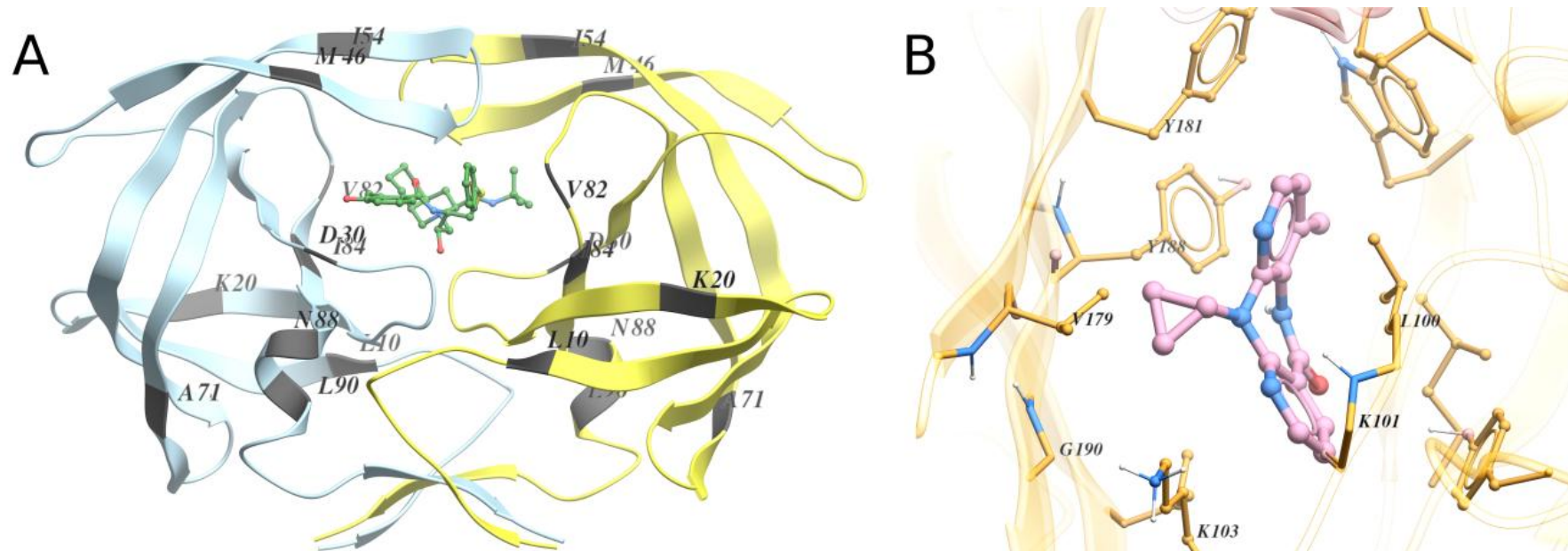


Figure 4.3. A: Wild type protease (in yellow and blue) with an inhibitor (NFV, in green) (PDB code: 3EKX). We highlight the ten most important positions according to RF: 10, 90, 54, 46, 71, 88, 84, 30, 20 and 83. These positions are scattered throughout the protein and only a few belong to the drug binding site (*e.g.* 30, 82 and 84). Mutations at the binding site reduce the affinity for the inhibitor, but can impair the protease catalytic activity as a collateral damage. Mutations in distant residues are typically concurrent with these binding site mutations and often have a compensatory role (*e.g.* stabilizing the protease structure or restoring the catalytic activity). Position 30 appears to be important only in the case of the NFV drug, while the other positions are found in all (or almost all) protease inhibitors. This agrees with the literature (Iyidogan & Anderson, 2014). B: Binding pocket of the reverse transcriptase (in yellow) with an NNRTI (NVP, in pink) (PDB code: 3V81). We highlight the five most important positions for NVP according to RF: 103, 181, 190, 188 and 101. All these positions reside in the NNRTI binding pocket of the enzyme, and also appear in the other NNRTIs analyzed. Thus, in EFV, we find 100 (but not 181) in the top 5; and in ETR, we have 179 instead of 188 (also highlighted). Positions 103 and 101 are located near the entry of the inhibitor binding pocket and, when mutated, interfere with the entrance of the inhibitor to the binding site. Y181 and Y188 have a crucial contribution the NVP binding via stacking interactions between its side chains and the inhibitor aromatic groups. G190 mutations lead to resistance through steric hindrance, because of the substitution by a more voluminous side chain. L100 effect is also related to steric hindrance (Iyidogan & Anderson, 2014).

As for predictive performance, weighting was more effective in integrase and reverse transcriptase inhibitors than in protease inhibitors. In NRTI and NNRTI databases, weighted kernels outperformed RF in all cases, whereas their unweighted counterparts did not. This was particularly the case for 3TC, DDI (Figure 4.1B), EVG (Figure 4.1D) and especially NVP (Figure 4.1C), where weighting decreased the Jaccard kernel error by around 50%. In contrast, the effect of weighting was less marked in the PI database: similar errors were obtained (*e.g.* Figure 4.1A) for all drugs but TPV, where the error actually increased. In the INI database, weighting decreased dramatically the error in RAL and EVG drugs but not in DTG. In summary, Jaccard was the best weighted kernel followed by Overlap, RBF and Linear.

4.3.2 Factors affecting prediction error

To investigate the relevance of each factor in prediction, we fitted the following linear model to NMSE obtained in each replicate across all kernels and drugs (40 replicates x 21 drugs x 8 kernels):

$$NMSE \sim N + \mathcal{K} + W + GINI + \epsilon$$

where N is the drug data size (Table 4.1), \mathcal{K} is a class variable with the kernel used (Linear, RBF, Overlap or Jaccard), $W = 0$ or 1 depending on whether the kernel was unweighted or weighted, respectively, and $GINI$ is the standardized Gini index of RF weights. Table 4.2 summarizes the coefficients and their significance. We found that all factors are significant and behave additively (interactions were not significant; results not shown). As expected NMSE decreases with N but, interestingly, also with Gini index, *i.e.*, prediction improves when there are only a few positions of large effect. Categorical kernels were consistently better than noncategorical ones and Jaccard was the best option in all cases. Weighting protein positions significantly lowers the error, although only in reverse transcriptase and integrase inhibitors (as also observed in Figure 4.1 and Suppl mat, figures S1-S9).

Table 4.2. Linear model coefficient estimates and p-values.

	All drugs	PIs	NRTIs + NNRTIs	INIs
N increment	$-3.1 \cdot 10^{-4} ***$	$-3.0 \cdot 10^{-5} ***$	$-1.8 \cdot 10^{-4} ***$	$-1.3 \cdot 10^{-3} ***$
Unweighted → Weighted	$-3.0 \cdot 10^{-2} ***$	$3.0 \cdot 10^{-3}$	$-3.4 \cdot 10^{-2} ***$	$-3.1 \cdot 10^{-2} ***$
Gini Index increment	$-4.9 \cdot 10^{-3} ***$	$-6.0 \cdot 10^{-2} ***$	$-5.4 \cdot 10^{-2} ***$	$1.7 \cdot 10^{-2} **$
Jaccard → Linear	$4.5 \cdot 10^{-2} ***$	$3.1 \cdot 10^{-2} ***$	$5.0 \cdot 10^{-2} ***$	$9.4 \cdot 10^{-2} ***$
Jaccard → RBF	$3.5 \cdot 10^{-2} ***$	$1.3 \cdot 10^{-2} ***$	$3.8 \cdot 10^{-2} ***$	$4.7 \cdot 10^{-2} ***$
Jaccard → Overlap	$1.9 \cdot 10^{-2} ***$	$9.3 \cdot 10^{-3} ***$	$3.1 \cdot 10^{-2} ***$	$3.6 \cdot 10^{-2} ***$

Legend: Significance codes are 0 *** 0.001 ** 0.01 * 0.05

To visualize the impact of Gini index not ascribable to the effects of data size (N) and the kernel used (\mathcal{K}), we plotted the residuals of model $NMSE \sim N + \mathcal{K}$ against $GINI$ (Figure 4.4

panels A, B and C). For protease inhibitors, the Gini effect is confined to TPV drug (red dots in Figure 4.4A). The effect is rather linear for reverse transcriptase inhibitors, although NMSE variability was larger than average for RPV (red dots), the drug with lowest N . In the case of integrase inhibitors, Gini takes values in a narrow range and does not seem to have an impact on prediction. As in the case of RPV, large variability in NMSE values is observed in DTG (blue dots), which is the drug with second lowest sample size.

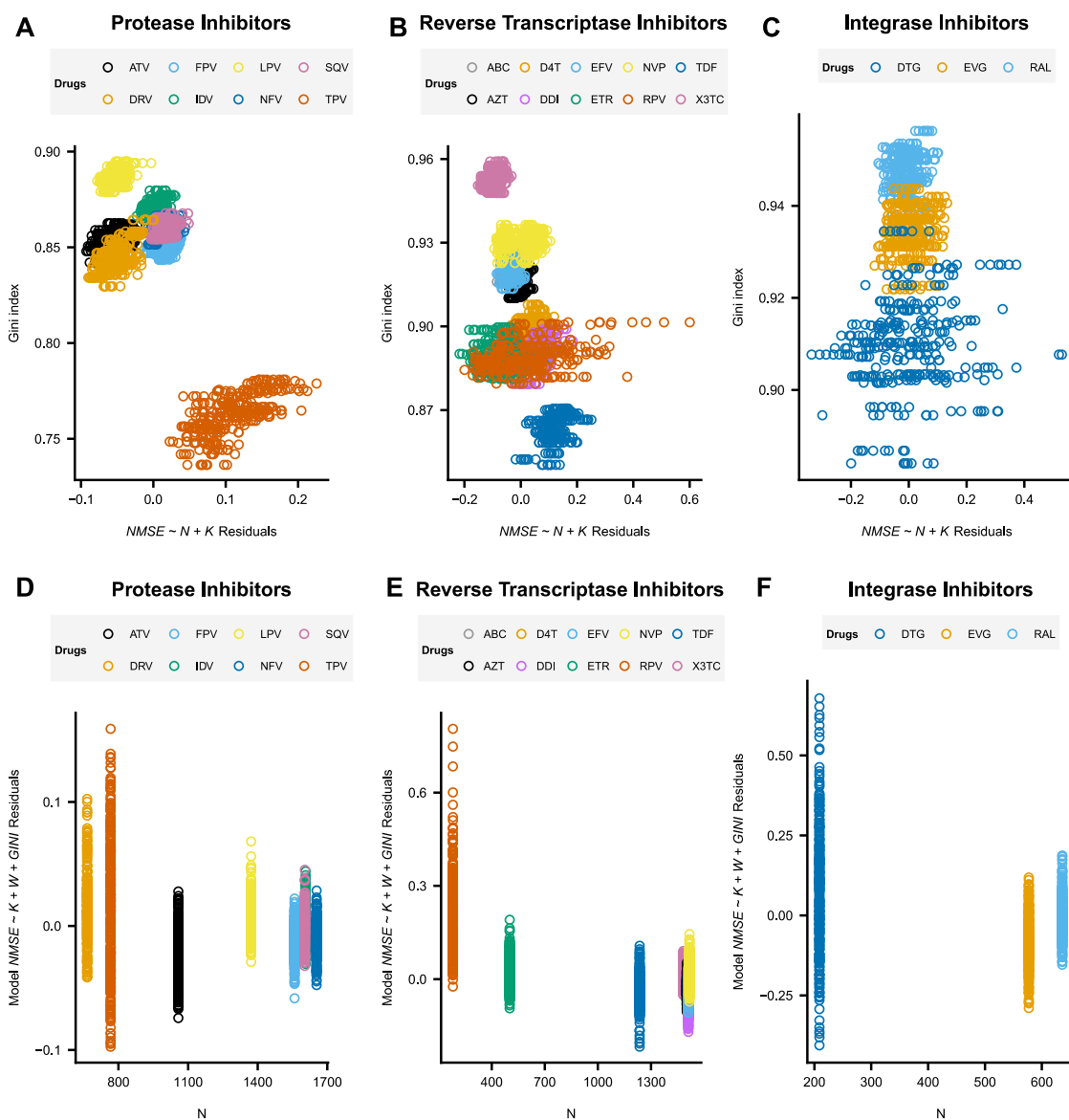


Figure 4.4. A, B and C: NMSE residuals (*observed* – *fitted* values) of the linear model containing only data size (N) and kernel (K) vs. Gini index. Each color represents a different drug. Note different scale for the Gini index between panels. D, E and F: Residuals (*observed* – *fitted* values) of the linear model containing K , W and $GINI$ vs. data size (N). Each color represents a different drug.

Sample size is one of the most important factors in any experimental design, and the main one influencing total cost. Figure 4.4 panels D, E and F show the residuals of model $NMSE \sim \mathcal{K} + W + GINI$ vs. N . Although Table 4.2 shows that the NMSE decreases with

sample size for all drugs and proteins, a clear trend appears only for reverse transcriptase inhibitors. In this case, a law of diminishing returns is observed, and adjusted NMSE decrease with N is very small for $N > \approx 600$.

4.3.3 Kernel PCA

Even if weighting increases prediction accuracy overall, the effect was markedly different when we compare reverse transcriptase and integrase with protease (Table 4.2). In the latter protein, weighted kernels were not clearly superior. To further investigate this issue, we performed a PCA on the Jaccard kernel. Figure 4.5 shows the results of for FPV (a protease inhibitor, panels A and B) and NVP (a reverse transcriptase inhibitor, panel C and D), both with unweighted and weighted Jaccard kernels. The remaining figures can be found at Suppl mat, figures S28-S46. Unweighted kPCA results, overall, in a good, spectrum-like separation between resistant and susceptible isolates for protease inhibitors, whereas weighted kernels can improve dramatically the separation in the case of reverse transcriptase. The integrase inhibitors RAL and EVG behave similarly to reverse transcriptase inhibitors, while DTG (which has a very small sample size) do not achieve a good separation either in the weighted or the unweighted kPCAs.

4.3.4 Stacked models

We compared the performances of four methods (SVM plus weighted Linear, RBF, Overlap and Jaccard kernels) with those of their stacked counterparts in Suppl mat, tables S1 (mean NMSE) and S2 (NMSE standard error). Intriguingly, we found that the stacked versions of SVM with weighted kernels have similar performances to those of the individual models. This suggests that all the information of the sequence has been already extracted in the first step, and so stacking the models was of no additional value.

4.3.5 Performance comparison to other approaches

Figure 4.6 shows the performance comparison between our best method (SVM with weighted Jaccard kernel) with the ANN1 and ANN2 (see subsection 4.2.5). ANN2 tends to have better performance than ANN1, especially in drugs with small sample size, but also presents greater standard errors in some drugs. In the case of protease inhibitors (panel A) both ANN1 and ANN2 are only marginally worse than the weighted Jaccard SVM, with the exception of the FPV drug. In the case of reverse transcriptase and the integrase inhibitors (panels B, C and D), the difference between the performance of weighted Jaccard and the ANN increases. The

latter method presents higher NMSE and larger standard errors, especially for 3TC, DDI, TDF, the NNRTIs, and the INIs.

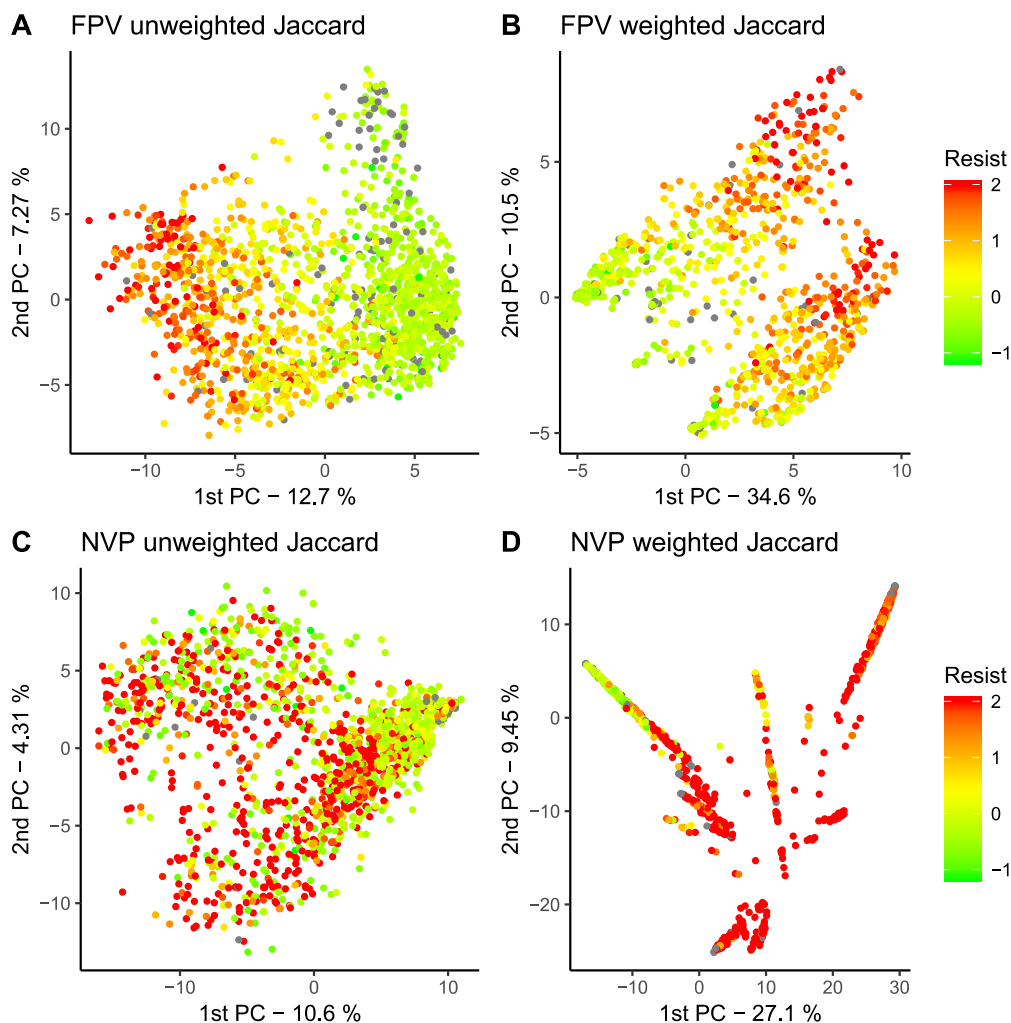


Figure 4.5. The Jaccard kPCA in a protease inhibitor (FPV, panels A and B) and a reverse transcriptase inhibitor (NVP, panels C and D). Panels A and C correspond to unweighted Jaccard, and B and D to weighted Jaccard. Dot color represents the actual log-resistance value for each specific drug; in red the more resistant, and in green the least resistant. Sequences with missing resistance value are in gray.

4.4 Discussion

Recent results on predicting HIV drug resistance as a regression problem can be found in Sheik Amamuddy *et al.* (2017) and Shen *et al.* (2016). Shen *et al.* (2016) used RF and computed the 5-fold cross-validation R^2 . Sheik Amamuddy *et al.* (2017) used ANN and computed the R^2 of the test set without replicates. The two approaches were based in a previous version of the Stanford dataset (version date: 2014-9-28) and share a similar treatment of amino acid mixtures based on sequence expansions. We did a comparison with

the ANN, which to our knowledge achieved the best performance so far in this dataset (Sheik Amamuddy *et al.*, 2017). We observed that weighted Jaccard outperforms ANN in all drugs, and that the ANN prediction performances were worse than those originally reported (which had R^2 values ranging between of 0.85 and 0.99). It has to be stressed, however, that we used different versions of the dataset (the version used by Sheik Amamuddy *et al.* (2017), for instance, did not contain information about the INIs) and that we followed very different strategies concerning pre-processing. In Sheik Amamuddy *et al.* (2017), a pre-processing with removal of outliers and rare variant filtering is performed, which can result in a loss of generalizability, as is acknowledged by the authors. Another reason for the discrepancy is probably the treatment of allele mixtures, as we discuss next.

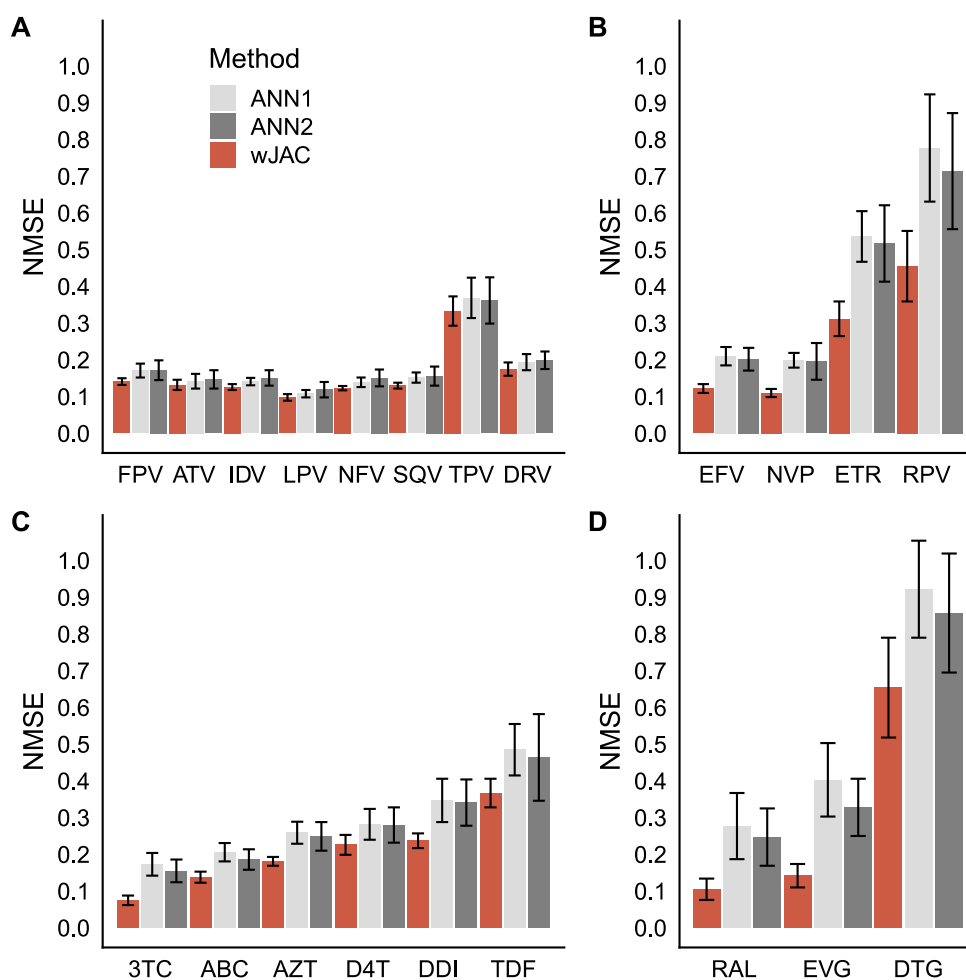


Figure 4.6. Mean NMSE values and their corresponding standard errors for the SVM + weighted Jaccard kernels (red), ANN1 (light gray) and ANN2 (dark gray). PIs are shown in panel A, NRTIs in panel C, NNRTIs in panel B and INIs in panel D.

In this work, we present a novel approach to predict drug resistance in HIV, using kernel functions that directly address the presence of allele mixtures and the categorical nature of

the data. Previous work handled these two issues using several pre-processing strategies. Categorical data are systematically recoded into numeric data, usually in the form of dummy data or, in Sheik Amamuddy *et al.* (2017), assigning an integer to each category. Here, we have shown that addressing the categorical nature of the data and the presence of allele mixtures does lower the test error in comparison to the dummy variable approach (Table 4.2). In fact, even the simplest categorical kernel (*i.e.* the Overlap kernel) improves prediction upon the standard RBF kernel, although the extent of the improvement depends on the specific drug. It has to be stressed that recoding the categorical data into dummy variables increases the dimensionality of the problem, thus increasing computation needs and leading to sparse datasets. As this effect depends on the number of different categories of the variables, categorical methods may be more useful when data has more than few categories. Coding the different alleles as an integer does not increase the dimensionality either, but introduces an order without biological meaning among the amino acids.

The treatment of amino acid mixtures is more challenging. In the data analyzed we observed that it is a widespread phenomenon: about 60% of the sequences had at least one mixture. Mixtures introduce ambiguity in the genotype-phenotype correlation since it makes impossible to know the actual sequences of strains. Also, the quasispecies distribution may have undergone undefined modifications during the *in vitro* assay (Perrin & Telenti, 1998). Previous approaches to deal with this issue included keeping the most frequent amino acid of the mixture (Tarasova *et al.*, 2018) and sequence expansion (Sheik Amamuddy *et al.*, 2017; Shen *et al.*, 2016; Yu *et al.*, 2014). The latter strategy consists in expanding the data to sequences with single amino acids at each mixture location until all possible combinations have been exhausted. These “derived” sequences share the resistance value, *i.e.*, the resistance of the original sequence. This approach dramatically enlarges data size (in the aforementioned works, minimum by a 10x factor in the protease inhibitors and almost a 30x in the reverse transcriptase inhibitors). This might be one of the main reasons for the discrepancy between the ANN performance computed in this work and in Sheik Amamuddy *et al.* (2017). Without expansion, the data size ranges between 200-1500, but the number of (dummy) variables is almost 2000 in the PIs, and more than 4000 in the other drugs. The higher number of variables compared to observations might have adversely affected the ANN performance in comparison to the original work and, also, in comparison to SVM, as the latter are less prone to over-fitting. Furthermore, the expansion potentially biases the dataset by over representing sequences with mixtures (especially those with a larger number of mixtures and/or alleles per mixture) and it can generate HIV variants not found in the patient. Expansion also increases the difficulty of the training/test splitting because all expansions of the same sequence must be placed either in the training set or in the test set; otherwise, the

independence of both sets is lost. In our work, we preferred keeping only one amino acid of the mixture, which is allegedly the most conservative pre-processing choice. This differs from *e.g.* Tarasova *et al.* (2018), because we keep one amino acid at random, while they pick the most frequent one, which is sound if mixtures are considered a technical artifact. However, in case of HIV, this event mostly reflects the coexistence of actual HIV variants in the body of the patient (Iyidogan & Anderson, 2014; Shafer *et al.*, 2001; Schmidt, 2000; Perrin & Telenti, 1998) and the ambiguity lies in the resistance value delivered via the *in vitro* test. In any case, part of the original information is lost by picking one of the alleles of the mixture. This does not happen when using the Jaccard kernel, which naturally handles allele mixtures. We have shown that Jaccard is clearly the best among kernels assessed and that also improves the RF results, in most cases by a large margin. Both Overlap and Jaccard are basic kernel functions, but our kernel definition (4.7) is general enough to replace them for more sophisticated categorical kernels, perhaps with improved prediction performance.

An additional theoretical proposal was to weigh kernel positions according to its inferred influence on drug resistance. Here we employed RF decrease in impurity as weights but numerous options are equally justified and so additional research on this topic is warranted. Using RF we were able to identify, from protein sequence alone, important positions for the drug resistance that have a structural meaning (Figure 4.3). We observed a distinct effect of weighting in protease inhibitors and transcriptase reverse inhibitors that correlates with the distribution of the importances. At least part of this behavior might be due to differences in the mutational pattern between the two enzymes in regards to drug resistance. In the reverse transcriptase, the major resistance mutations tend to be located in specific positions, particularly at the drug binding sites of the N-terminal side, weakening the affinity between drug and enzyme. As early as 1998, it was noted that a single mutation of the reverse transcriptase may confer high resistance to drugs like 3TC and NVP (Perrin & Telenti, 1998), whereas the virus acquires resistance to protease inhibitors by accumulating mutations. First, primary resistance mutations arise at the active site pocket and the surrounding residues. But, as these mutations often cause conformational changes, additional secondary mutations that compensate the impaired catalytic activity and stabilize the protease tend to be selected in turn. There are at least 36 important residues (out of a total of 99) involved in protease drug resistance mutations and (unlike reverse transcriptase) they are distributed along the whole sequence (Iyidogan & Anderson, 2014). These differences may explain why RF, and therefore the weighted categorical kernels, performed better at the NRTI and NNRTI databases. Further, the estimate of the variable importance is more reliable when few

relevant protein positions have a large impact on resistance. In contrast, the compensatory secondary mutations of the protease probably introduce some degree of correlation between protein positions, which may explain why weighting in PI database does not result in a clear improvement of performance.

4.5 Conclusions

Machine learning is an effective approach to predict HIV drug resistance, and a straightforward alternative to the much slower and expensive *in vitro* assay. Results show that kernels that take into account both the categorical nature of the data and the presence of mixtures consistently result in the best prediction model. As for the introduction of position weights, we found that the amount of improvement was a function of the number of positions with large effect on drug resistance, which may be related to the known different mutational patterns regarding drug resistance among the viral proteins. Using more sophisticated categorical kernels and/or kernels able to take into account structural information may improve even more the resistance prediction.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The datasets analyzed during the current study are available in the Genotype-Phenotype Stanford HIV Drug Resistance Database repository, <https://hivdb.stanford.edu/pages/genopheno.dataset.html>. Structural data can be found at <https://www.rcsb.org/structure/3ekx> and <https://www.rcsb.org/structure/3v81>. Code used in this manuscript is available at https://bitbucket.org/elies_ramon/catkern.

Competing interests

MPE is a member of the editorial board (Associate Editor) of BMC Bioinformatics.

Acknowledgements

A partial version of this work was presented at the 7th International Work-Conference IWBBIO 2019 in Granada, Spain (May 8-10, 2019), and is available on the conference proceedings (LNBI Proceedings Part II).

Funding

This work was funded by project AGL2016-78709-R (Ministry of Economy and Competitiveness, Spain) to MPE. ER has funding from a FI-AGAUR PhD studentship grant (Generalitat de Catalunya). We acknowledge financial support from the Spanish Ministry of Economy and Competitiveness, through the “Severo Ochoa Programme for Centres of Excellence in R&D” 2016-2019 (SEV-2015-0533)”, and from the EU through the BFU2016-77236-P (MINECO/AEI/FEDER, EU).

Authors' contributions

LBM and MPE conceived and supervised research. ER developed code and performed research. ER and MPE wrote the manuscript with help from LBM. All authors read and approved the final manuscript.

CHAPTER 5

STRUCTURAL AND FUNCTIONAL ASSOCIATIONS IN HIV DRUG RESISTANCE PREDICTION WITH KERNEL METHODS

Elies Ramon¹ and Lluís Belanche-Muñoz²

¹ Centre for Research in Agricultural Genomics (CRAG), CSIC-IRTA-UAB-UB, Campus UAB, 08193, Bellaterra, Barcelona, Spain.

² Computer Science Department, Technical University of Catalonia, Carrer de Jordi Girona 1-3, 08034, Barcelona, Spain.

(in preparation)

Abstract

A definitive cure to HIV infection does not exist yet and, thus, patients rely in antiretroviral therapy for life. In this scenario, the emergence of drug resistance is an important concern. The automatic prediction of resistance from HIV sequences is a fast tool for physicians to choose the best possible medical treatment. In a previous work, we devised two kernel functions that took into account the categorical nature and the presence of allelic mixtures in HIV sequence data. Here, we expand the aforementioned kernels to consider that sequence positions may be associated. As background linkage disequilibrium has been reported to be low in HIV due to its high recombination rate, we focused on functional associations between protein positions. Mutated protein sequences were modeled as graphs, with edges weighted by the Euclidean distance (obtained from three-dimensional crystal structures) between residues. A kernel for graphs, the exponential random walk kernel, was combined with the previous categorical kernels to generate kernels *ad hoc* for this data. Despite the potential advantages of this approach, an improvement on predictive ability was not observed.

Keywords: HIV, Drug Resistance Prediction, Protein Structure, Random Walk Kernel, Allele mixture, PI, NRTI, NNRTI, Support Vector Machine, kernel PCA

5.1 Introduction

Actual HIV treatment is based on antiretroviral drugs that target several key stages of the virus replication cycle. Resistance occurs in all classes of antiretrovirals and can pose a threat to the treatment effectiveness. One of the main causes of the emergence of resistant variants is the HIV mutation rate, which is as high as 10^{-4} - 10^{-5} per viral genome and replication cycle. As patients have to follow the therapy for life, drug resistance is a serious concern (Anstett *et al.*, 2017).

In a previous work (see Chapter 4) we used SVM coupled to categorical kernels to predict, from HIV mutated sequences, the resistance of virus isolates to 21 drugs from four different classes: PI (protease inhibitors), INI (integrase inhibitors), NRTI and NNRTI (nucleoside and non-nucleoside reverse transcriptase inhibitors, respectively). We performed a sequence-based prediction that assumed that each one of the sequence positions was completely independent from the others. This approach did not take into account potential cases of linkage disequilibrium or LD, *i.e.* non-random associations between alleles from different *loci*. These associations may arise for different causes; for instance, when the analyzed *loci* are

placed in the same sequence, so specific sets of alleles are co-inherited *en bloc* from a common ancestor. This is called genetic linkage or background LD (Wang & Lee, 2007), and it is known to decay with distance by the action of recombination.

Associations can also be due to structural and functional interactions among protein residues. Protein three-dimensional folding is determined by its sequence and, in turn, determines protein function. Thus, there exist some constraints posed to specific combinations of alleles that completely disrupt the protein structure. Other associations may arise because of selective pressure. Contrarily to background LD, functional associations can be observed between positions that are distant in the sequence but that have some kind of interaction once the protein is folded.

In the case of HIV, recombination occurs at a very high rate, between 10^{-5} - 10^{-4} breakpoints per site per generation, depending on the report (Song *et al.*, 2018). Each HIV virion contains two single-stranded RNA genomes, and it is well known that reverse transcriptase switches between the two during reverse transcription (Schlub *et al.*, 2014). It is reasonable to expect that these high levels of recombination in HIV, together with the short generation time of virions, greatly accelerate the disappearance of background LD. In this direction, Wang and Lee (2007) reported that pairwise associations within *pol* gene are mainly due to functional interactions, and that background LD was present in a much lower amount. The *pol* gene encodes reverse transcriptase, integrase and protease, which catalyze reactions as important for HIV infection as RNA \rightarrow DNA reverse transcription, integration in human genome and cleavage of precursor polypeptides into mature viral proteins. It is widely acknowledged that three-dimensional structure is key for enzymatic activity, as it determines the kind of reactions catalyzed and, also, allows the enzyme to be very selective about its substrates. As antiretroviral drugs are intended to target specific proteins and block their activity, structure also has a great importance in drug effectiveness. For instance, most PIs are competitive inhibitors that mimic the transition state of protease natural substrates, and were the first success of structure-based drug design (De Clercq, 2009). Interestingly enough, Wang and Lee (2007) found that most associations detected in HIV data from therapy-experienced patients vanish in sequences from untreated HIV carriers, which suggest that these interactions are mainly due to drug selection pressure.

In this work, we aim to expand the two categorical kernels over sequences of Chapter 4, the Overlap and the Jaccard kernel, to include structural information. The main difference between these two kernels is that the latter can handle amino acid mixtures, which are very widespread in HIV sequence data. We will take advantage of the modularity in kernel design to combine them with a type of kernels based on random walks in graph data (Vishwanathan

et al., 2010; Borgwardt *et al.*, 2005; Gärtner *et al.*, 2003). These kernels for graphs have been applied successfully to general protein function prediction (Borgwardt *et al.*, 2005). In our case, we will model the three-dimensional folding of HIV viral proteins as graphs and, then, apply over them the extensions of the Overlap and Jaccard kernels for structural data.

5.2 Material and Methods

5.2.1 Protein structures, sequence datasets and data pre-processing

The mutated HIV sequences were obtained from Genotype-Phenotype Stanford HIV Drug Resistance Database (version date: 2019-2-20). Protease sequences and their relative resistance to eight protease inhibitors are in the PI dataset. NRTI and NNRTI datasets contain data of reverse transcriptase sequences and their resistance to two different classes of inhibitors (six NRTIs and four NNRTIs). Finally, the INI dataset contains integrase sequences and resistance to five inhibitors. Both the assessed drugs and the pre-processing of HIV sequences was exactly the same than in Chapter 4.

Structural data was retrieved from RCSB Protein Data Bank for protease (PDB code: 3OXC) and reverse transcriptase (2WOM). We did not find complete crystal structures of integrase, and so the INI dataset was not included in subsequent analyses. Euclidean distance (in Å) between all pairs of protein residues was computed from the (x,y,z) coordinates of their alpha carbons (C α). Then, a graph was generated for each one of the HIV mutated sequences. Each protein position was assigned a different node, labeled with the specific allele of the HIV isolate at that position. Edges connected every pair of two nodes and were labeled with the Euclidean distance between each pair. Possible structural changes in the neighborhood of mutated protein residues were not assessed, and so the same weights were used in all the protease sequences on one hand, and reverse transcriptase on the other. In return, the set of nodes differed among graphs, as each one contained a different HIV sequence.

5.2.2 Random walk kernel

The pairwise similarity between graphs was computed using the exponential random walk kernel (Gärtner *et al.*, 2003):

$$\text{Exp}(G, G') = \sum_{i,j=1}^{|\mathbf{V}_x|} \sum_{k=0}^{\infty} \frac{1}{k!} \gamma^k [\mathbf{A}_x^k]_{ij} \quad (5.1)$$

where G and G' are the graph representations of the same enzyme (protease or reverse transcriptase) coming from two different HIV isolates, \mathbf{A}_x is the adjacency matrix of the direct product graph $G_x = G \times G'$, \mathbf{V}_x is the set of nodes of the direct product graph, and $\gamma \in (0,1)$ is the decay hyperparameter.

To compute \mathbf{V}_x , let V and V' be the set of nodes of G and G' , *i.e.* the protein sequences of two HIV isolates. \mathbf{V}_x can be obtained in a straightforward way by comparing all nodes of both graphs and keeping only those with identical labels (*i.e.* as with an Overlap kernel):

$$\mathbf{V}_x = \{(v_i, v'_i) \in V \times V' : (\text{label}(v_i) = \text{label}(v'_i))\} \quad (5.2)$$

with $v_i \in V$, $v'_i \in V'$, and $\text{label}(\cdot)$ being the allele at the i -th node (protein position). $|\mathbf{V}_x|$, the number of nodes of G_x , is at most $|V| \times |V'|$.

On other hand, the adjacency matrix of a graph is the matrix that contains the weights of its edges. In our case, $[\mathbf{A}]_{ij} = \omega_{ij}$ contains the Euclidean distance between nodes i and j . \mathbf{A}_x can be obtained as the Kronecker product of the adjacency matrices of G and G' , *i.e.* $\mathbf{A}_x = \mathbf{A} \otimes \mathbf{A}'$ (Vishwanathan *et al.*, 2010). For \mathbf{A}_x and \mathbf{V}_x to match, we kept only the ij entries of \mathbf{A}_x such that nodes i and j exist in \mathbf{V}_x , which following (5.2) is the case if $\text{label}(v_i) = \text{label}(v'_i)$ and $\text{label}(v_j) = \text{label}(v'_j)$. Again, the rank of \mathbf{A}_x is at most $|V| \times |V'|$.

The exponential random walk kernel allows using more sophisticated kernels than the Overlap for comparing the nodes of G and G' (Borgwardt *et al.*, 2005). In that case, \mathbf{A}_x in (5.1) should be replaced by:

$$[\mathbf{A}_x]_{ij} = \kappa_{step} \left((v_i, v_j), (v'_i, v'_j) \right) \quad (5.3)$$

Where κ_{step} summarizes the information of nodes i and j and the edge connecting them:

$$\begin{aligned} \kappa_{step} \left((v_i, v_j), (v'_i, v'_j) \right) \\ = \kappa_{node}(v_i, v'_i) \cdot \kappa_{node}(v_j, v'_j) \cdot \kappa_{edge} \left((v_i, v_j), (v'_i, v'_j) \right) \end{aligned} \quad (5.4)$$

We took advantage of the fact that the protein sequences from different isolates are aligned and have equal length. Thus, we first applied the Overlap kernel to ensure that the two nodes v_i, v_j correspond to the same protein position (that is: $i = j$). Then, either the Overlap or the Jaccard kernels for protein sequences were used over the labels on the nodes. The

Exponential-Overlap kernel (expOv) checks for complete equality between the nodes of two proteins from two different isolates and was computed as follows:

$$\kappa_{node}(v_i, v'_i) = Ov(i, j) \cdot Ov(v_i, v_j) \quad (5.5)$$

Instead, the Exponential-Jaccard kernel (expJac) had the advantage of handle the potential amino acid mixtures:

$$\kappa_{node}(v_i, v'_i) = Ov(i, j) \cdot Jac(v_i, v_j) \quad (5.6)$$

We can obtain the edges by applying the Kronecker product over the adjacency matrices of G and G' :

$$\kappa_{edge}((v_i, v_j), (v'_i, v'_j)) = [\mathbf{A} \otimes \mathbf{A}']_{ij} \quad (5.7)$$

However, $Ov(i, j)$ sets most rows and columns to zero. These all-zero rows and columns can be eliminated of \mathbf{A}_x , which now has a maximum rank of $|V|$. Thus, we only need to compute the submatrix of (5.7) for which $i = j$. This can be easily computed as the element-wise product of the adjacency matrices \mathbf{A} and \mathbf{A}' :

$$\kappa_{edge}((v_i, v_j), (v'_i, v'_j)) = [\mathbf{A}]_{ij} \cdot [\mathbf{A}']_{ij} = [\mathbf{A}^2]_{ij} \quad (5.8)$$

as the same adjacency matrix is shared across all protease sequences, on one hand, and all reverse transcriptase sequences, on the other.

5.2.3 Kernel function implementation

At first sight, it seems that a naive implementation of (5.1) has exponential asymptotic complexity due to $\frac{1}{k!} \gamma^k \mathbf{A}_x^k$ summation (even when an upper limit for k is set). This can be substantially improved using dynamic programming, but nonetheless remains the most prohibitive step of the kernel computation, even when a “reduced” \mathbf{A}_x as in (5.8) is used.

In fact, the summation term in (5.1) is a Neumann series and can be computed in an exact form (Kress, 2012). As a Neumann series, it converges in:

$$\sum_{k=0}^{\infty} \frac{1}{k!} \gamma^k \mathbf{A}_x^k = (\mathbf{I} - \gamma \mathbf{A}_x)^{-1} \quad (5.9)$$

Where \mathbf{I} is the identity matrix with the same dimension of \mathbf{A}_x . The cost of matrix inversion is roughly cubic. Another possibility consists in diagonalizing \mathbf{A}_x . This is a squared and symmetric matrix, so its spectral decomposition is:

$$\mathbf{A}_x = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \quad (5.10)$$

where \mathbf{U} is the square matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues. Then, as matrix product is continuous, we can take \mathbf{U} off the sum:

$$\sum_{k=0}^{\infty} \frac{1}{k!} \gamma^k \mathbf{A}_x^k = \mathbf{U} \left(\sum_{k=0}^{\infty} \frac{1}{k!} \gamma^k \mathbf{\Lambda}^k \right) \mathbf{U}^T = \mathbf{U}\mathbf{\Delta}\mathbf{U}^T \quad (5.11)$$

Diagonalizing \mathbf{A}_x greatly facilitates the calculations, as the powers of $\mathbf{\Lambda}$ can be obtained by the power of each entry on the main diagonal. If this approach is combined with (5.9):

$$\mathbf{\Delta} = \sum_{k=0}^{\infty} \frac{1}{k!} \gamma^k \mathbf{\Lambda}^k = (\mathbf{I} - \gamma\mathbf{\Lambda})^{-1} \quad (5.12)$$

Again, the inverse of a diagonal matrix is the inverse of each one of its elements (in this case, the eigenvalues of \mathbf{A}_x), so $[\mathbf{\Delta}]_{ii} = \frac{1}{1-\gamma\Lambda_{ii}}$ and $[\mathbf{\Delta}]_{i \neq j} = 0$. Computing the eigenvalues and eigenvectors of \mathbf{A}_x also has cubic complexity. A third possibility (Gärtner *et al.*, 2003) is:

$$\sum_{k=0}^{\infty} \frac{1}{k!} \gamma^k \mathbf{A}_x^k = e^{\gamma\mathbf{A}_x} = \mathbf{U}e^{\gamma\mathbf{\Lambda}} \mathbf{U}^T \quad (5.13)$$

In this case, the diagonal contains the exponential of the product of γ and \mathbf{A}_x eigenvalues.

5.2.4 Experimental set-up and kernel visualization

All analyses were run in R. To combine the exponential random walk kernel with the Overlap (expOv) and the Jaccard kernels (expJac) we computed κ_{node} using (5.5) and (5.6), respectively. Implementation of the exponential random walk kernel was done as in (5.12) using the R packages Rcpp and RcppArmadillo. The former package provides an API to easily write C++ functions and running them in R, and the latter allows accessing to the linear algebra library Armadillo. That way the computation of the exponential kernel was speed up.

Resistance prediction was done by coupling the expOv and the expJac kernels to SVM. Experimental set-up was mostly the same than in Chapter 4. The difference was that we

randomly sampled the amino acid mixtures only once prior to computing (5.5). The reason was the high cost of computing 40 versions of the expOv kernel.

We used kernel PCA (kPCA) to visually check whether sequences that were considered more similar by the kernel were also similar in their drug resistance. kPCA was computed over the whole dataset. For both SVM and kPCA, results of the exponential random walk kernel were compared to our previous results.

5.2.5 LD measures

To further interpret the SVM and kPCA results, we computed the LD among protein positions. Given two multi-allelic *loci* A and B , the disequilibrium between any possible pair of alleles of these *loci* is:

$$\mathbf{D}_{ij} = p(A_i B_j) - p(A_i) p(B_j) \quad (5.14)$$

where $p(A_i)$ is the observed frequency of i -th allele in *locus* A , $p(B_j)$ is the observed frequency of j -th allele in *locus* B , and $p(A_i B_j)$ the observed frequency of the haplotype $A_i B_j$. The correlation between A_i and B_i is obtained from \mathbf{D}_{ij} , doing:

$$\mathbf{r}_{ij}^2 = \frac{\mathbf{D}_{ij}^2}{p(A_i) \cdot (1 - p(A_i)) p(B_i) \cdot (1 - p(B_i))} \quad (5.15)$$

Out LD measure was r^2 , the overall correlation between protein positions (Zhao *et al.*, 2007):

$$\mathbf{r}_{AB}^2 = \sum_{i=1}^k \sum_{j=1}^m p(A_i) p(B_j) \mathbf{r}_{ij}^2 \quad (5.16)$$

A p-value was computed using the statistic $T2$, as proposed by Zaykin *et al.* (2008):

$$T2 = \frac{N(k-1)(m-1)}{k \cdot m} \sum_{i=1}^k \sum_{j=1}^m r_{ij}^2 \sim X^2_{(k-1)/(m-1)} \quad (5.17)$$

Correlation measures were computed for protease (PI dataset) and reverse transcriptase (NRTI+NNRTI, removing sequences that appeared in both datasets). To deal with mixtures, we repeated the analysis 40 times, with different versions of the datasets randomly sampling one allele at a time. Results were averaged, considering that nonsignificant p-values were equal to $r^2 = 0$.

5.3. Results

The prediction performance for two PI and two reverse transcriptase inhibitors is shown in Figure 5.1. NMSE for the rest of drugs can be found in Suppl mat, figures S47-S53 (Annexes). Within the two kernels over graphs, the expJac achieved better results than the expOv kernel, as happened in the first study (Chapter 4). However, the inclusion of structural information via the exponential kernel did not result in an improvement of the Normalized Mean Squared Error (NMSE). Only in a single drug, ETR (Figure S53, left), a small improvement was observed. In the remaining cases, the kernels over graphs delivered equivalent or slightly worse performances than their Overlap and Jaccard counterparts over sequences.

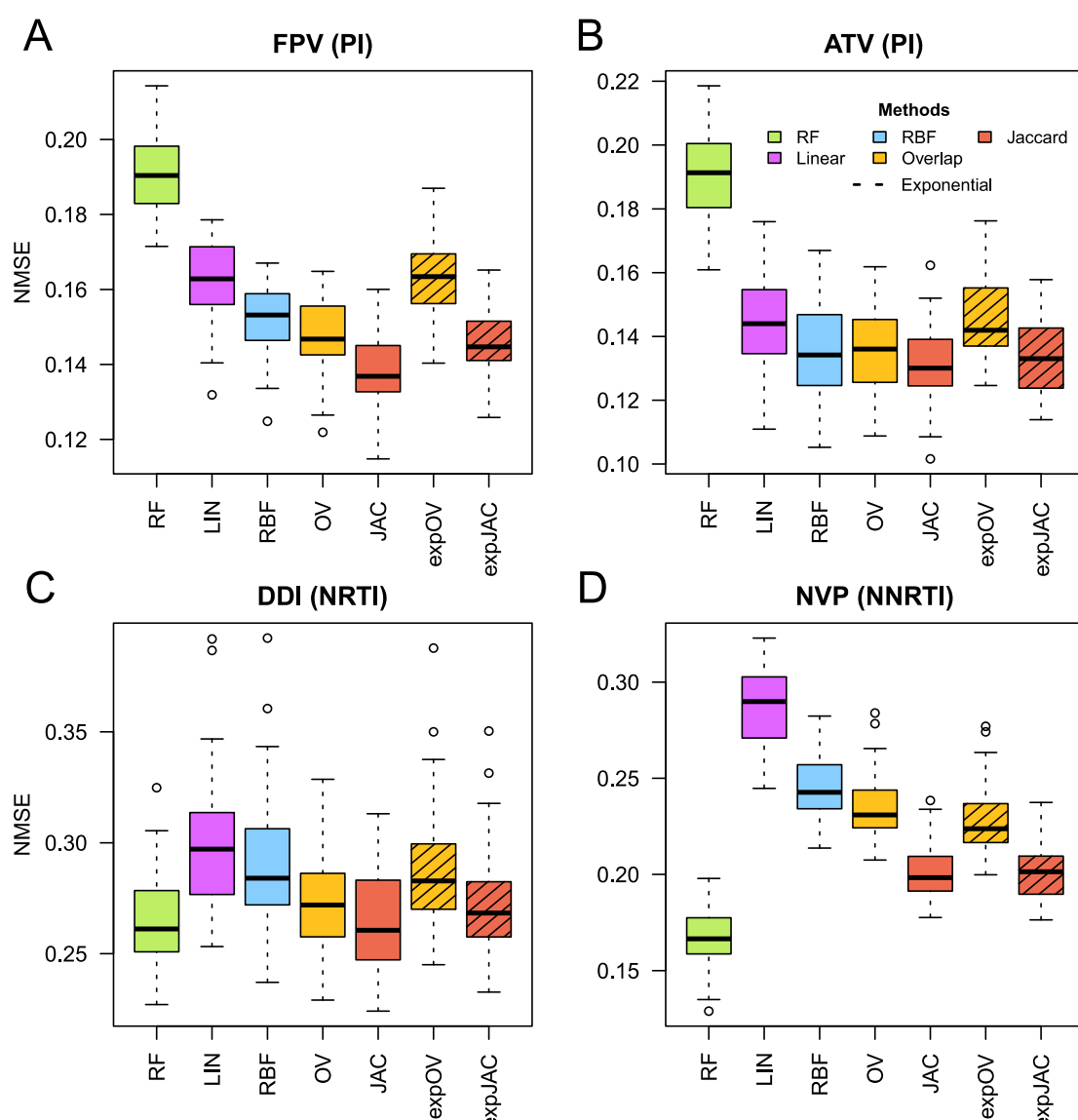
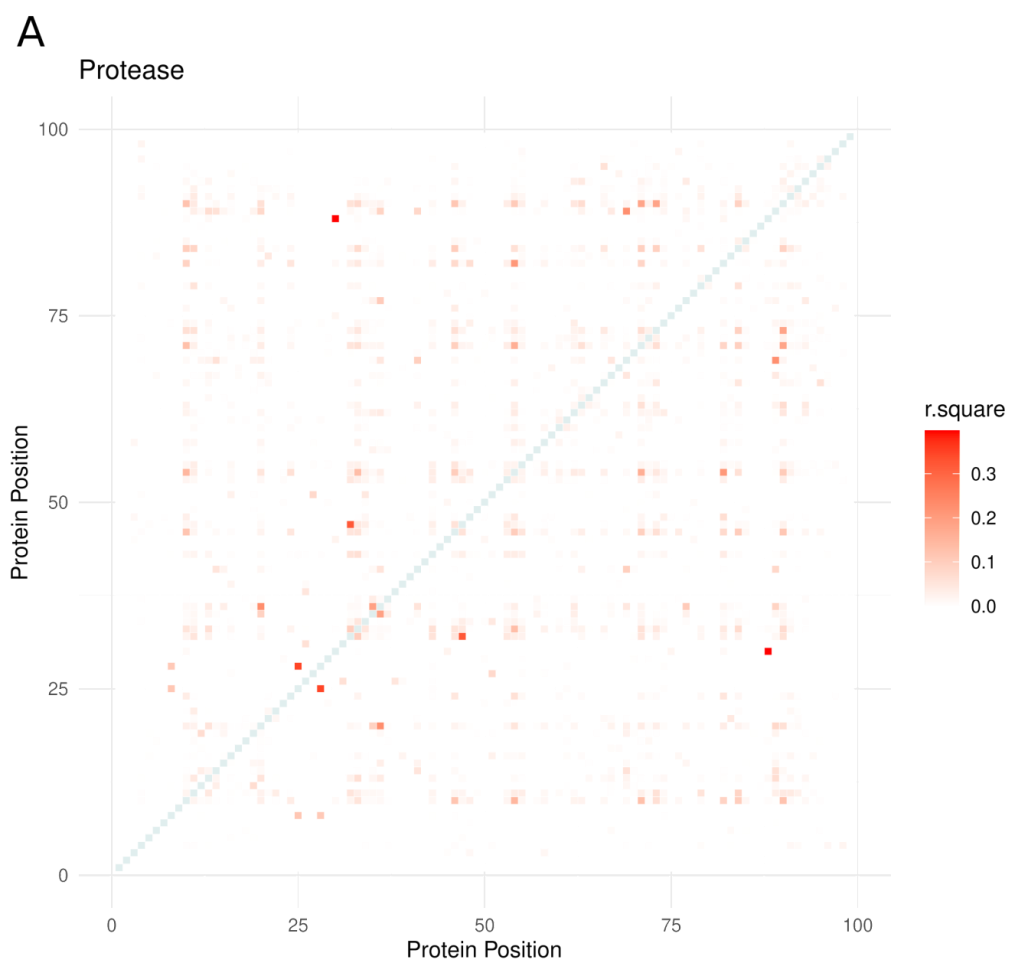


Figure 5.1. NMSE distributions for two PIs (FPV and ATV, panels A and B), an NRTI (DDI, panel C) and an NNRTI (NVP, panel D). Plain colors correspond to the unweighted kernels over sequences from Chapter 4, while the striped lines correspond to the exponential random walk kernels. RF results are also included as an additional benchmark. Note that NMSE scale varies between panels.

In Suppl mat, figures S54-S71, we show the kPCA for all drugs. Results are quite similar to the unweighted kernels over sequences of Chapter 4 (see Suppl mat, figures S28-S46): they present a spectrum-like separation from resistant to susceptible HIV isolates without clear-cut clusters, especially in PIs. Overall, separation by drug resistance is better in that drug class than in reverse transcriptase inhibitors, mirroring the results of the SVM analysis.

Results of LD r^2 measure in protease and reverse transcriptase sequence data are in Figure 5.2. Correlation values were, overall, weak. The ten positions more strongly correlated (in average) to other positions were 54, 90, 71, 46, 10, 33, 84, 20, 36 and 32. All of them are associated to drug resistance (Iyidogan & Anderson, 2014). In the case of reverse transcriptase, r^2 reached a higher maximum value than with protease (0.71 vs. 0.40), but the matrix was very sparse. The positions more strongly correlated (in average) to other positions were 41, 210, 215, 118, 67, 44, 208, 116, 219 and 151. Among them, at least seven are known to be resistance-associated positions, which is remarkable as reverse protease has few positions involved in resistance.



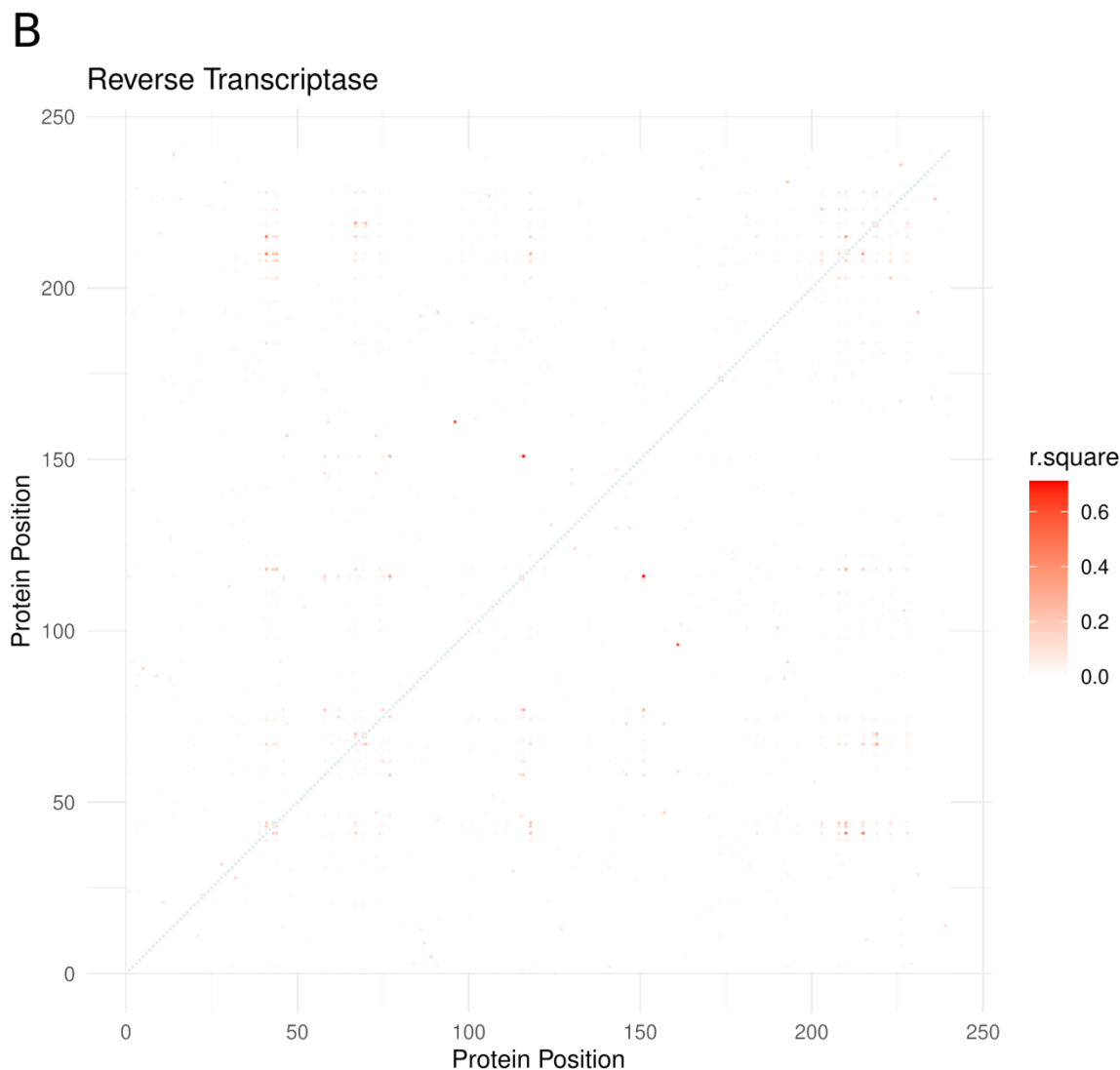


Figure 5.2. r^2 values (averaged over the 40 replicates) of protease (A) and reverse transcriptase (B).

5.4 Comparison with other works

Some previous works have also combined structural and sequential information to predict drug resistance in HIV. For instance, in Masso *et al.* (2014), resistance to an INI (EVG) was predicted training a RF and a SVM with 115 protein sequences. They first identified 1417 different integrase chains in PDB and modeled them as a convex hull of non-overlapping tetrahedra (Delaunay tessellation), so the four vertex of a tetrahedron are nearest neighbors residues in the protein. For each combination of four amino acids, a relative frequency of occurrence was calculated as the proportion of tetrahedra observed in all 1417 protein tessellations. A score for each tetrahedron was obtained by contrasting this observed frequency with the frequency expected by chance. Then, the tessellation was computed for a

partial crystal structure of wild type integrase, so each one of the 115 HIV variants was threaded onto it, re-labeling the protein position if a mutation was present. By adding up the scores of all tetrahedra that shared a particular residue, the environment score was obtained per protein position. Finally, each integrase variant was represented as a feature vector of structure-based attributes. They reported a regression performance that achieved, at best, an $r=0.78$ and a $MSE=0.20$.

In Shen *et al.* (2016), the crystal structure of a protease and a reverse transcriptase was modeled using the Delaunay triangulation. Resistance to the same eight PIs, six NRTIs and four NNRTIs evaluated in the present work was assessed. For each HIV variant, a feature vector was built by adding the distances between $C\alpha$ atoms along each arc of the Delaunay triangulation, given that the arc connected a specific pair of amino acids. The vector was 210 features long (as the canonic amino acids generate 210 unique pairs). Their best results were obtained with RF, with an R^2 that ranged between 0.772-0.995.

Instead, Khalid and Sezerman (2016) approach was based on using a large quantity of different data, including one-hot encoding of the sequences, hydrophobicity measures, evolutionary conservation, frequency occurrence count, solvent accessibility, amino acid volume information and contact energies of the interacting protein residues. As the number of features was high they performed feature selection prior to the model training. They used RF and SVM to classify HIV variants in susceptible or resistant to six PIs, four NRTIs and three NNRTIs. Best results were achieved by the SVM, with cross-validation accuracies between 95-96% (depending on the drug), and test set accuracies between 98-99.2%.

Direct comparison of our results with the three aforementioned reports is not trivial. Khalid and Sezerman (2016) posed the resistance prediction as a classification problem, with no regression results reported. In the cases of and Masso *et al.* (2014) and Shen *et al.* (2016) it is important to stress that they report cross-validation performances and not the performance of an independent test.

Another question is if adding structural information to the sequence data effectively improves the resistance prediction. In contrast with our work, this is unsolved in Masso *et al.* (2014) and Shen *et al.* (2016), as their encodings completely tangle both kinds of data. In the case of Khalid and Sezerman (2016), the feature selection procedure highlighted the effect of several high-impact protein positions, as well as structural effects like the amino acid volume or contact energies between multiple protein residues.

5.5. Discussion

The goal of the present study was expanding the two categorical kernels over sequences of Chapter 4 to consider dependence between protein positions. We embedded them into the exponential random walk kernel, and the two *ad hoc* kernels generated (expOv and expJac) were used over graphs containing structural information about the proteins (specifically, distance between residues).

Despite the potential advantages of this approach, a clear improvement on predictive ability was not observed. One of the causes may be the choice of the kernel. At least a previous work (Borgwardt *et al.*, 2005) reports the utility of random walk kernels in predicting function from proteins represented as graphs. However, in that study, proteins of different families (*i.e.* with a very low sequence and structure identity) were assessed. It is possible that other types of kernels for graphs, *e.g.* diffusion kernels (Schölkopf *et al.*, 2004) or information theoretic kernels (Bai & Hancock, 2011), give better results when applied to this particular problem.

Aside from the kernel chosen, it is quite possible that a large part of problem stems from an oversimplified representation of protein structure. A major weakness of our approach was that only one crystal structure per protein was used. Then, only the graphs' nodes varied between HIV variants, while distances on the edges remained always the same. This is a clear simplification, as it is well known that amino acid changes disturb the protein structure, at least, in the neighborhood of the mutation. For example, resistance-associated mutations in protease lead to structural rearrangements (Iyidogan & Anderson, 2014). Also, in the case of NNRTIs, mutations block the entrance or change the shape of the inhibitor binding pocket. Thus, our results may be improved by inferring the specific structure of each HIV variant. However, it should be stressed that determining the mechanistic impact of single mutations is not easy at all. Complex techniques involving molecular dynamics simulation and energy minimization may be used to this effect (Masso *et al.*, 2014).

On the other hand, LD between pairs of protein residues showed that positions with strong average associations are either drug resistance-associated positions or positions close to them. This was found also by Wang and Lee (2007) and points to functional interactions due to drug selection pressure. The number of correlation pairs in protease was higher than in reverse transcriptase, as expected by the different mutational pattern of the two enzymes (see section 4.4). In any case, with some exceptions, overall correlations were very weak. This may be misleading, as r^2 considers the frequencies of the alleles involved. Because of the high mutation rate of HIV, in most protein positions we observed high frequencies of the wild type

allele and a large number of alternative alleles in low frequency. This can cause small r^2 values (Wray, 2005).

Taking into account all the above-mentioned, it seems that there exist some degree of association (and at least part of it is related to drug selective pressure) between residues, but it has not been correctly grasped by our models. Not only more accurate measures of distance between mutated residues are needed; probably, using only distance information is not enough. As shown in Khalid and Sezerman (2016), combining data as diverse as amino acid volumes, chemical interactions, contact energies, etc. may draw a more complete picture of the enzyme structure. Fortunately, the great modularity of the kernel approach allows using different kernels to evaluate each aspect of data and, then, performing an integration. This can be achieved with Multiple Kernel Learning or by including a larger range of kernels for nodes and edges in the exponential random walk kernel. That way, there may be room for improving the results presented in this chapter.

CHAPTER 6

***kernInt*: A KERNEL FRAMEWORK FOR INTEGRATING SUPERVISED AND UNSUPERVISED ANALYSES IN SPATIO-TEMPORAL METAGENOMIC DATASETS**

Elies Ramon¹, Lluís A. Belanche², Francesc Molist³, Raquel Quintanilla⁴, Miguel Perez-Enciso^{1,5}, Yulixaxis Ramayo-Caldas⁴

¹ Plant and Animal Genomics, Statistical and Population Genomics Group, CSIC-IRTA-UAB-UB Consortium, Centre for Research in Agricultural Genomics (CRAG), 08193 Bellaterra, Spain.

² Computer Science Department, Technical University of Catalonia, Carrer de Jordi Girona 1-3, 08034 Barcelona, Spain.

³ Schothorst Feed Research B.V., Lelystad, The Netherlands.

⁴ Animal Breeding and Genetics Program, IRTA, 08140 Caldes de Montbui, Spain

⁵ Catalan Institution for Research and Advanced Studies (ICREA), Passeig de Lluís Companys 23, 08010 Barcelona, Spain.

(submitted)

Abstract

The advent of next-generation sequencing technologies allowed relative quantification of microbiome communities and their spatial and temporal variation. In recent years, supervised analysis (*i.e.* prediction of a phenotype of interest) from taxonomic abundances has become increasingly common in the microbiome field. However, a gap exists between supervised and classical unsupervised analyses, based on computing ecological dissimilarities for visualization or clustering. Despite this, both approaches face common challenges, as the compositional nature of next generation sequencing data or the integration of spatial and temporal factors. Here we propose a kernel framework to unify unsupervised and supervised microbiome analyses, including the retrieval of microbial signatures (taxa importances). We define two compositional kernels (Aitchison-RBF and compositional linear) and discuss how to transform noncompositional measures into kernels. Spatial data is integrated with Multiple Kernel Learning, while longitudinal data is evaluated by specific kernels. We illustrate our framework through a single point soil dataset, a human dataset with a spatial component, and a previously unpublished longitudinal dataset concerning pig production. The proposed framework and the case studies data are freely available in the *kernInt* package at <https://github.com/elies-ramon/kernInt>.

Keywords: microbiome, metagenomics, kernel, supervised, unsupervised, spatio-temporal, SVM, kPCA.

6.1 Introduction

The microbiome is defined as the ensemble of microorganisms and their genomes in a given environment. Microorganisms are present in ecological niches as diverse as soil, oceans, freshwater, plants and animals, but a large fraction of these taxa cannot be cultivated with culture-dependent methods. The advent of the next generation sequencing (NGS) revolutionized this field by allowing the massive sequencing and quantification of microbial habitats.

Proper analysis of microbiome data is challenging for a variety of reasons. Abundance data obtained with NGS is multivariate, sparse and compositional in nature (Gloor *et al.*, 2017). Also, microbial communities are very dynamic biological systems, thus justifying spatial or time-course studies (Bodein *et al.*, 2019). The first approach on the field used statistical tools from standard ecological studies (Gloor *et al.*, 2017). For example, one of the first steps in

nearly all microbiome studies consists in computing alpha and beta-diversities. Beta diversity measures, like Bray-Curtis or Unifrac, quantify the difference in diversity between samples from different habitats. They are used for clustering analysis or, more commonly, for visualization techniques like PCoA (Principal Coordinates Analysis) or MDS (Multidimensional Scaling). However, this approach has been challenged, as the abundance data obtained by NGS has a particular nature. The total number of reads delivered is constrained by the sequencing capacity of the instrument; thus, only relative frequencies are informative. Data consisting in proportions with an uninformative sum is consequently compositional and has a specific mathematical treatment (Gloor *et al.*, 2017). In the case of metagenomics, extensive research is being done to translate current statistical techniques to this paradigm (Rivera-Pinto *et al.*, 2018; Gloor *et al.*, 2017; Silverman *et al.*, 2017). One example is the proposal of using the compositional Aitchison distance instead of the classic beta-diversity measures.

In machine learning, the aforementioned visualization, clustering and ordination techniques belong to the so-called unsupervised learning. Supervised learning, which is focused on prediction, is not so widespread in microbiome analysis yet, but the number of studies using this kind of approach is rapidly growing in the last years (Zhou & Gallins, 2019). Supervised methods include Random Forests (RF), Artificial Neural Networks (ANN), Support Vector Machines (SVM), k-Nearest Neighbors and ridge regression (Namkung, 2020; Qu *et al.*, 2019; Zhou & Gallins, 2019). Among the aforementioned, RF are popular in the microbiome context and tend to outperform other methods (Namkung, 2020; Zhou & Gallins, 2019). ANN have shown excellent performance in some cases but are susceptible to overfitting, especially if sample size is greatly exceeded by the number of taxa, as is often the case in metagenomics and metataxonomics. A desirable feature for supervised methods is the identification of microbial signatures (*i.e.* taxa that are predictive of a certain phenotype) that may enable biological interpretation of the results. RF are endowed with variable importance measures that can be used to this effect, while there is not such straightforward heuristic for ANN – although several possible strategies exist (Ibrahim, 2013)–. Another supervised method, *selbal* (Rivera-Pinto *et al.*, 2018), is focused on the identification of microbial signatures based on balances (a log contrast of geometric means of data from two groups of taxa), and has the particularity of being purely compositional.

As microbial communities are highly dynamic systems, it is important to address their spatial and/or temporal variation (Berg *et al.*, 2020). In spatial-structured studies, repeated samples of different sites (*e.g.* body sites, depth layers) are obtained from the same individuals or entities, thus raising the question of how to integrate them. A more general challenge is the integration coming from datasets of different sources (*e.g.* ‘omics’), which may have different

data types. Several statistical methods have been proposed to solve this question in the microbiome field. Some examples are Link-HD (Zingaretti *et al.*, 2020), mixKernel (Mariette & Villa-Vialaneix, 2018) and MOFA (Argelaguet *et al.*, 2018), all focused in the unsupervised learning setting. In most supervised methods, this integration is usually performed at the input data level (*early integration*), for example by concatenating the datasets; or after the model is built (*late integration*), combining their scores as in ensemble methods. However, early integration may be not possible if data nature differs across sources (Schölkopf *et al.*, 2004). The case of the longitudinal studies (which follow the evolution over time of microbial communities) is more complex. Typically, longitudinal data is modeled by fitting a function (*e.g.* polynomial interpolation, splines) to the data points over time. To date, there exist few analytical tools for this kind of data in the microbiome field. Two examples can be found at Bodein *et al.* (2019) and Coenen *et al.* (2020), but they are restricted to unsupervised analysis. Difficulties like the compositionality of data or how to accommodate the spatial and temporal dimensions affect supervised and unsupervised methods alike. However, there is a gap between the most widely used supervised methods and the unsupervised analyses typical of the microbiome field. Carrying out all analyses in a common mathematical framework would provide a new, holistic view to microbiome studies. With all this in mind we propose a generic and flexible kernel framework as a way to unify supervised and unsupervised microbiome analyses, while paying special attention to data compositionality and spatial and temporal integration. Kernel methods are a family within machine learning methods that share the use of kernel functions or, simply, kernels. Some of these methods have been already applied to some specific problems or areas within microbiome analysis (Zhou & Gallins, 2019; Mariette & Villa-Vialaneix, 2018; Zhan *et al.*, 2017) but their potential has not been fully exploited. In this work, we propose two new compositional kernels and discuss how to translate noncompositional, but nonetheless widespread, beta-diversity matrices to the kernel framework. We perform supervised and unsupervised analyses from the same kernel matrix, and show how to extract microbial signatures. Spatial and longitudinal data are also treated with specific kernel tools. This kernel framework is illustrated with three case studies: a single point soil metagenomic dataset, a human dataset with a spatial component, and a previously unpublished longitudinal dataset concerning pig gut microbiota. An R package implementing the proposed methods, along with the case studies data, is freely available at <https://github.com/elies-ramon/kernInt>.

6.2 Material and methods

6.2.1 Kernels for microbiome data

A real symmetric two-place function is a kernel iff, for every finite set of objects x_1, \dots, x_N , it generates a symmetric matrix that is positive semi-definite (Schölkopf *et al.*, 2004, Shawe-Taylor & Cristianini; 2004). Probably the most widely known and used kernel functions are the linear and RBF (Radial Basis Function) kernels, both defined for real vectors.

Intuitively, a kernel can be understood as a measure of the similarity between x_i and x_j . As objects x_1, \dots, x_N are never represented explicitly, kernels can be designed for nonstandard data types if a notion of what is considered “similar” in that given context exists (Schölkopf *et al.*, 2004). Also, as similarity measures, kernels are related to the beta-diversity dissimilarities widely used in microbiome analyses. We now present two compositional and two noncompositional kernels. In this work, we are restricted to measures that can be obtained from taxonomic abundance tables only, but further insights can be found in the Discussion.

Compositional kernels

Here we define two kernels analogous to the linear and RBF kernels, but specific for compositional data. We introduce the Aitchison-RBF kernel as:

$$cRBF(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \sum_{k=1}^D \left(\log\left(\frac{x_{ik}}{\text{geo}(\mathbf{x}_i)}\right) - \log\left(\frac{x_{jk}}{\text{geo}(\mathbf{x}_j)}\right)\right)^2\right) \quad (6.1)$$

where \mathbf{x}_i and \mathbf{x}_j represent the taxonomic abundances in two different individuals, D is the number of different taxa, $\text{geo}(\cdot)$ is the geometric mean, and $\gamma > 0$ is a hyperparameter that has to be tuned. This nonlinear kernel derives from the Aitchison distance, which is Euclidean and therefore (6.1) is a valid kernel. The logarithm term can be identified as the compositional clr-transformation (Gloor *et al.*, 2017) over the original data.

Analogously, we define the compositional linear kernel as:

$$cLin(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^D \log\left(\frac{x_{ik}}{\text{geo}(\mathbf{x}_i)}\right) \log\left(\frac{x_{jk}}{\text{geo}(\mathbf{x}_j)}\right) \quad (6.2)$$

Although cRBF is related to Aitchison distance and has the advantage of nonlinearity, cLin is easier to interpret and allows the retrieval of the microbial signatures.

Noncompositional kernels

The most widely beta-diversity measures are Bray-Curtis, Unifrac and Jensen-Shannon (Gloor *et al.*, 2017). Bray-Curtis and Jensen-Shannon are computed from taxonomic tables, while

Unifrac additionally needs a phylogenetic tree. The Jensen-Shannon is metric and is paired with a kernel that is already described in Bai and Hancock (2011) as the Jensen-Shannon Kernel (JSK):

$$JSK(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{1}{2} \left[\sum_{k=1}^D x_{ik} \ln \left(\frac{2x_{ik}}{x_{ik} + x_{jk}} \right) + \sum_{k=1}^D x_{jk} \ln \left(\frac{2x_{jk}}{x_{ik} + x_{jk}} \right) \right] \quad (6.3)$$

provided that \mathbf{x}_i and \mathbf{x}_j contain relative frequencies. The Bray-Curtis dissimilarity is semimetric, and so we propose using a similar distance (Gardener, 2014), Jaccard, instead. The Jaccard distance is paired with a well-known kernel (Bouchard *et al.*, 2013) and has a variant suitable for quantitative data. The quantitative Jaccard (also known as Ružička) kernel is defined in Gardener (2014) as:

$$qJac(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^D \frac{\min(x_{ik}, x_{jk})}{\max(x_{ik}, x_{jk})} \quad (6.4)$$

6.2.2 Kernel methods and framework

Kernel methods share the use of symmetric and positive semi-definite matrices (*i.e.* kernel matrices), and not the original data, as input. This places all different analyses in a common mathematical ground, which we refer as the kernel framework. We will use SVM, a classical method that can perform both regression and classification, for phenotype prediction. For the unsupervised analyses we will use kPCA (kernel Principal Components Analysis), a kernelized version of the standard algorithm.

6.2.3 Spatial data

The kernel framework is particularly well suited for the integration of spatial or heterogeneous data types (Mariette & Villa-Vialaneix, 2018; Schölkopf *et al.*, 2004). This is because the integration can be done directly at the kernel matrices level. Let $\mathbf{K}_1, \dots, \mathbf{K}_M$ be the kernel matrices computed from M different sources of data coming from the same individuals. Then, we can obtain a consensus kernel matrix \mathbf{K}^* :

$$\mathbf{K}^* = \sum_{z=1}^M \beta_z \mathbf{K}_z \quad (6.5)$$

with the restriction $\beta_z \geq 0$. The optimal β_z values can be selected through an optimization process, which is known as MKL (Multiple Kernel Learning) (Schölkopf *et al.*, 2004). In unsupervised scenarios, a consensus matrix \mathbf{K}^* can be obtained by choosing the β values that maximize average similarity of \mathbf{K}^* with all \mathbf{K}_z matrices (Mariette & Villa-Vialaneix, 2018).

6.2.4 Temporal data

A time series is an ordered set of repeated samples indexed by time, in the form $\{x_i, t_i\}$. The natural way to summarize this type of data is through a function, which can be obtained using polynomial interpolation or splines. When data contains the time series of several individuals it is commonly referred as longitudinal data. Kernels specific for this data compare the evolution over time among individuals, so that information is used afterwards for phenotype prediction or unsupervised tasks.

The functional RBF kernel (Chen *et al.*, 2013) translates the RBF kernel to accept functions as input. Let $f(t)$ and $g(t)$ be continuous functions, so that they represent the time series of two different individuals between the time interval $[t_a, t_b]$. Then, the kernel definition is:

$$fRBF(f, g) = \exp(-\gamma \int_{t_a}^{t_b} |f(t) - g(t)|^2 dt) \quad (6.6)$$

In an analogous way, the functional linear kernel is defined as:

$$fLin(f, g) = \int_{t_a}^{t_b} f(t) g(t) dt \quad (6.7)$$

These kernels allow irregular sampling intervals and missing time points, but suffer of the cost of computing numerically the integral (*e.g.* if an algebraic solution is not possible). The calculation of fRBF and fLin can be simplified if the modeling of the time series as continuous functions is skipped. Let the original finite sets of points indexed by time for both individuals be denoted directly as $f(t)$ and $g(t)$. fRBF and fLin for discrete functions are now defined as:

$$fRBF(f, g) = \exp(-\gamma \sum_{i=1}^T (f(t_i) - g(t_i))^2) \quad (6.8)$$

$$fLin(f, g) = \Delta t \sum_{i=1}^T f(t_i) g(t_i) \quad (6.9)$$

where T is the total number of time points and Δt the time increment. This can be a sound approach in cases with few data points, when the modeling is less reliable. However, contrarily to (6.6) and (6.7), these expressions cannot deal with irregular sampling times or missing data.

In multivariate scenarios, for instance microbiome data, many variables are simultaneously sampled over time. Let f_k and g_k model taxon k in two individuals, being D the total number of taxa. The aforementioned kernels can be combined as in:

$$fLin'(f, g) = \sum_{k=1}^D fLin(f_k, g_k) \quad (6.10)$$

$$fRBF'(f, g) = \prod_{k=1}^D fRBF(f_k, g_k) \quad (6.11)$$

It should be noted that the kernel approach allows the simultaneous integration of temporal and spatial (via MKL) data.

6.2.5 Microbial signature

In a broad sense, the *microbial signature* is the collection of taxa associated with a trait of interest and that has a high predictive value in the context of a given model (Rivera-Pinto *et al.*, 2018). It can be retrieved from a linear SVM using the orientation of the separating hyperplane (Guyon *et al.*, 2002): if the plane is orthogonal to a particular feature dimension, then that feature is maximally informative. This method takes into account the correlation between taxa. As cLin is a translation of the linear kernel for compositional data, using (6.2) we can retrieve the microbial signatures, which should be understood as the taxa importances after the clr-transformation. The same occurs when assessing the variable influence on the principal components in kPCA. A general permutation technique is proposed in Mariette and Villa-Vialaneix (2018), but using cLin permits obtaining the taxa influence in the same straightforward way than standard PCA.

The linearity also permits extending the microbial signature retrieval when using SVM to the longitudinal and spatial cases. When performing MKL, as long as the cLin kernel is strictly applied to all sampled sites, the global importance of a given taxon among all sites can be computed as the weighted sum (using the optimal β coefficients) of its partial importance in each site. In the longitudinal case, the global importance of each taxon k can be obtained from (6.9) by addition of the partial importances over all T time points.

6.2.6 Case studies and data preprocessing

We illustrate our framework with three case studies: a single point dataset, a dataset with a spatial component, and a longitudinal dataset. The latter is previously unpublished while the rest of the data is public.

Soil dataset

Bacterial composition of soil varies significantly at a biogeographical scale, and is related to chemical and environmental factors. Here we reanalyzed a single point dataset by Lauber *et al.* (2009), who used 16S small-subunit ribosomal (16S rRNA) gene pyrosequencing to profile the bacterial communities of different soils across North and South America. Authors

reported that soil pH was significantly correlated with beta-diversity distances between samples. They also found correlation with alpha diversity, which was highest in soils with near-neutral pHs. To perform our analysis, we retrieved the taxonomic abundances as well as the associated metadata from Qiita <https://qiita.ucsd.edu/> (ID: 103). The number of OTUs (Operational Taxonomic Unit) was 7,396, while the number of soil samples was 89. As a part of the preprocessing, we excluded sample number 89, with only 1 read, which was also not included in the original paper.

Smokers dataset

Charlson *et al.* (2010) analyzed the impact of cigarette smoking on the global airway microbial population. Bacterial communities were profiled using 454 pyrosequencing of the 16S rRNA gene in four airway sites: the left and right sides of nasopharynx and oropharynx. Authors reported that composition was primarily determined by airway site, with individuals exhibiting minimal lateral or temporal variation. They used RF to predict the smoking status from the taxonomic abundances. We retrieved the dataset (metadata and taxonomic abundances) from Qiita (ID: 524) to perform our analysis. Of the original 70 individuals, we discarded those that reported airway illness or antibiotic usage in the three months prior to sampling. Thus, we analyzed the same 62 individuals of the original work. Number of different OTUs was 2,817.

Pig dataset

Here we present an original dataset, which evaluates the relationship of pre-weaning diarrhea with the early gut microbiota colonization in piglets. The experiment was conducted at Schothorst Feed Research B.V. facilities (Netherlands) where management, environmental and housing factors were controlled for all animals throughout the whole study. Gut microbiota was profiled in 153 piglets during their first week of life. Between days 8 and 21 (weaning day), 79 out of the 153 piglets had diarrhea and were treated with antibiotics. Piglets sampling (swabs) was done within 5 minutes after farrowing (day 0) and at days 3 and 7 post-farrowing. DNA was extracted from faecal samples and profiled using Illumina sequencing of 16S rRNA gene in each of the three time points. The cleaned sequences were processed into Amplicon Sequence Variants (ASVs). Further details are described in Supplementary methods S0. Analyses were carried out at the ASV (3,577 ASVs were obtained) and at the Genera taxonomic levels.

6.2.7 Experimental set-up

Analyses across the three datasets included a comparison with the original reports (for Soil and Smokers datasets), as well as contrast with results from RF. The cLin and cRBF kernels

were applied directly to the raw counts, as they handle data in an inherently compositional manner. Following Quinn *et al.* (2018), a number under the detection limit was added to all dataset to handle zeroes. An alternative normalization of data, the cumulative sum scaling (CSS) (Paulson *et al.*, 2013) was performed prior to applying the noncompositional Jensen-Shannon and Jaccard kernels. That way the compositional and noncompositional kernels could be compared. In the rest of cases (RF and longitudinal functions) we used the compositional clr-transformation over data. RF were obtained using the R package ‘randomForest’ (Liaw & Wiener, 2002), and the kernel methods with ‘kernInt’.

Unsupervised analyses (*e.g.* kPCA) were computed over the whole datasets. In the Smokers dataset, we additionally computed the similarity among kernel matrices of different body sites with the mixKernel package (Mariette & Villa-Vialaneix, 2018). For the supervised analyses, each dataset was split at random into the training set (80% of data) and test set (20%). Optimal hyperparameters and β coefficients for MKL were obtained by 5×5 cross-validation on the training set. Then, the final model was built using the whole training set. We repeated the whole process 40 times, each time with different 80/20 randomly split training/test partitions, to obtain an error distribution. Performance over the test set was computed using Normalized Mean Squared Error (NMSE) for regression and Accuracy for classification.

For the Pig dataset additional considerations had to be taken into account. To make sure that the training and test sets were completely independent, piglets from the same litter (full sibs) were always placed either in one or other set. Performance of fLin and fRBF was contrasted to those of RF and their analogous nonlongitudinal kernels (cLin and cRBF) using all available days at once. For the nonlongitudinal methods, 80% of the piglets were used to train the model, using their three time points data in separate rows, with time included as an additional variable. The remaining piglets were reserved to test the model, but using only one of their time points (either day 0, 3, or 7) chosen at random and discarding the rest. This way, both longitudinal and non-longitudinal approaches had the same test set size. Longitudinal kernels fLin and fRBF were computed using (6.9) and (6.8), as only three time points were available and we preferred not to interpolate the days in-between. Also, using the expression for discrete functions we could obtain the microbial signatures. The information of all taxa was combined as in (6.10) and (6.11), and the training/test partitions were carried out as in the normal case. In a second step, the dataset was decomposed by sampling times and the analysis was carried out for days 0, 3 and 7 separately using RF, cLin and cRBF in the usual way.

Microbial signatures from SVM were obtained from the hyperplane normal vector \mathbf{w} , so the importance of taxon k was computed as $(w_k)^2$ (Guyon *et al.*, 2002). In RF, we used the mean decrease in node impurity (for regression tasks) and mean decrease in Gini index (for classification). Both RF and SVM give absolute values of taxa importance, so they were converted to relative values.

6.3 Results

6.3.1 Soil data

The cLin kPCA over the bacterial abundances is shown in Figure 6.1A. The remaining kPCAs, which gave a similar profile, can be found at Suppl mat, Figure S72. Soil samples are clearly separated by their pH, in agreement with the original results. The U-shaped projection is typical of banded metagenomic data, *i.e.* data structured by a gradual transition with few overlapping OTUs at the endpoints (Suppl mat, Figure S73). The peak diversity in near-neutral soils in contrast with extreme pHs may also have some effect (Suppl mat, Figure S74).

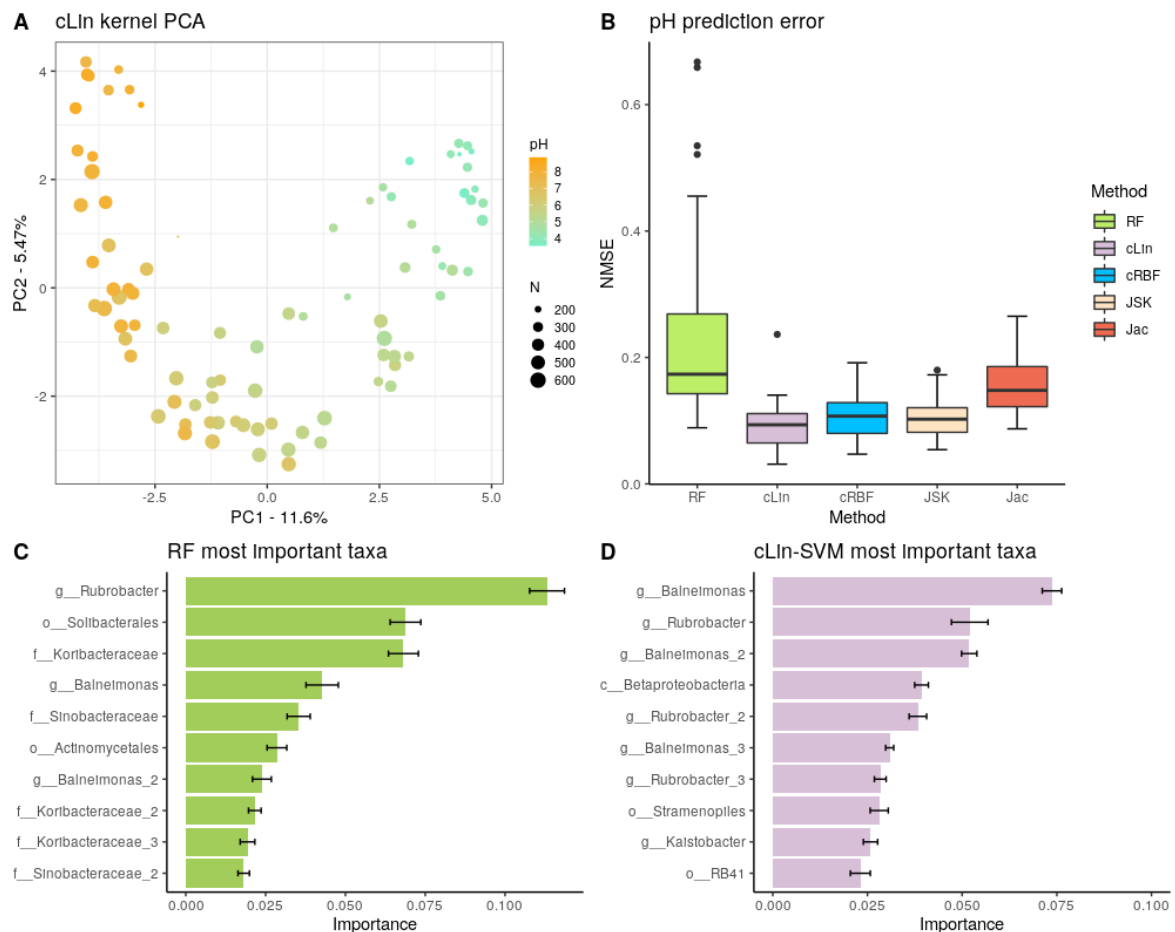


Figure 6.1. A) Compositional linear kPCA over the 88 soils. Color represents pH, while point size stands for the number of different observed taxa. B) pH prediction error distribution over the 40 replicates. C)

and D) Top relevant taxa for pH prediction according to RF and cLin-SVM. Standard error across the 40 replicates is marked with error bars.

In addition, we used SVM with the four kernels described above to predict the pH of each soil site from the bacterial abundances. This was not done in the original work and so we used RF, a nonkernel, alternative method, as benchmark. Results are shown in Figure 6.1B. The best compositional kernel was cLin, having a median error of ≈ 0.09 ; and the best noncompositional one was JSK, with a median error of ≈ 0.10 . In comparison, RF had a higher median error, almost the double of cLin, around 0.17.

To go further in the interpretation of the results, we analyzed the microbial signatures retrieved from RF and cLin-SVM. The distribution of the importances was highly skewed. For subsequent analyses we kept only 5% of the taxa, which accounted for around the 90% (RF) and 95% (SVM) of total importance, with the two methods having 42% of OTUs in common. Top ten relevant taxa are shown in Figure 6.1C (RF) and D (SVM). In agreement with the kPCA results, prediction is primarily driven by few OTUs of extreme pH ecosystems (*e.g.* genera *Rubrobacter* and *Balneimonas* on the basic side, orders *Solibacterales* and *RB41* on the acid side).

6.3.2 Smokers data

We predicted smoking status from the taxonomic abundances. At first models were built using the four sites separately, as in the original study. Authors used RF and reported a median accuracy of 64% on the right and 65% on the left oropharynx (*i.e.* throat), and 71% on the right and 68% on the left nasopharynx (Suppl mat, Figure S75). In our case, the worst kernel was cLin (Suppl mat, Figure S76), which nonetheless gave similar accuracies to those reported in the original paper. The best kernel was the Jaccard kernel (Figure 6.2A), which improved substantially the RF accuracies, especially in the throat (median accuracies: 79% right side and 75% left side). Then, we combined the spatial-structured samples of the same individuals to test if accuracy increased when using an integrative approach. Using MKL, we combined kernel matrices by airway (nasopharynx on one hand and oropharynx on the other) and, finally, we integrated all sites. This decreased the error substantially and delivered the best classification result, with a median accuracy of 92%. Also, we compared this result with performing an early integration approach, concatenating the 4 datasets of each individual previously to compute the kernel matrix. This approach gave a median accuracy of 83%. The results for the rest of kernels can be found in Figure S76. In all cases, integration of the four

datasets using our MKL proposal increased the accuracy, and doing so at the kernel matrix levels was better or equivalent to concatenating the four datasets.

We show the top ten most important taxa across the four sampling sites in Figure 6.2B. The importance distribution is not as skewed as in the Soil dataset: here the top 5% taxa accounted for the 62% of overall importance. *Neisseria* sp. large impact in discriminating smokers from nonsmokers was already reported in the original work, especially in oropharynx models. The rest of highlighted taxa in Figure 6.2B were also noted to have a role, either in models from nasopharynx alone or from both airways sites (Charlson *et al.*, 2010). This mostly agrees with our results when the sampling sites are analyzed separately (Suppl mat, Figure S77).

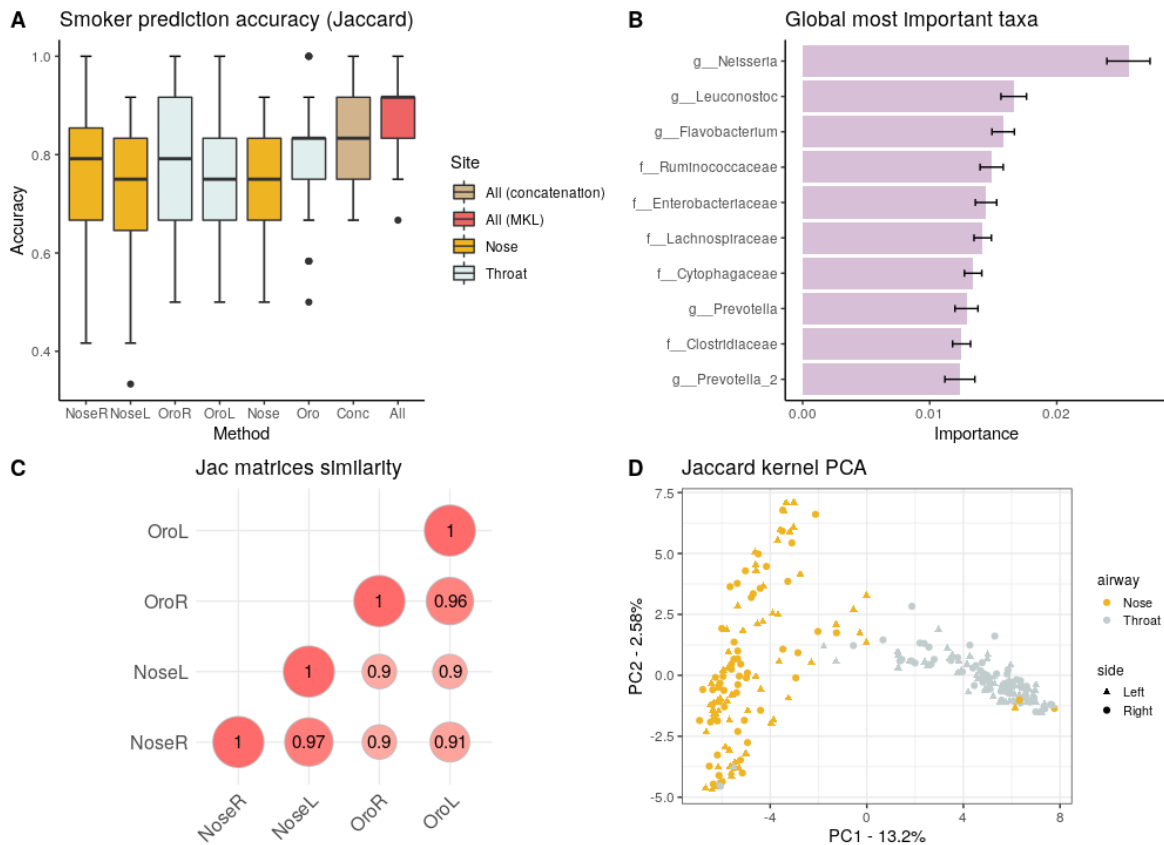


Figure 6.2. Analysis using the Jaccard kernel. A) Nonsmoker/smoker accuracies from taxonomic data: NoseL, NoseR, OroL and OroR models are obtained from single datasets data, Conc from the concatenation of the datasets, and Oro, Nose and All models from MKL. B) Top ten of the global cLin-SVM importances across all body sites. C) Similarity across the kernel matrices of the four sites (Nasopharynx Right and Left and Oropharynx Right and Left). D) Jaccard kPCA of the taxonomic abundances. Color code represent airway site, whereas shape indicates the laterality of the samples.

Following the original work, differences in bacterial communities among the body sites were also analyzed. We present results for the Jaccard kernel in Figure 6.2C and D, while the other kernels are in Figure S78. Panel C shows the similarity across kernel matrices derived from left

and right nasopharynx and oropharynx. The highest similarity was achieved within matrices of the same airway site but different laterality. As in the original paper (Figure S75), using a kPCA (Figure 6.2D) we could discriminate between nasopharynx and oropharynx sites (first PC) but not between left and right.

6.3.3 Pig data

Evolution of gut microbiota from 153 healthy piglets over their first week of life was used to predict the occurrence of pre-weaning diarrhea. In Figure 6.3A we compared the performance of the longitudinal kernels (fLin and fRBF) versus their analogous nonlongitudinal kernels (cLin and cRBF) plus RF when using all available days at once.

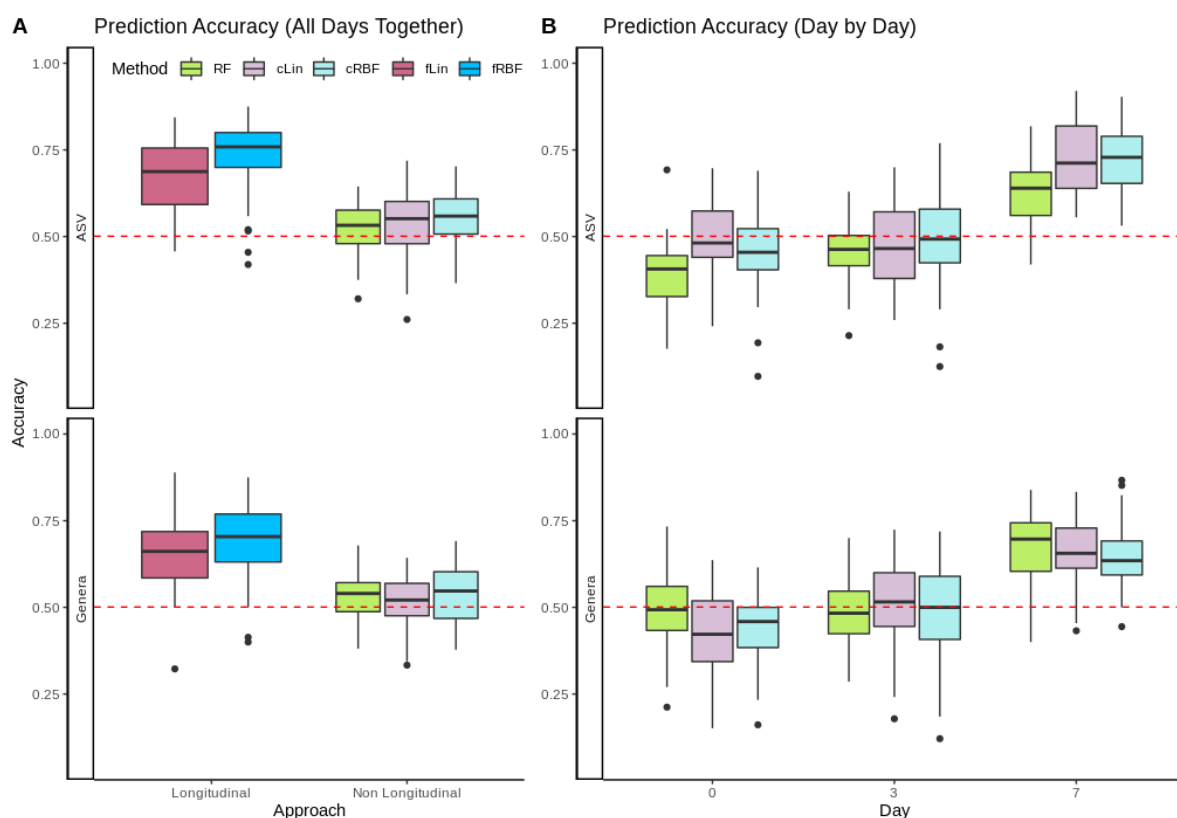


Figure 6.3 A) Accuracy for RF, non-longitudinal kernels (cRBF, cLin) and longitudinal kernels (fLin, fRBF) in the prediction of neonatal diarrhea from all available days. B) Accuracy for RF and non-longitudinal cLin and cRBF kernels from metagenomic data of days 0, 3 and 7 post-birth separately. In both panels, the red dashed line marks the accuracy of the random model.

It can be observed that the longitudinal approach clearly outperformed the nonlongitudinal approach at both Genera and ASVs levels. fRBF had a better performance than fLin, and worked best at the ASV level (with a median accuracy around 76%) than in Genera data (median accuracy around 70%). Although aggregating taxa to the genus level is a relatively

common practice –see *e.g.* Rivera-Pinto *et al.* (2018)–, in our case using a coarser taxonomic resolution decreased the accuracy. Within the non-longitudinal approach, we obtained similar accuracies using RF and kernels, and both were close to the median accuracy of the random model (50.1%). To further understand the results, the analysis was carried out in days 0, 3 and 7 separately using RF, cLin and cRBF kernels. Figure 6.3B reveals that all models from days 0 and 3 had no predictive power. Accuracy increased dramatically after day 7 to a maximum of 73% for cRBF (ASV level), only slightly worse than its analogous longitudinal kernel fRBF.

In a second step, we discarded all models without predictive power and analyzed the kPCA and microbial signatures. Figure 6.4A and B show the fLin and cLin (day 7) kPCA, while fRBF and cRBF are in Suppl Mat, Figure S79.

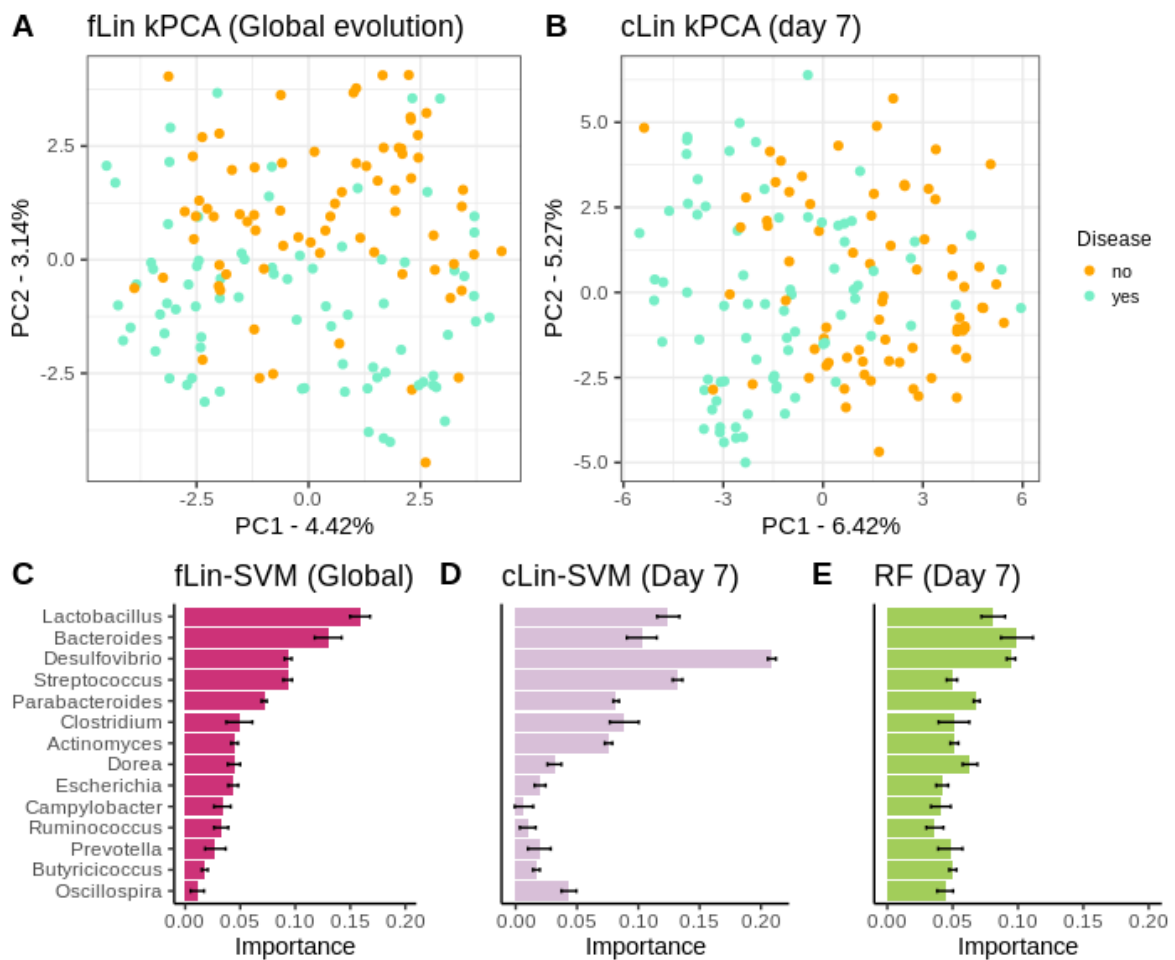


Figure 6.4. Above: kPCA of fLin in panel A and of cLin (day 7) in panel B. Below: microbial signatures at the Genera level. Global importance for the first week is in panel C. Importances for the day 7 according to the cLin kernel and RF are in panels D and E. Standard error across the 40 replicates is marked with error bars.

In all cases, a partial separation between healthy and sick piglets, with a large area of overlap, is observed. Genera relevance on prediction of pre-weaning diarrhea is shown in panels C, D and E. We discuss the microbial signature at the Genera level, as around 2/3 of the ASVs lack species assignment (Suppl mat, Figure S80). According to fLin, beneficial genera like *Lactobacillus* and *Bacteroides* had the higher overall importance during the first week. In day 7, it was striking the great importance given by cLin to genus *Desulfovibrio*, and secondarily to *Streptococcus*. RF also highlighted the butyrate-producing genus *Dorea*. Distribution of the microbial signature at the ASV level was skewed, but again much less than in the Soil dataset case. The top 5% ASVs accounted between 46% and 58% of the total importance, with an overlap between RF and cLin in day 7 of 2/3 of the ASVs.

6.4 Discussion

The kernel framework allows performing a great diversity of analyses in a common ground. However, within the microbiome field, previous application of kernel methods has been mostly restricted to specific areas. Zhan *et al.* (2017) proposed a kernel-based semi-parametric regression method for testing the association of the human microbiota communities with multiple phenotypes. In turn, Mariette and Vila-Vialaneix (2018) combined metagenomic data and environmental measures of the TARA ocean expedition using unsupervised MKL integration. In some reports that compare the performance of different supervised methods in microbiome data, SVM often appear along RF or ANN (Namkung, 2020; Qu *et al.*, 2019; Zhou & Gallins, 2019). These methods were used in an isolated way, without exploiting the kernel framework ability to integrate a great range of analyses while giving a unitary view. Another advantage of this framework is that it can handle virtually any data type. However, to our best knowledge, it has not been previously applied to longitudinal microbiome studies. Finally, in previous works there was a lack of kernels that took into account the compositional nature of metagenomic datasets.

Throughout this work, we summarized the microbiome analyses in three branches: unsupervised learning (represented by kPCA), supervised learning (SVM) and identification of phenotype-associated microbial signatures. The Soil case study clearly illustrated how all three types are intertwined and complementary. In agreement with the original publication, both SVM and kPCA results showed that taxonomic abundances and pH are strongly related. This granted a quite low prediction error (up to a median NMSE of 0.09) but, by itself, does not explain the underlying mechanism connecting microbial abundance and pH. Microbial

signature revealed that the SVM learning is driven by few taxa of opposite pH ecosystems. For instance, *RB41* belong to the phylum *Acidobacteria*. The *Rubrobacter* genus contains well known extremophiles and, like the *Balneimonas* (renamed *Microvirga*) genus, has preference for clearly alkaline soils (Chen *et al.*, 2018; Dahal & Kim, 2017). Furthermore, the arch in the kPCA projection indicated that communities from acid and basic habitats did not overlap (Morton *et al.*, 2017). Taken together, these complementary views point that soil microbial structure is shaped by a gradual niche differentiation strongly modulated by the pH. This agrees with previous findings on this dataset (Morton *et al.*, 2017; Lauber *et al.*, 2009) but appears in a more concise and unified way using the kernel framework.

In comparison to other methods, the kernel framework did not only allow a holistic view of data, but also gave good results in each learning area. In our unsupervised analyses, when comparing to the original MDS (Soil dataset) and PCoA (Smokers dataset), the main structure (ordination by pH in the former, and by body site in the latter) was conserved in kPCA. On the other hand, microbial signatures obtained with SVM had a biological interpretation. In general, the most important taxa retrieved from SVM coincided with those of RF (40-65% of overlap depending on the dataset). Concerning supervised analyses, SVM were consistently better (or at least equivalent) to RF in all three case studies. This disagrees with some previous reports in the microbiome area, *e.g.* Zhou and Gallins (2019). However, it should be noted that SVM performance depends on the kernel used, and these reports used generic linear and RBF kernels. Even when using kernels specific for metagenomic data, we observed differences among their mean NMSE or accuracies as large as fifteen percentage points. At the same time, our results suggest that there is not a single kernel that systematically achieves the best performance in every problem. We found that cLin was the best one in the first case study, qJac in the second and fRBF in the third. In this scenario, we consider that the linear-like kernels like cLin are a safe starting point. They allow for the retrieval of the microbial signatures, are faster to compute and easier to interpret than nonlinear kernels, and with high-dimensional data ($> 10^3 - 10^4$ taxa) they tend to match the RBF kernel (usually considered the gold standard) in performance (Hsu *et al.*, 2003; Keerthi & Lin, 2003). Phylogenetic kernels were beyond the scope of this work, nor the available datasets had the phylogenetic trees needed to compute them. However, they can be derived from (6.1) and (6.2) by replacing the clr term with other transformations, *e.g.* the PhILR transformation (Silverman *et al.*, 2017). A phylogeny-based kernel was also proposed in Xiao and Chen (2017).

We illustrated the integration of spatial-structured samples with the Smokers dataset. The analysis in the original work was carried out in each sampling site independently, with a

maximum median accuracy of 71%. Here we showed how combining the body sites using MKL increased the median accuracy to 92%. Therefore, our results remark the relevance of using an integrative approach to improve the accuracy of phenotype prediction when spatial-structured samples of the same individuals are available.

In addition to the package and framework proposal, we presented a novel dataset profiling the microbiota evolution and pre-weaning diarrhea incidence in 153 piglets. Through this dataset we illustrated the kernel framework application to time-structured samples. Pre-weaning diarrhea is an important issue in pig production, as the antibiotic treatment increases both the emergence of resistances and the economic costs. It is already known that gut colonization starts immediately after birth, and it evolves from a highly variable to a more stable and homogeneous ecosystem over the first weeks. However, most of the current studies in pig production ignore early dynamics in gut microbiota (Massacci *et al.*, 2020; Han *et al.*, 2018; Mach *et al.*, 2015). We wanted to test if pre-weaning diarrhea could be anticipated as soon as the first week of life. In this sense, our results suggest that the first stages of intestinal microbiota convey some valuable information indeed. kPCAs showed a partial separation between piglets affected of diarrhea versus healthy piglets, and by using longitudinal kernels we achieved a moderate accuracy of 76%. However, it was unclear if this accuracy was to be attributed to a different taxa evolution in the two groups over the first week, or to a single time point with a great predictive value. The day-by-day prediction clarified this issue, and showed that day 7 achieved a median accuracy of 73% while the rest of points lacked predictive power. Even so, longitudinal kernels were able to slightly improve prediction (76% vs. 73% at the ASV level, and 69% vs. 64% using Genera), so global taxa evolution may also have a small role.

This is also seen in the underlying microbial signatures of the global first week (fLin) versus day 7 (cLin). To be noted, in day 7 the most important genus was sulfate-reducing bacteria *Desulfovibrio*, which is known to have a relevant role during pig gut colonization (Mach *et al.*, 2015). Instead, the global model was mainly led by *Lactobacillus* and *Bacteroides*. Relationship of both genera to pre-weaning diarrhea is well sustained in literature. *Lactobacillus* spp. are well known probiotic bacteria, while members of *Bacteroides* genus are associated with increased infants gut microbial diversity (Stewart *et al.*, 2018). Furthermore, both play an important role on mammals' gut microbial colonization (Wexler & Goodman, 2017; Sawicki *et al.*, 2017) and are dominant in healthy pigs compared with diarrhea-affected piglets (Song *et al.*, 2017), which gives confidence in the reliability of our findings.

In summary, our kernel framework successfully unifies the most important analyses in the microbiome field, takes into account the compositionality of data, and is flexible enough to integrate spatial and temporal dimensions of the datasets.

Declarations

Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author Contributions

YRC, MPE and RQ, and ER contributed to conception and design of the study. FM was in charge of the pig data sampling. YCR and MPE supervised the overall research, while LBM supervised the machine learning part. ER performed all analysis and wrote the first draft of the manuscript. YRC, MPE and LBM revised and re-wrote sections of the manuscript. All authors contributed to manuscript revision, read, and approved the submitted version.

Funding

This work was funded by projects AGL2016–78709-R and AGL2017–88849-R awarded by the Spanish Ministry of Economy and Competitiveness. ER has funding from a FI-AGAUR PhD studentship grant, with the support of the Secretaria d'Universitats i Recerca de la Generalitat de Catalunya and the European Social Fund. YRC was funded by Marie Skłodowska-Curie grant (P-Sphere) agreement No 6655919 (EU). We acknowledge further financial support from the Spanish Ministry of Economy and Competitiveness through the “Severo Ochoa Programme for Centres of Excellence in R&D” 2016–2019 (SEV-2015-0533)”, and from the EU through the BFU2016–77236-P (MINECO/AEI/FEDER, EU).

Acknowledgments

The authors warmly thank all technical staff from Schothorst Feed Research.

CHAPTER 7

DISCUSSION AND FUTURE RESEARCH

This work uses specific kernel tools to perform phenotype prediction and pattern inference from biological data. The main contribution of this thesis is showing how the flexibility and modularity of kernel functions enable a “personalized” approach to different problems. This is illustrated across the different chapters, as in all of them we use kernels adapted to the nature of data: sequences in Chapter 4, graphs in Chapter 5, and abundance data in Chapter 6. For the former two cases, we propose new kernels based on the Overlap and Jaccard kernels: the “RBF-like” versions and the ones embedded into the exponential random walk kernel. In the latter chapter, we propose the Aitchison-RBF and the compositional linear kernels. Results were always contrasted with those of previously described kernels and well-established machine learning (ML) methods: Random Forest (RF) and Artificial Neural Networks (ANN). Additionally, Chapter 6 deals with space and time-structured datasets, for which we propose using Multiple Kernel Learning (MKL) and specific kernels for time series, respectively.

Another important subject in this thesis is the use of the kernel framework to give both a supervised (via Support Vector Machines or SVM) and unsupervised (kernel Principal Components Analysis or kPCA) view over data. This is a novel approach in microbiome analysis and thus has the utmost importance in Chapter 6, but also plays a secondary role in chapters 4 and 5. Finally, a relevant question was how previous knowledge can be encoded into a model to improve prediction; or conversely, how the machine learning model can provide valuable insights into a specific problem. Examples of prior knowledge inclusion were the weighting of all kernels of Chapter 4 or the use of structural information in Chapter 5. Instead, knowledge retrieval from the model was performed in Chapter 6 to obtain microbial signatures. Our main contributions were implemented in two R packages (see ‘catkern’ and

'kernInt' vignettes in the Annexes), which contain the main kernels proposed, the integration of space and time-structured datasets, the supervised and unsupervised view, and the automatic recovering of microbial signatures. Though in the present PhD thesis we dealt with HIV drug resistance prediction and metagenomic analysis, the tools we provide can also help to tackle other related problems.

Strengths and limitations of our work are reviewed next. In section 7.1, some general considerations valid for all chapters are discussed. In section 7.2, we give a complete picture of our approach to HIV genomic data particularities (chapters 4 and 5) and suggest future research lines. Lastly, in section 7.3, we do the same with the application of the kernel framework to microbiome analysis in Chapter 6.

7.1 General remarks

Data-driven approaches stress the importance of adapting models to data. Even so, these approaches are not completely agnostic and without assumptions. Assumptions are taken, for instance, in the very moment the problem is posed and a particular machine learning method, with its own set of strengths and weaknesses, is chosen. Also, data gathering and pre-processing are constrained by implicit assumptions, *e.g.* which data is considered relevant to the problem in question, or how this data is represented (Libbrecht & Noble, 2015). During this process, valuable information may be lost and/or noise might be added inadvertently by the person conducting the study. It is important to be aware of these pitfalls, as data pre-processing is an important step in the machine learning workflow.

In this work, we have tried to minimize pre-processing by adapting the kernel to the characteristics of the problem analyzed. Some reasons for doing so were already advanced in Chapter 4: it reduces the risk of inserting spurious patterns and is, in general, more computationally efficient. Furthermore, the structure of the original data may convey useful information on itself. Thus, we relied in the intrinsic properties of each dataset (for instance, data type and structure), as well as in additional knowledge obtained from previous reports and studies. The idea was that prior knowledge can help to frame the learning process in a reasonable way and to discard unfruitful pre-processing and model choices.

Kernels have a rich literature about how previous knowledge can be encoded in an implicit or explicit form into the kernel function. Probably, the more exhaustive work about this subject in the biology field is *Kernel Methods for Computational Biology* or KMCB (Schölkopf *et al.*, 2004). That book is concerned, among other objectives, '*to operate on structured data types with the use of kernel functions*' (p.33). Although our goals coincide, there is little overlapping between the kernels they use and those assessed in the present work. KMCB deals primarily

with kernels for strings, kernels over parametric statistical models (*e.g.* the Fisher kernel), and some kernels for graphs. Kernels specific for counts, time series and categorical data are absent –in general, there is a surprising lack of categorical kernels in the field (Belanche & Villegas, 2013)–. Furthermore, albeit SVM has reached a wide audience and is now a quite standard method, some of its advantages as a kernel method (particularly, the adaptability provided by using specific kernels) are often underused. We could assess, when reviewing previous work on metagenomic and HIV data, that SVM is usually coupled with default kernels. That, in turn, leads to recode the original data into numeric vectors. For example, in the case of drug resistance prediction in HIV, Yu *et al.* (2014), Rhee *et al.* (2006) and Beerenwinkel *et al.* (2003) used the linear kernel, Rhee *et al.* (2006) also used a polynomial kernel, Masso *et al.* (2014) used the RBF kernel, and Khalid and Sezerman (2016) do not specify the kernel used. Khalid and Sezerman (2016), Rhee *et al.* (2006), and Beerenwinkel *et al.* (2003) stated that HIV sequences were recoded into binary variables, while Yu *et al.* (2014) and Masso *et al.* (2014) proposed encodings that combine sequence and structural data and, ultimately, generate a numeric vector. In the case of general reviews concerning phenotype prediction from metagenomic data, Zhou and Gallins (2019) used the linear kernel and Namkung (2020) the RBF kernel. Counts were typically normalized using diverse strategies *e.g.* rarefaction or CSS, or were transformed with *clr* or *ilr* (Zhou & Gallins, 2019; Gloor *et al.*, 2017). Furthermore, in the microbiome field, we did not find any mention of kernels for spatial or temporal-structured datasets.

Using kernels adapted to the data nature can be considered as an “implicit” way to make use of our prior knowledge of the dataset (Libbrecht & Noble, 2015). Additionally, we guided the learning process explicitly in some cases. That was done by introducing the protein position importances into the kernels of Chapter 4 and, in a sense, by using structural information in Chapter 5. The weighted kernels of Chapter 4 dramatically improved the resistance prediction for NRTIs, NNRTIs, and some INIs (but not for PIs), while the graph kernel of Chapter 5 did not succeed in extracting valuable information from protein structure. It seems that the improvement observed in reverse transcriptase and integrase inhibitors is caused because few positions have a great impact on drug resistance. Instead, at least 1/3 of protease positions are involved in resistance and, in addition, there exists some degree of association among them (Iyidogan & Anderson, 2014). In this case, a larger dataset is probably needed to give a reliable estimate of the importance of each position. On the other hand, as discussed in Chapter 5, a more complete picture of structural properties that does not stop at residue distances may be able to capture the functional association between positions. Overall, this highlights the difference between having prior knowledge about a biological mechanism and

being able to put it in numbers in an effective manner. When quantification is not reliable, or when an essential part of the information is missing, it is probably better to stick to simpler models than introduce noise inadvertently.

In Chapter 6, we took a different approach: instead of encoding information into the model, we retrieved it as the importance given to each variable. Although the main purpose of our models is prediction, the obtention of variable importances is desirable in fields like microbiome analysis. It can also be applied to feature selection. Many algorithms to do so in SVM exist –for a review, see Sanz *et al.* (2018)–. However, retrieving the variable importances from an SVM is only trivial if the linear kernel is used. In the other cases, complex techniques that often include the removal or modification of a variable at a time are used. Here we have shown that, for some kernels, variable importance can be obtained in a rather straightforward manner. The key lies in knowing the map used implicitly by the kernel. In Chapter 6 we applied this approach for retrieving the microbial signatures of the three case studies reviewed, and the results we got were quite consistent with previous literature and with the variable importances given by RF. The weaknesses of this strategy are, mainly, two: first, it is very difficult to generalize, as some kernels implicitly map data into an infinite-dimensional space –for the RBF kernel, though, some approximations have been proposed; see Liu *et al.* (2011)–. The second limitation is that the input data has to be explicitly mapped into feature space, so the appeal of the kernel trick is lost.

Finally, we chose using simultaneously a supervised and an unsupervised view of the problems assessed. “Seeing” the same data (and the same kernel matrix) through the prism of different methods is another advantage of the kernel approach that is not always capitalized. At the end, SVM prediction is evaluated numerically (*e.g.* with NMSE, accuracy, etc.), but sometimes the reasons behind a specific performance outcome may be obscure. Keeping kPCA results side-to-side with those of SVM has several useful applications: it may help interpreting the SVM behavior, provide a complementary view of data, show some visual pattern that reinforce SVM results, or assess how different kernels grasp data. In Chapter 4 and 5, kPCA was more intended to bolster SVM results. We found that it mirrored the differential pattern between protease and reverse transcriptase inhibitors, and also anticipated the improving of performance in weighted kernels. In Chapter 6, it was a more integral part of the process, as the aim was to obtain a holistic view by analyzing kPCA, SVM and microbial signatures at once. kPCA not only reinforced the SVM results, but also provided some valuable information on its own (*e.g.* the band nature of the Soil dataset).

7.2 The approach to HIV data particularities

Viruses are the only known organisms that can use RNA as genetic material. There exist RNA viruses in the strict sense and others that, like HIV, switch between RNA and DNA during their replication cycle. In any case, both of them tend to present a genetic variability per generation time much higher than any organism purely based on DNA (Moya *et al.*, 2000). They have strikingly high mutation rates, around 100 times greater than those of DNA-based viruses (Peck & Luring, 2018). Indeed, a large number of deleterious mutants is generated at each generation, but this drawback is compensated by small replication times, high virion yields and large populations. RNA-based viruses evolve at very high rates within their infected hosts (Lythgoe & Fraser, 2012), creating a complex and dynamic constellation of genetically-linked variants: the quasispecies. Furthermore, high rates of recombination (as those of HIV) further increase variability in the viral population by redistributing new alleles and forming complex mutational patterns (Iyidogan & Anderson, 2014). All together, these traits allow a fast exploration of potential sequence variants to overcome selective pressure, *e.g.* host immune system or drug therapy.

The Stanford HIV Drug Resistance Database reflects the extremely high genetic variability of HIV. Considering only missense mutations, the number of polymorphic positions ranged from 93% in protease (1700 sequences analyzed) to 75% in integrase (652 sequences analyzed). The number of different observed alleles was as high as 16 in some protease positions. Mean number of mutations per sequence, with respect to the consensus, ranged between 10 in protease –in agreement with Nalam and Schiffer (2008) data from therapy-experienced patients– and 7.31 in integrase. That said, most positions were dominated by wild type alleles, while alternative ones tended to be present in low frequencies or even once (averaging the three proteins, this was seen in $\approx 73\%$ of positions). This further remarks the singularity of each HIV isolate. Finally, almost the 60% of samples contained mixtures from 2 to 4 amino acids in at least one position, pointing to the presence of quasispecies.

Thus, HIV sequence datasets have very unusual features: high range of alternative alleles (many of them in very low frequencies), mixtures of an arbitrary number of these alleles in a given *locus*, and uniqueness of the sequences. These particularities complicate classic genetic analyses, as the r^2 estimation of LD in Chapter 5. They also pose several challenges to predictive models. Most supervised methods cannot deal with categorical variables, and so (as reviewed in Chapter 4 and section 7.1) protein sequences are typically converted to binary “dummy” variables by one-hot encoding. But what happens when more than a negligible fraction of mutations is observed once? Some of these instances will be placed in the test set

and not in the training set. If the model is built from the alleles observed in the training set, binary variables for these rare mutations will not be generated and a proper prediction in the test set cannot be done. Rare variant filtering has been applied in other works (Sheik Amamuddy *et al.*, 2017; Khalid & Sezerman, 2016); however, this approach plays down the emergence of novel mutations, which is an important mechanism for HIV to evade the antiretroviral therapy. An alternative solution is to generate as many dummy variables as canonic amino acids, thus expanding the dataset features in a 20x factor. This increases computational costs and operates against methods like ANN, which are more vulnerable to the curse of dimensionality. As for amino acid mixtures, we do not consider that they are well handled with sequence expansion, as is extensively discussed in Chapter 4. Other strategies, as keeping only one allele, lead to losing part of the information and do not take into account that HIV variants coexist within infected hosts as quasispecies.

Contrarily to other works (Sheik Amamuddy *et al.*, 2017; Khalid & Sezerman, 2016; Shen *et al.*, 2016; Yu *et al.*, 2014; Masso *et al.*, 2014), we decided to proceed with minimal dataset pre-processing. To do so, we proposed kernel functions capable to handle data with so much intrinsic variability: the Overlap and the Jaccard kernels. As said in subsection 3.3.2, the plain Overlap kernel gives the same kernel matrix that the linear kernel applied over data transformed by one-hot encoding, but has the computational advantages of the kernel trick. We modified the standard formulation of the Overlap and Jaccard by introducing an exponentiation, which allowed the nonlinear approach of the standard RBF kernel, but without increasing the dataset dimensionality.

In both chapters 4 and 5, the Jaccard kernel stood out among the other kernels because it could also accept amino acid mixtures as input. That allowed for more fidelity to the HIV quasispecies structure while sparing a great deal of dataset pre-processing. Although the Jaccard kernel is equivalent to the Overlap kernel in absence of mixtures, it systematically improved the Overlap kernel performance, sometimes by a large margin. It was the best kernel in all drugs but one, thus confirming that the amino acid mixtures need to be taken into account by the resistance prediction models.

Research on the most relevant and widespread mutations has been one of the most important contributions to the inference of HIV drug resistance. That knowledge is the foundation of rule-based algorithms, but it is not particularly exploited in ML prediction models. A possible reason is that many methods do not provide a straightforward way to encode additional information. In our case, that was possible by modifying the definition of all kernels so each position was assigned a weight. But we encountered several issues to directly use the rule-based information as input to our kernels: (i) this information is binary in

catalogs like Wensing *et al.* (2017), *i.e.* important/not important, and we wanted a weighting that acknowledged that there may be different levels of importance, (ii) software like Stanford HIVdb provide the fold-decrease in susceptibility per specific mutation or group of mutations, but not per protein position. Thus, we obtained this information instead for the RF mean decrease in node impurity. We already showed in Chapter 4 that the variable importance profiles obtained with RF mostly agrees with the major drug resistance-associated residues found in catalogues. Weighting was not of much use in the PIs, but it delivered a dramatic improvement of prediction in most INIs and reverse transcriptase inhibitors. We found a significant relationship between this result and the inequality of the position importances' distribution. The reasons behind this trend seem to be related to two well-known different mutational strategies among viral proteins: few high-impact residues cause reverse transcriptase to become resistant to its inhibitors, while protease tends to accumulate a greater number of mutations.

Kernels of Chapter 4 assumed perfect independence between protein positions and did not consider any type of LD. That is completely opposite to the treatment of sequence data that is done, for instance, in KMCB. They mostly use *string kernels*, which are based on comparing the substrings (*e.g.* of length k) or motifs present in two sequences. That approach has been successfully applied to problems as homology detection. However, as they look for substrings or motifs, string kernels implicitly assume that only close positions within the sequence are associated. That is compatible with background LD but, in HIV, this kind of association is continuously scrambled by the high recombination and mutation rates of the virus. It has been reported (Wang & Lee, 2007) that (at least in datasets containing HIV isolates from treatment-experienced patients) most associations are caused by functional interactions between protein residues. In addition, our analysis of LD in the Stanford datasets shown that positions with more strong associations in average are also known to be involved in drug resistance.

Henceforth, in Chapter 5, we decided to embed the Overlap and the Jaccard kernels into a kernel for graphs: the exponential random walk kernel. The underlying idea was that close positions within the folded protein are more likely to interact. We modeled each protease and reverse transcriptase isolate as a graph. To do so, we used empirical data obtained with X-ray crystallography, so the edges were the Euclidean distance between each pair of protein positions, while each allele of the sequence was the label of a node. The implementation was optimized as much as possible to make the exponential kernel computation feasible. However, at the end, we did not succeed in improving the prediction performances of the unweighted kernels over sequences of Chapter 4, while the computation time was much

more costly (even with the optimized implementation). The most likely reasons for this outcome are: (i) oversimplification of the graph model, as the same values on the edges were used for all HIV isolates, (ii) not using other types of information (apart of distances) related to enzyme structure and (iii) failing to consider that some functional associations might appear between positions that are not close in space.

Thus, at the end, our better effort for predicting drug resistance in HIV was the Jaccard kernel over sequences, either in its weighted or unweighted form. In average, it achieved a median NMSE around 0.13 (all PIs), 0.21 (all NRTI+NNRTIs), and 0.14 (all INIs). However, there were outliers with quite worse prediction errors. The most striking case, taking into account the PI context, is TPV. DRV had lower sample size and is, like TPV, a robust second-generation PI, but gave us much better performances. A previous review that analyzed several resistance prediction algorithms also found a suboptimal performance with TPV, while they obtained, in general, good results with DRV (Stürmer *et al.*, 2011). TPV is the only nonpeptidomimetic PI and has a different mode of binding to protease. It is known to have a particular resistance profile, and also that it is unaffected by most multidrug-resistant variants and that protease needs to accumulate multiple mutations to become resistant (Ali *et al.*, 2010). Indeed, TPV presented the lower Gini index (around 0.76) of all 21 drugs analyzed and was, thus, the one with the most “equal” distribution of position importances.

Our approach to drug resistance prediction has several limitations that should be highlighted. First, our kernels could only handle sequences that differed by point mutations. Therefore, sequences with indels and nonsense mutations were removed off the dataset. Although they accounted for $\leq 5\%$ of sequences, they are in no case irrelevant, as it is known that some changes in protein length are related to drug resistance. An example is the T69 insertion complex in reverse transcriptase, which induces cross-resistance to NRTIs (Iyidogan & Anderson, 2014). The ability to deal with gaps on data is an advantage of some strings kernels over the Overlap and Jaccard kernels we used.

But the most important limitation in our kernels’ definition is probably that they do not leverage any intrinsic properties to the specific amino acid placed in a given position. Their only property is to be completely equal (or completely different) to the amino acid placed in that exact position in other viral sequences. However, the physicochemical properties of amino acids (*e.g.* charge, hydrophobicity, size), make them unequal in diverse degrees. For instance, the substitution of D for E is typically more conservative (as both are negatively-charged and have similar volume) than D for W (see Figure 7.1).

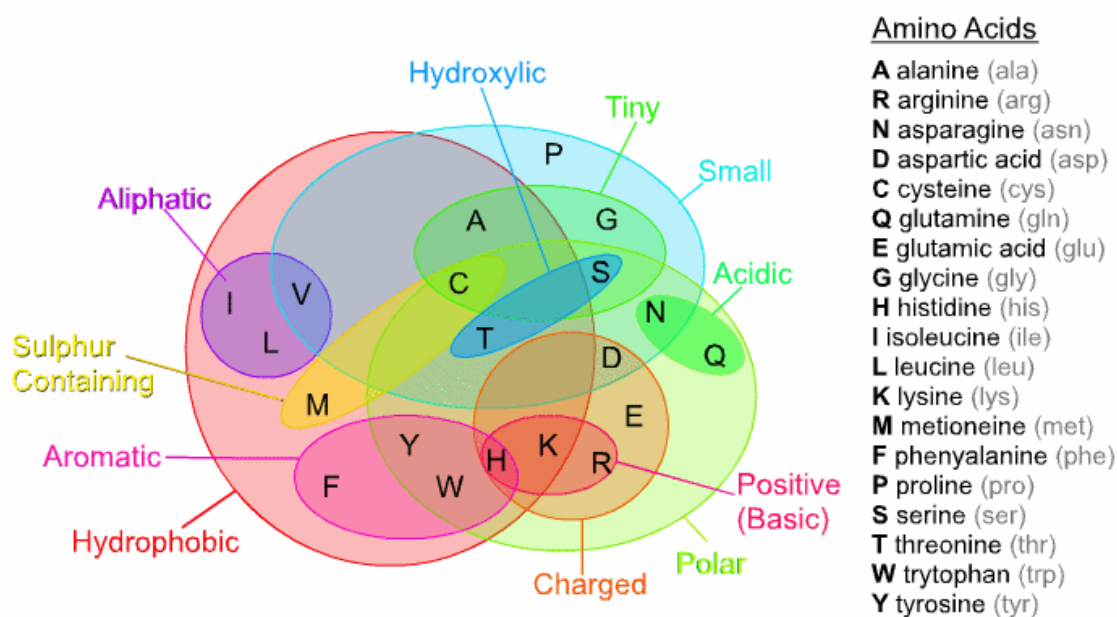


Figure 7.1. A possible classification of the canonic amino acids according to their physicochemical properties. Taken from Esquivel *et al.* (2013). License: [CC BY 3.0](https://creativecommons.org/licenses/by/3.0/).

A similar claim can be done when computing the importance of each protein position. Although it is clear that some positions are key in acquiring resistance (as seen in the dramatic improving of prediction for some drugs in Chapter 4), not all amino acid substitutions may be equally important. As said before, this is acknowledged by catalogs of resistance-associated positions, which always highlight the key alleles that decrease (or increase) the susceptibility to a drug. An example: it is well known that the K70R substitution confers cross-resistance to all NRTIs, while K70E confers resistance to TDF (Wensing *et al.*, 2019). In our version of the Stanford NRTI dataset we found, apart from R and E, a high frequency of the wild type allele K, plus 6 additional mutations: A, G, N, Q, S and T (all of them in frequencies $\leq 1\%$). It is possible that they are irrelevant to drug resistance or that some of them may have a small impact. In any case, when weighting the kernels of Chapter 4, the importance (obtained via RF) was not given to the alleles R and E, but globally to position 70.

The same problem reappears in Chapter 5. It is evident that substituting a given amino acid for another induces a spatial rearrangement in the neighborhood of that residue and, in some cases, in the whole protein. However, potential coordinate changes in mutated proteins were not looked upon. Furthermore, we did not assess additional data like physicochemical properties of amino acids, which –as suggested by studies like Khalid & Sezerman (2016)– may give a more nuanced view of protein structure.

The exponential random walk kernel that we used in Chapter 5 did not result in an improvement of prediction performance. Yet, this kernel provides an excellent framework to

include different types of data apart from sequences and inter-residue distances. Information about contact energies, volumes, physicochemical properties, etc. that individualize the amino acid residues can also be included with little effort. This can be achieved by including additional kernels for evaluating the nodes and/or edges in (5.4); see Borgwardt *et al.* (2005) for an example. That way, the aforementioned limitations of the present work can be overcome. We expect that a better grasp of the functional interactions between positions enhances the prediction of drug resistance, at least in protease, as it seems reasonable due to its particular mutational pattern.

7.3 Synthesis of current microbiome analysis

The advent of NGS technologies in the first decade of 21st century was a turning point in genomics. These technologies allowed, for the first time, to transcend the individual organism to give a whole view on the genes and genomes of a full microbial community (National Research Council (US), 2007). Before NGS, individual microorganisms had to be isolated and cultured prior to be studied. However, most of the microbial diversity on earth is not cultivable, and therefore was invisible for these methodologies (Handelsman, 2004). Metagenomics and other ‘omics’ have provided great volumes of sequence data that highlight both the ubiquity of ecological niches for microorganisms (which include plant and animal hosts) and the complexity of interactions within the community and with their hosts. Today, it is widely acknowledged that microbiota (*i.e.* the set of all microorganisms of a given environment) plays a critical role in complex diseases like diabetes, asthma, allergies or inflammatory bowel disease (Berg *et al.*, 2020). Thus, phenotype prediction from microbial data (*e.g.* metagenomics) has important implications in agriculture, livestock production and human health (Zhou & Gallins, 2019). For instance, human microbiome is becoming a key target of personalized medicine (Berg *et al.*, 2020). Due to the inherent complexity of this goal, studies have started to apply supervised ML to predict host characteristics from microbial data (Zhou & Gallins, 2019).

In Chapter 6 we propose using a kernel approach to handle microbiome analyses. A visual summary of the process is shown in Figure 7.2. In contrast to other ML methods, kernel methods start similarly to most traditional ecological analyses: computing a kernel matrix (in ecology, a distance or dissimilarity matrix is used instead). Furthermore, as shown in subsection 3.3.4, some ecological dissimilarities/distances can be turned into kernels. Thus, the kernel approach may result more familiar to ecologists than other ML methods. The main difference between community ecology analyses and the kernel approach is that, in the former, the dissimilarity matrix is used for unsupervised learning only (*e.g.* clustering,

visualization). Instead, in the latter, there exists a wide range of methods for supervised as well as unsupervised settings. In Chapter 6, this advantage is exploited to unify different kinds of microbiome analyses under the umbrella of the kernel framework. SVM were used as a representative of the former analysis and kPCA of the latter. That way, we do not simply “aggregate” methods in parallel: both share the same kernel matrix, and hence are bonded in some degree, though they have different purposes (prediction on one hand, data visualization and/or feature extraction on the other) and may emphasize different aspects of the dataset.

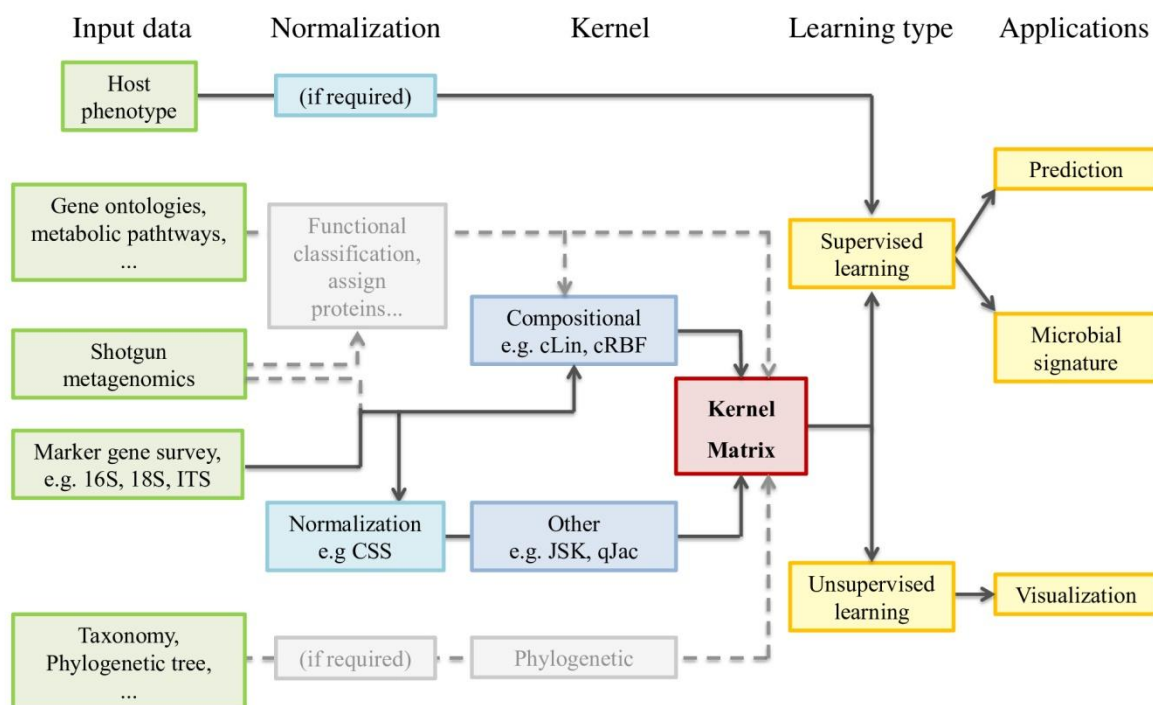


Figure 7.2. Metagenomic analysis workflow when using the kernel framework. The pivotal position of the kernel matrix is clearly observed. In grey, several tasks not performed during the present work but that merit future research.

The kernel framework is not only flexible in allowing a great range of different analyses, but also in the almost infinite ways it provides to approach data. This can be achieved at any time by changing the kernel functions used, while the rest of the process remains the same. This versatility is useful in an emerging field as microbiome analysis, where there is not a general agreement in subjects like which is the best normalization technique (Costea *et al.*, 2014; McMurdie & Holmes, 2014; Paulson *et al.*, 2013) or distance (Parks & Beiko, 2013). Some reports point that the best approach depends upon the specific problem and data at hand (Weiss *et al.*, 2017). In Chapter 6 we propose four different kernels for microbiome data but, contrarily to chapters 4 and 5, the aim was not performing an exhaustive comparison to find

the best one, nor was it feasible with only three case studies. Instead, the goal was to give a few examples of kernels, each one of them having its own motivation and strengths. Two of them inherently take into account that data is compositional (thus, we called them “compositional kernels”), while the other two were obtained from widespread beta-diversity measures in community ecology. However, we acknowledge that there are limitations in our treatment of microbiome data, as we discuss next.

NGS data is considered compositional because it consists of counts bounded by an uninformative sum, which in this case is the library size (*i.e.* the total reads per sample). We say that library size is uninformative because it does not contain information about the population. Instead, it is arbitrarily fixed by the sequencing process and it may vary by orders of magnitude across samples (McMurdie & Holmes, 2014). As the number of total reads is constrained, an increase in a taxon leads to an overall decrease of the rest (Gloor *et al.*, 2017; McMurdie & Holmes, 2014). Thus, absolute counts are irrelevant, as they only account for the precision of the estimate, and difference between taxa is only meaningful in proportion. Expressing compositional data as log-ratios (*e.g.* clr-transformation) acknowledges this dependence and also enables mapping the data into real space.

However, NGS data is extraordinarily sparse, and in that point it differs from compositional data in a strict sense. The proportion of zeroes in most OTU tables is around 90% (Weiss *et al.*, 2017). In our case, the proportions were as high as 95% in the Soil dataset, 96% in the Smokers dataset, and 90% in the ASV table of the Pig dataset (which fell to 65% when aggregating by genera). Of course, it is not possible to know if a zero means true absence or abundance below the detection limit. How to deal with zeroes is an open topic of research in compositional analysis (Quinn *et al.*, 2018; Weiss *et al.*, 2017) as, by definition, log-ratios do not handle them well. That is a weakness of our compositional kernels (instead, zeroes pose no problems for the quantitative Jaccard kernel). We did not apply a very innovative approach to the subject: we added a pseudo-count to all elements of the dataset (Quinn *et al.*, 2018) prior to computing the compositional kernels. There exist other replacement strategies involving pseudo-counts, but again, there is not a consensus on how choosing them. It is known, however, that pseudo-counts may alter slightly the ratios between taxa. A more sophisticated approach is replacing zeroes using a compositionally valid Dirichlet distribution, but this assumes that every pair of taxa has a negative correlation (Quinn *et al.*, 2018; Weiss *et al.*, 2017). More research is needed to design compositional kernels that do not require pre-processing zeroes in NGS count data tables.

In addition, the scope of our work was limited because the public datasets that we used in Chapter 6 only had information about taxonomic abundances. Thus, although we succeeded

in analyzing three distinct datasets in structure (one single point dataset, a longitudinal dataset and a dataset with spatial-related samples) and in application (soil studies, human health and livestock production), we did not study functional data, nor we included information of the phylogenetic tree in the kernel computation (see Figure 7.2). Replacing the clr-transformation in (3.39) by the PhILR transformation (Silverman et al., 2017) may allow the generation of alternative compositional linear and RBF kernels that take into account phylogenetic data. However, some problems with the retrieval of microbial signatures may arise, as the ilr-transformation generates $D - 1$ features from D taxa, and does not allow for insights into the relationships between single features in the dataset (Gloor *et al.*, 2017).

Finally, regarding the spatial and temporal-structured datasets from Chapter 6, we obtained a great increase in prediction performance in the former when integrating the related samples using MKL. Instead, the integration via longitudinal kernels in the latter dataset (*i.e.* the Pig case study) resulted only in a slight improvement over the prediction using day 7. This is in contrast with results shown in the ‘kernInt’ vignette (Annexes), in which we apply a longitudinal kernel to the Berkeley Growth dataset (presented in subsection 1.3.2). There, the results are much better when using the fRBF kernel over all data (median accuracy = 0.95) than when using only data at age 18 (median accuracy around 0.85). Then, what are the reasons behind the results of the Pig dataset? One possibility is that the number of time points (only three, while the Berkeley Growth dataset has 31) is too low to have a good grasp of potential differences in microbial colonization between healthy and ill pigs. It is also possible that microbiota of piglets in days 0 and 3 post-birth is still too unstable, so maybe a sampling design that includes later points (*e.g.* second week of life) could be more informative. Also, it has to be stressed that we used general kernels for longitudinal data, but without fully address that we were dealing with a compositional time series. We share this limitation with other related works, *e.g.* Bodein *et al.* (2019). In our case, the abundance data was mapped out of the simplex with the clr-transformation, and only then we applied the fLin and fRBF kernels for discrete functions. Ideally, the time series function would model absolute frequencies, but compositional data only has information about relative frequencies. It seems that is not possible to successfully recover the original absolute counts from a compositional sample (Quinn *et al.*, 2018). To our best knowledge, a specific compositional approach to longitudinal data does not exist yet.

CHAPTER 8

CONCLUSIONS

- I. Current treatment for HIV infection is chronic and, thus, the emergence of drug resistance is a serious concern. We used RF and SVM models to predict resistance to 21 drugs from mutated protease, integrase and reverse transcriptase sequences. We propose specific kernels for HIV data that acknowledge the presence of amino acid mixtures and/or the categorical nature of the protein positions. Taking into account both characteristics (as done in this thesis with the Jaccard kernel) consistently results in the best option to predict drug resistance in HIV.
- II. Incorporating prior information of the importance of protein positions onto the kernel further improves prediction in reverse transcriptase and integrase, but not in protease. The amount of improvement grows with the inequality of the importance distribution, as measured by the Gini index. Thus, the effect of incorporating the position weights depends on the specific mutational pattern leading to drug resistance, which is different among the three viral proteins.
- III. Conversely, including information about distances between residues on the folded enzymes does not increase the predictive ability of the categorical kernels assessed, *i.e.* the Overlap and the Jaccard kernels. The inclusion of more diverse structural data (*e.g.* amino acid volumes and physicochemical properties, contact energies...) may be necessary to properly grasp the functional interactions between protein positions.
- IV. Our kernel framework successfully unifies the most important analyses in the microbiome field: phenotype prediction (SVM), data visualization (kPCA) and retrieval

of microbial signatures (*i.e.* taxa importances) from the SVM. This approach does not only give a holistic view of data, but also delivers good results in each learning area.

- V. The kernel framework allows diverse approaches to microbiome data by using different kernels; for instance, kernels derived from widely used beta-diversity measures, or the kernels for compositional data proposed in this thesis: the Aitchison-RBF and the compositional linear kernels. Furthermore, this framework can handle spatial and temporal-structured microbiome study designs by using multiple kernel learning or specific kernels for time series, respectively.

REFERENCES

- Ahnert S. E. (2017). Structural properties of genotype-phenotype maps. *Journal of the Royal Society, Interface*, 14(132), 20170275. <https://doi.org/10.1098/rsif.2017.0275>
- Aitchison, J., Barceló-Vidal, C., Martín-Fernández, J. A., & Pawlowsky-Glahn, V. (2000). Logratio analysis and compositional distance. *Mathematical Geology*, 32(3), 271-275.
- Ali, A., Bandaranayake, R. M., Cai, Y., King, N. M., Kolli, M., Mittal, S., Murzycki, J. F., Nalam, M. N., Nalivaika, E. A., Ozen, A., Prabu-Jeyabalan, M. M., Thayer, K., & Schiffer, C. A. (2010). Molecular Basis for Drug Resistance in HIV-1 Protease. *Viruses*, 2(11), 2509–2535. <https://doi.org/10.3390/v2112509>
- Alonso-Atienza, F., Rojo-Álvarez, J. L., Rosado-Muñoz, A., Vinagre, J. J., García-Alberola, A., & Camps-Valls, G. (2012). Feature selection using support vector machines and bootstrap methods for ventricular fibrillation detection. *Expert Systems with Applications*, 39(2), 1956-1967.
- Alpaydm, E. (2010). Introduction to machine learning (2nd ed.). MIT press
- Amamuddy, O. S., Bishop, N. T., & Bishop, Ö. T. (2017). Improving fold resistance prediction of HIV-1 against protease and reverse transcriptase inhibitors using artificial neural networks. *BMC bioinformatics*, 18(1), 1-7.
- Anstett, K., Brenner, B., Mesplede, T., & Wainberg, M. A. (2017). HIV drug resistance against strand transfer integrase inhibitors. *Retrovirology*, 14(1), 36. <https://doi.org/10.1186/s12977-017-0360-7>
- Argelaguet, R., Velten, B., Arnol, D., Dietrich, S., Zenz, T., Marioni, J. C., ... & Stegle, O. (2018). Multi-Omics Factor Analysis—a framework for unsupervised integration of multi-omics data sets. *Molecular systems biology*, 14(6), e8124.
- Bai, L., & Hancock, E. R. (2011, August). Graph clustering using the jensen-shannon kernel. In *International Conference on Computer Analysis of Images and Patterns* (pp. 394-401). Springer, Berlin, Heidelberg.
- Beerenwinkel, N., Daumer, M., Oette, M., Korn, K., Hoffmann, D., Kaiser, R., ... & Walter, H. (2003). Geno2pheno: Estimating phenotypic drug resistance from HIV-1 genotypes. *Nucleic acids research*, 31(13), 3850-3855.

- Beerenwinkel, N., Schmidt, B., Walter, H., Kaiser, R., Lengauer, T., Hoffmann, D., ... & Selbig, J. (2002). Diversity and complexity of HIV-1 drug resistance: a bioinformatics approach to predicting phenotype from genotype. *Proceedings of the National Academy of Sciences*, 99(12), 8271-8276.
- Belanche Muñoz, L. A., & Villegas, M. (2013). Kernel functions for categorical variables with application to problems in the life sciences. In *Artificial intelligence research and development: proceedings of the 16 International Conference of the Catalan Association of Artificial Intelligence* (pp. 171-180).
- Berg, G., Rybakova, D., Fischer, D., Cernava, T., Vergès, M. C. C., Charles, T., ... & Kazou, M. (2020). Microbiome definition re-visited: old concepts and new challenges. *Microbiome*, 8(1), 1-23.
- Bhadra, S., Kaski, S., & Rousu, J. (2017). Multi-view kernel completion. *Machine Learning*, 106(5), 713-739.
- Binkowski, M., Marti, G., & Donnat, P. (2018, July). Autoregressive convolutional neural networks for asynchronous time series. In *International Conference on Machine Learning* (pp. 580-589).
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Bodein, A., Chapleur, O., Droit, A., & Lê Cao, K. A. (2019). A generic multivariate framework for the integration of microbiome longitudinal studies with other data types. *Frontiers in genetics*, 10.
- Bonet, I. (2015). Machine learning for prediction of HIV drug resistance: a review. *Current Bioinformatics*, 10(5), 579-585.
- Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V., Smola, A. J., & Kriegel, H. P. (2005). Protein function prediction via graph kernels. *Bioinformatics (Oxford, England)*, 21 Suppl 1, i47-i56. <https://doi.org/10.1093/bioinformatics/bti1007>
- Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152).
- Bouchard, M., Jusselme, A. L., & Doré, P. E. (2013). A proof for the positive definiteness of the Jaccard index matrix. *International Journal of Approximate Reasoning*, 54(5), 615-626.
- Cao, J., & Xiong, L. (2014). Protein sequence classification with improved extreme learning machine algorithms. *BioMed research international*, 2014, 103054. <https://doi.org/10.1155/2014/103054>
- Chen, H., Tang, F., Tino, P., & Yao, X. (2013, August). Model-based kernel for efficient time series analysis. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 392-400).
- Chen, R. W., Wang, K. X., Wang, F. Z., He, Y. Q., Long, L. J., & Tian, X. P. (2018). *Rubrobacter indicoceni* sp. nov., a new marine actinobacterium isolated from Indian Ocean sediment. *International Journal of Systematic and Evolutionary Microbiology*, 68(11), 3487-3493.

- Charlson, E. S., Chen, J., Custers-Allen, R., Bittinger, K., Li, H., Sinha, R., Hwang, J., Bushman, F. D., & Collman, R. G. (2010). Disordered microbial communities in the upper respiratory tract of cigarette smokers. *PLoS one*, *5*(12), e15216. <https://doi.org/10.1371/journal.pone.0015216>
- Chial, H. (2008). Rare genetic disorders: learning about genetic disease through gene mapping, SNPs, and microarray data. *Nature education*, *1*(1), 193.
- Coenen, A. R., Hu, S. K., Luo, E., Muratore, D., & Weitz, J. S. (2020). A Primer for Microbiome Time-Series Analysis. *Frontiers in genetics*, *11*, 310. <https://doi.org/10.3389/fgene.2020.00310>
- Collins, F. S., Morgan, M., & Patrinos, A. (2003). The Human Genome Project: lessons from large-scale biology. *Science (New York, N.Y.)*, *300*(5617), 286–290. <https://doi.org/10.1126/science.1084564>
- Costea, P. I., Zeller, G., Sunagawa, S., & Bork, P. (2014). A fair comparison. *Nature methods*, *11*(4), 359–359.
- Costello, E. K., Lauber, C. L., Hamady, M., Fierer, N., Gordon, J. I., & Knight, R. (2009). Bacterial community variation in human body habitats across space and time. *Science (New York, N.Y.)*, *326*(5960), 1694–1697. <https://doi.org/10.1126/science.1177486>
- Crowder, C., & Austin, D. (2005). Age ranges of epiphyseal fusion in the distal tibia and fibula of contemporary males and females. *Journal of forensic sciences*, *50*(5), 1001–1007.
- Dahal, R. H., & Kim, J. (2017). *Microvirga soli* sp. nov., an alphaproteobacterium isolated from soil. *International Journal of Systematic and Evolutionary Microbiology*, *67*(1), 127–133.
- Davies, P. C., Rieper, E., & Tuszynski, J. A. (2013). Self-organization and entropy reduction in a living cell. *Bio Systems*, *111*(1), 1–10. <https://doi.org/10.1016/j.biosystems.2013.10.005>
- De Clercq, E. (2009). The history of antiretrovirals: key discoveries over the past 25 years. *Reviews in medical virology*, *19*(5), 287–299.
- Esquivel, R. O., Molina-Espíritu, M., Salas, F., Soriano, C., Barrientos, C., Dehesa, J. S., & Dobado, J. A. (2013). Decoding the building blocks of life from the perspective of quantum information. In *Advances in Quantum Mechanics*. IntechOpen.
- Gardener, M. (2014). *Community ecology: analytical methods using R and Excel*. Pelagic Publishing Ltd.
- Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines* (pp. 129–143). Springer, Berlin, Heidelberg.
- Genotype-Phenotype Stanford University HIV Drug Resistance Database (n.d.) Retrieved May 30, 2019, from <https://hivdb.stanford.edu/pages/genopheno.dataset.html>.
- German Advisory Committee Blood (Arbeitskreis Blut), Subgroup 'Assessment of Pathogens Transmissible by Blood'. Human Immunodeficiency Virus (HIV) [Blood G. A. C]. (2016). Human immunodeficiency virus (HIV). *Transfusion Medicine and Hemotherapy*, *43*(3), 203.

- Gligorijević, V., & Pržulj, N. (2015). Methods for biological data integration: perspectives and challenges. *Journal of the Royal Society, Interface*, *12*(112), 20150571. <https://doi.org/10.1098/rsif.2015.0571>
- Gloor, G. B., Macklaim, J. M., Pawlowsky-Glahn, V., & Egozcue, J. J. (2017). Microbiome Datasets Are Compositional: And This Is Not Optional. *Frontiers in microbiology*, *8*, 2224. <https://doi.org/10.3389/fmicb.2017.02224>
- Gundry, R. L., White, M. Y., Murray, C. I., Kane, L. A., Fu, Q., Stanley, B. A., & Van Eyk, J. E. (2010). Preparation of proteins and peptides for mass spectrometry analysis in a bottom-up proteomics workflow. *Current protocols in molecular biology*, *90*(1), 10-25.
- Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, *46*(1-3), 389-423.
- Han, G. G., Lee, J. Y., Jin, G. D., Park, J., Choi, Y. H., Kang, S. K., ... & Choi, Y. J. (2018). Tracing of the fecal microbiota of commercial pigs at five growth stages from birth to shipment. *Scientific reports*, *8*(1), 1-9.
- Handelsman, J. (2004). Metagenomics: application of genomics to uncultured microorganisms. *Microbiology and molecular biology reviews*, *68*(4), 669-685.
- Heider, D., Senge, R., Cheng, W., & Hüllermeier, E. (2013). Multilabel classification for exploiting cross-resistance information in HIV-1 drug resistance prediction. *Bioinformatics*, *29*(16), 1946-1953.
- Holmes, S., & Huber, W. (2019). *Modern statistics for modern biology*. Cambridge University Press.
- Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification. Retrieved June 12, 2020, from <https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>
- Ibrahim, O. M. (2013). A comparison of methods for assessing the relative importance of input variables in artificial neural networks. *Journal of applied sciences research*, *9*(11), 5692-5700.
- Iyidogan, P., & Anderson, K. S. (2014). Current perspectives on HIV-1 antiretroviral drug resistance. *Viruses*, *6*(10), 4095–4139. <https://doi.org/10.3390/v6104095>
- Joint United Nations Programme on HIV/AIDS [UNAIDS]. (2018). Global HIV & AIDS statistics—2018 fact sheet. *UNAIDS.org* Retrieved February 2, 2019, from http://www.unaids.org/sites/default/files/media_asset/UNAIDS_FactSheet_en.pdf.
- Keerthi, S. S., & Lin, C. J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural computation*, *15*(7), 1667-1689.
- Khalid, Z., & Sezerman, O. U. (2016). Prediction of HIV drug resistance by combining sequence and structural properties. *IEEE/ACM transactions on computational biology and bioinformatics*, *15*(3), 966-973.
- Kress, R. (2012). *Linear Integral Equations* (Vol. 82). Springer Science & Business Media.

- Lauber, C. L., Hamady, M., Knight, R., & Fierer, N. (2009). Pyrosequencing-based assessment of soil pH as a predictor of soil bacterial community structure at the continental scale. *Applied and environmental microbiology*, 75(15), 5111–5120. <https://doi.org/10.1128/AEM.00335-09>
- Liaw, A., & Wiener, M. (2015). randomForest: Breiman and Cutler's random forests for classification and regression. *R package version*, 4, 6-10.
- Li, P. (2015). Min-max kernels. *arXiv preprint*. [arXiv:1503.01737v1](https://arxiv.org/abs/1503.01737v1) [stat.ML]
- Li, J., Chen, L., Zhang, Y. H., Kong, X., Huang, T., & Cai, Y. D. (2018). A Computational Method for Classifying Different Human Tissues with Quantitatively Tissue-Specific Expressed Genes. *Genes*, 9(9), 449. <https://doi.org/10.3390/genes9090449>
- Libbrecht, M. W., & Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6), 321-333.
- Lipschutz, S., & Lipson, M. (2009). Linear Algebra (4th ed.). Schaum's Outlines.
- Liu, Q., Chen, C., Zhang, Y., & Hu, Z. (2011). Feature selection for support vector machines with RBF kernel. *Artificial Intelligence Review*, 36(2), 99-115.
- Lythgoe, K. A., & Fraser, C. (2012). New insights into the evolutionary rate of HIV-1 at the within-host and epidemiological levels. *Proceedings of the Royal Society B: Biological Sciences*, 279(1741), 3367-3375.
- Mach, N., Berri, M., Estellé, J., Levenez, F., Lemonnier, G., Denis, C., ... & Rogel-Gaillard, C. (2015). Early-life establishment of the swine gut microbiome and impact on host phenotypes. *Environmental microbiology reports*, 7(3), 554-569.
- Maldonado, S., & Weber, R. (2009). A wrapper method for feature selection using support vector machines. *Information Sciences*, 179(13), 2208-2217.
- Mariette, J., & Villa-Vialaneix, N. (2018). Unsupervised multiple kernel learning for heterogeneous data integration. *Bioinformatics*, 34(6), 1009-1015.
- Massacci, F. R., Berri, M., Lemonnier, G., Guettier, E., Blanc, F., Jardet, D., ... & Rogel-Gaillard, C. (2020). Late weaning is associated with increased microbial diversity and *Faecalibacterium prausnitzii* abundance in the fecal microbiota of piglets. *Animal Microbiome*, 2(1), 1-13.
- Masso, M., Chuang, G., Hao, K., Jain, S., & Vaisman, I. I. (2014). Structure-based predictors of resistance to the HIV-1 integrase inhibitor Elvitegravir. *Antiviral research*, 106, 5-13.
- Masso, M., & Vaisman, I. I. (2013). Sequence and structure based models of HIV-1 protease and reverse transcriptase drug resistance. *BMC genomics*, 14(S4), S3.
- Mateu-Figueras, G., Pawlowsky-Glahn, V., & Egozcue, J. J. (2011). The principle of working on coordinates. *Compositional data analysis*, 29-43.

- Mazzocchi F. (2015). Could Big Data be the end of theory in science? A few remarks on the epistemology of data-driven science. *EMBO reports*, 16(10), 1250–1255. <https://doi.org/10.15252/embr.201541001>
- McMurdie, P. J., & Holmes, S. (2014). Waste not, want not: why rarefying microbiome data is inadmissible. *PLoS Comput Biol*, 10(4), e1003531.
- Morton, J. T., Toran, L., Edlund, A., Metcalf, J. L., Lauber, C., & Knight, R. (2017). Uncovering the horseshoe effect in microbial analyses. *Msystems*, 2(1).
- Moya, A., Elena, S. F., Bracho, A., Miralles, R., & Barrio, E. (2000). The evolution of RNA viruses: a population genetics view. *Proceedings of the National Academy of Sciences*, 97(13), 6967-6973.
- Nalam, M. N., & Schiffer, C. A. (2008). New approaches to HIV protease inhibitor drug design II: testing the substrate envelope hypothesis to avoid drug resistance and discover robust inhibitors. *Current opinion in HIV and AIDS*, 3(6), 642–646. <https://doi.org/10.1097/COH.0b013e3283136cee>
- Namkung, J. (2020). Machine learning methods for microbiome studies. *Journal of Microbiology*, 58(3), 206-216.
- National Human Genome Research Institute (n.d.) The Cost of Sequencing a Human Genome. Retrieved August 18, 2020, from <https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost>
- National Research Council. (2007). *The new science of metagenomics: revealing the secrets of our microbial planet*. National Academies Press.
- Neftci, E. O., & Averbek, B. B. (2019). Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, 1(3), 133-143.
- Nguyen, T., Bhatti, A., Yang, S., & Nahavandi, S. (2016). RNA-Seq count data modelling by grey relational analysis and nonparametric Gaussian process. *PloS one*, 11(10), e0164766
- Orgogozo, V., Morizot, B., & Martin, A. (2015). The differential view of genotype-phenotype relationships. *Frontiers in genetics*, 6, 179. <https://doi.org/10.3389/fgene.2015.00179>
- Parks, D. H., & Beiko, R. G. (2013). Measures of phylogenetic differentiation provide robust and complementary insights into microbial communities. *The ISME journal*, 7(1), 173-183.
- Pasomsub, E., Sukasem, C., Sungkanuparph, S., Kijirikul, B., & Chantratita, W. (2010). The application of artificial neural networks for phenotypic drug resistance prediction: evaluation and comparison with other interpretation systems. *Jpn. J. Infect. Dis*, 63(2), 87-94.
- Paulson, J. N., Stine, O. C., Bravo, H. C., & Pop, M. (2013). Differential abundance analysis for microbial marker-gene surveys. *Nature methods*, 10(12), 1200-1203.
- Pawlowsky-Glahn, V., Egozcue, J. J., & Tolosana Delgado, R. (2007). Lecture notes on compositional data analysis.

- Peck, K. M., & Lauring, A. S. (2018). Complexities of viral mutation rates. *Journal of virology*, 92(14).
- Pekalska, E., Paclik, P., & Duin, R. P. (2001). A generalized kernel approach to dissimilarity-based classification. *Journal of machine learning research*, 2(Dec), 175-211.
- Perrin, L., & Telenti, A. (1998). HIV treatment failure: testing for HIV resistance in clinical practice. *Science*, 280(5371), 1871-1873.
- Price, K. S., Svenson, A., King, E., Ready, K., & Lazarin, G. A. (2018). Inherited cancer in the age of next-generation sequencing. *Biological research for nursing*, 20(2), 192-204.
- Qu, K., Guo, F., Liu, X., Lin, Y., & Zou, Q. (2019). Application of machine learning in microbiology. *Frontiers in Microbiology*, 10, 827.
- Quinn, T. P., Erb, I., Richardson, M. F., & Crowley, T. M. (2018). Understanding sequencing data as compositions: an outlook and review. *Bioinformatics (Oxford, England)*, 34(16), 2870–2878. <https://doi.org/10.1093/bioinformatics/bty175>
- Rhee, S. Y., Taylor, J., Wadhwa, G., Ben-Hur, A., Brutlag, D. L., & Shafer, R. W. (2006). Genotypic predictors of human immunodeficiency virus type 1 drug resistance. *Proceedings of the National Academy of Sciences*, 103(46), 17355-17360
- Riemenschneider, M., Senge, R., Neumann, U., Hüllermeier, E., & Heider, D. (2016). Exploiting HIV-1 protease and reverse transcriptase cross-resistance information for improved drug resistance prediction by means of multi-label classification. *BioData mining*, 9(1), 10.
- Riemenschneider, M., Hummel, T., & Heider, D. (2016). SHIVA-a web application for drug resistance and tropism testing in HIV. *BMC bioinformatics*, 17(1), 1-6.
- Rivera-Pinto, J., Egozcue, J. J., Pawlowsky-Glahn, V., Paredes, R., Noguera-Julian, M., & Calle, M. L. (2018). Balances: a New Perspective for Microbiome Analysis. *mSystems*, 3 (4): e00053–18.
- Rivero, R., Lemence, R., & Kato, T. (2017). Mutual kernel matrix completion. *IEICE TRANSACTIONS on Information and Systems*, 100(8), 1844-1851.
- Roser, M., Appel, C., & Ritchie, H. (2013) - "Human Height". Published online at OurWorldInData.org. Retrieved June 17,2020, from: <https://ourworldindata.org/human-height>
- Sanz, H., Valim, C., Vegas, E., Oller, J. M., & Reverter, F. (2018). SVM-RFE: selection and visualization of the most relevant features through non-linear kernels. *BMC bioinformatics*, 19(1), 433. <https://doi.org/10.1186/s12859-018-2451-4>
- Sawicki, C. M., Livingston, K. A., Obin, M., Roberts, S. B., Chung, M., & McKeown, N. M. (2017). Dietary fiber and the human gut microbiota: application of evidence mapping methodology. *Nutrients*, 9(2), 125.

- Schlub, T. E., Grimm, A. J., Smyth, R. P., Cromer, D., Chopra, A., Mallal, S., ... & Davenport, M. P. (2014). Fifteen to twenty percent of HIV substitution mutations are associated with recombination. *Journal of virology*, *88*(7), 3837-3849.
- Schmidt, B., Walter, H., Moschik, B., Paatz, C., Van Vaerenbergh, K., Vandamme, A. M., ... & Korn, K. (2000). Simple algorithm derived from a geno-/phenotypic database to predict HIV-1 protease inhibitor resistance. *Aids*, *14*(12), 1731-1738.
- Schölkopf, B., Tsuda, K., & Vert, J. P. (2004). *Kernel methods in computational biology*. MIT press.
- Shafer, R. W., Dupnik, K., Winters, M. A., & Eshleman, S. H. (2001). A guide to HIV-1 reverse transcriptase and protease sequencing for drug resistance studies. *HIV sequence compendium*, *2001*, 1.
- Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.
- Shen, C., Yu, X., Harrison, R. W., & Weber, I. T. (2016). Automated prediction of HIV drug resistance from genotype data. *BMC bioinformatics*, *17*(8), 563-569.
- Silverman, J. D., Washburne, A. D., Mukherjee, S., & David, L. A. (2017). A phylogenetic transform enhances analysis of compositional microbiota data. *Elife*, *6*, e21887.
- Song, H., Giorgi, E. E., Ganusov, V. V., Cai, F., Athreya, G., Yoon, H., ... & Jiang, C. (2018). Tracking HIV-1 recombination to resolve its contribution to HIV-1 evolution in natural infection. *Nature communications*, *9*(1), 1-15.
- Song, D., Peng, Q., Chen, Y., Zhou, X., Zhang, F., Li, A., ... & Wang, L. (2017). Altered gut microbiota profiles in sows and neonatal piglets associated with porcine epidemic diarrhea virus infection. *Scientific reports*, *7*(1), 1-10.
- St-Pierre, A. P., Shikon, V., & Schneider, D. C. (2018). Count data in biology-Data transformation or model reformation?. *Ecology and evolution*, *8*(6), 3077–3085. <https://doi.org/10.1002/ece3.3807>
- Stewart, C. J., Ajami, N. J., O'Brien, J. L., Hutchinson, D. S., Smith, D. P., Wong, M. C., ... & Muzny, D. (2018). Temporal development of the gut microbiome in early childhood from the TEDDY study. *Nature*, *562*(7728), 583-588.
- Stürmer, M., Stephan, C., Gute, P., Knecht, G., Bickel, M., Brodt, H. R., Doerr, H. W., Gürtler, L., Lecocq, P., & van Houtte, M. (2011). Comparison of drug resistance scores for tipranavir in protease inhibitor-naïve patients infected with HIV-1 B and non-B subtypes. *Antimicrobial agents and chemotherapy*, *55*(11), 5362–5366. <https://doi.org/10.1128/AAC.00611-11>
- Tarasova, O., Biziukova, N., Filimonov, D., & Poroikov, V. (2018). A computational approach for the prediction of HIV resistance based on amino acid and nucleotide descriptors. *Molecules*, *23*(11), 2751.

- Tipping, M. E. (2000). The relevance vector machine. In *Advances in neural information processing systems* (pp. 652-658).
- Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology*, 49(11), 1225-1231.
- Tuddenham, R. D., & Snyder, M. M. (1954) Physical growth of California boys and girls from birth to eighteen years. *University of California publications in child development*, 1, 183-364.
- UniProt (n.d.) UniProtKB help - Where do the protein sequences come from? Retrieved July 9, 2020, from https://www.uniprot.org/help/sequence_origin
- Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., & Borgwardt, K. M. (2010). Graph kernels. *The Journal of Machine Learning Research*, 11, 1201-1243.
- Wang, Q., & Lee, C. (2007). Distinguishing functional amino acid covariation from background linkage disequilibrium in HIV protease and reverse transcriptase. *PloS one*, 2(8), e814. <https://doi.org/10.1371/journal.pone.0000814>
- Wang, D., & Larder, B. (2003). Enhanced prediction of lopinavir resistance from genotype by use of artificial neural networks. *The Journal of Infectious Diseases*, 188(5), 653-660.
- Weise, M., De-Levi, S., Barnes, K. M., Gafni, R. I., Abad, V., & Baron, J. (2001). Effects of estrogen on growth plate senescence and epiphyseal fusion. *Proceedings of the National Academy of Sciences*, 98(12), 6871-6876.
- Weiss, S., Xu, Z. Z., Peddada, S., Amir, A., Bittinger, K., Gonzalez, A., ... & Hyde, E. R. (2017). Normalization and microbial differential abundance strategies depend upon data characteristics. *Microbiome*, 5(1), 27.
- Wensing, A. M., Calvez, V., Ceccherini-Silberstein, F., Charpentier, C., Günthard, H. F., Paredes, R., ... & Richman, D. D. (2019). 2019 update of the drug resistance mutations in HIV-1. *Topics in antiviral medicine*, 27(3), 111.
- Wensing, A. M., Calvez, V., Günthard, H. F., Johnson, V. A., Paredes, R., Pillay, D., ... & Richman, D. D. (2016). 2017 update of the drug resistance mutations in HIV-1. *Topics in antiviral medicine*, 24(4), 133.
- West-Eberhard, M. J. (2003). *Developmental plasticity and evolution*. Oxford University Press.
- Wexler, A. G., & Goodman, A. L. (2017). An insider's perspective: Bacteroides as a window into the microbiome. *Nature microbiology*, 2(5), 1-11.
- Wood, A. R., Esko, T., Yang, J., Vedantam, S., Pers, T. H., Gustafsson, S., Chu, A. Y., Estrada, K., Luan, J., Kutalik, Z., Amin, N., Buchkovich, M. L., Croteau-Chonka, D. C., Day, F. R., Duan, Y., Fall, T., Fehrmann, R., Ferreira, T., Jackson, A. U., Karjalainen, J., ... Frayling, T. M. (2014). Defining the role

- of common variation in the genomic and biological architecture of adult human height. *Nature genetics*, 46(11), 1173–1186. <https://doi.org/10.1038/ng.3097>
- Wooley, J. C., Lin, H. S., & National Research Council. (2005). On the Nature of Biological Data. In *Catalyzing Inquiry at the Interface of Computing and Biology*. National Academies Press (US).
- Wray, N. R. (2005). Allele frequencies and the r^2 measure of linkage disequilibrium: impact on design and interpretation of association studies. *Twin Research and Human Genetics*, 8(2), 87-94.
- Xia, Z., Wu, L. Y., Zhou, X., & Wong, S. T. (2010). Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces. *BMC systems biology*, 4 Suppl 2(Suppl 2), S6. <https://doi.org/10.1186/1752-0509-4-S2-S6>
- Xiao, J., & Chen, J. (2017). Phylogeny-based kernels with application to microbiome association studies. In *New Advances in Statistics and Data Science* (pp. 217-237). Springer, Cham.
- Xing, L., Lesperance, M., & Zhang, X. (2019) Simultaneous prediction of multiple outcomes using revised stacking algorithms. *arXiv preprint*. [arXiv:1901.10153v1](https://arxiv.org/abs/1901.10153v1) [q-bio.QM]
- Yip, K. Y., Cheng, C., & Gerstein, M. (2013). Machine learning and genome annotation: a match meant to be?. *Genome biology*, 14(5), 205. <https://doi.org/10.1186/gb-2013-14-5-205>
- Yu, X., Weber, I. T., & Harrison, R. W. (2014). Prediction of HIV drug resistance from genotype with encoded three-dimensional protein structure. *BMC genomics*, 15(S5), S1.
- Zaykin, D. V., Pudovkin, A., & Weir, B. S. (2008). Correlation-based inference for linkage disequilibrium with multiple alleles. *Genetics*, 180(1), 533-545.
- Zhan, X., Tong, X., Zhao, N., Maity, A., Wu, M. C., & Chen, J. (2017). A small-sample multivariate kernel machine test for microbiome association studies. *Genetic epidemiology*, 41(3), 210-220.
- Zhao, H., Nettleton, D., & Dekkers, J. C. (2007). Evaluation of linkage disequilibrium measures between multi-allelic markers as predictors of linkage disequilibrium between single nucleotide polymorphisms. *Genetics Research*, 89(1), 1-6.
- Zhou, D. X. (2020). Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48(2), 787-794.
- Zhou, Y. H., & Gallins, P. (2019). A review and tutorial of machine learning methods for microbiome host trait prediction. *Frontiers in Genetics*, 10, 579.
- Zingaretti, L. M., Renand, G., Morgavi, D. P., & Ramayo-Caldas, Y. (2020). Link-HD: a versatile framework to explore and integrate heterogeneous microbial communities. *Bioinformatics*, 36(7), 2298-2299.

ANNEXES

Supplementary material Chapter 4

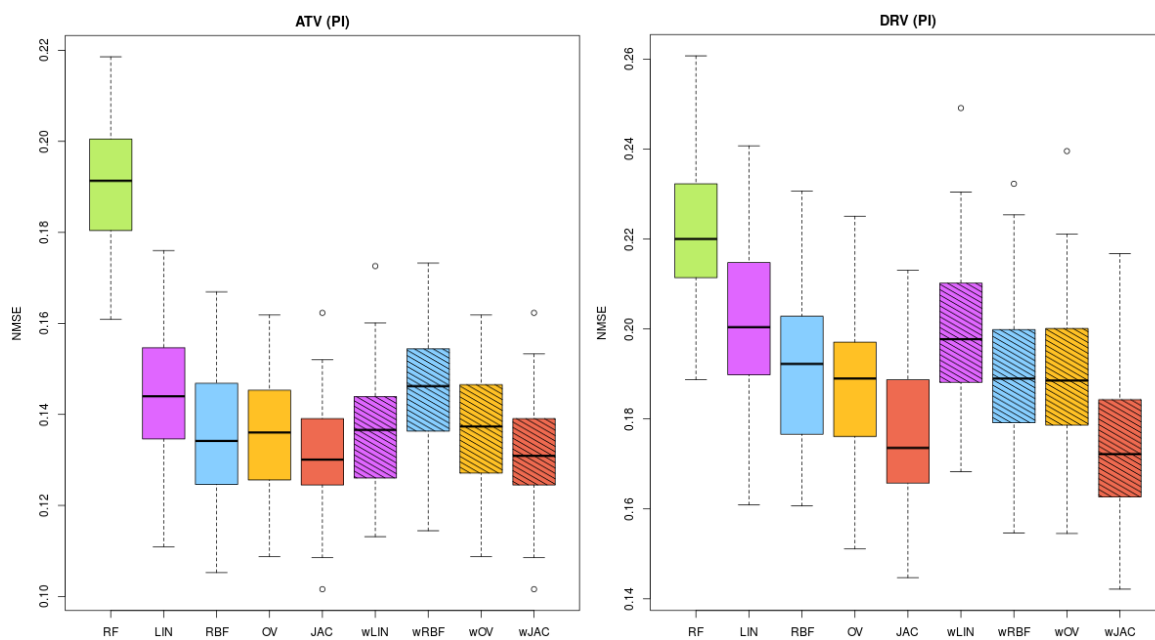


Figure S1. NMSE distribution for ATV and DRV (protease inhibitors). Same legend as that of Figure 4.1.

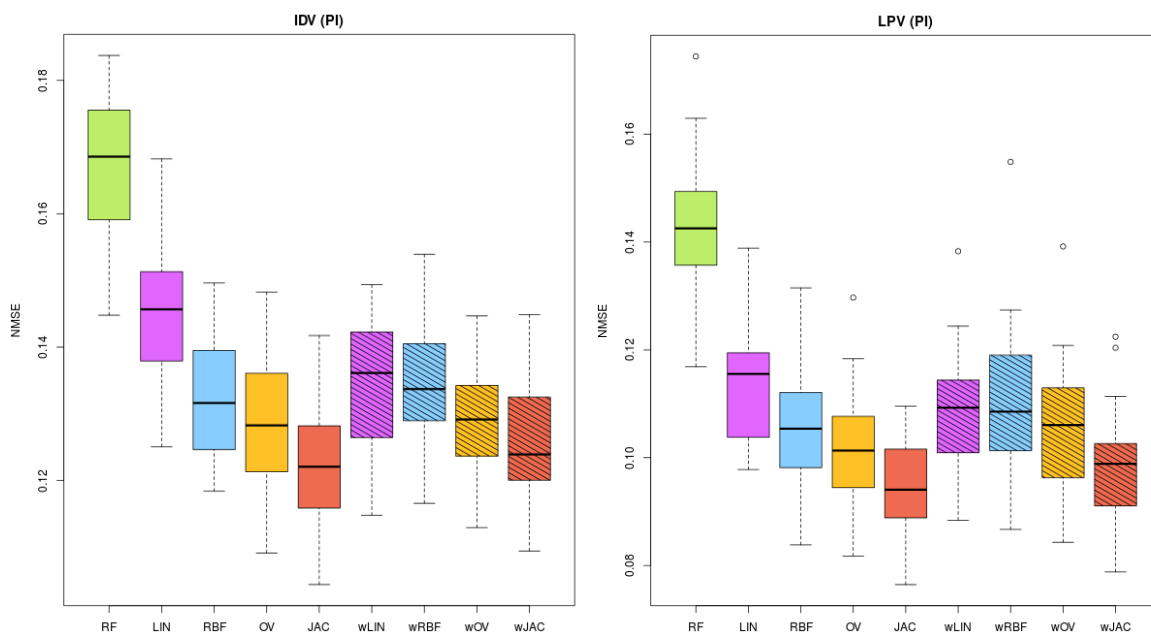


Figure S2. NMSE distribution for IDV and LPV (protease inhibitors). Same legend as that of Figure 4.1.

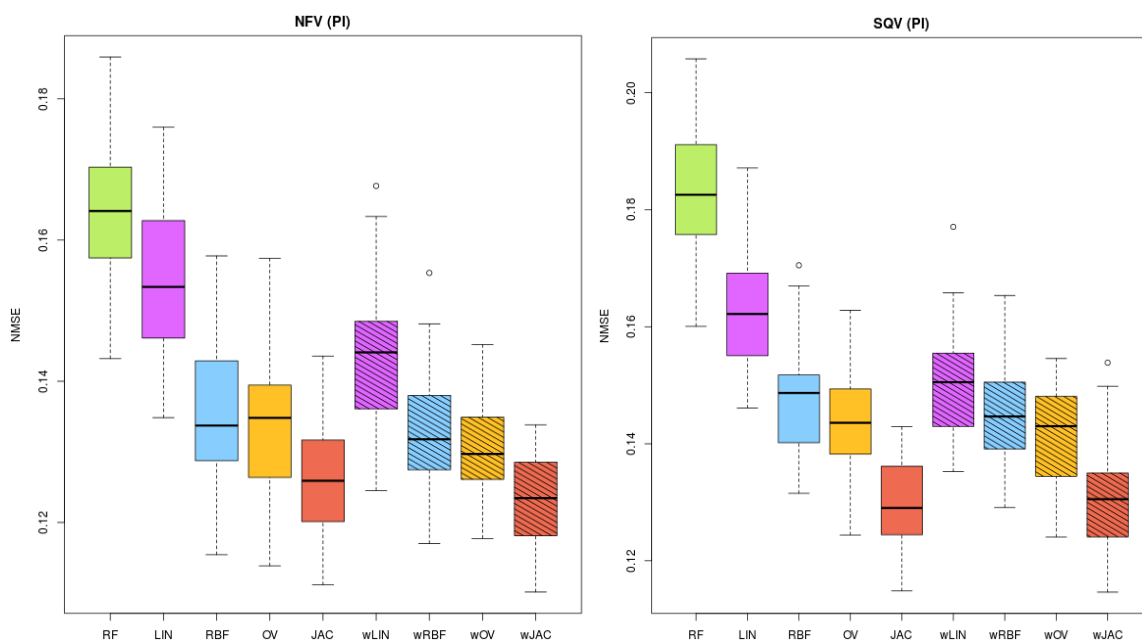


Figure S3. NMSE distribution for NFV and SQV (protease inhibitors). Same legend as that of Figure 4.1.

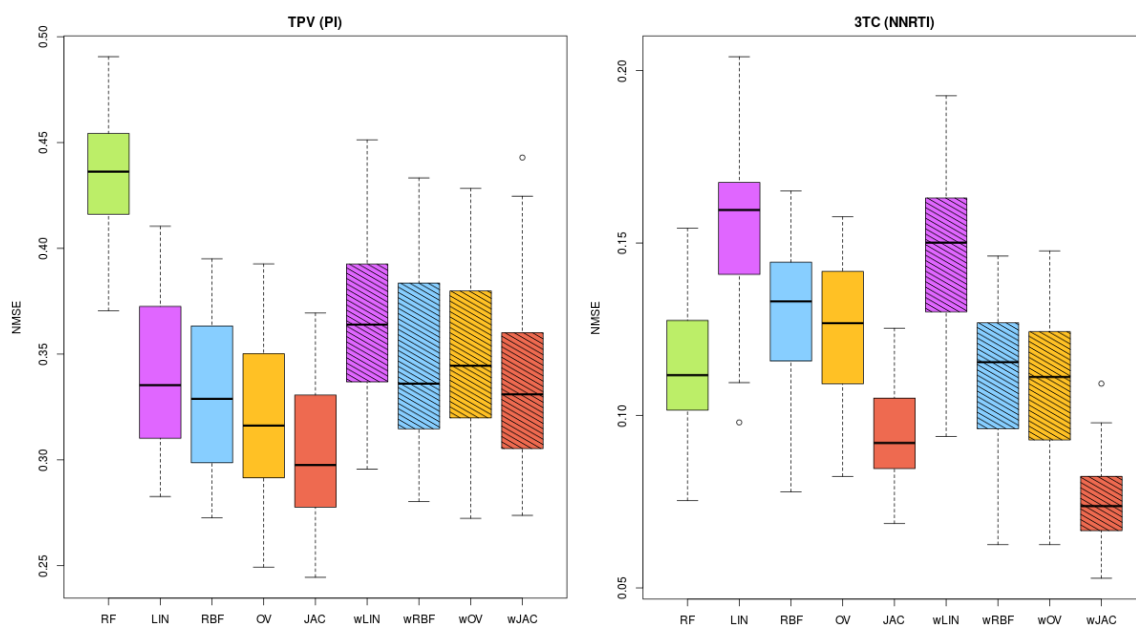


Figure S4. NMSE distribution for TPV (protease inhibitor) and 3TC (reverse transcriptase inhibitor). Same legend as that of Figure 4.1.

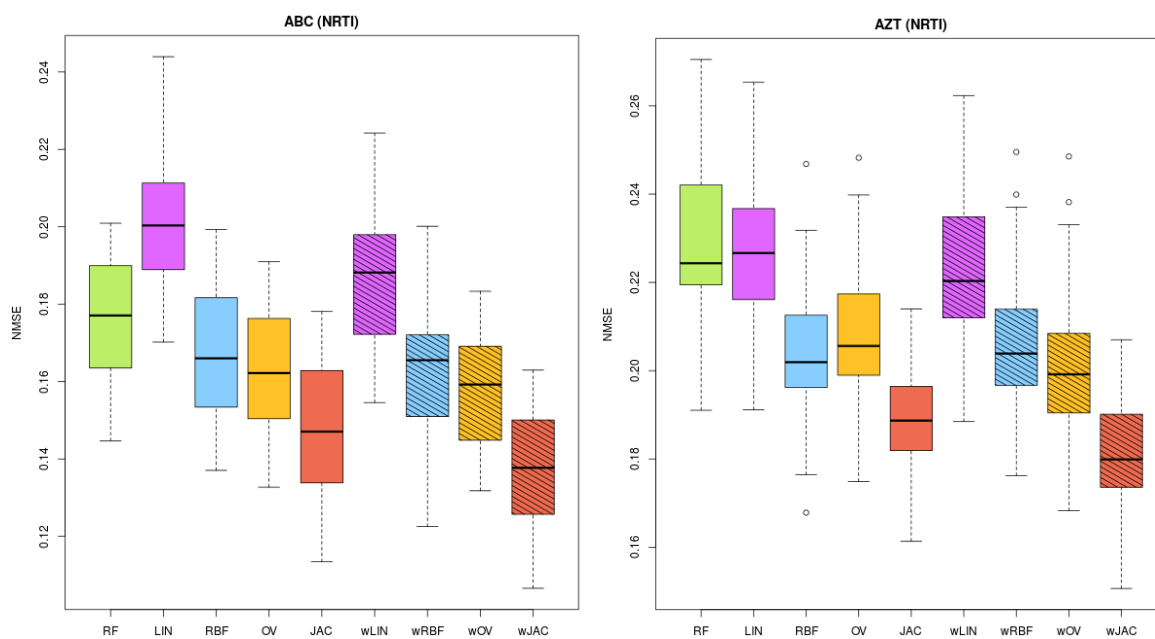


Figure S5. NMSE distribution for ABC and AZT (reverse transcriptase inhibitors). Same legend as that of Figure 4.1.

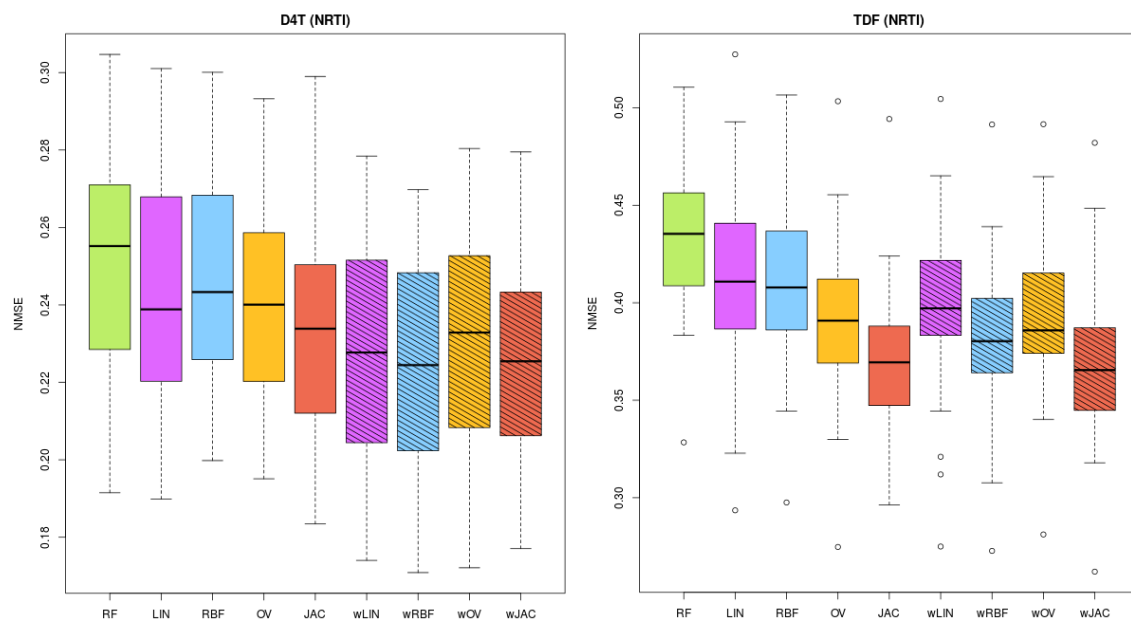


Figure S6. NMSE distribution for D4T and TDF (reverse transcriptase inhibitors). Same legend as that of Figure 4.1.

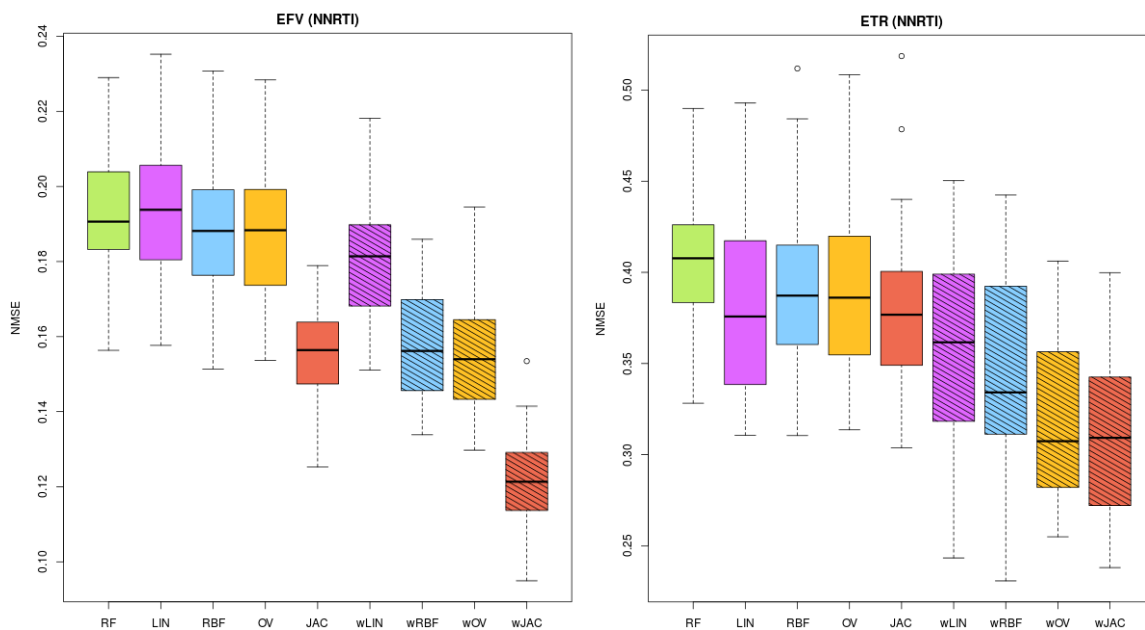


Figure S7. NMSE distribution for EFV and ETR (reverse transcriptase inhibitors). Same legend as that of Figure 4.1.

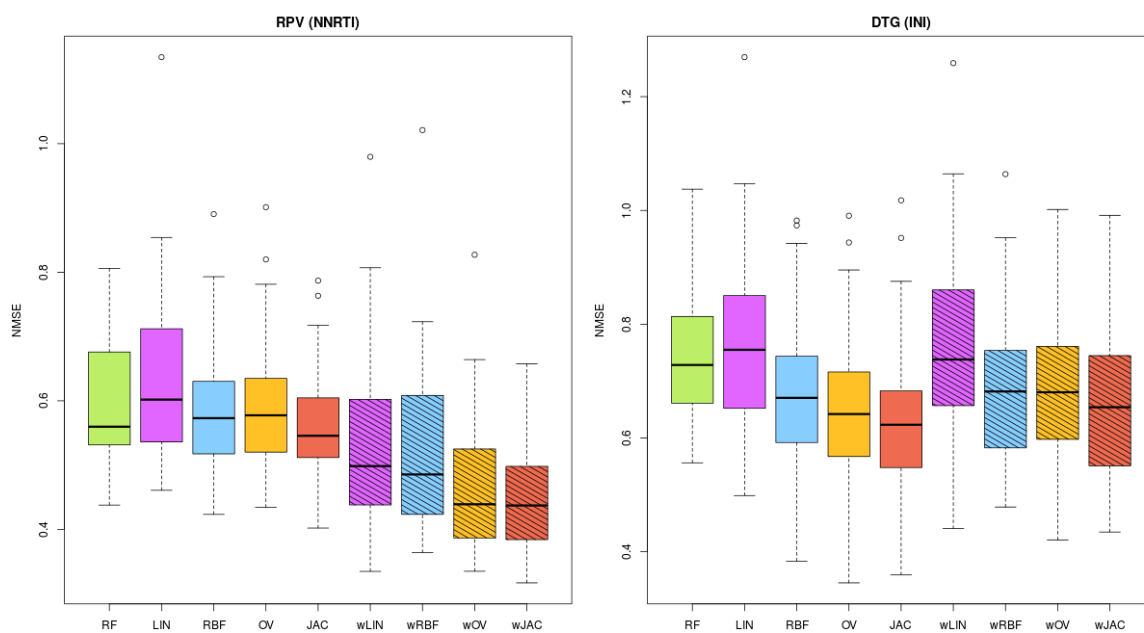


Figure S8. NMSE distribution for RPV (reverse transcriptase inhibitor) and DTG (integrase inhibitor). Same legend as that of Figure 4.1.

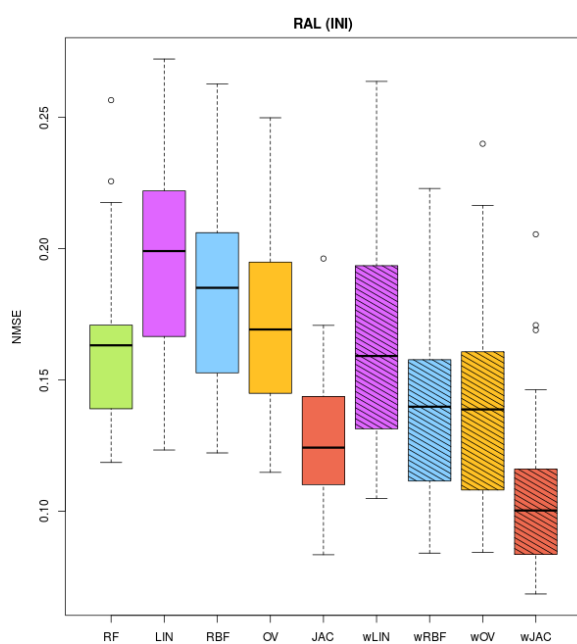


Figure S9. NMSE distribution for RAL (integrase inhibitor). Same legend as that of Figure 4.1.

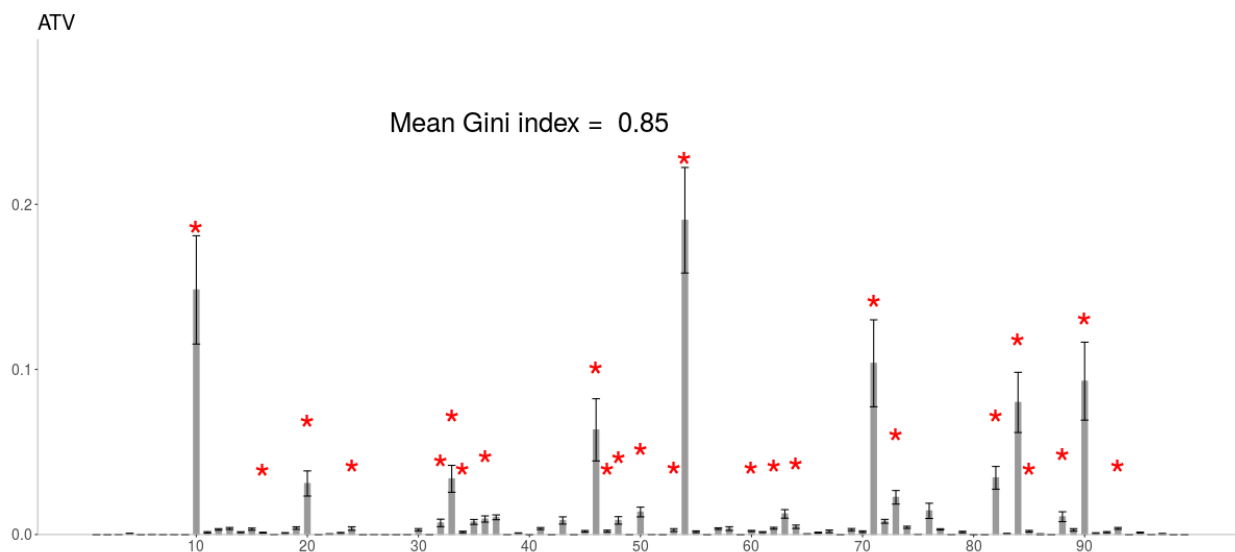


Figure S10. RF relative importance of each protein position, averaged over 40 replicates, for ATV (protease inhibitor). Asterisks mark the major drug-related positions reported in the literature.

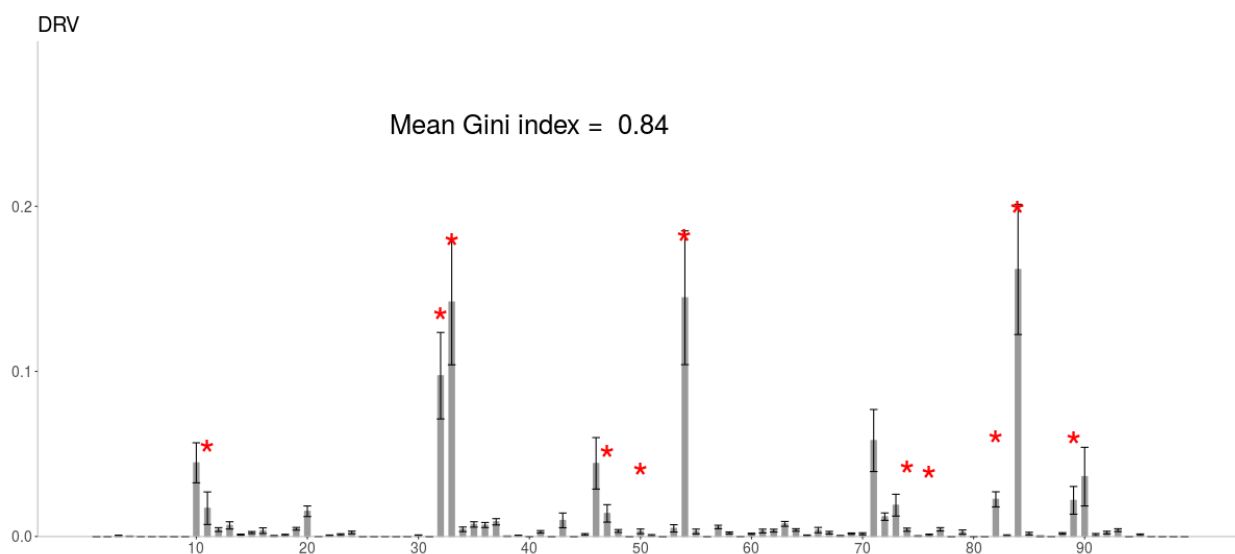


Figure S11. RF relative importance of each protein position, averaged over 40 replicates, for DRV (protease inhibitor). Asterisks mark the major drug-related positions reported in the literature.

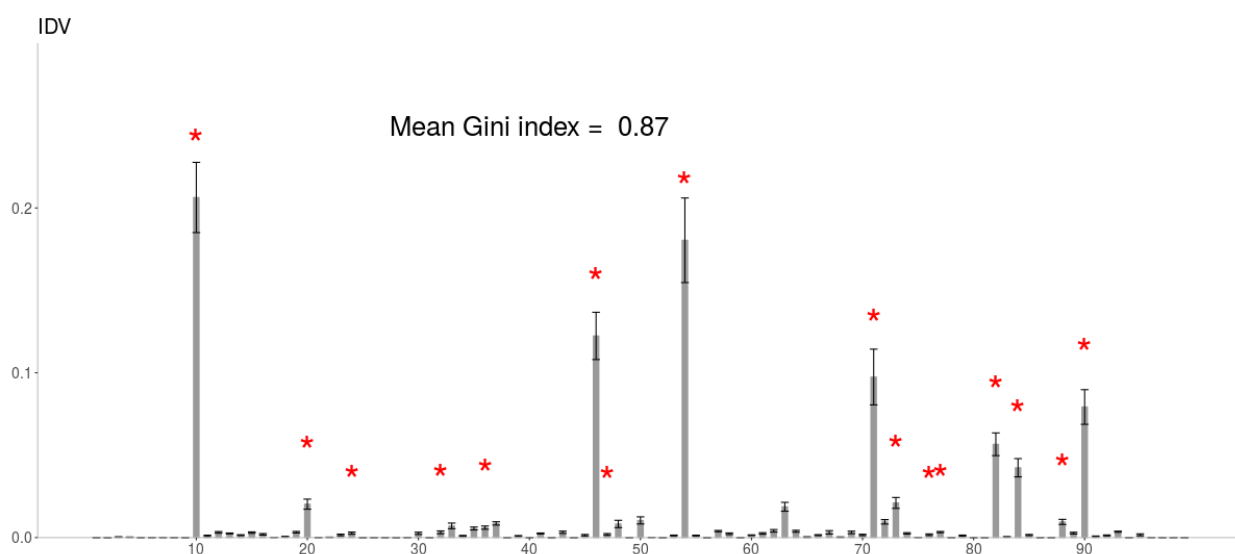


Figure S12. RF relative importance of each protein position, averaged over 40 replicates, for IDV (protease inhibitor). Asterisks mark the major drug-related positions reported in the literature.

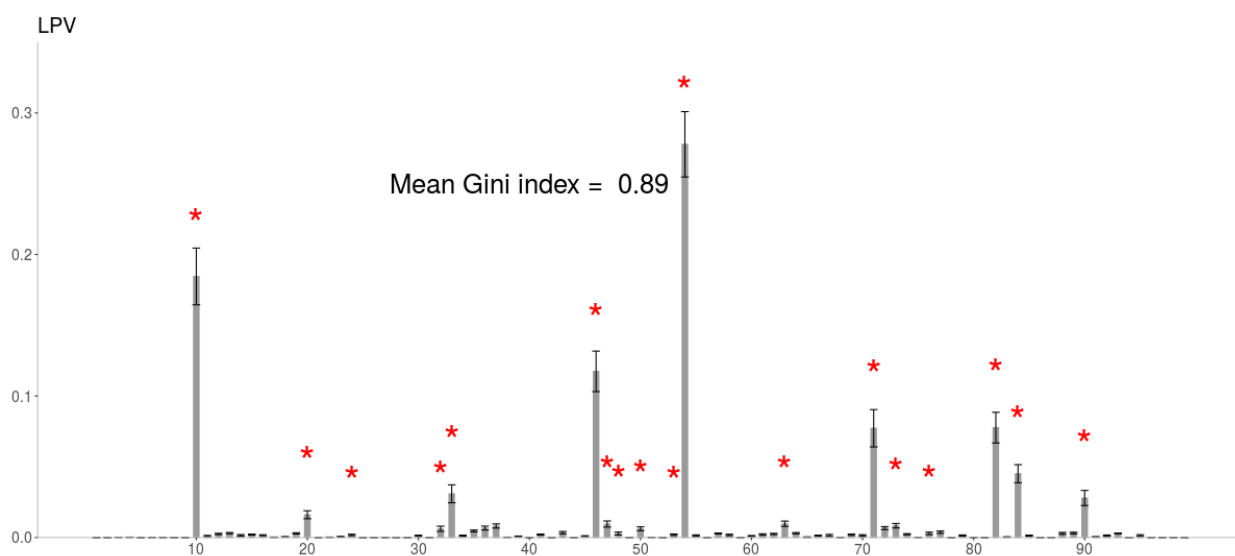


Figure S13. RF relative importance of each protein position, averaged over 40 replicates, for LPV (protease inhibitor). Asterisks mark the major drug-related positions reported in the literature.

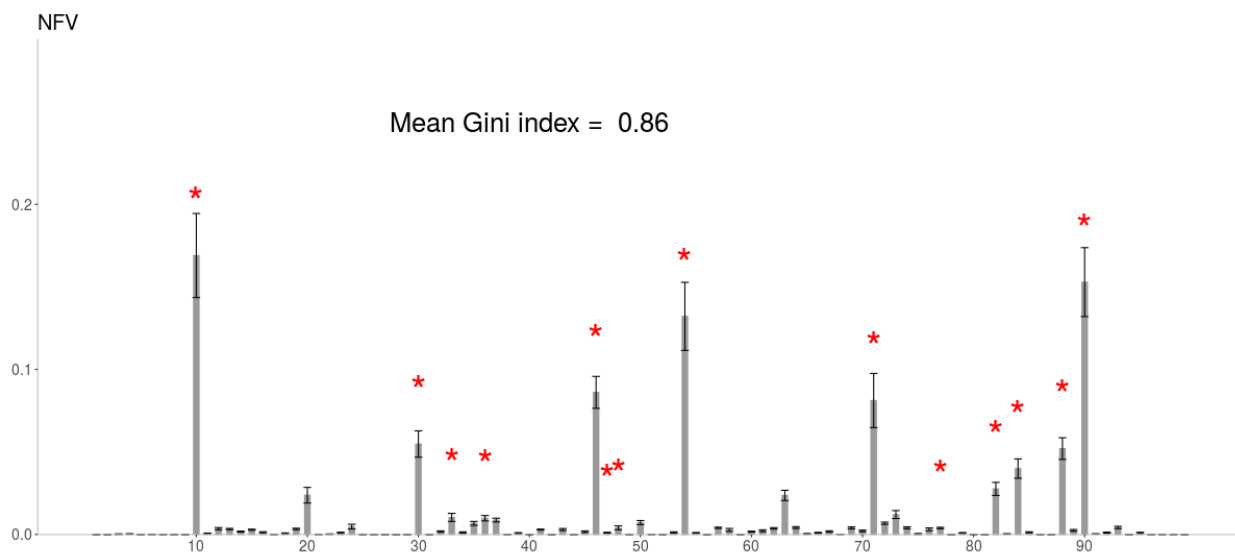


Figure S14. RF relative importance of each protein position, averaged over 40 replicates, for NFV (protease inhibitor). Asterisks mark the major drug-related positions reported in the literature.

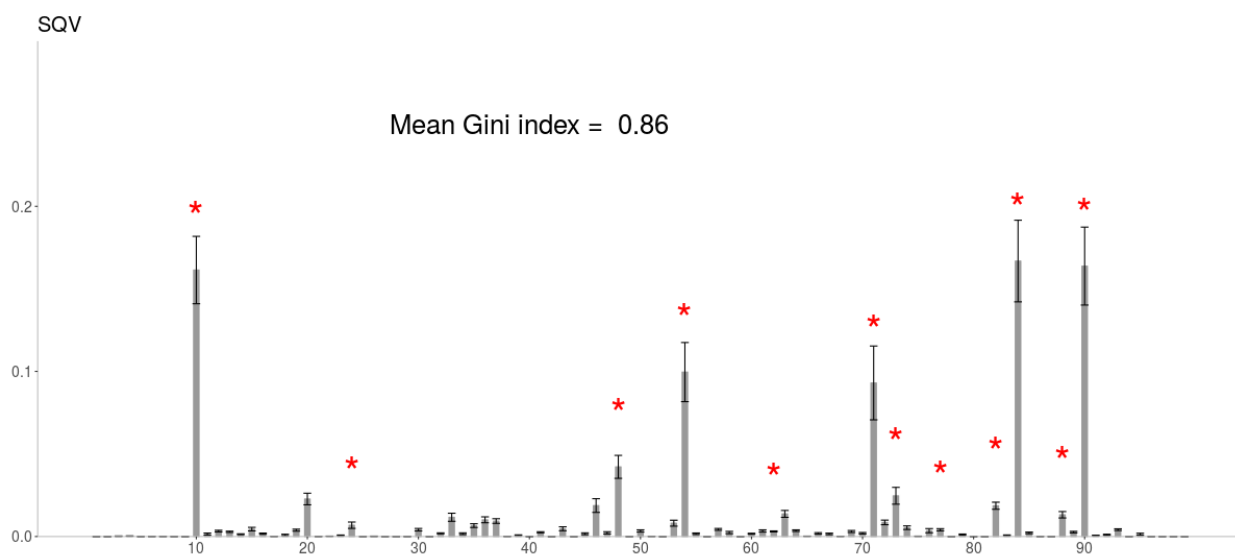


Figure S15. RF relative importance of each protein position, averaged over 40 replicates, for SQV (protease inhibitor). Asterisks mark the major drug-related positions reported in the literature.

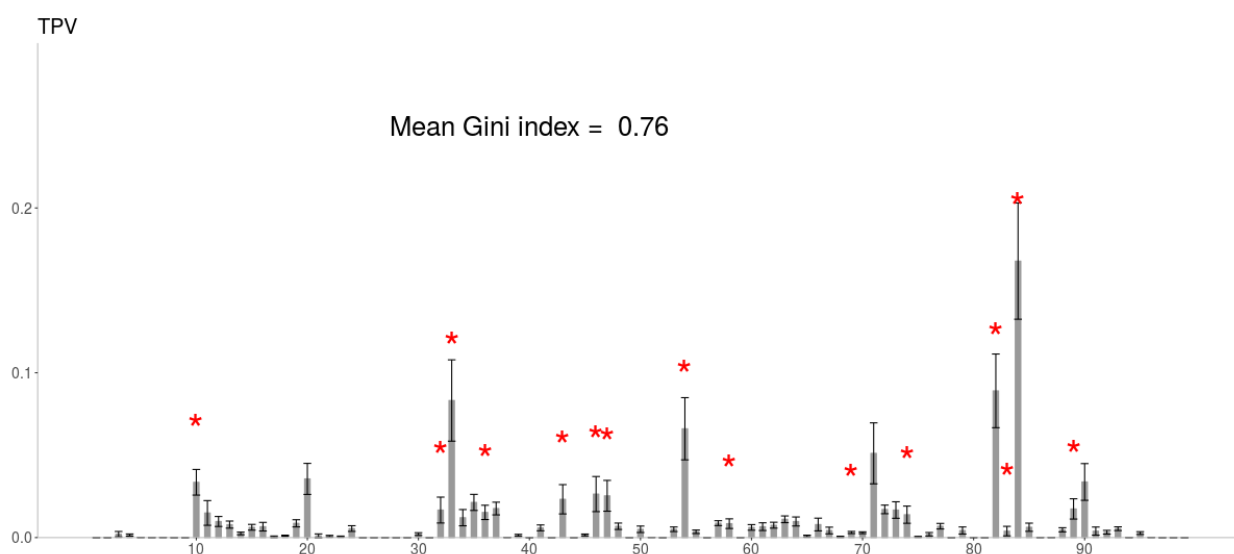


Figure S16. RF relative importance of each protein position, averaged over 40 replicates, for TPV (protease inhibitor). Asterisks mark the major drug-related positions reported in the literature.

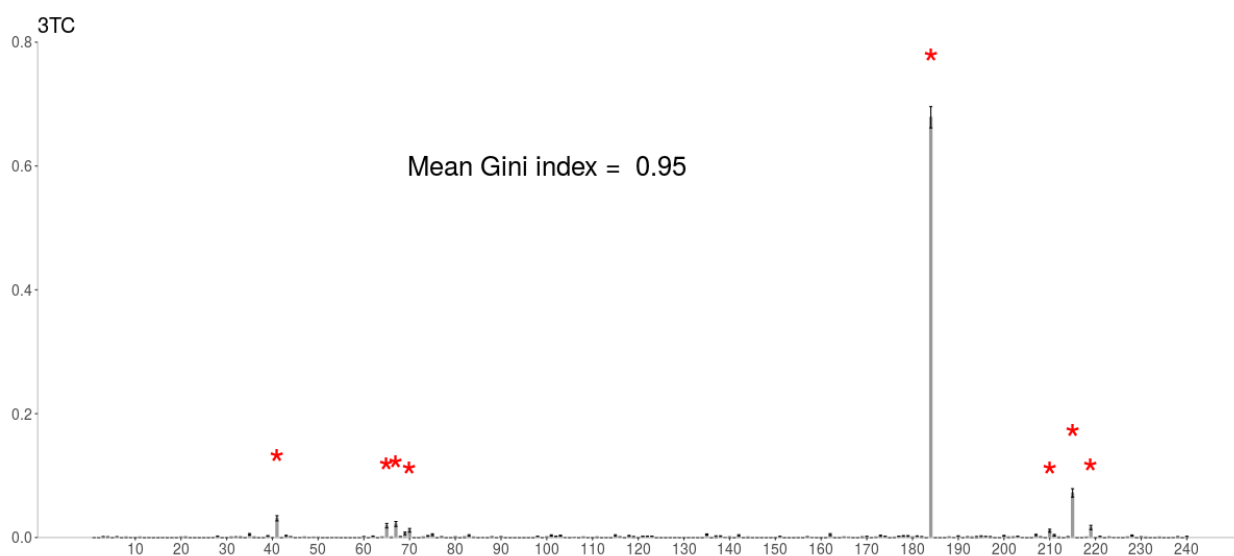


Figure S17. RF relative importance of each protein position, averaged over 40 replicates, for 3TC (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

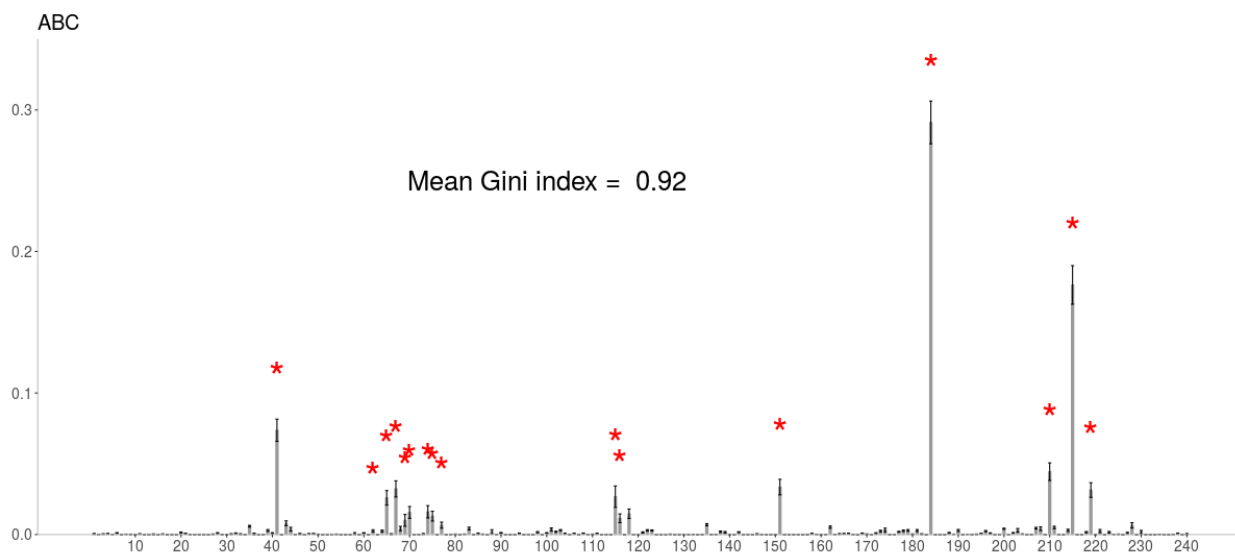


Figure S18. RF relative importance of each protein position, averaged over 40 replicates, for ABC (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

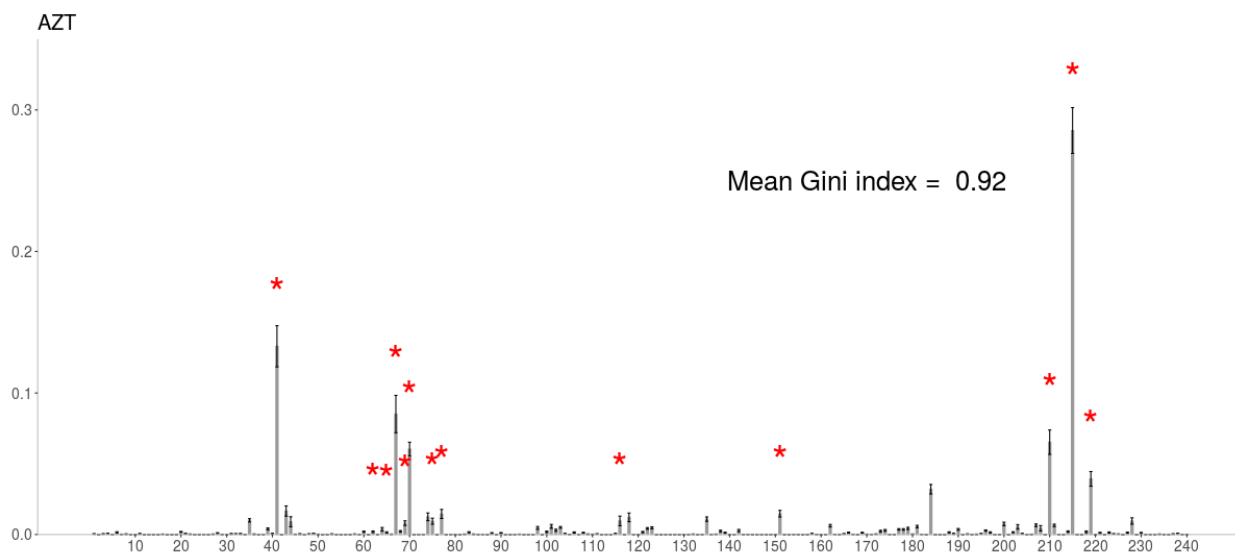


Figure S19. RF relative importance of each protein position, averaged over 40 replicates, for AZT (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

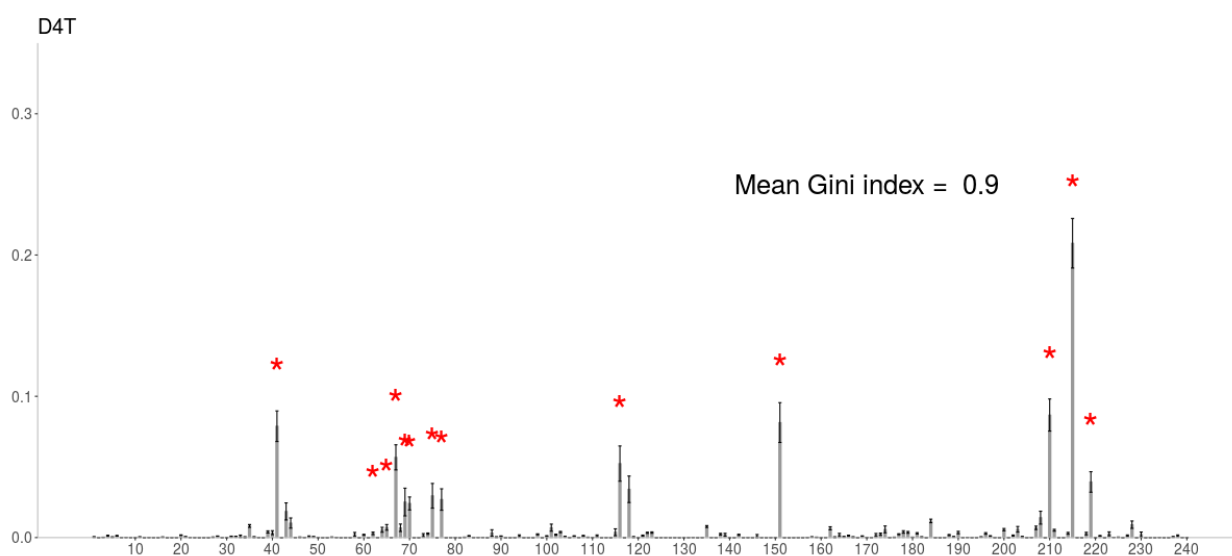


Figure S20. RF relative importance of each protein position, averaged over 40 replicates, D4T (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

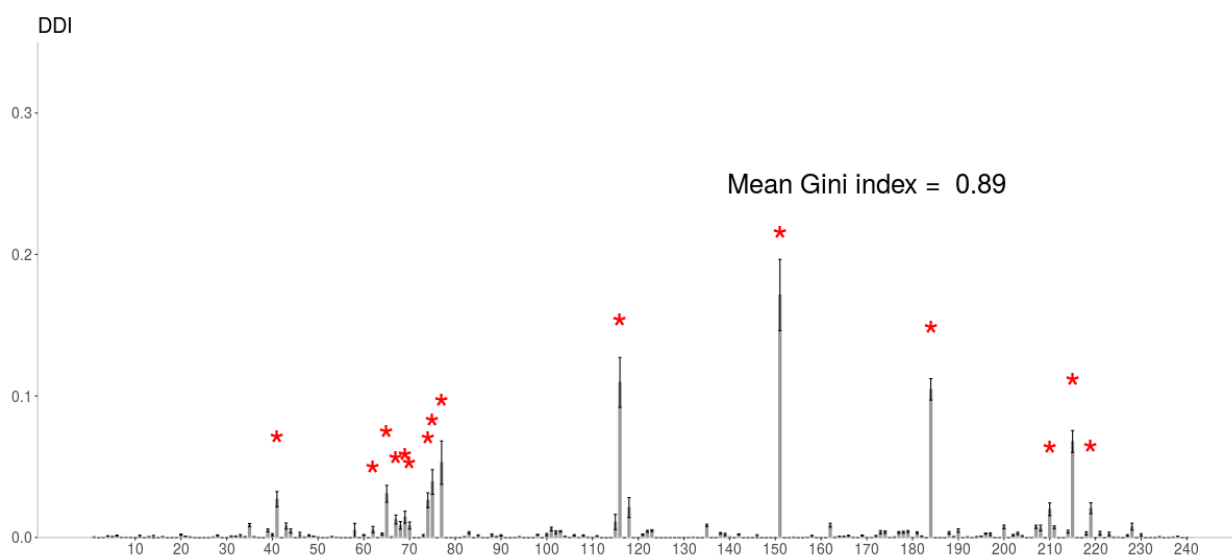


Figure S21. RF relative importance of each protein position, averaged over 40 replicates, for DDI (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

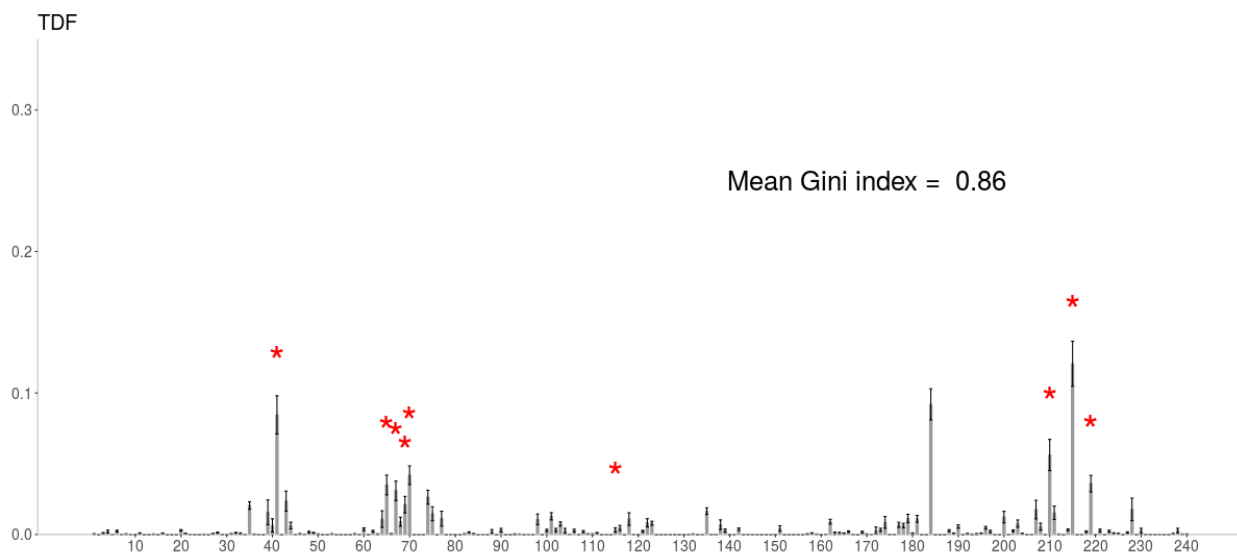


Figure S22. RF relative importance of each protein position, averaged over 40 replicates, for TDF (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

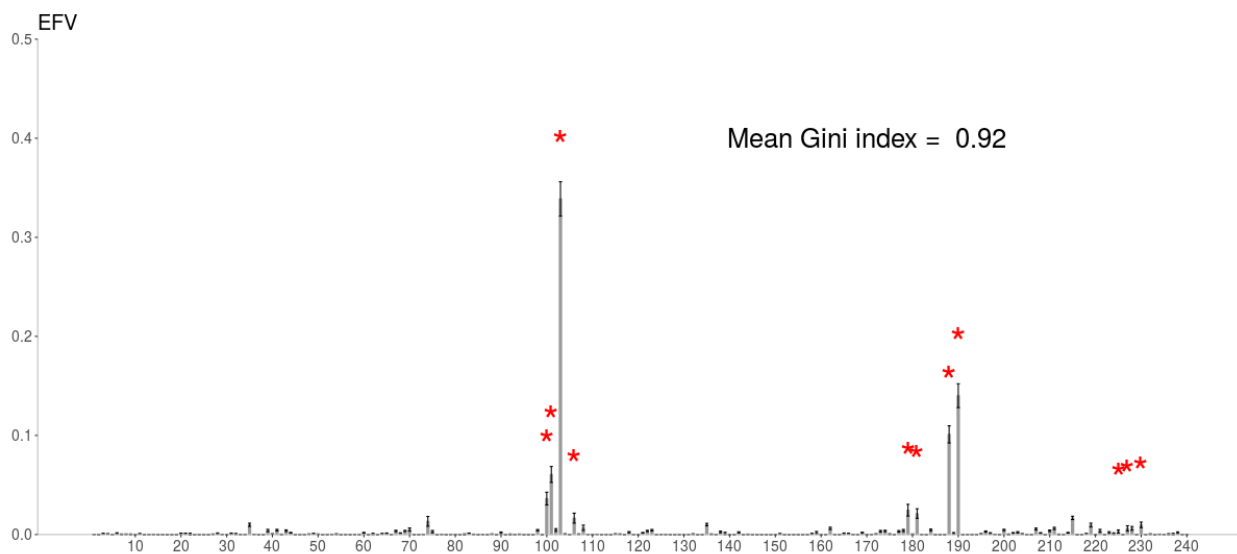


Figure S23. RF relative importance of each protein position, averaged over 40 replicates, for EFV (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

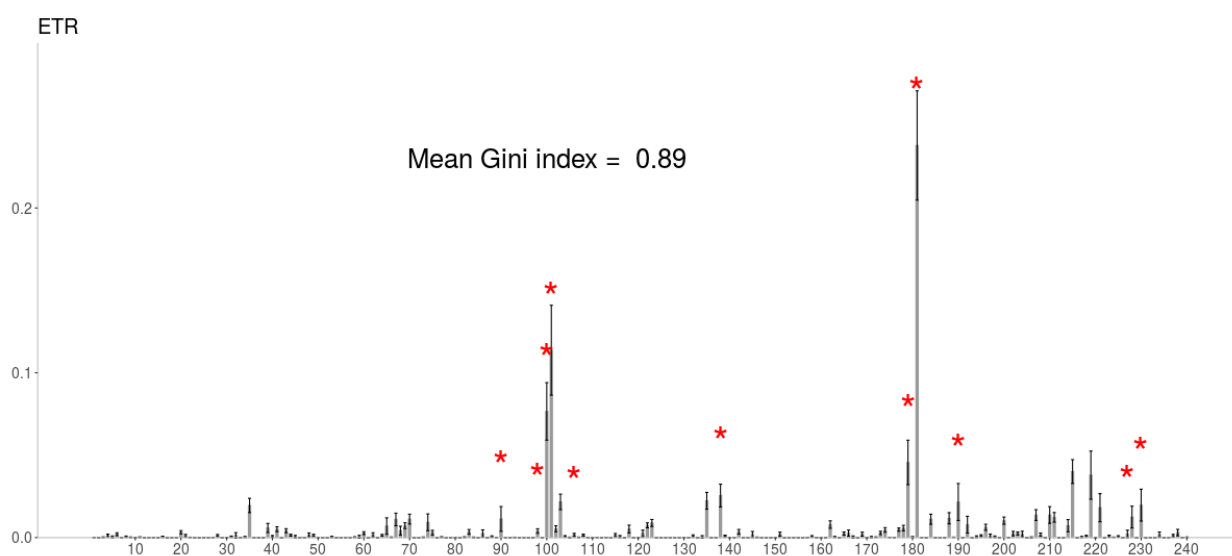


Figure S24. RF relative importance of each protein position, averaged over 40 replicates, for ETR (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

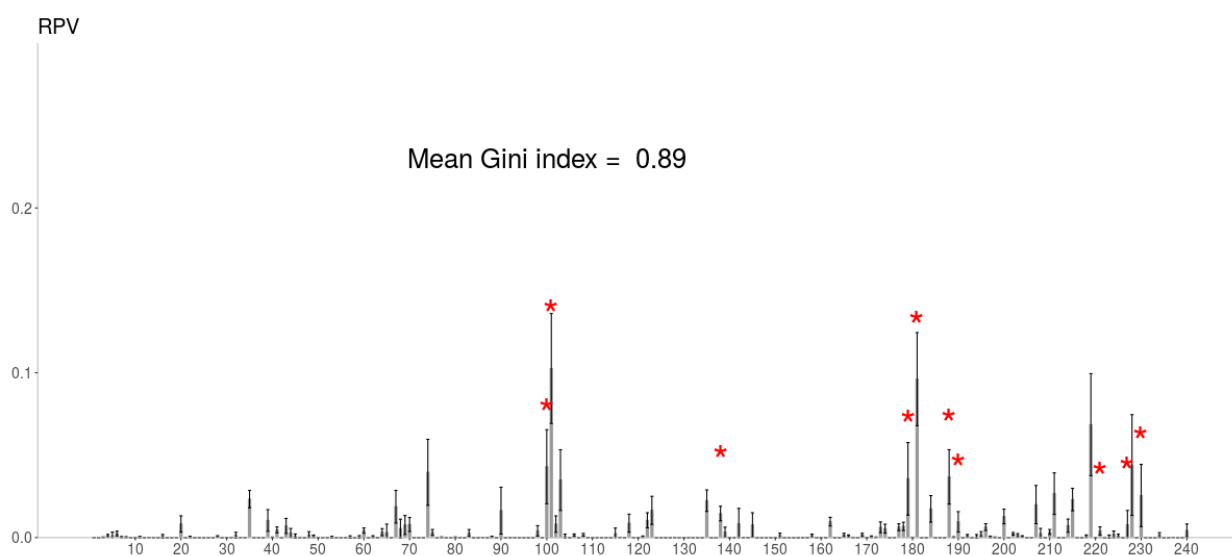


Figure S25. RF relative importance of each protein position, averaged over 40 replicates, for RPV (reverse transcriptase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

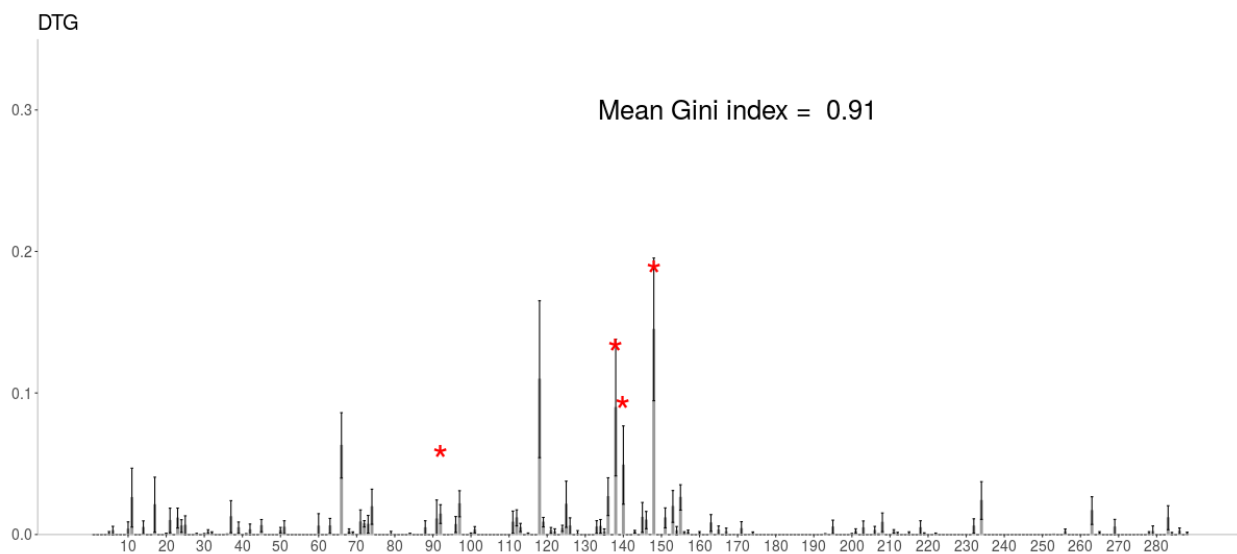


Figure S26. RF relative importance of each protein position, averaged over 40 replicates, for DTG (integrase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

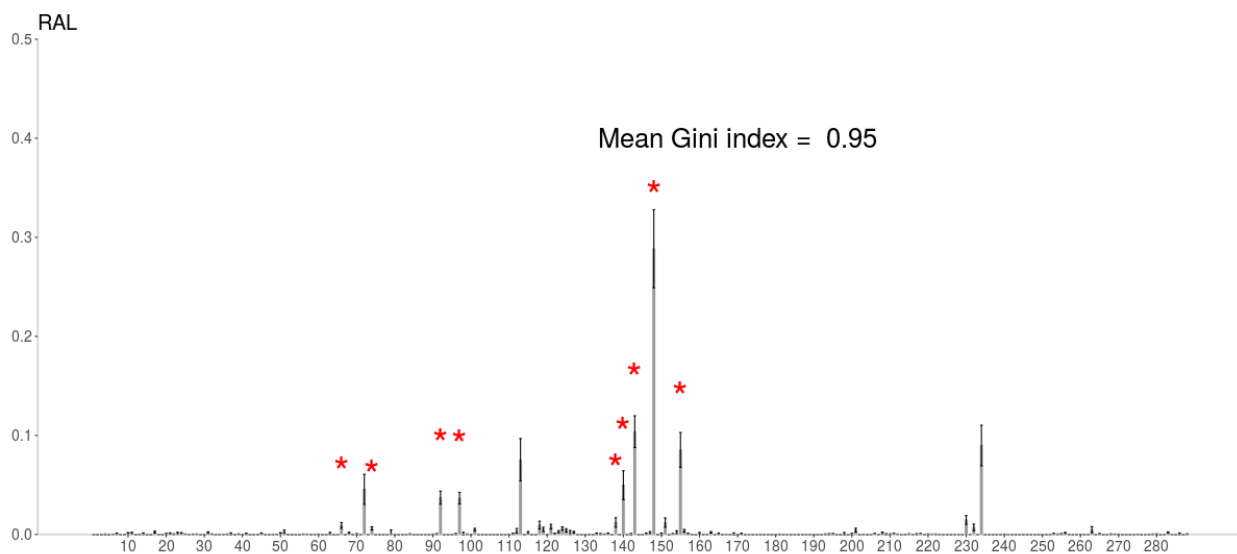


Figure S27. RF relative importance of each protein position, averaged over 40 replicates, for RAL (integrase inhibitor). Asterisks mark the major drug-related positions reported in the literature.

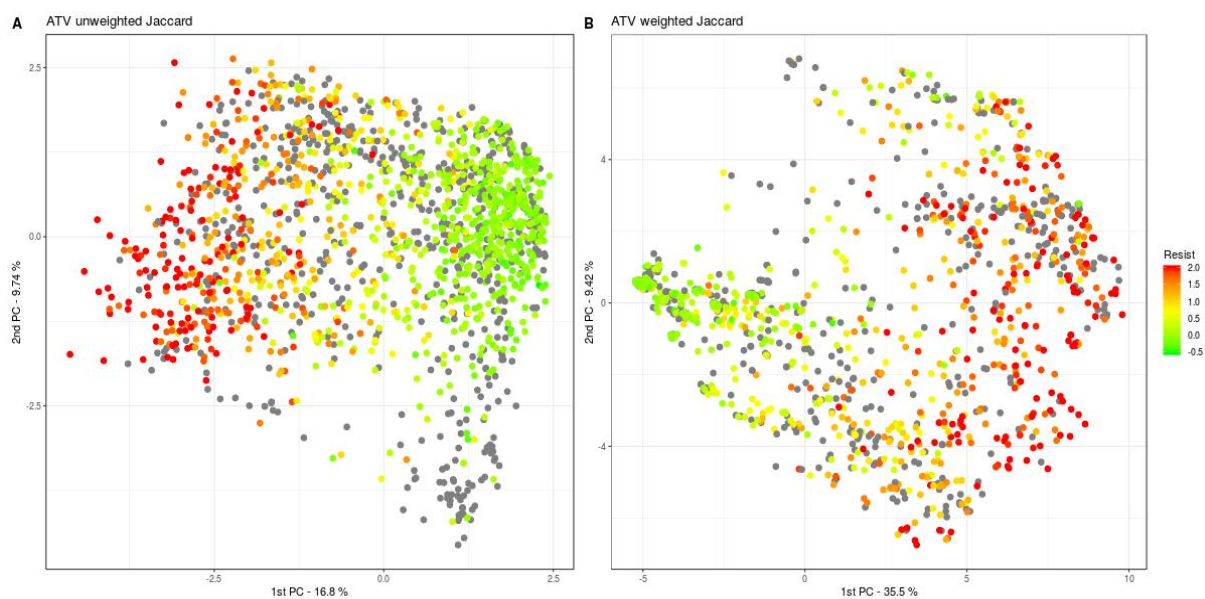


Figure S28. The unweighted (A) and weighted (B) Jaccard kPCA for ATV (protease inhibitor). Gray dots represent sequences with missing resistance value.

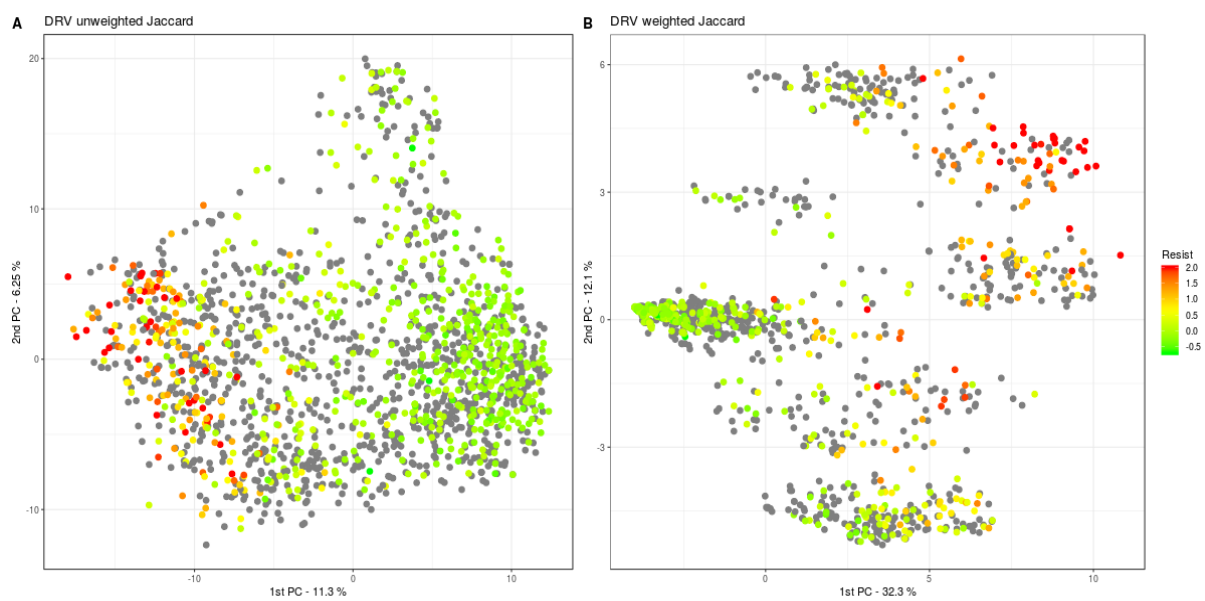


Figure S29. The unweighted (A) and weighted (B) Jaccard kPCA for DRV (protease inhibitor). Gray dots represent sequences with missing resistance value.

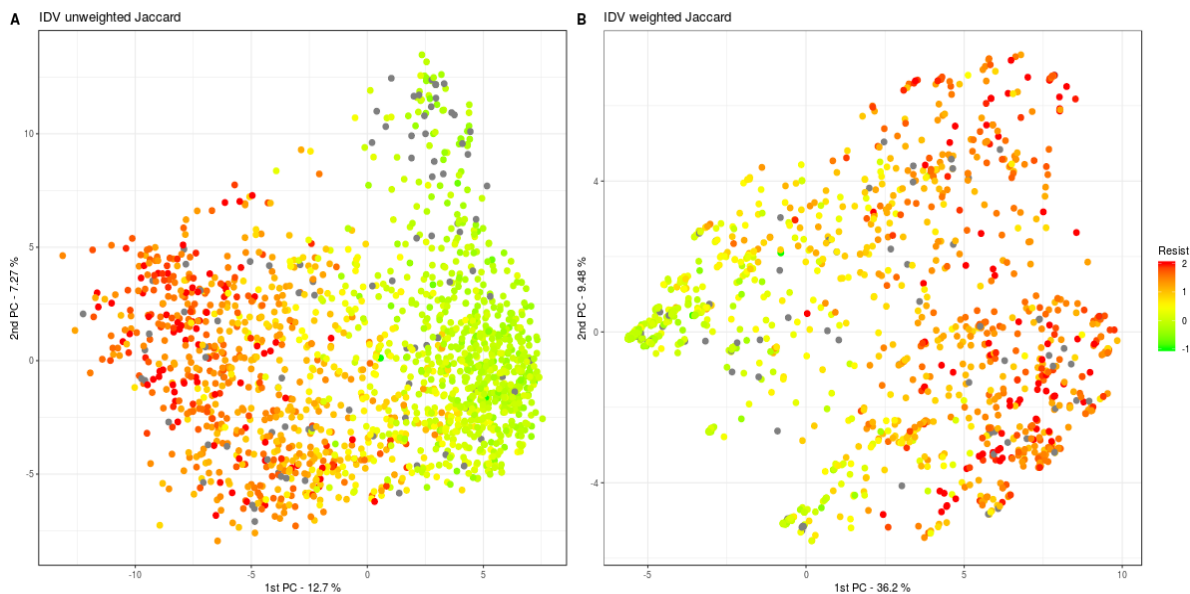


Figure S30. The unweighted (A) and weighted (B) Jaccard kPCA for IDV (protease inhibitor). Gray dots represent sequences with missing resistance value.

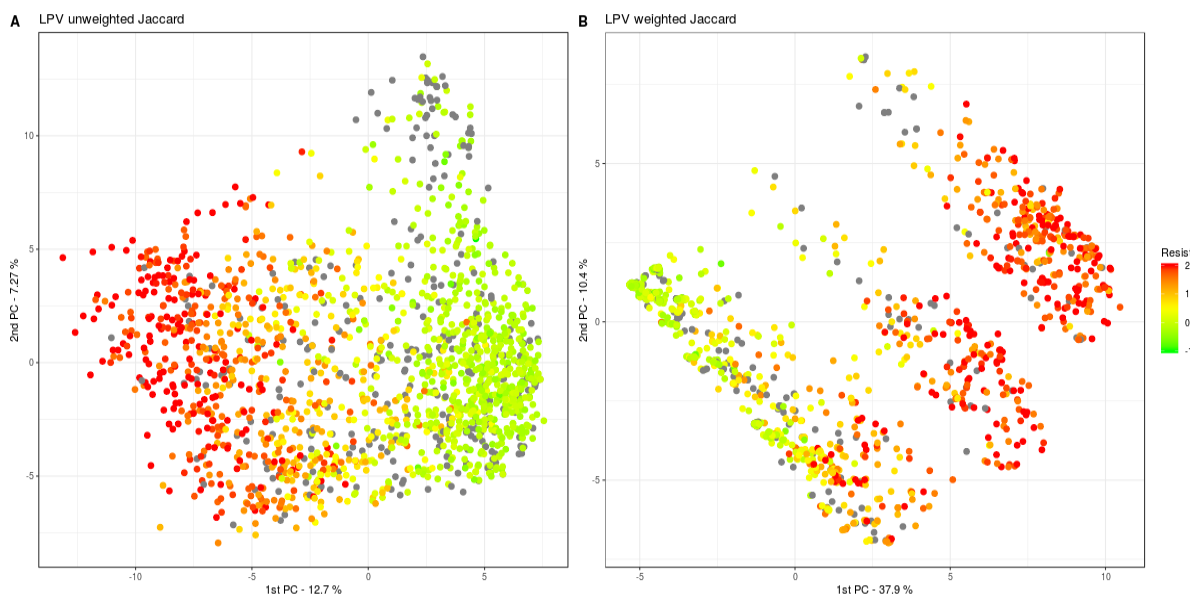


Figure S31. The unweighted (A) and weighted (B) Jaccard kPCA for LPV (protease inhibitor). Gray dots represent sequences with missing resistance value.

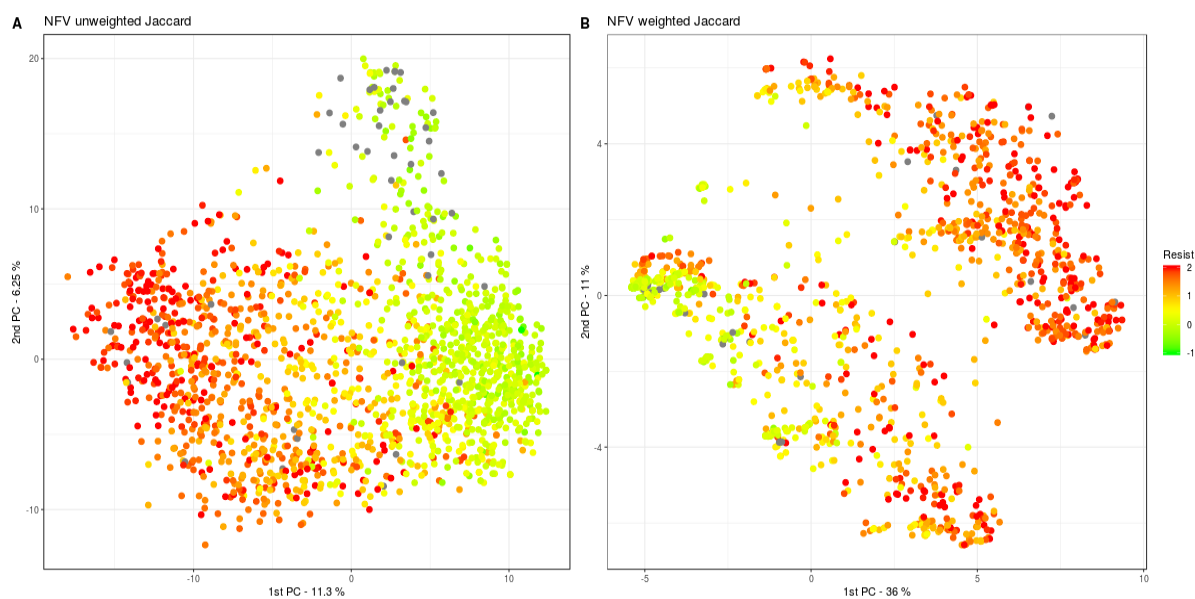


Figure S32. The unweighted (A) and weighted (B) Jaccard kPCA for NFV (protease inhibitor). Gray dots represent sequences with missing resistance value.

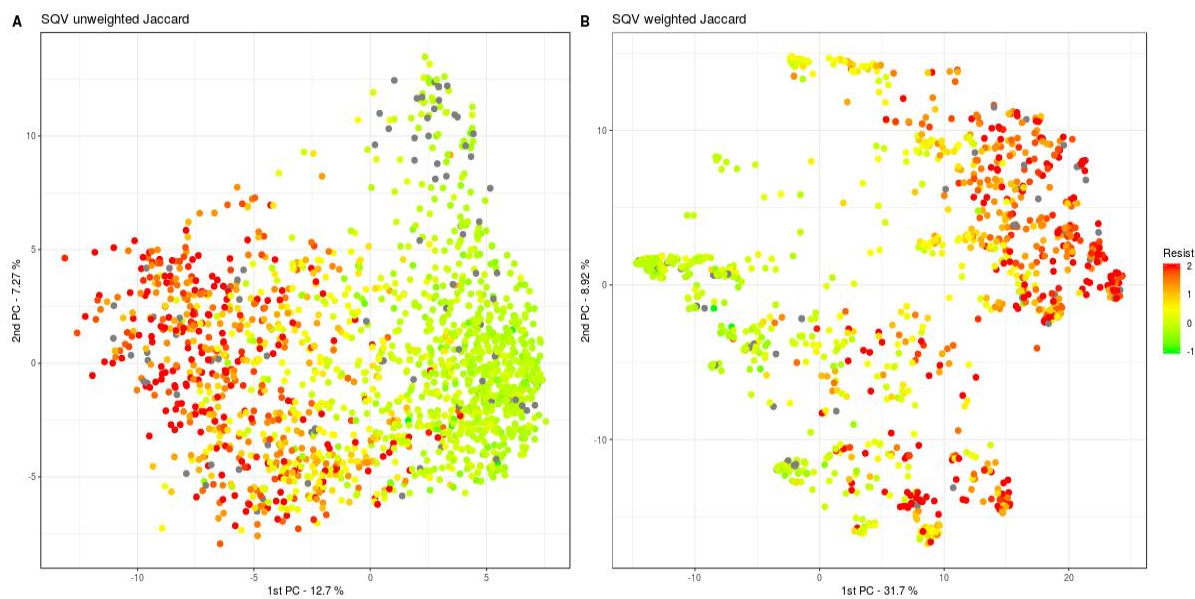


Figure S33. The unweighted (A) and weighted (B) Jaccard kPCA for SQV (protease inhibitor). Gray dots represent sequences with missing resistance value.

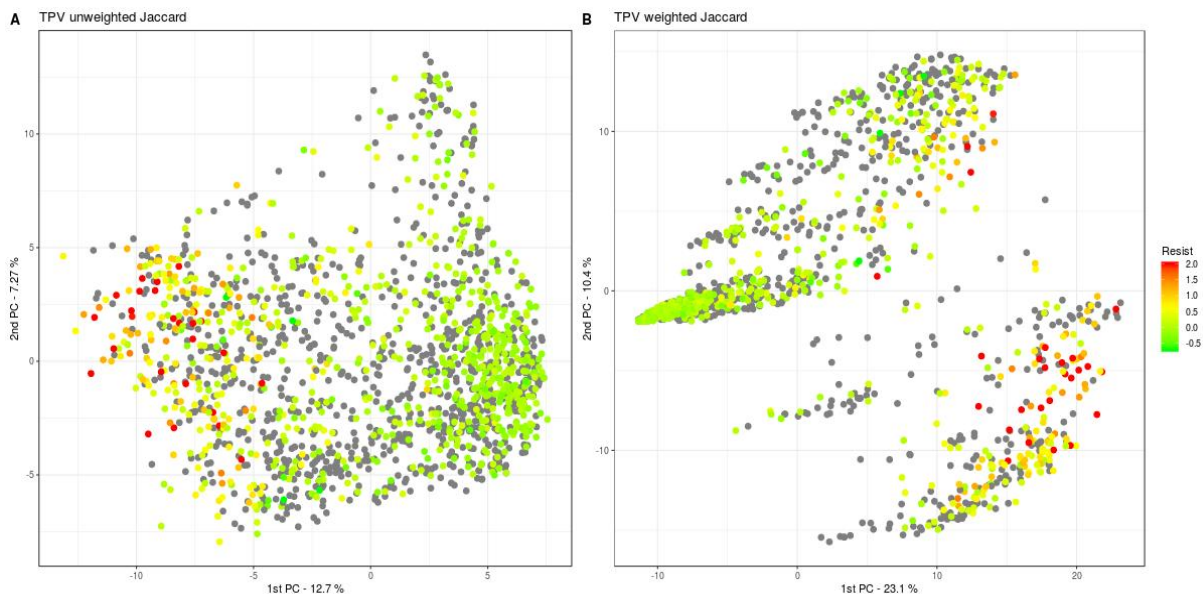


Figure S34. The unweighted (A) and weighted (B) Jaccard kPCA for TPV (protease inhibitor). Gray dots represent sequences with missing resistance value.

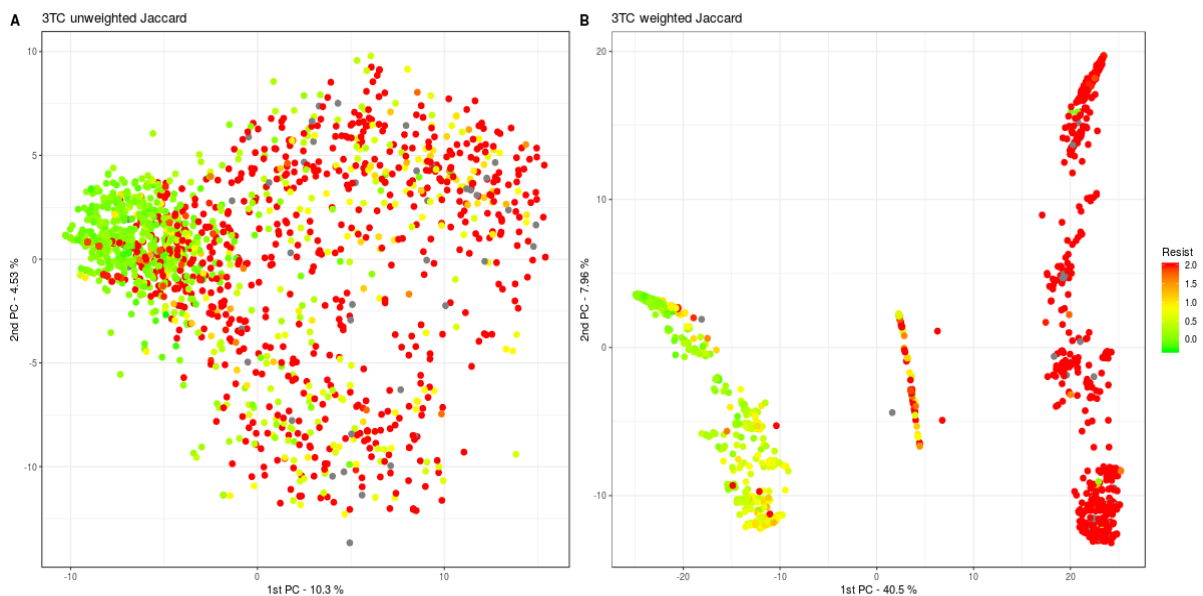


Figure S35. The unweighted (A) and weighted (B) Jaccard kPCA for 3TC (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

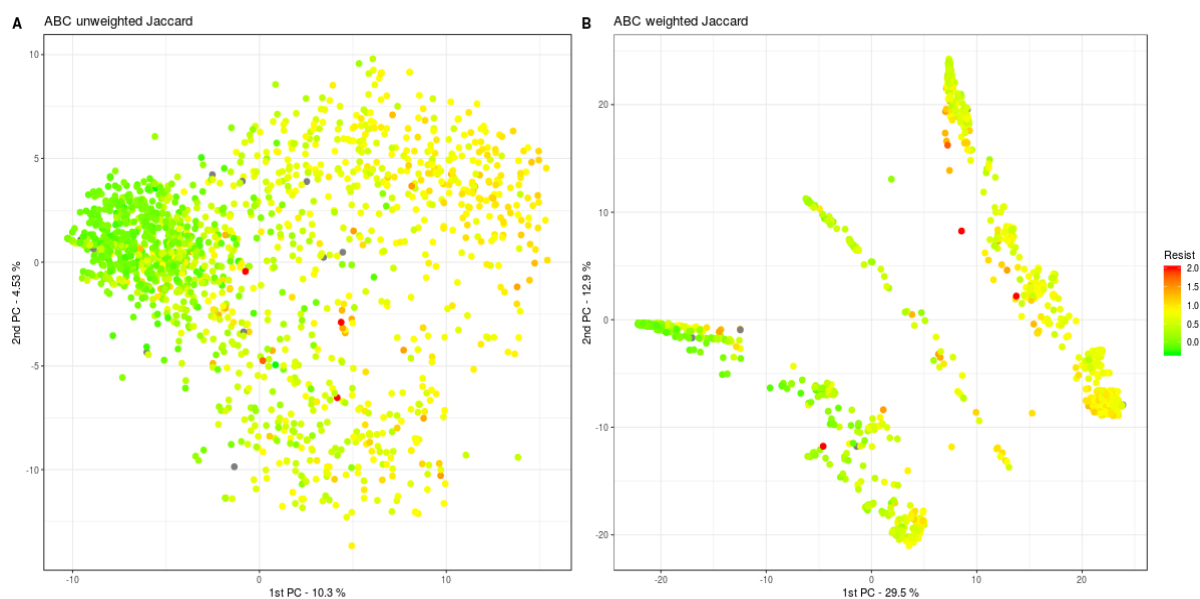


Figure S36. The unweighted (A) and weighted (B) Jaccard kPCA for ABC (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

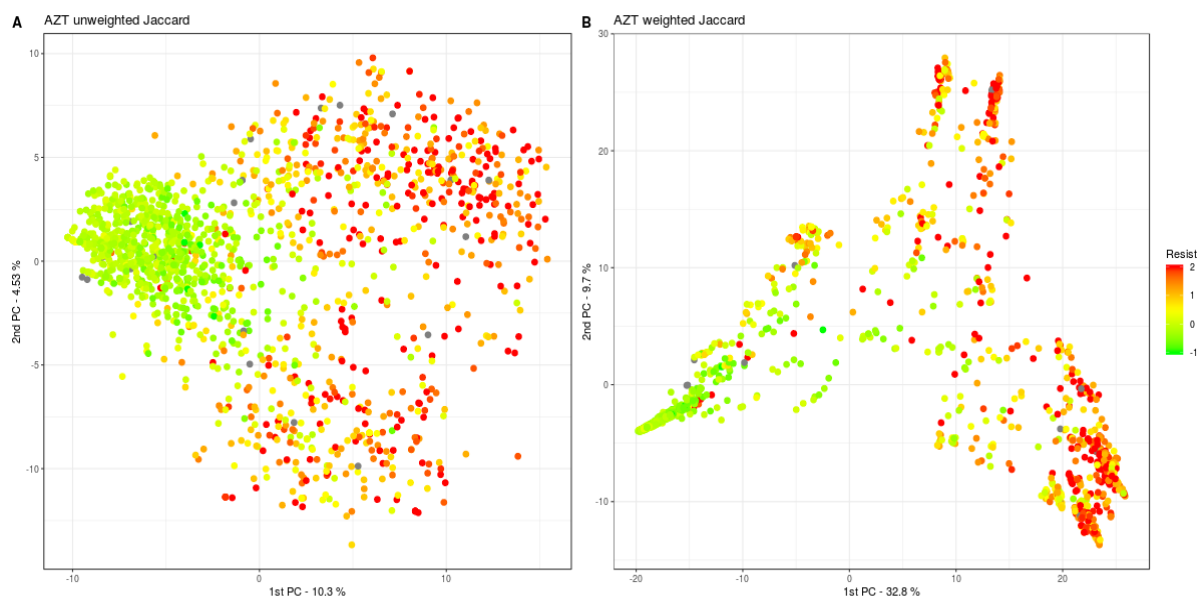


Figure S37. The unweighted (A) and weighted (B) Jaccard kPCA for AZT (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

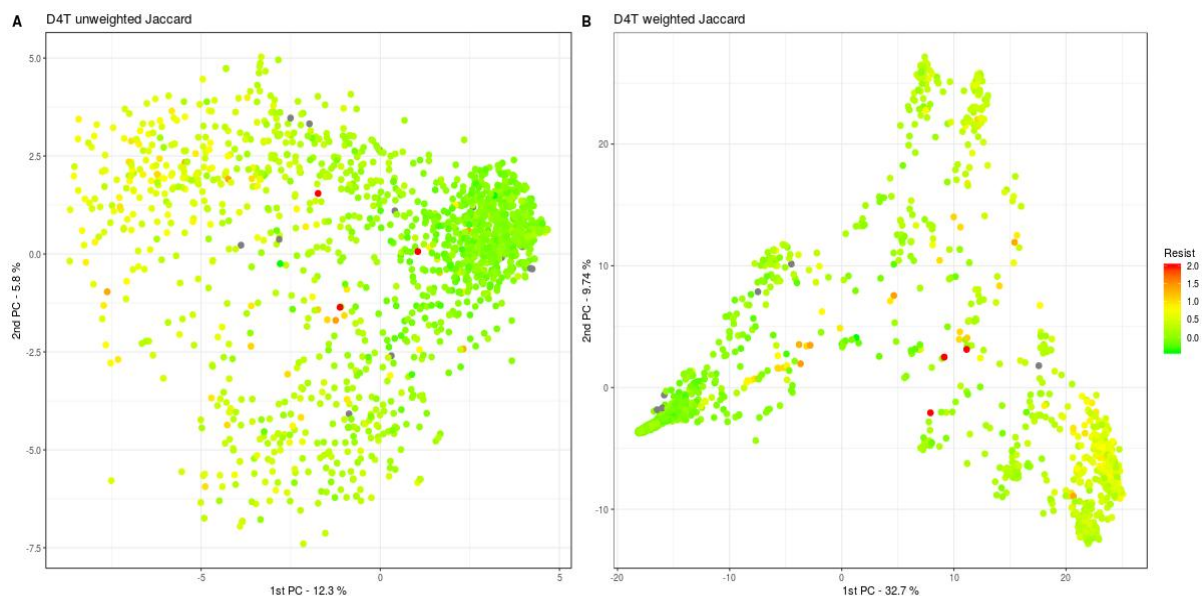


Figure S38. The unweighted (A) and weighted (B) Jaccard kPCA for D4T (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

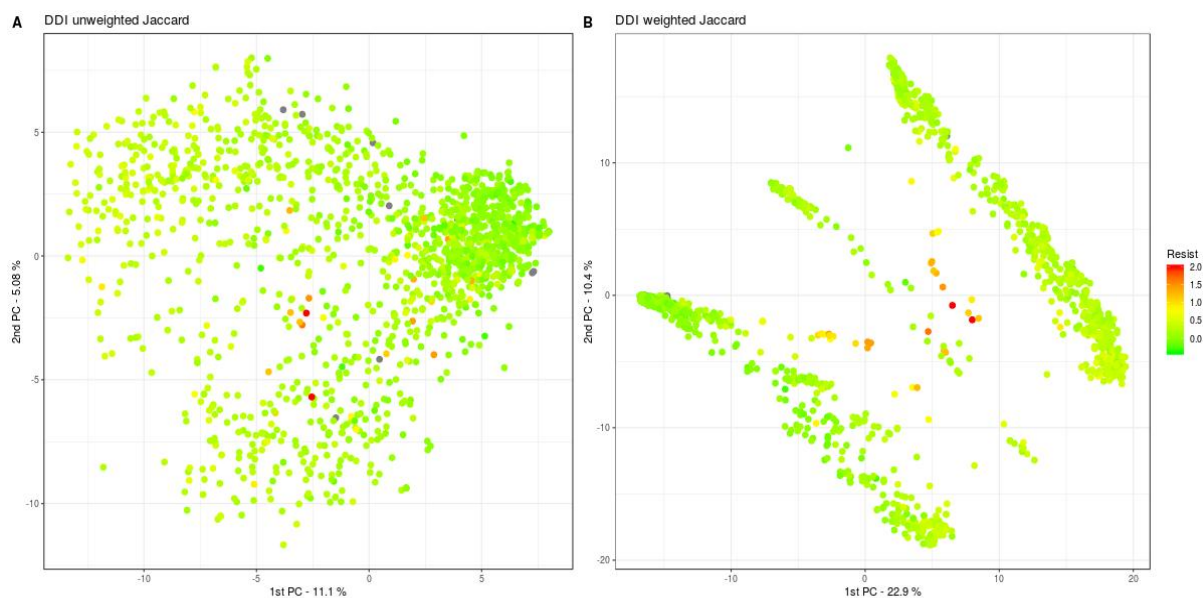


Figure S39. The unweighted (A) and weighted (B) Jaccard kPCA for DDI (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

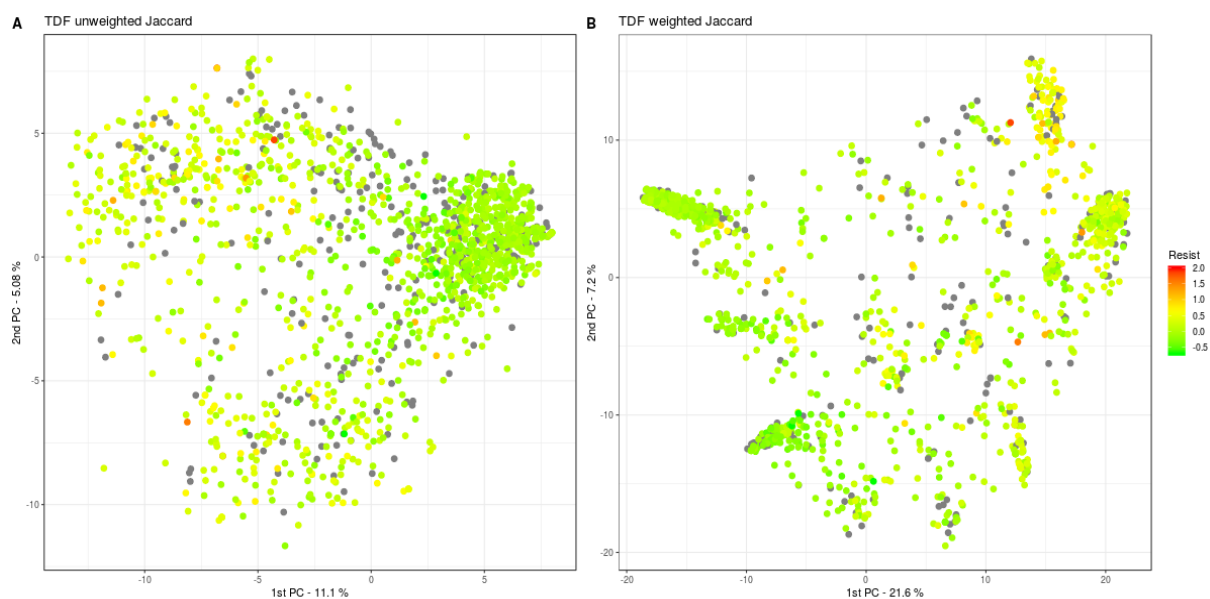


Figure S40. The unweighted (A) and weighted (B) Jaccard kPCA for TDF (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

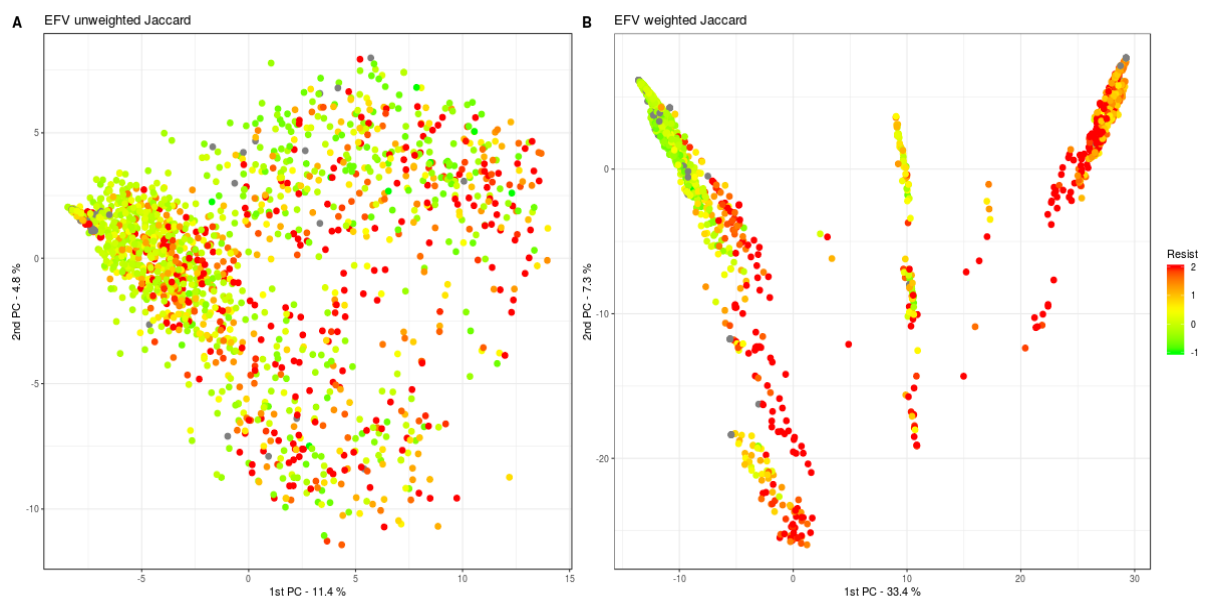


Figure S41. The unweighted (A) and weighted (B) Jaccard kPCA for EFV (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

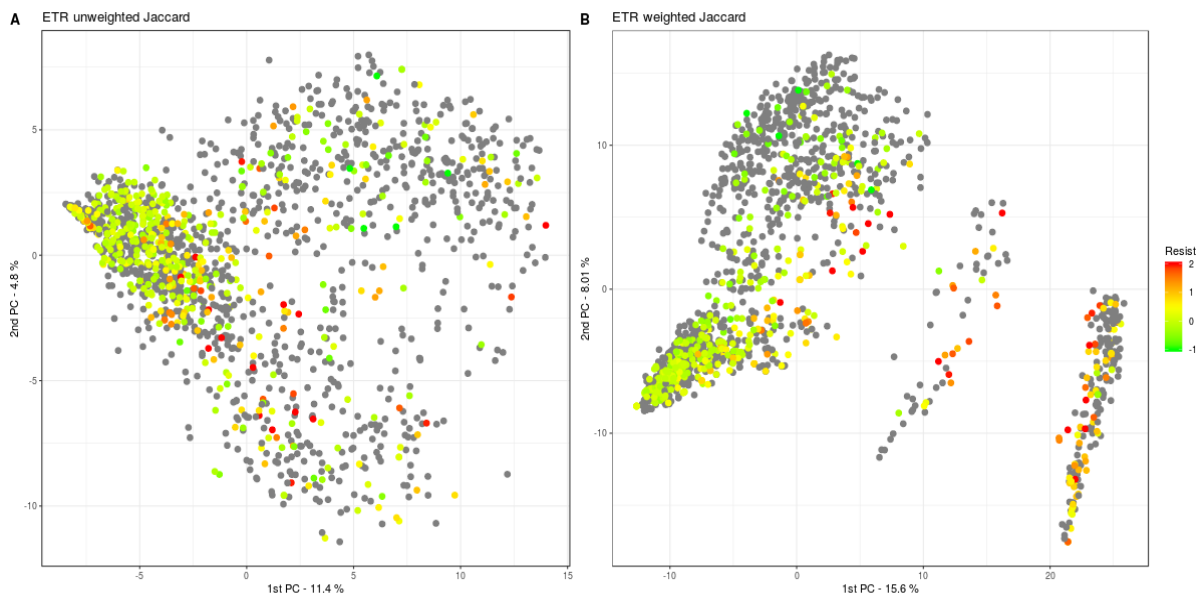


Figure S42. The unweighted (A) and weighted (B) Jaccard kPCA for ETR (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

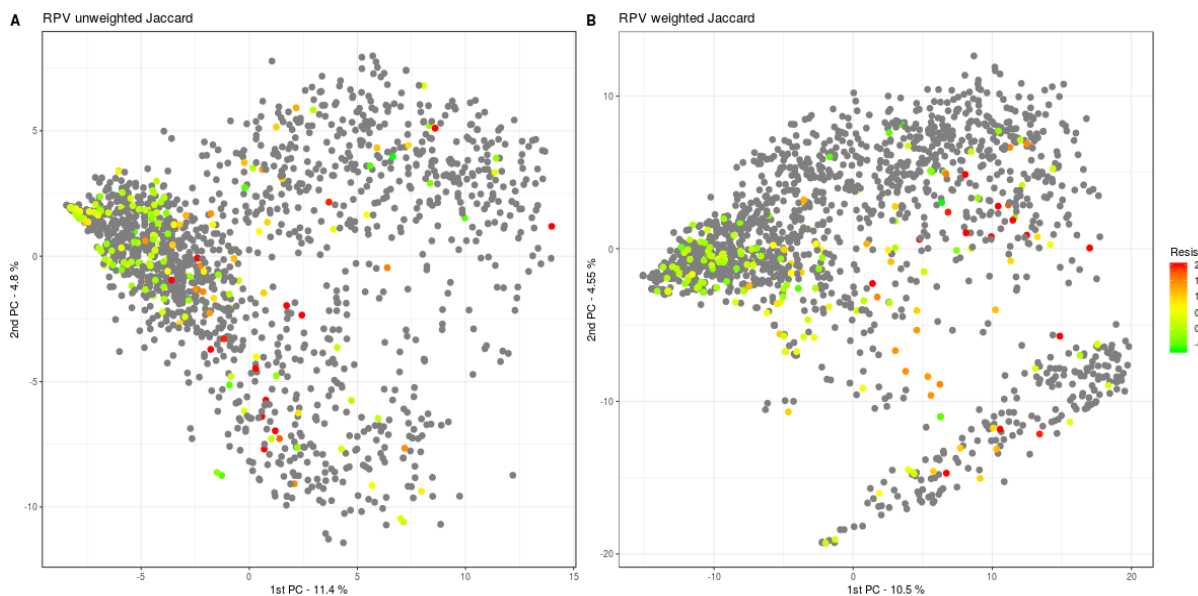


Figure S43. The unweighted (A) and weighted (B) Jaccard kPCA for RPV (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

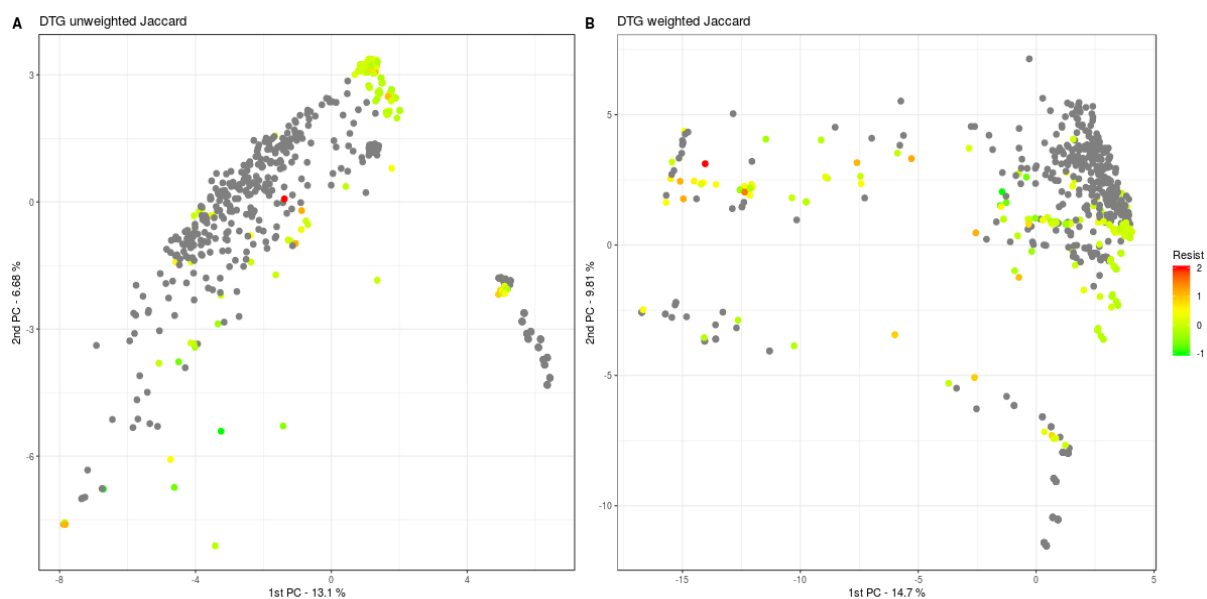


Figure S44. The unweighted (A) and weighted (B) Jaccard kPCA for DTG (integrase inhibitor). Gray dots represent sequences with missing resistance value.

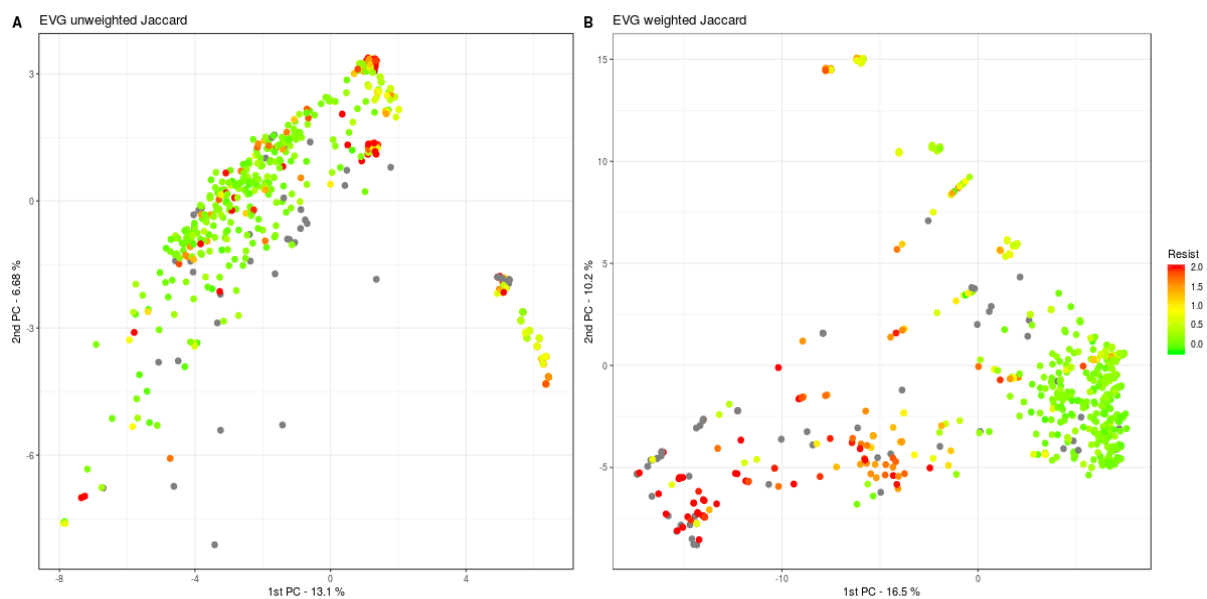


Figure S45. The unweighted (A) and weighted (B) Jaccard kPCA for EVG (integrase inhibitor). Gray dots represent sequences with missing resistance value.

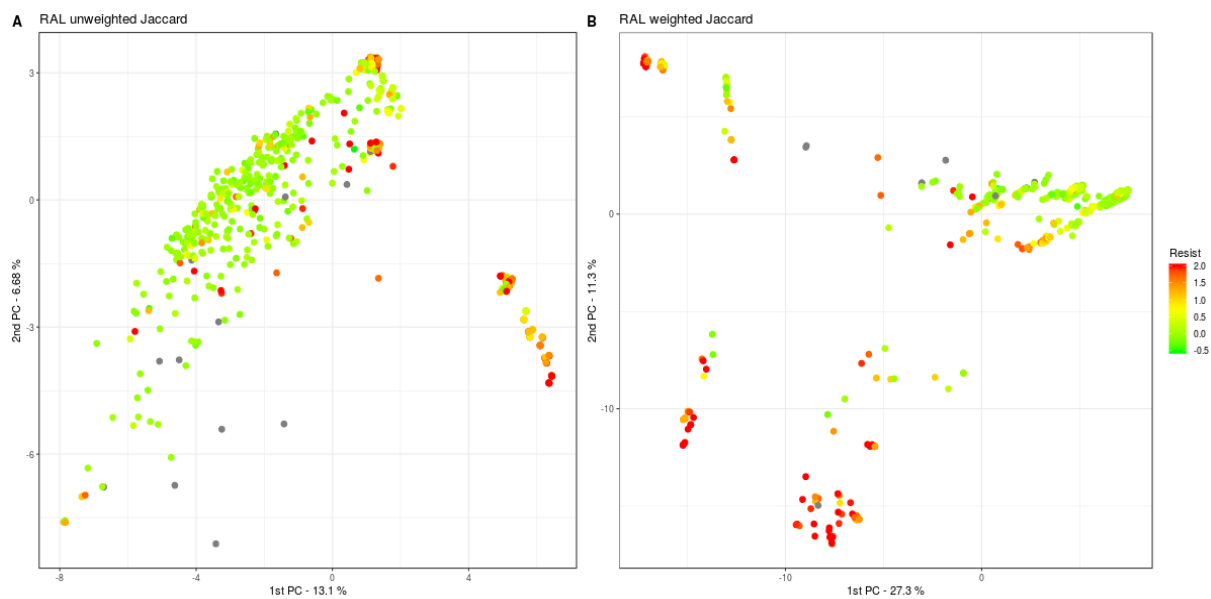


Figure S46. The unweighted (A) and weighted (B) Jaccard kPCA for RAL (integrase inhibitor). Gray dots represent sequences with missing resistance value.

Table S1. Mean NMSE for all 21 analyzed drugs. wLIN, wRBF, wOV and wJAC stand for (individual) weighted Linear, RBF, Overlap and Jaccard models. sLIN, sRBF, sOV and sJAC correspond to their stacked counterparts.

	wLIN	sLIN	wRBF	sRBF	wOV	sOV	wJAC	sJAC
ATV	0.136	0.136	0.146	0.149	0.137	0.137	0.132	0.132
DRV	0.200	0.197	0.190	0.192	0.190	0.196	0.175	0.182
FPV	0.155	0.162	0.156	0.165	0.150	0.158	0.141	0.148
IDV	0.135	0.141	0.134	0.142	0.129	0.136	0.126	0.134
LPV	0.108	0.112	0.110	0.118	0.105	0.113	0.098	0.102
NFV	0.143	0.138	0.133	0.136	0.130	0.134	0.123	0.126
SQV	0.151	0.146	0.145	0.148	0.142	0.146	0.130	0.133
TPV	0.365	0.398	0.346	0.387	0.348	0.377	0.333	0.358
3TC	0.146	0.134	0.112	0.182	0.108	0.163	0.075	0.091
ABC	0.186	0.189	0.162	0.187	0.158	0.177	0.138	0.153
AZT	0.222	0.229	0.206	0.218	0.201	0.217	0.181	0.194
D4T	0.229	0.270	0.223	0.255	0.230	0.257	0.226	0.251
DDI	0.256	0.279	0.250	0.281	0.247	0.276	0.237	0.267
TDF	0.397	0.454	0.381	0.475	0.390	0.453	0.367	0.431
EFV	0.181	0.168	0.157	0.161	0.154	0.165	0.122	0.126
ETR	0.360	0.386	0.345	0.366	0.320	0.331	0.312	0.321
NVP	0.269	0.165	0.164	0.155	0.144	0.155	0.110	0.112
RPV	0.529	0.524	0.516	0.522	0.466	0.474	0.455	0.461
DTG	0.759	0.779	0.677	0.726	0.686	0.729	0.654	0.679
EVG	0.275	0.238	0.212	0.212	0.186	0.196	0.142	0.144
RAL	0.165	0.153	0.139	0.141	0.140	0.154	0.105	0.110

Table S2. NMSE standard error for all 21 analyzed drugs. Abbreviations as in Table S1.

	wLIN	sLIN	wRBF	sRBF	wOV	sOV	wJAC	sJAC
ATV	0.013	0.016	0.014	0.017	0.014	0.014	0.014	0.015
DRV	0.018	0.029	0.018	0.024	0.018	0.024	0.018	0.017
FPV	0.011	0.012	0.010	0.013	0.009	0.010	0.009	0.009
IDV	0.009	0.010	0.008	0.008	0.008	0.007	0.008	0.008
LPV	0.010	0.013	0.013	0.016	0.011	0.012	0.009	0.011
NFV	0.009	0.009	0.009	0.009	0.007	0.008	0.006	0.008
SQV	0.009	0.012	0.008	0.011	0.008	0.009	0.008	0.008
TPV	0.038	0.047	0.040	0.050	0.043	0.048	0.040	0.044
3TC	0.022	0.026	0.021	0.038	0.020	0.043	0.013	0.016
ABC	0.016	0.019	0.016	0.020	0.014	0.015	0.015	0.015
AZT	0.017	0.022	0.017	0.021	0.018	0.021	0.012	0.016
D4T	0.028	0.032	0.027	0.030	0.027	0.035	0.027	0.033
DDI	0.023	0.029	0.023	0.034	0.022	0.022	0.020	0.025
TDF	0.041	0.053	0.039	0.100	0.039	0.065	0.039	0.065
EFV	0.016	0.017	0.013	0.017	0.014	0.017	0.012	0.013
ETR	0.053	0.055	0.052	0.061	0.042	0.045	0.047	0.047
NVP	0.023	0.021	0.019	0.020	0.017	0.017	0.011	0.012
RPV	0.132	0.130	0.127	0.147	0.106	0.114	0.096	0.106
DTG	0.159	0.175	0.129	0.166	0.128	0.130	0.136	0.166
EVG	0.046	0.048	0.036	0.044	0.036	0.039	0.032	0.033
RAL	0.040	0.040	0.033	0.035	0.037	0.041	0.029	0.032

Supplementary material Chapter 5

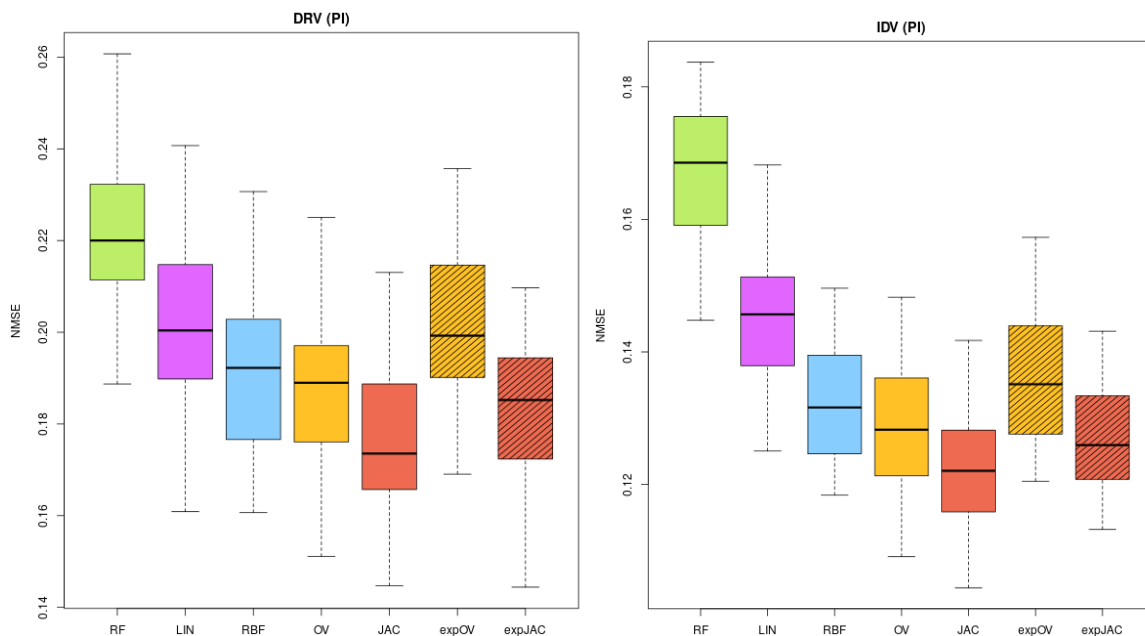


Figure S47. NMSE distribution for DRV and IDV (protease inhibitors). Same legend as that of Figure 5.1.

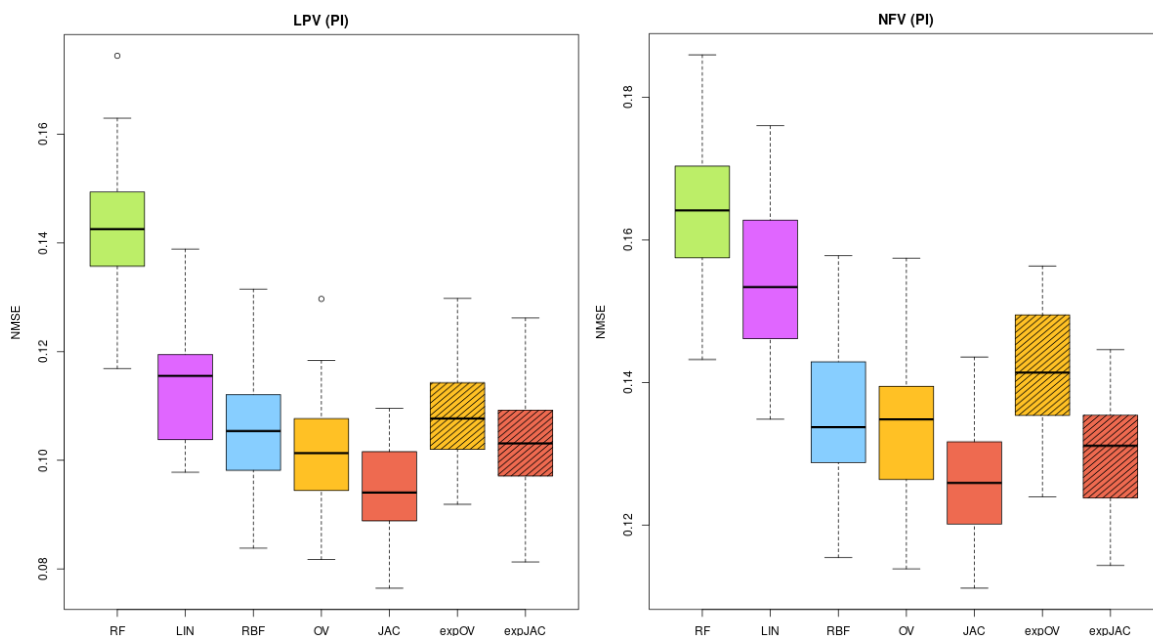


Figure S48. NMSE distribution for LPV and NFV (protease inhibitors). Same legend as that of Figure 5.1.

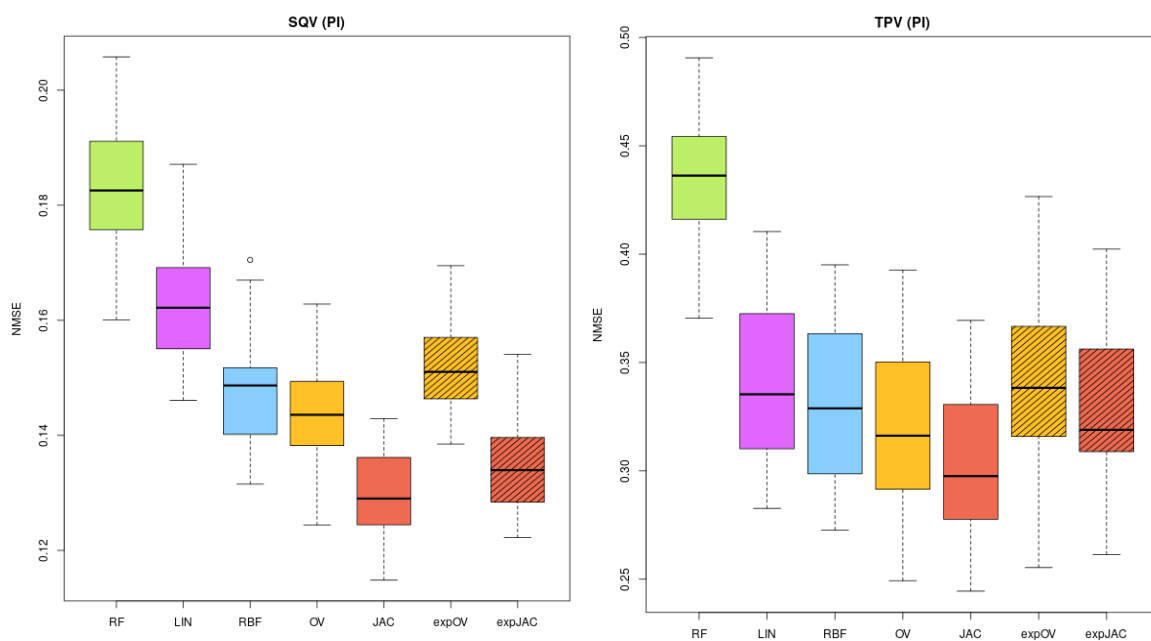


Figure S49. NMSE distribution for SQV and TPV (protease inhibitors). Same legend as that of Figure 5.1.

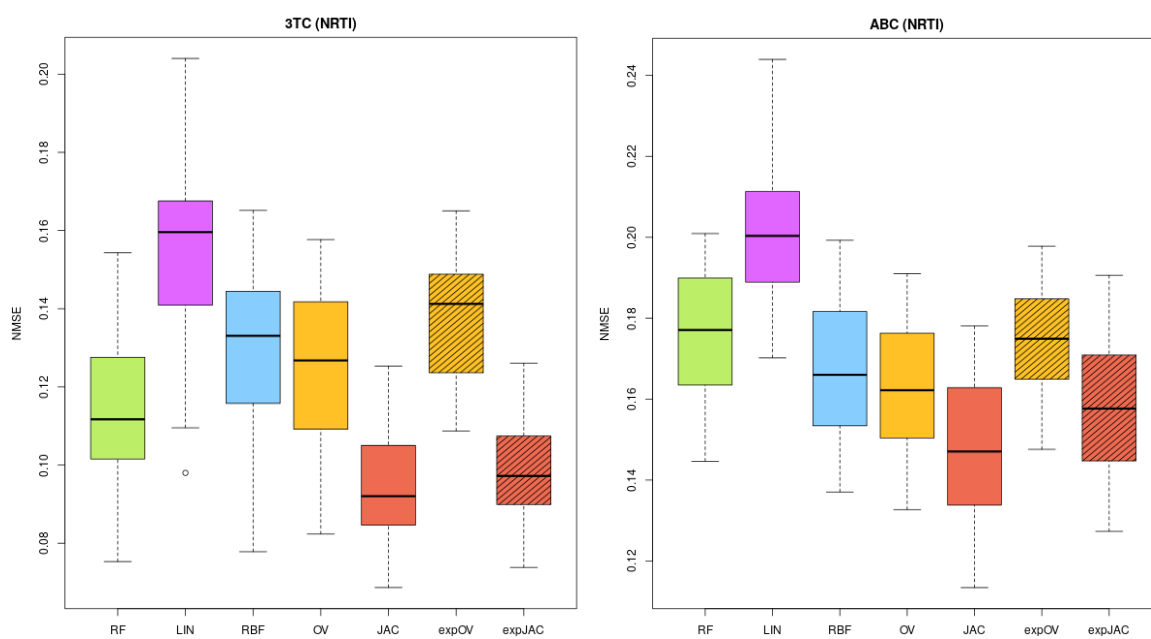


Figure S50. NMSE distribution for 3TC and ABC (reverse transcriptase inhibitors). Same legend as that of Figure 5.1.

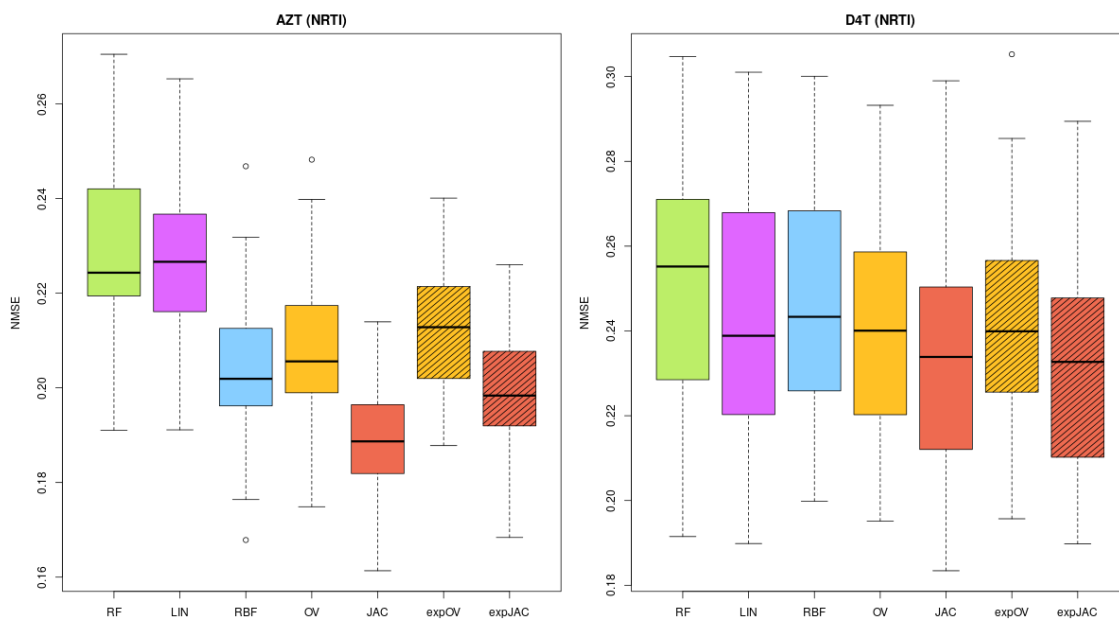


Figure S51. NMSE distribution for AZT and D4T (reverse transcriptase inhibitors). Same legend as that of Figure 5.1.

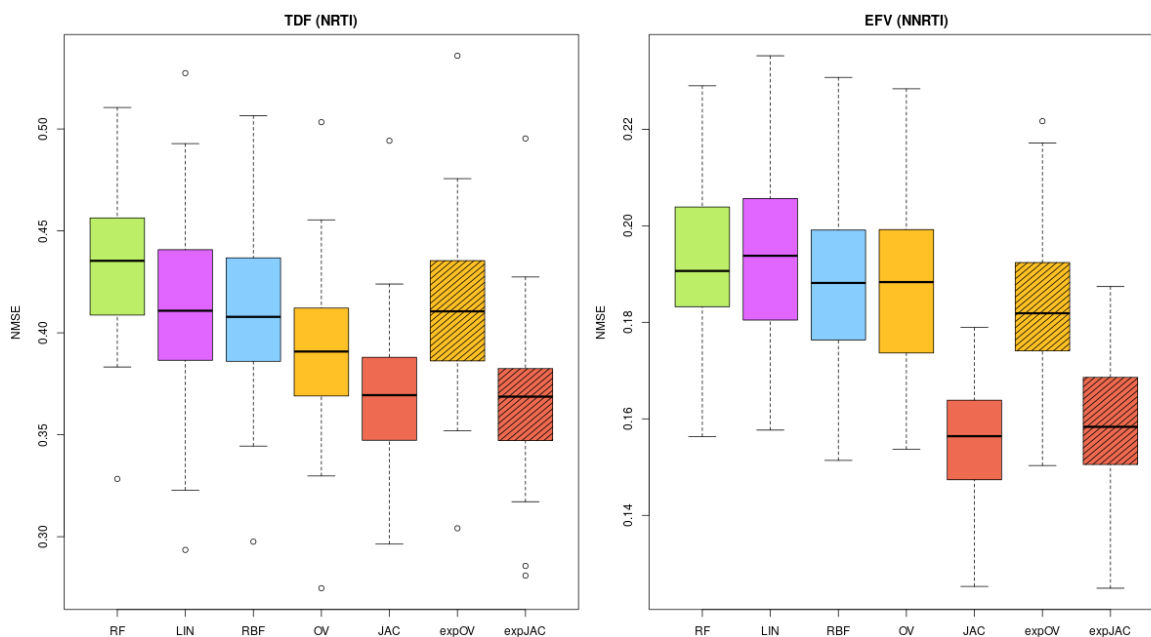


Figure S52. NMSE distribution for TDF and EFV (reverse transcriptase inhibitors). Same legend as that of Figure 5.1.

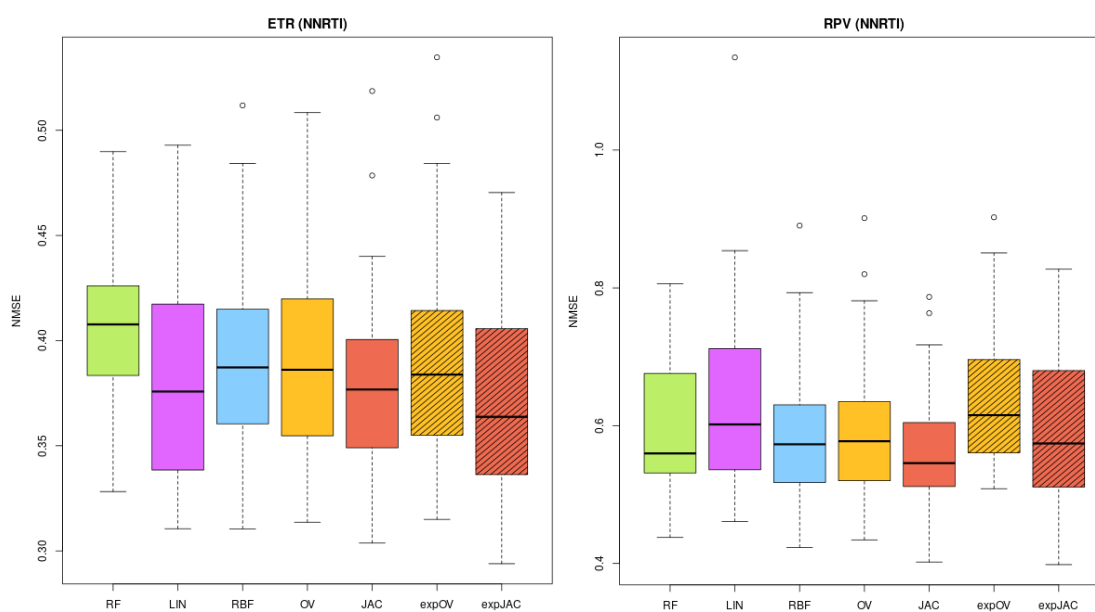


Figure S53. NMSE distribution for ETR and RPV (reverse transcriptase inhibitors). Same legend as that of Figure 5.1.

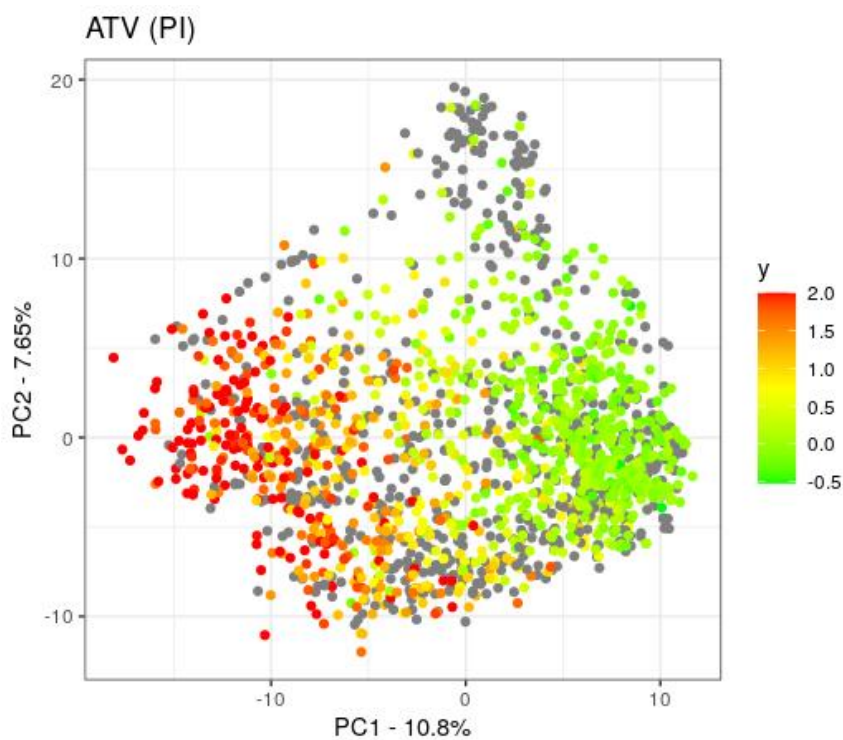


Figure S54. The expJac kPCA for ATV (protease inhibitor). Gray dots represent sequences with missing resistance value.

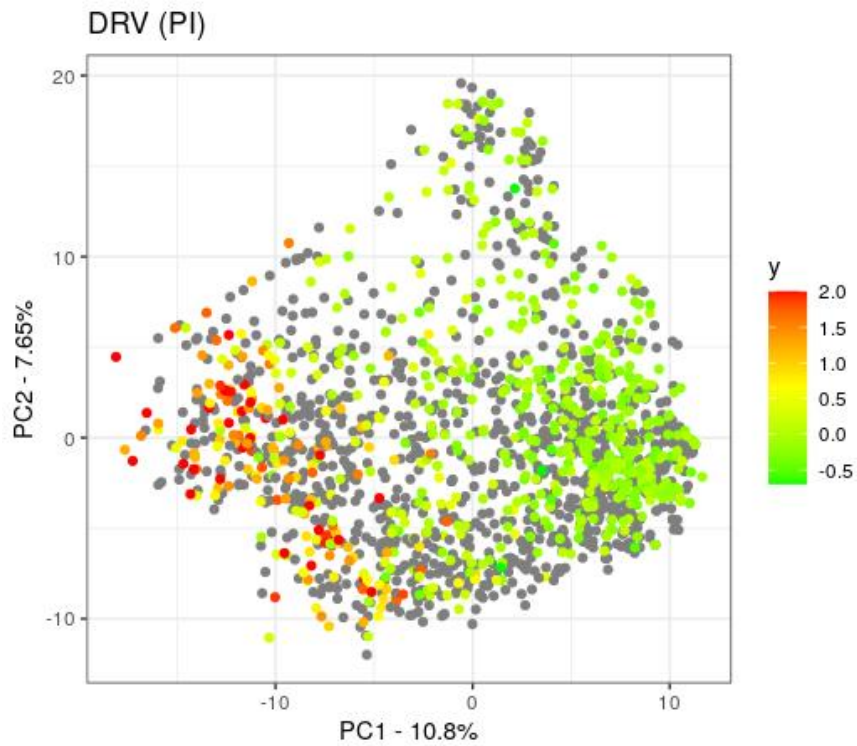


Figure S55. The expJac kPCA for DRV (protease inhibitor). Gray dots represent sequences with missing resistance value.

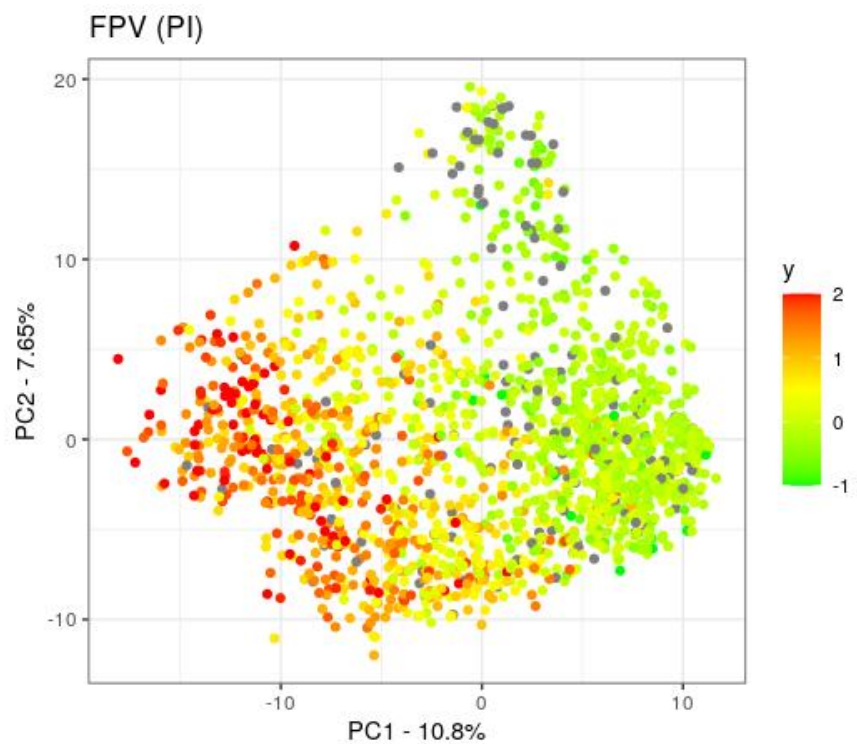


Figure S56. The expJac kPCA for FPV (protease inhibitor). Gray dots represent sequences with missing resistance value.

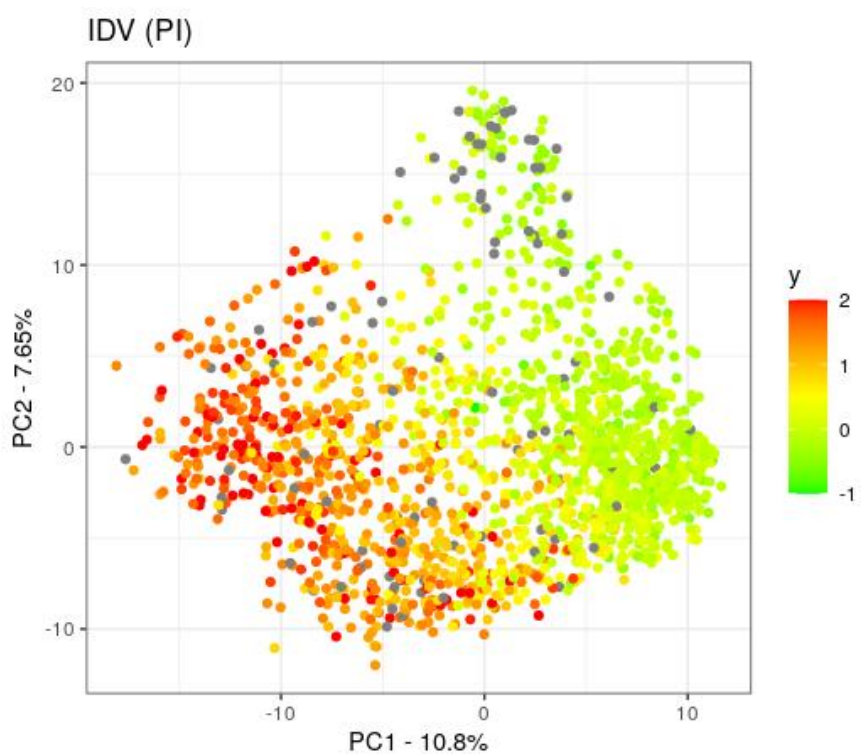


Figure S57. The expJac kPCA for IDV (protease inhibitor). Gray dots represent sequences with missing resistance value.

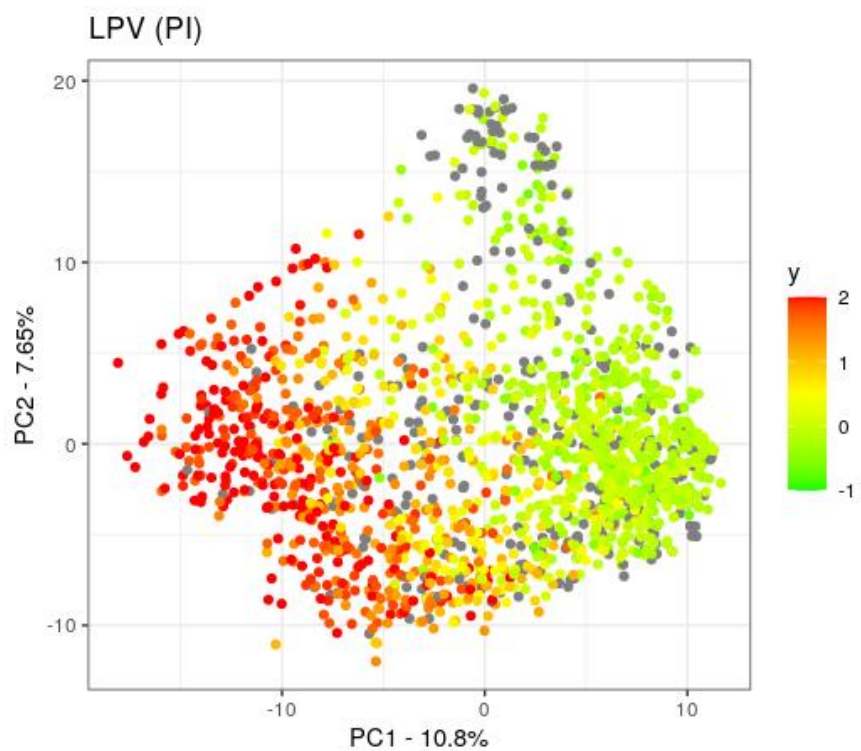


Figure S58. The expJac kPCA for LPV (protease inhibitor). Gray dots represent sequences with missing resistance value.

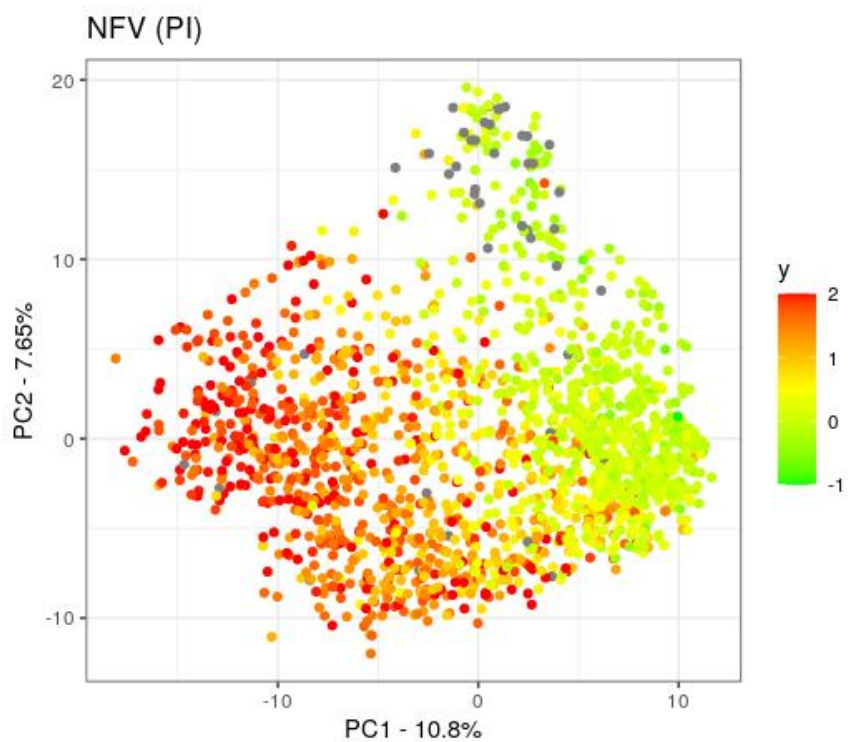


Figure S59. The expJac kPCA for NFV (protease inhibitor). Gray dots represent sequences with missing resistance value.

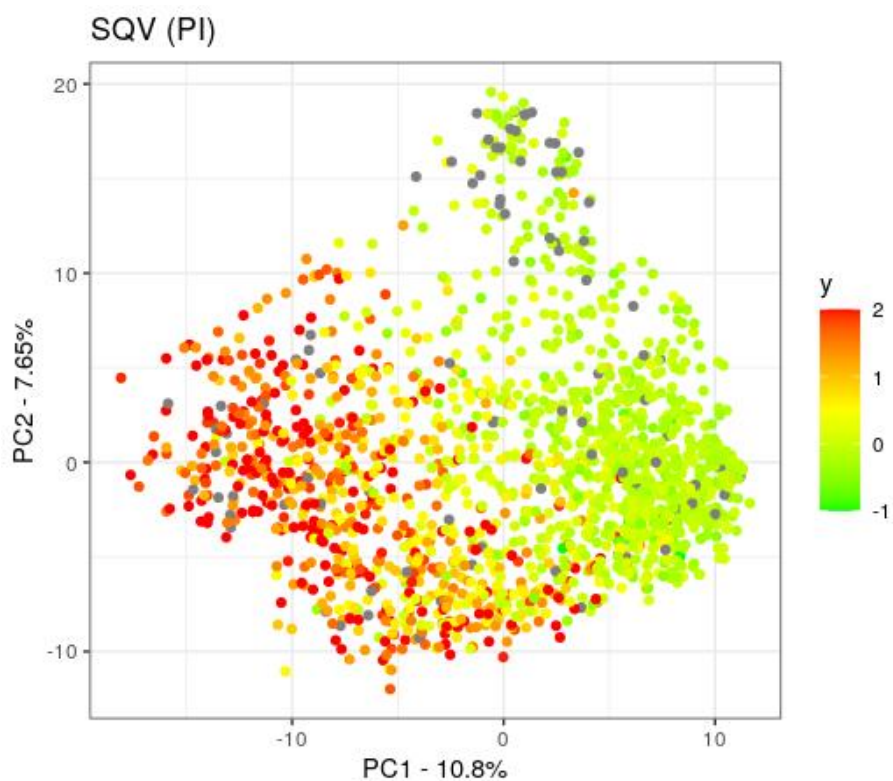


Figure S60. The expJac kPCA for SQV (protease inhibitor). Gray dots represent sequences with missing resistance value.

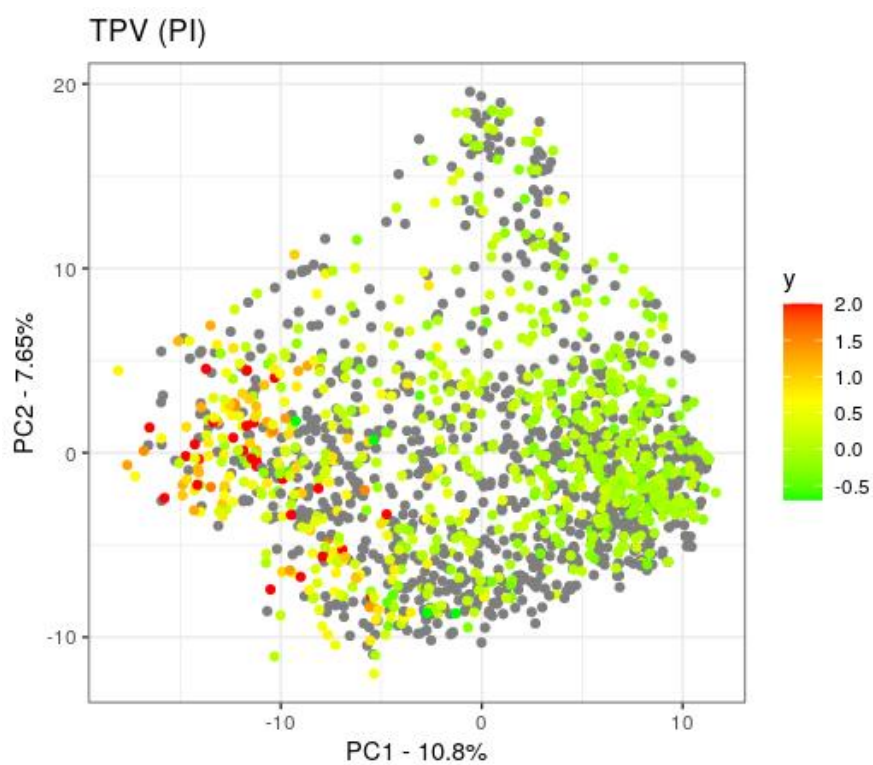


Figure S61. The expJac kPCA for TPV (protease inhibitor). Gray dots represent sequences with missing resistance value.

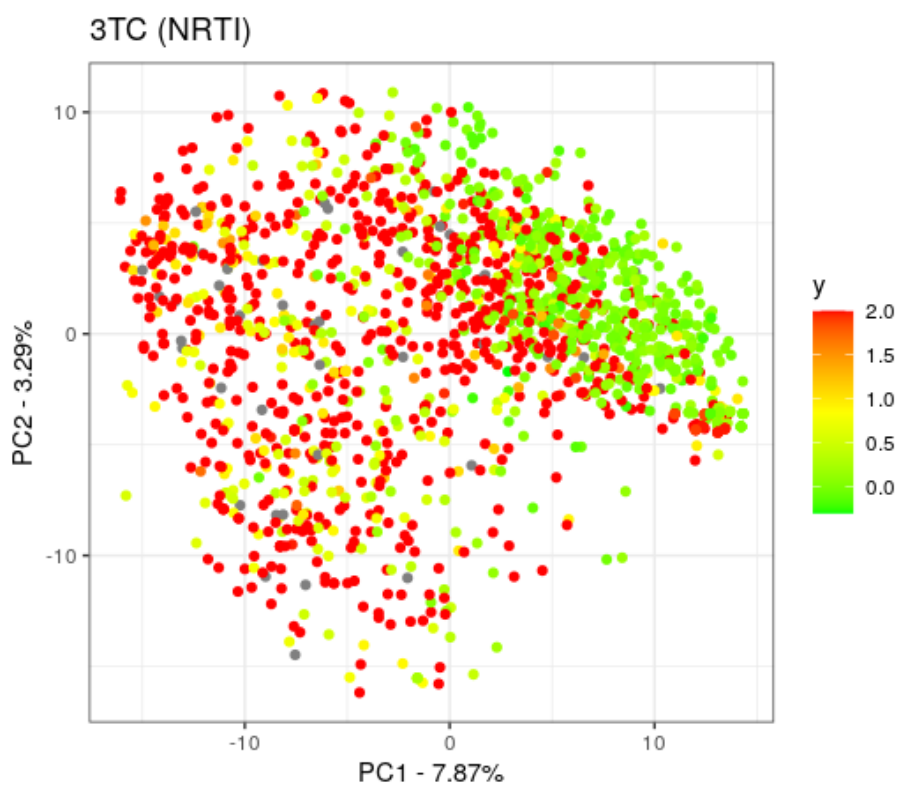


Figure S62. The expJac kPCA for 3TC (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

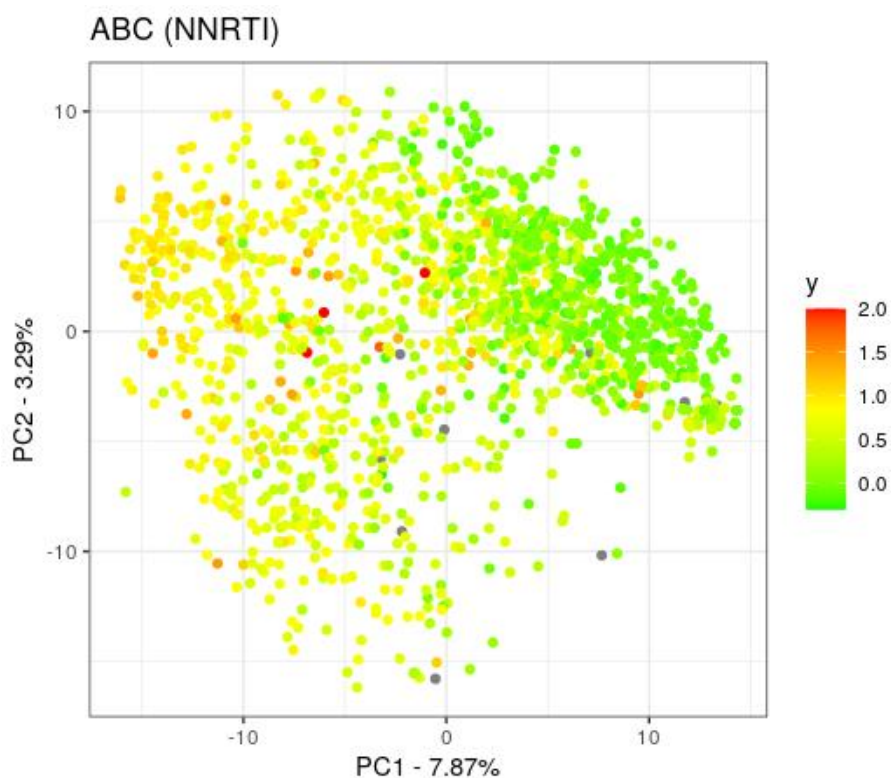


Figure S63. The expJac kPCA for ABC (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

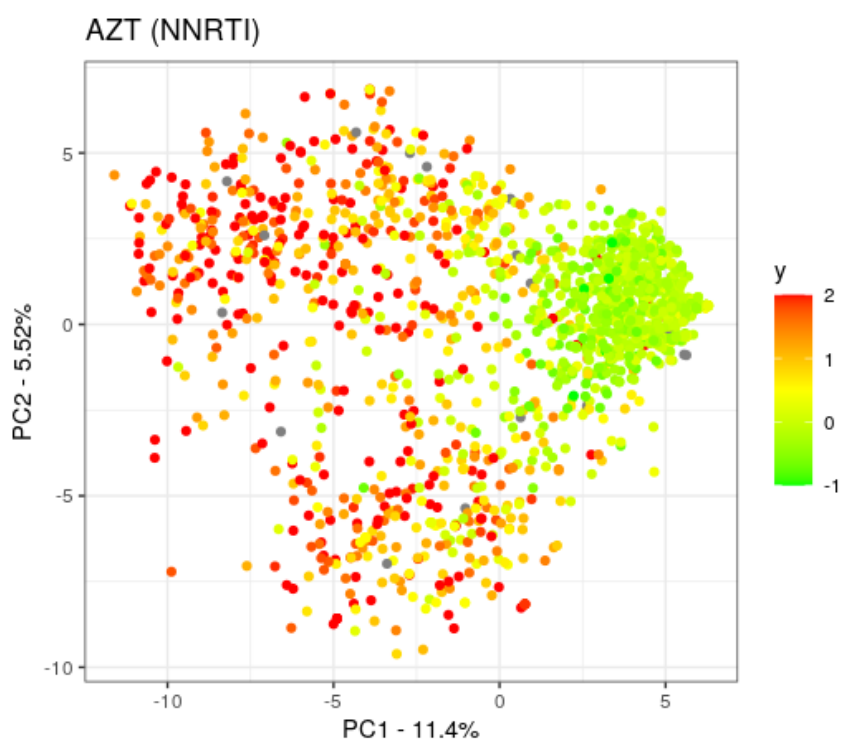


Figure S64. The expJac kPCA for AZT (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

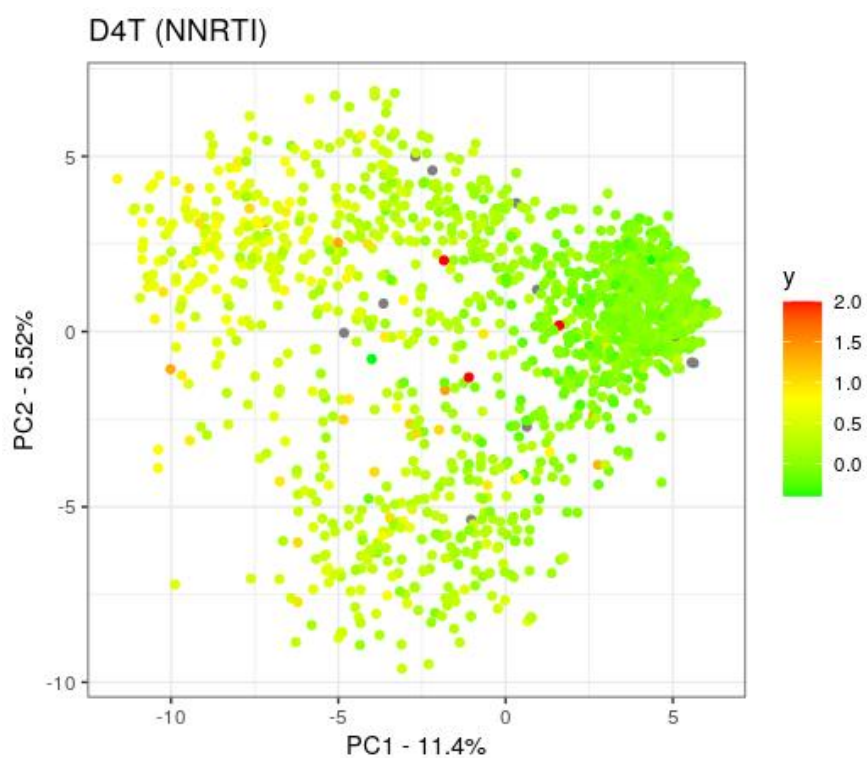


Figure S65. The expJac kPCA for D4T (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

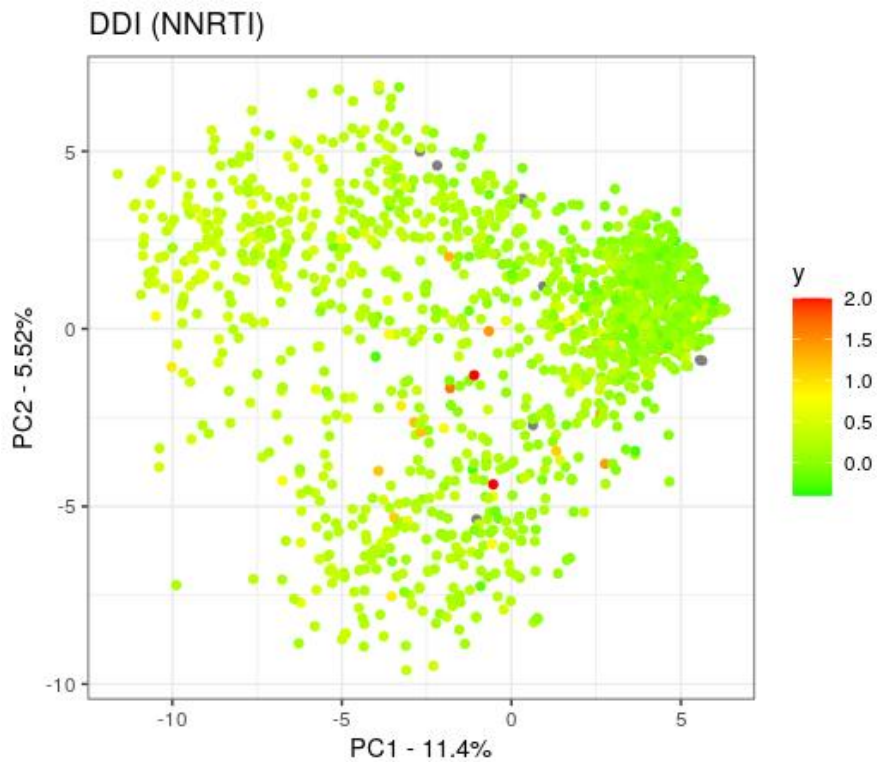


Figure S66. The expJac kPCA for DDI (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

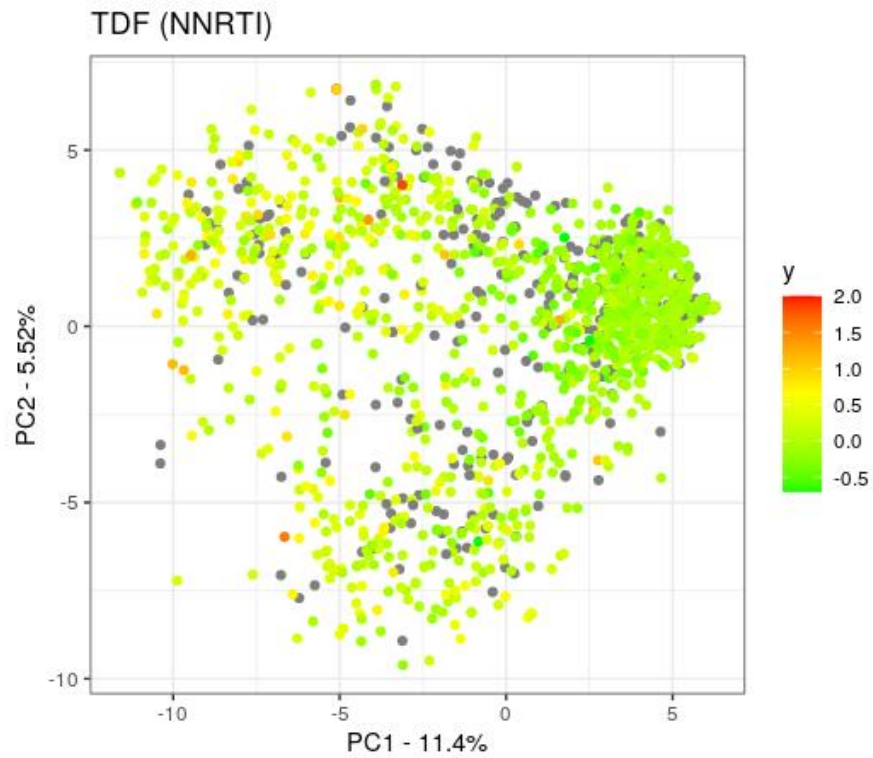


Figure S67. The expJac kPCA for TDF (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

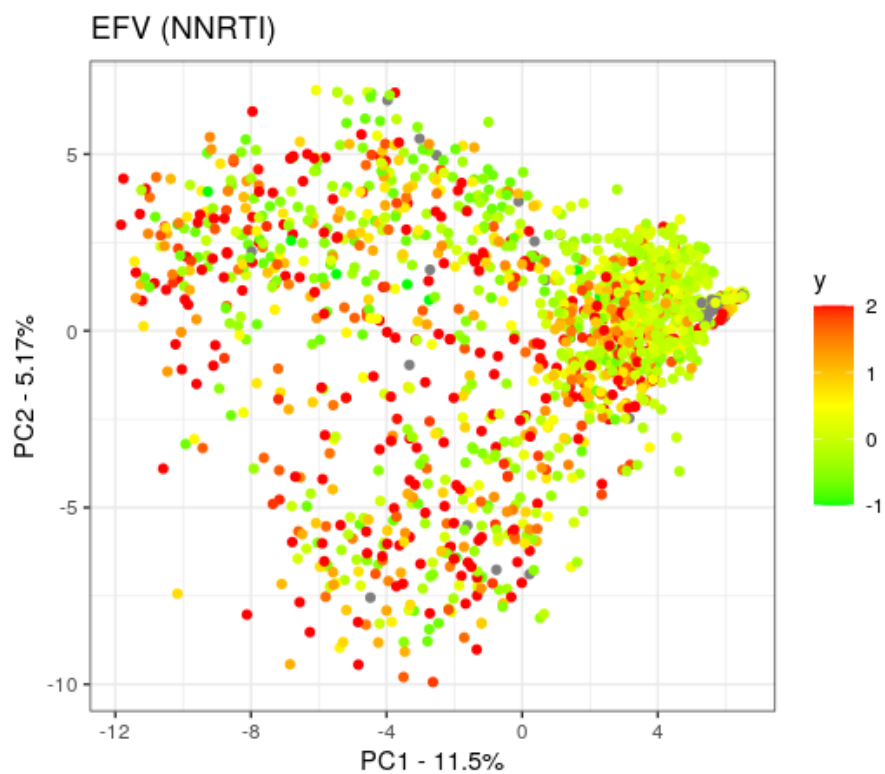


Figure S68. The expJac kPCA for EFV (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

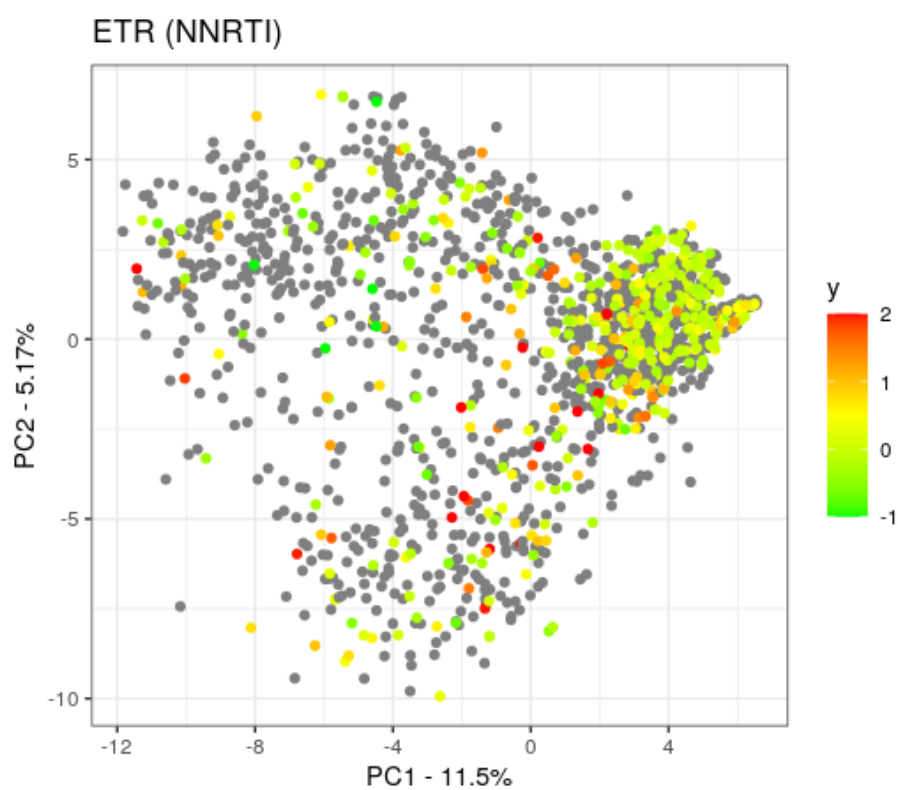


Figure S69. The expJac kPCA for ETR (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

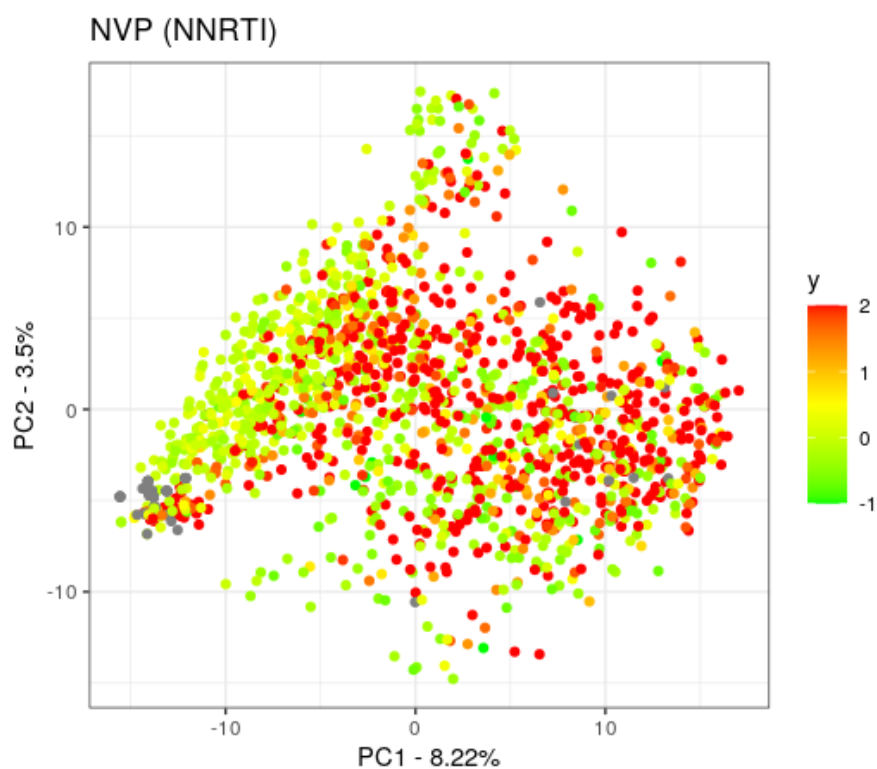


Figure S70. The expJac kPCA for NVP (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

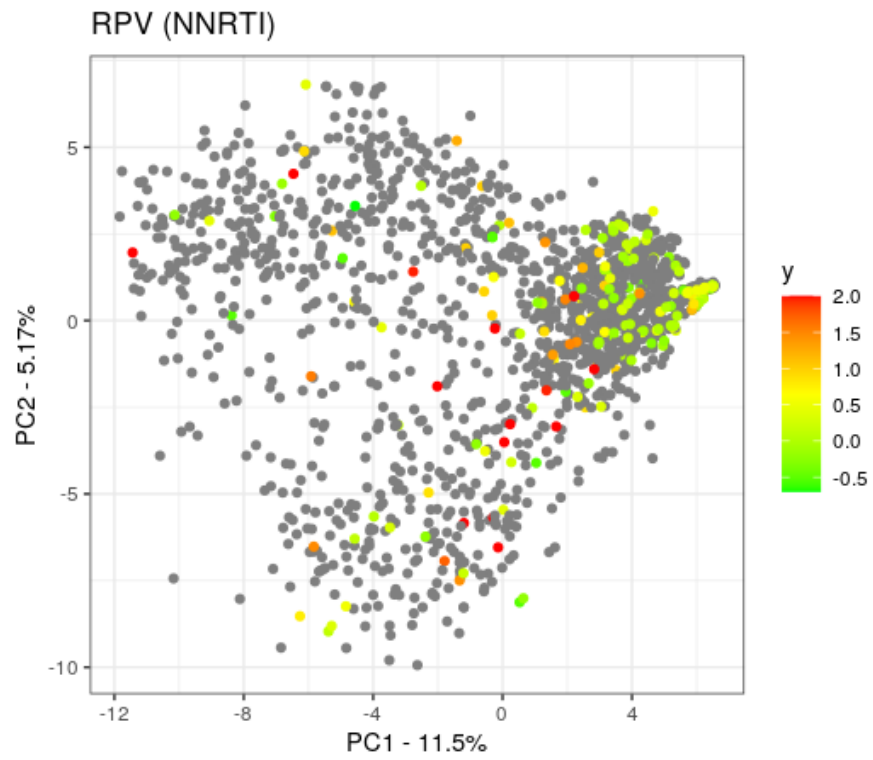


Figure S71. The expJac kPCA for RPV (reverse transcriptase inhibitor). Gray dots represent sequences with missing resistance value.

Supplementary material Chapter 6

Supplementary Figures:

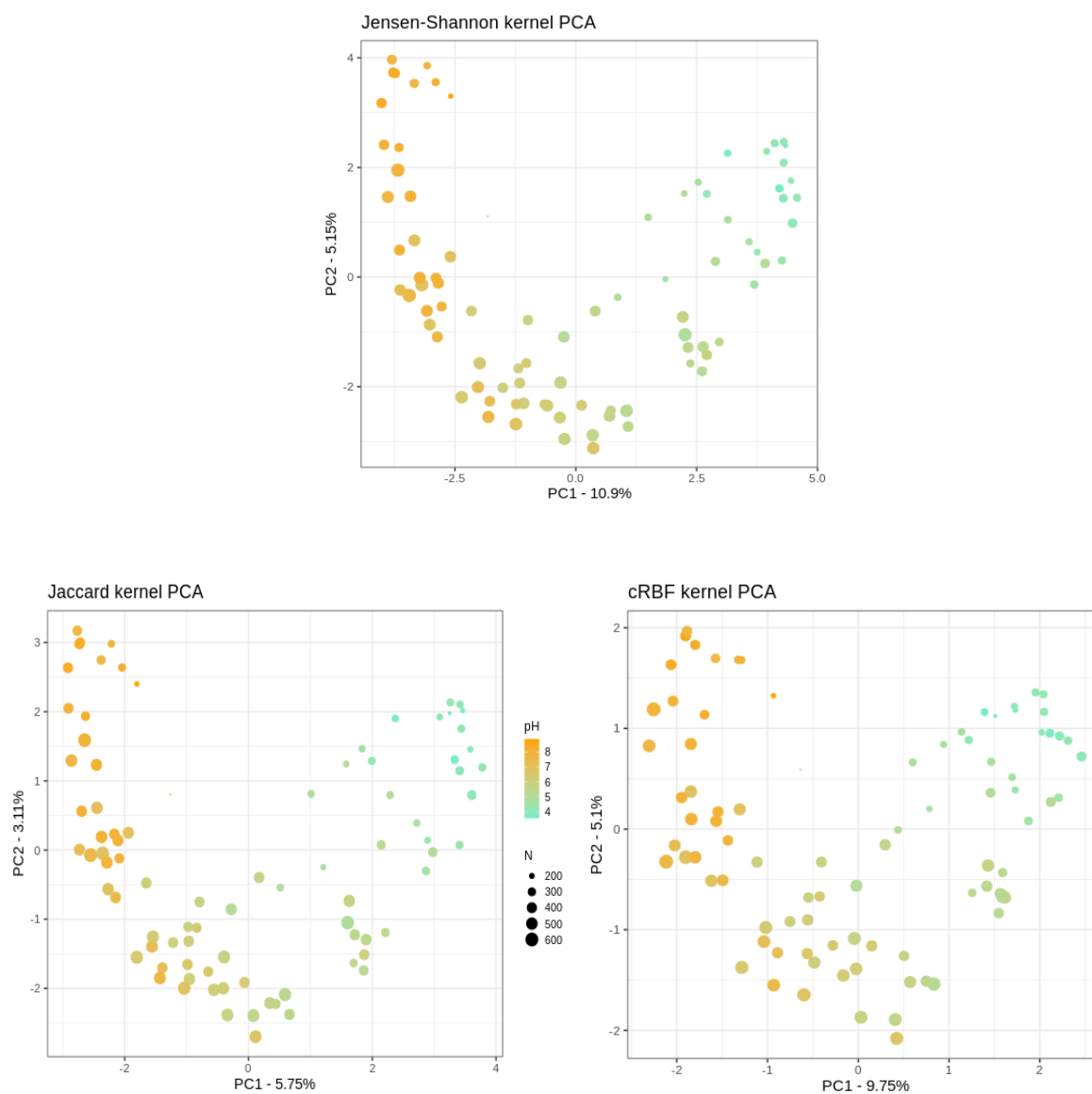


Figure S72. Soil kPCAs for the cRBF kernel, JSK and the Jaccard kernel. Legend as in Figure 6.1A.

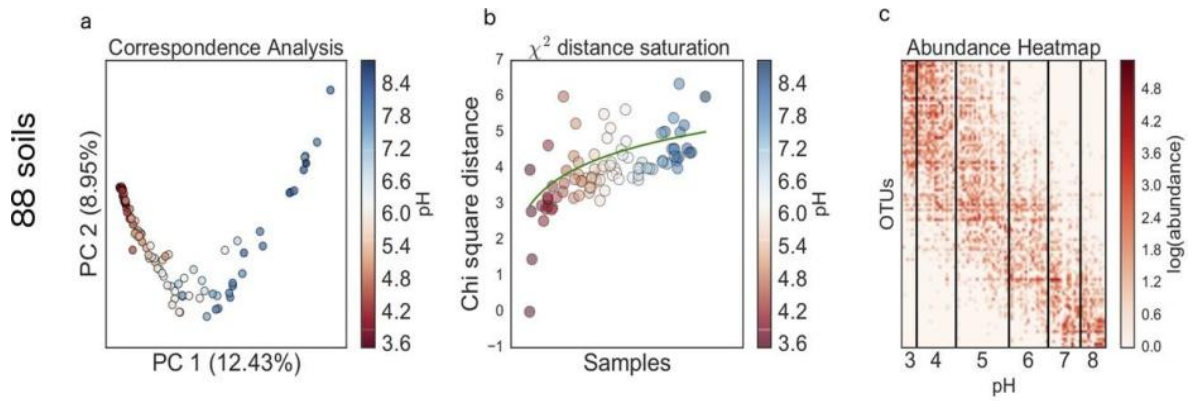


Figure S73. Morton *et al.* (2017) revisit Lauber *et al.* (2009) data. The Correspondence Analysis in panel a shows a clear U-shaped effect. Panel c demonstrates the band nature of the Soil dataset.

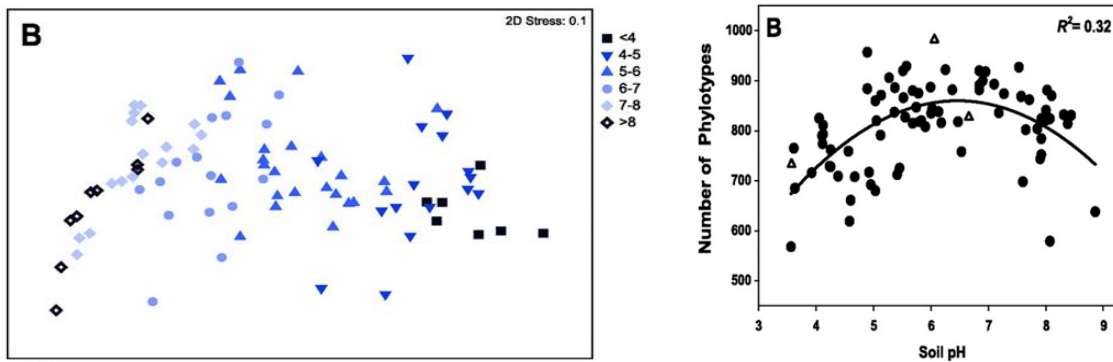


Figure S74. Original results by Lauber *et al.* (2009). Left: MDS plot derived from Unifrac distances with shape indicating soil pH. The arch pattern is visible but less steep than in figures 6.1A, S1 and S2 because of the coarser taxonomic resolution in the original study (Morton *et al.*, 2017). Right: soil pH correlation to number of phylotypes.

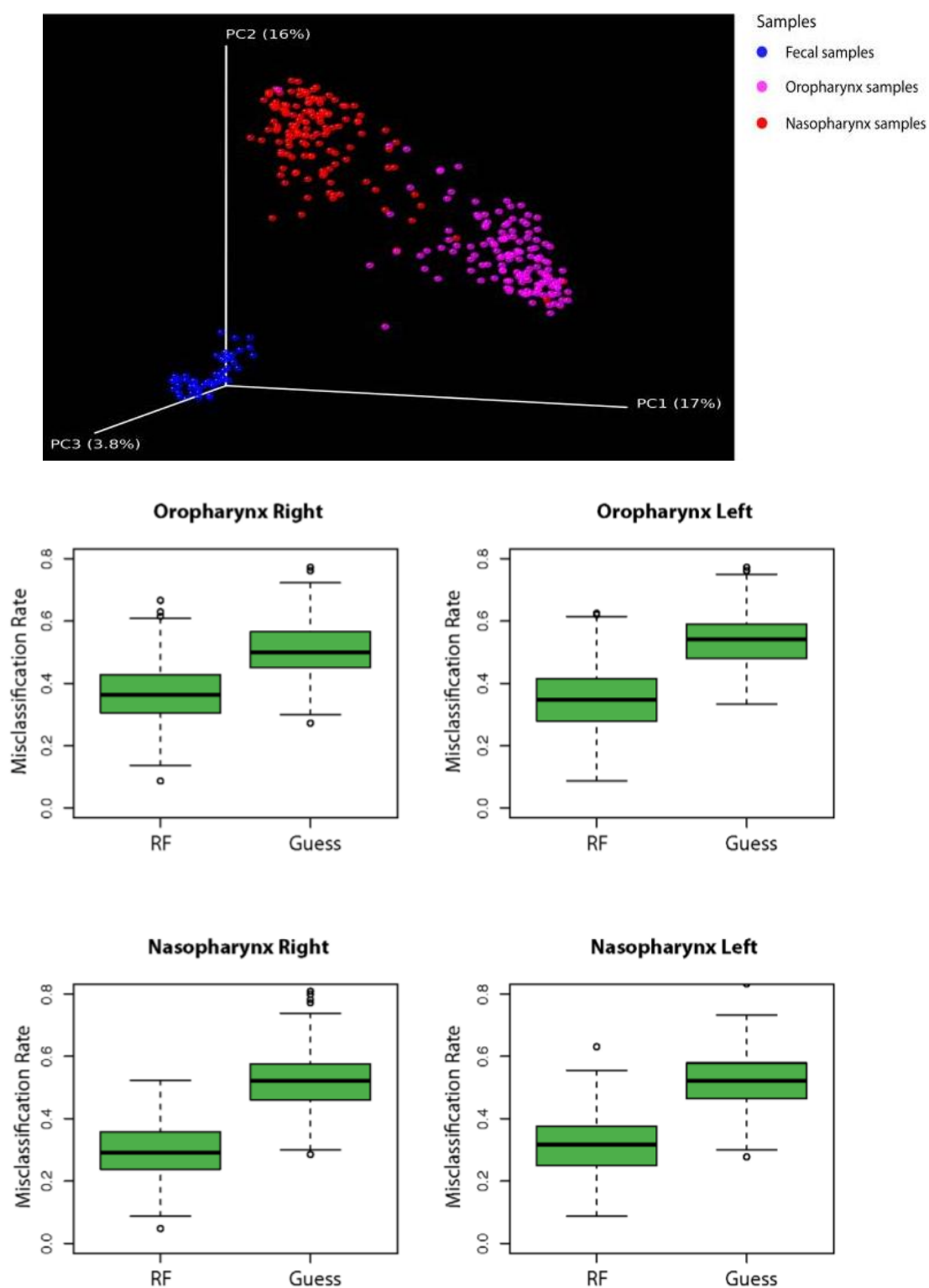


Figure S75. Original results of Charlson *et al.* (2010). Above: PCoA comparison of the taxonomic abundances. Colors denote body sites: oropharynx (red), nasopharynx (pink) and fecal (blue). Below: RF misclassification (1-Accuracy) versus misclassification of the random model (Guess).

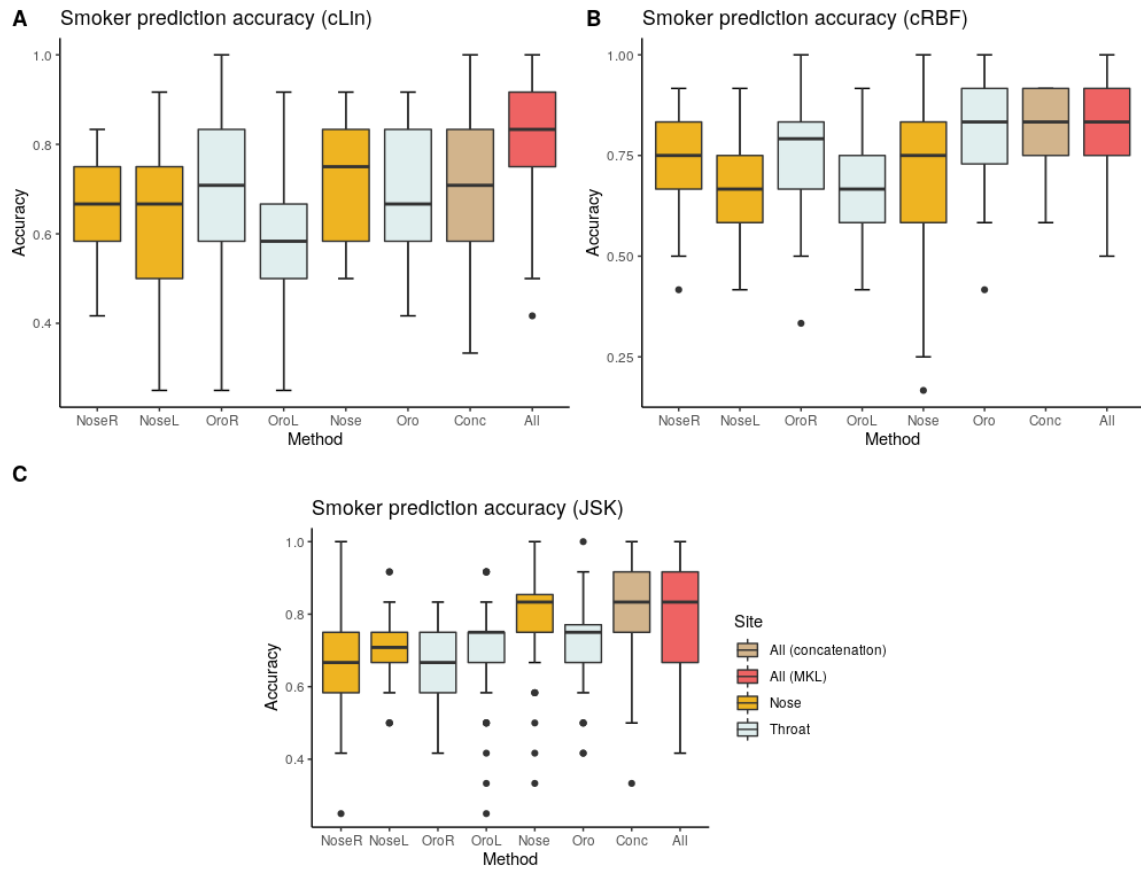


Figure S76. Smoker/nonsmoker prediction accuracy of cLin, cRBF and JSK+SVM. Legend as in Figure 6.2A.

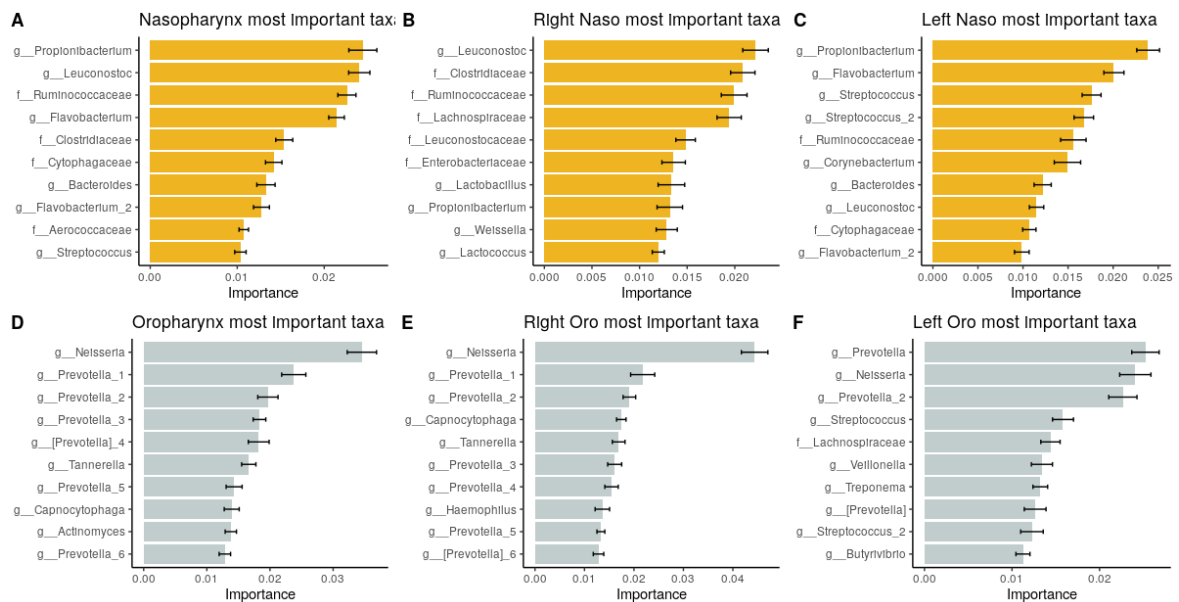


Figure S77. Top ten relevant taxa for the nasopharynx (A) and oropharynx (D) MKL models, and for the four body sampling sites separately (B, C, E, F).

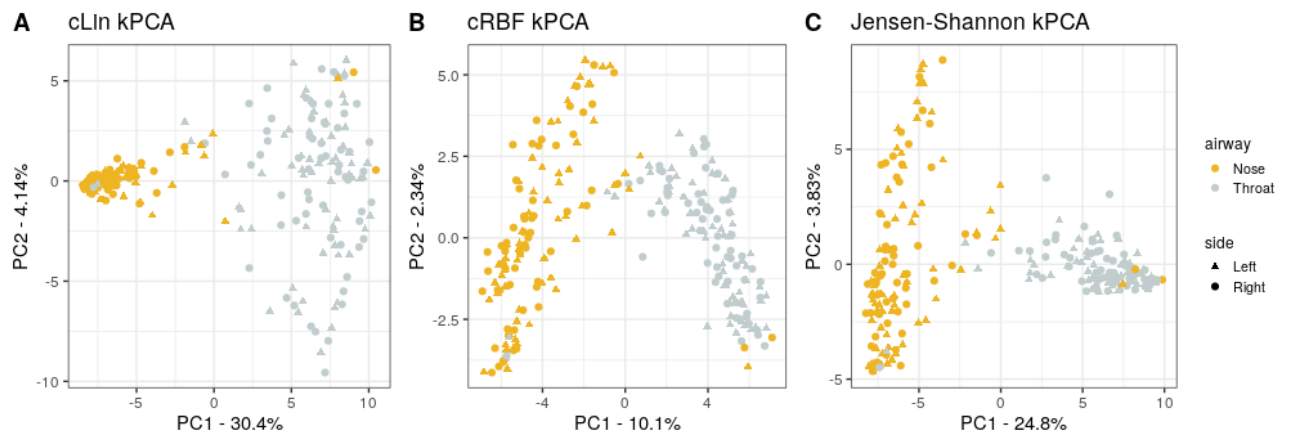


Figure S78. cLin, cRBF and Jensen-Shannon kPCA. Legend as in Figure 6.2D.

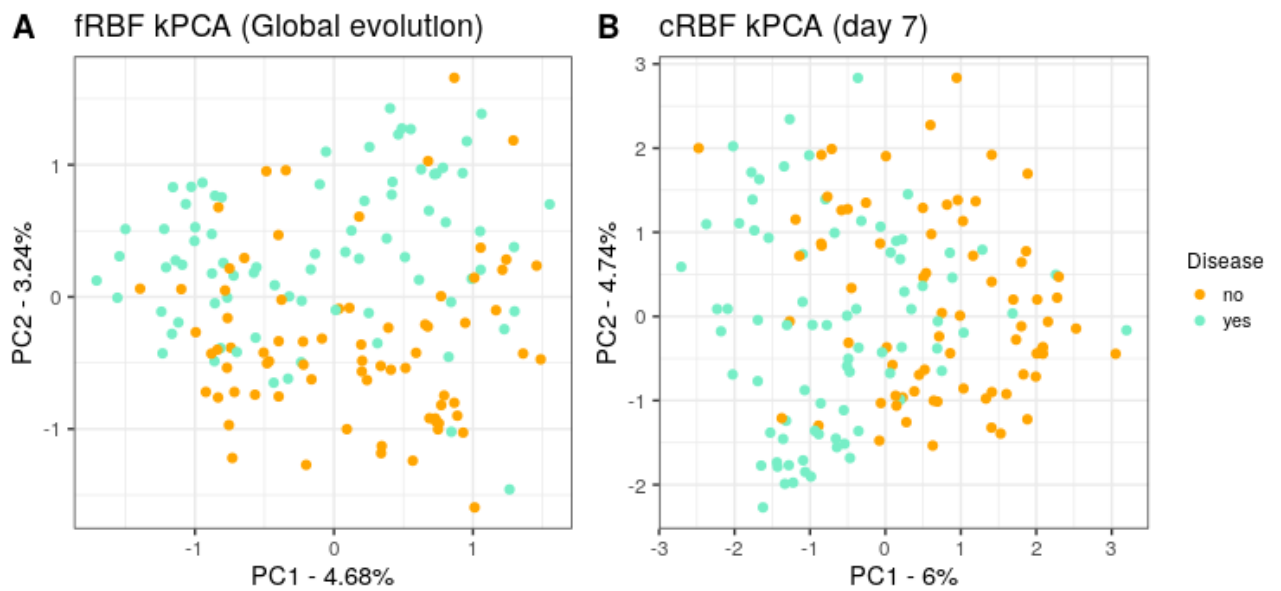


Figure S79. fRBF and cRBF kPCA for the ASV data. Legend as in Figure 6.4 A and B.

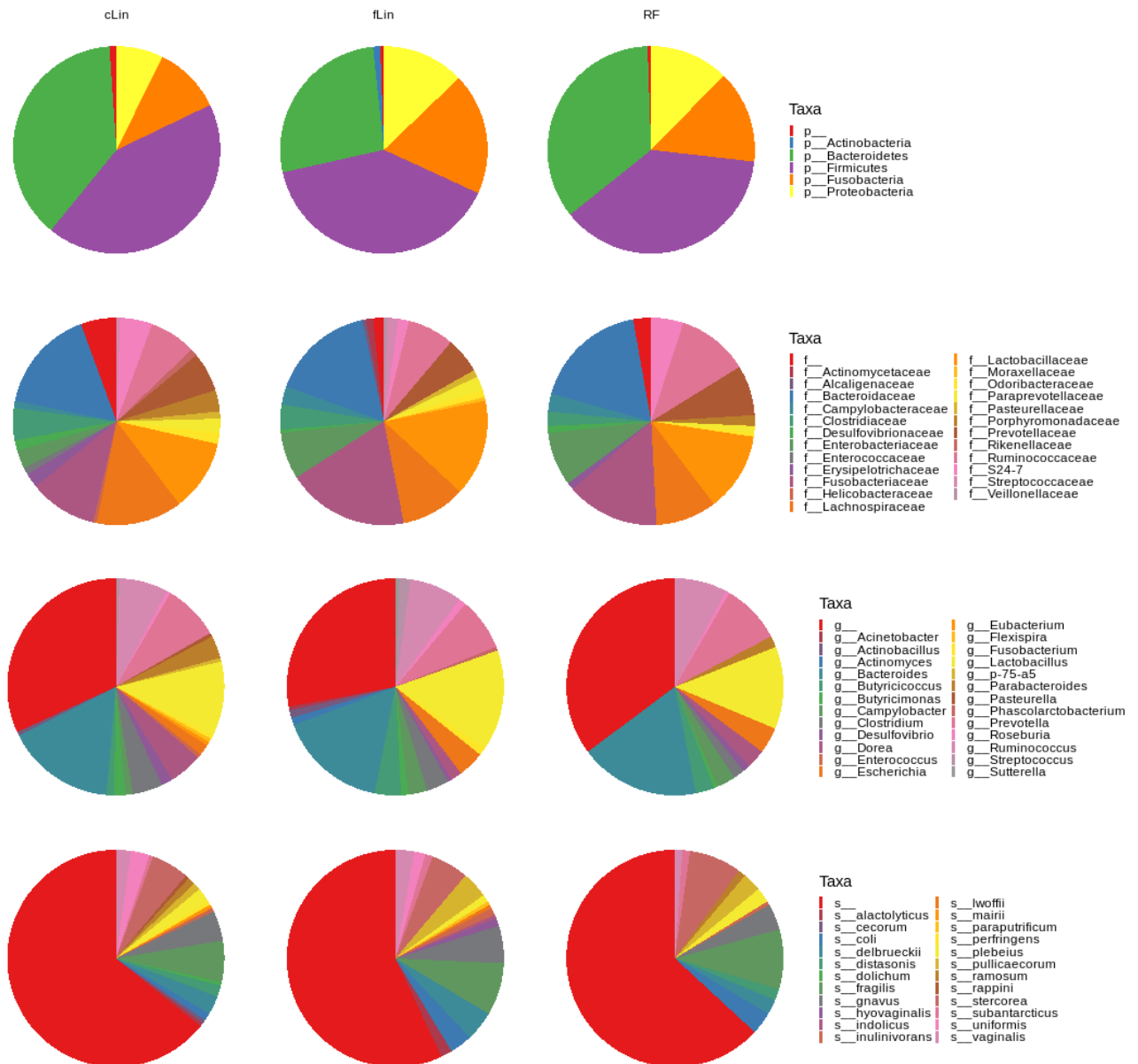


Figure S80. Pig dataset: importance distribution at the Phyla, Family, Genera and Species level of the top 5% for cLin and RF (day 7) and fLin (global).

Supplementary methods S0 – Pig Data obtention and preprocessing

The experiment was conducted at Schothorst Feed Research B.V. facilities where management, environmental and housing factors were controlled for all animals throughout the whole study. Piglets sampling (swabs) was done in healthy piglets across three age strata: within 5 minutes after farrowing (*i.e.* day 0) and at days 3 and 7 post-farrowing. DNA from faecal samples was extracted at IRTA laboratory with the DNeasy PowerSoil Kit (QIAGEN). Extracted DNA was sent to the University of Illinois Keck Center for Fluidigm sample preparation and Illumina sequencing of 16S rRNA gene. Primers targeting the V3 – V4 region (F357 and -R805) were used to amplify a region of 552 base pairs

of the bacterial 16S rRNA gene and sequenced on one MiSeq flowcell for 251 cycles. Sequences corresponding to the V3-V4 region of the 16S rRNA gene were analysed using QIIME2 software (Bolyen *et al.* 2018). The workflow included a quality control step to remove sequences with Phred scores of < 30, trim sequences based on expected amplicon length, remove chimera and merge paired reads. The cleaned 16S rRNA gene sequences were processed into Amplicon Sequences Variants (ASVs) and classified to the lowest possible taxonomic rank using QIIME2 (Bolyen *et al.* 2018) against the GreenGenes Database release 2013-08 (DeSantis *et al.* 2006). Moreover, samples with less than 900 reads were excluded and not considered in posterior analysis. A total of 153 piglets and 3.29·10⁶ reads were retained for subsequent data analyses (Figure 2) after filtering out the low quality reads and samples with less the 900 reads. As expected, all negative control samples (42) were excluded in the quality control, which support the quality of DNA extraction and sequencing processes. Due to the low DNA concentration and the low bacterial biomass some samples from piglets microbiota at day 0 did not pass the filtering process. In the final step of the quality control we also filtered out singletons and doubletons, to finally obtain a total of 3,577 ASVs.

Kernel normalization

Following Shawe-Taylor & Cristianini (2004) kernel matrices were normalized with the cosine transformation:

$$[\hat{\mathbf{K}}]_{ij} = \hat{\kappa}(x_i, x_j) = \frac{\kappa(x_i, x_j)}{\sqrt{\kappa(x_i, x_i)\kappa(x_j, x_j)}}$$

We applied this normalization over the kernel for graphs matrices in Chapter 5 and the cLin kernel matrices of Chapter 6 (cRBF, qJac and JSK give values between [0,1]). The normalized version of a kernel uses the feature map $\hat{\phi}(x) = \frac{\phi(x)}{\|\phi(x)\|}$. That was taken into account in Chapter 6 when computing the variable importances using (3.49).

It can be proved that the proposed categorical kernels (4.7) of Chapter 4 are inherently standardized with the cosine transformation. For instance, if $\kappa(x_i, x_j)$ is the Overlap (3.26) or Jaccard kernels (3.29):

$$\kappa_{cat}(x_i, x_j) = \frac{e^{\gamma \sum_{k=1}^D w_k \kappa(x_{ik}, x_{jk})}}{\sqrt{e^{\gamma \sum_{k=1}^D w_k \kappa(x_i, x_i)} e^{\gamma \sum_{k=1}^D w_k \kappa(x_j, x_j)}}} = \frac{e^{\gamma \sum_{k=1}^D w_k \kappa(x_i, x_j)}}{\sqrt{e^{\gamma \cdot 1} e^{\gamma \cdot 1}}} = \frac{e^{\gamma \sum_{k=1}^D w_k \kappa(x_i, x_j)}}{e^{\gamma}}$$

provided that the sum of weights is equal to 1.

Hyperparameters

Hyperparameter optimization was done through grid search. Performance was measured by 10x10 cross-validation for all HIV sequence data, and 5x5 cross-validation error for the Soil, Smoker and Pig datasets.

For SVM, hyperparameters' values assessed in the course of this work are in Table S3.

For ANN (Chapter 4), we assessed two architectures: ANN1 and ANN3. In ANN1, only the architecture was optimized. The number of layers assessed was 1, 2 and 3, while the number of nodes per layer was 2, 4, 6, 8 or 10. In ANN2, there were three fixed architectures: 30, 20-10 and 30-20-10. For each architecture we assessed hyperparameter λ with possible values 10^{-5} , 10^{-4} , 10^{-3} and 10^{-2} . The best performing combination of architecture and λ was kept for building the final model. The maximum number of epochs was set to 1000.

DT were used in the step 2 of the combiner algorithms of Chapter 4. The minimum number of observations in any terminal leaf was 3. Maximum *complexity parameter* (cp) was 10^{-6} . Trees were pruned to a cp slightly greater than the optimum.

The number of RF trees was 400 in HIV data (Chapter 4) and 1000 in the microbiome data (Chapter 6). Minimum number of observation in each terminal leaf was 5 for regression (HIV and Soil data) and 1 for classification (Smoker and Pig data). Trees were grown to the maximum possible, as we did not set a cutoff for terminal leaves. The number of drawn candidate variables in each split was $D/3$ for regression and \sqrt{D} for classification.

Table S3. SVM candidate hyperparameters' values.

	HIV (Chapters 4 & 5)	Soil	Smoker	Pig
C	0.1, 1, 10, 100		1, 10, 25, 50, 100	
ϵ	10^{-4} , 10^{-3} , 10^{-2} , 10^{-1}	0.01		
γ (RBF and cRBF)	10^{-3} , 10^{-2} , 10^{-1} , 1		10^{-5} , 10^{-4} , 10^{-3} , 10^{-2}	10^{-4} , 10^{-3} , 10^{-2} , 10^{-1}
γ (Categorical kernels)	0.1, 1, 3, 5			
γ (Graph kernels)	10^{-6} , 10^{-5}			
γ (fRBF)				10^{-6} , 10^{-5} , 10^{-4} , 10^{-3}
MKL coefficients (2 sites)			{0.5,0.5},{0.25,0.75}, {0.75,0.25}	
MKL coefficients (4 sites)			{0.25,0.25,0.25,0.25}, {0.35,0.35,0.15,0.15}, {0.15,0.15,0.35,0.35}	

'catkern' vignette

Elies Ramon

2020-08-18

Purpose

The **catkern** package implements some useful kernel functions to handle categorical data. These functions were originally intended to analyze HIV sequence data, in the paper *HIV drug resistance prediction with weighted categorical kernel functions* by E. Ramon *et al.*

Installation

In R console:

```
install.packages("devtools")
devtools::install_bitbucket("elies_ramon/catkern")
```

Package Overview

Main features

- Computes kernel matrices from categorical datasets
- Plots the PCA of categorical datasets
- Permits the presence of ambiguities (*i.e.* more than one category) in the variables
- Allows for variable weighting
- Delivers the variable importance
- Can perform random sampling of ambiguities

These functions were originally intended to analyze HIV sequence data.

Functions provided

- `cmatrix()`
- `kmatrix()`
- `cplot()`
- `rfweight()`
- `mixSample()`

`cmatrix()` is the core function. Its input is a matrix or dataframe with categorical data, and its output a kernel matrix. Two kernel functions can be chosen: Overlap and Jaccard. The categorical variables can be weighted or not (thus giving equal importance to all of them). An external vector of weights can be provided as an argument; otherwise, weights will be automatically computed from the Random Forest variable importances.

`kmatrix()` is analogous to `cmatrix`, but its available kernels are non categorical: the linear and RBF kernels. The input can be numeric or categorical; in the case of categorical input, a one hot encoding is performed. As in `cmatrix`, the variables can be weighted or not. An external vector of weights can be provided as an argument; otherwise, weights will be automatically computed from the Random Forest variable importances.

`cpplot()` plots the kernel PCA of the data. If the target variable is provided, the points will be coloured, allowing the user to check if the individuals that are more similar in their response are also more similar according to the kernel.

`rfweight()` delivers a vector of variable importances. This vector corresponds to the Increase in Node Impurity, a statistic of variable importance computed by the Random Forest method. If needed, a plot of the importances can be delivered additionally.

`mixSample()` takes a categorical dataset with ambiguities as input. If the dataset contains ambiguities/mixtures of categories at some point, the function samples one of the category at random. Its output is the sampled dataset.

Example data

This package contains three categorical datasets with ambiguities: PI, NRTI and NNRTI ([source](#)). These datasets contain the protease (PI) and reverse transcriptase (NRTI and NNRTI) sequences of HIV variants, and its corresponding resistance to several drugs. The sequences with insertions, deletions and truncations in the original dataset have been removed. The datasets are lazy loaded and can be used anytime typing PI, NRTI or NNRTI.

Usage

Sampling a dataset

A dataset can have ambiguous or mixed categories in some positions; for example:

```
library(catkern)
Data <- na.omit(PI[, -c(1:2,4:9)])
print(Data[1:7,65:80])
```

P64	P65	P66	P67	P68	P69	P70	P71	P72	P73	P74	P75	P76	P77	P78	P79
I	E	I	C	G	H	K	A	IV	G	T	V	L	V	G	P
L	E	IV	C	G	H	K	V	I	S	T	V	L	V	G	P
I	E	I	C	G	H	K	T	I	G	T	V	L	V	G	P
I	E	I	C	G	H	K	V	I	S	T	V	L	V	G	P
I	E	I	C	G	H	K	A	I	G	T	V	L	I	G	P
I	E	I	C	G	Q	K	AV	I	G	T	V	L	I	G	P
V	E	I	C	G	H	K	I	I	T	T	V	L	V	G	AS

To take one of the categories at random and obtain a “clean” dataset, do:

```
smp1Data <- mixSample(dataset = Data, y=1)
print(smp1Data[1:7,65:80])
```

P64	P65	P66	P67	P68	P69	P70	P71	P72	P73	P74	P75	P76	P77	P78	P79
I	E	I	C	G	H	K	A	V	G	T	V	L	V	G	P
L	E	V	C	G	H	K	V	I	S	T	V	L	V	G	P
I	E	I	C	G	H	K	T	I	G	T	V	L	V	G	P
I	E	I	C	G	H	K	V	I	S	T	V	L	V	G	P
I	E	I	C	G	H	K	A	I	G	T	V	L	I	G	P
I	E	I	C	G	Q	K	V	I	G	T	V	L	I	G	P
V	E	I	C	G	H	K	I	I	T	T	V	L	V	G	A

where the y argument are the columns that don't have to be sampled (in this case, the target variable).

Computing the variable importance

We can compute a measure of variable importance as:

```
## Convert the sampled dataset from character to factor
smp1Data[, -1] <- lapply(smp1Data[, -1], as.factor)
weights <- rfweight(x = smp1Data[, -1], y = smp1Data[, 1], plot=TRUE)
weights <- weights/sum(weights) ## If we want the relative importances
```

`rfweight()` returns a vector. If `plot = TRUE`, a simple plot of the importances will be delivered as well.

Computing the kernel matrices

There are four kernel functions to choose from: two specific for categorical data (Overlap and Jaccard) and two non categorical kernels (Linear and RBF). In the former, the appropriate **catkern** function is `cmatrix()`; in the latter is `kmatrix()`. In any case, the usage of the two is way similar.

Categorical kernels:

The simpler call to `cmatrix()` is:

```
ovrMatrix <- cmatrix(data = smp1Data[, -1], kernel = "o")
```

There are two mandatory arguments: `data`, the dataset, and `kernel`, which is the desired kernel ("o" for Overlap and "j" for Jaccard). The target variable **must** be absent from the dataset. The output is a normalized kernel matrix, of dimensions $n \times n$.

The Jaccard kernel has an interesting property: as it is a kernel on sets, it can handle ambiguities/mixtures in the categories. This implies that we do not need to remove them from our dataset, so no information is lost. When there are not ambiguities, the Jaccard kernel is equivalent to Overlap kernel.

```
jacMatrix <- cmatrix(data = Data[, -1], kernel = "j")
```

In the previous invocations to `cmatrix()`, all variables had the same weight when computing the kernel matrix. But if we know that some variables are more important than others to the problem at hand, we can weight them. This is done using the `comp` argument. The two possible scenarios are:

· **Known importances:** If we already have a vector of variable importance, it can be passed as input to `cmatrix()`:

```
wOvMatrix <- cmatrix(data = smplData[, -1], kernel = "o", comp = "w", coefficient = as.vector(weights))
```

· **Unknown importances:** `cmatrix()` will call `rfweight()` to obtain the variable importances, and then will use them to weight the variables during the kernel computing.

```
wOvMatrix2 <- cmatrix(data = smplData[, -1], kernel = "o", comp = "w", y = smplData$ATV)
```

Finally, the last argument we can pass to the `cmatrix()` function is the `g` argument. When a value for `g` is provided, the Overlap and the Jaccard kernels computed are:

$$k(x,y) = \exp(\gamma k_c(x,y)) / \exp(\gamma)$$

where $k_c(x,y)$ are either the Jaccard or Overlap kernels, and γ is a hyperparameter. This modified Overlap and Jaccard are non linear and analogous to the RBF kernel.

```
gJacMatrix <- cmatrix(data = Example[, -1], kernel = "j", g = 1, y = data$ATV)
```

Non categorical kernels:

`kmatrix()` usage is pretty similar to `cmatrix()`. Main differences are:

- The kernel matrix is not computed by `kmatrix()` itself, but passed to the **kernlab** package.
- The linear and RBF kernels are intended to operate in numeric data. If `kmatrix()` receives a categorical dataset, this dataset **must** contain factors. Then, `kmatrix()` will perform a one hot encoding (*i.e.* convert the categorical variables to dummy variables).
- `g` argument is mandatory when RBF kernel is chosen.

An example:

```
rbfMatrix <- kmatrix(data = smplData[, -1], kernel = "r", g = 0.1, comp = "w", y = smplData$ATV)
```

Prediction with the kernel matrices

To use the kernel matrix for prediction, we strongly recommend using the R package **kernlab**. For example:

```
library(kernlab)
```

```

## Training/test indexes
n <- nrow(jacMatrix)
trInd <- sort(sample(n,0.7*n,replace=FALSE)) # Training: 70% dataset`
teInd <- (1:n)[-trInd] # Test indexes: 30% dataset`
trCmatrix <- jacMatrix[trInd,trInd] # Training kernel matrix`
teCmatrix <- jacMatrix[teInd,trInd] # Test kernel matrix`

## Train a regression SVM with our kernel matrix:
model <- ksvm(trCmatrix,y = Data$ATV[trInd],type="eps-svr",kernel="matrix")

## Predict
teCmatrix <- teCmatrix[,SVindex(model),drop=FALSE]
teCmatrix <- as.kernelMatrix(teCmatrix)
predict(model,teCmatrix)

```

Visualizing the kernel-PCA of the dataset

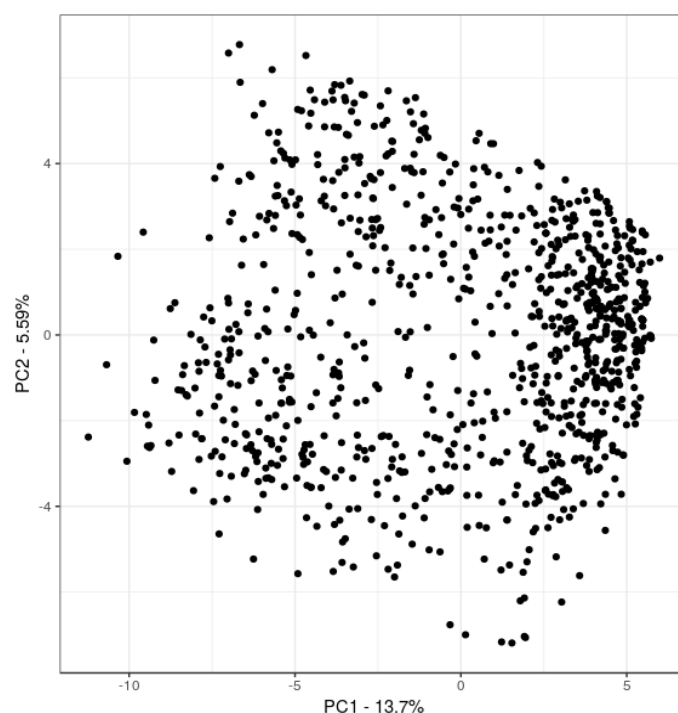
The kernel matrix obtained either with `cmatrix` or `kmatrix` can be used to obtain a kernel-PCA plot. This can be useful to visualize our categorical data and to detect outliers.

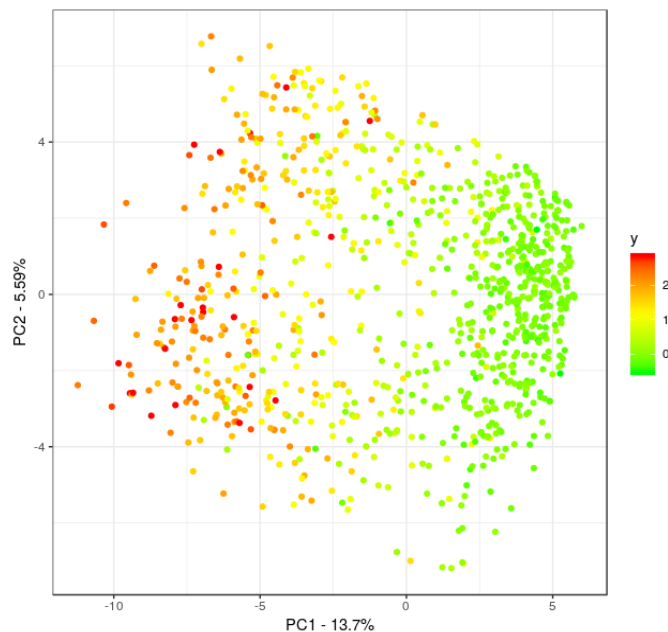
In case we have a target/response variable, we can pass it to the function (`y` argument). This information will be used to paint the dots, allowing us to check visually if the sequences that are considered more similar by our kernel function are also similar in regards to the response variable.

```

cplot(jacMatrix)
cplot(jacMatrix, y = Data$ATV, col = c("green","yellow","red"))

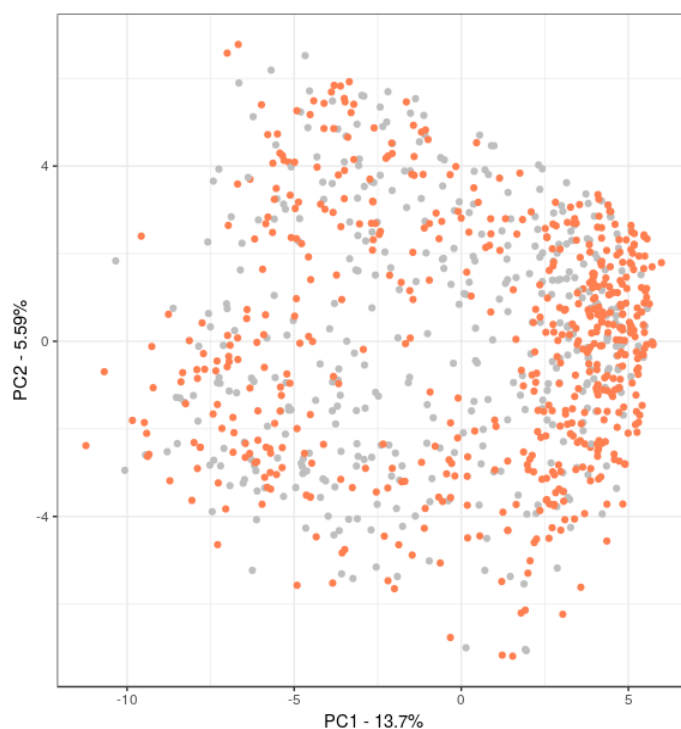
```

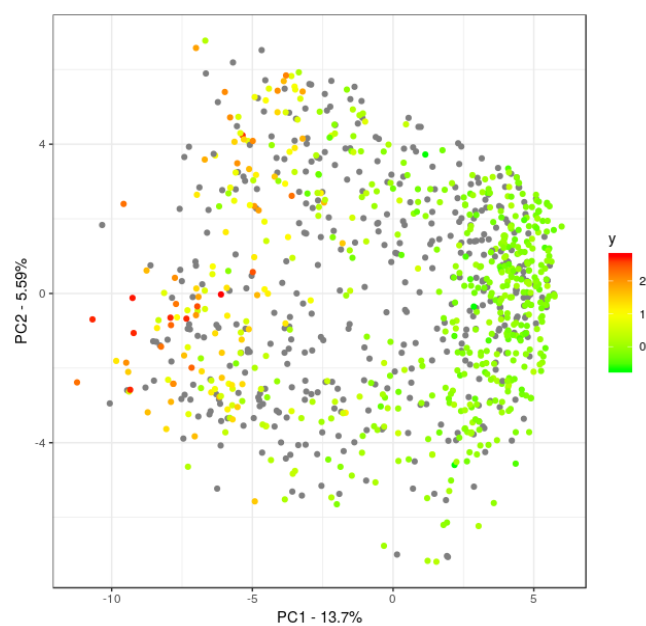




Also, we can paint with a different color the data with an unknown response value with `na.col`. This can be done either we are using a color spectrum or not:

```
# A different drug of the same database, with more NAs.
drv <- PI[complete.cases(PI[, -c(1:2, 4:9)]), ]$DRV
cplot(jacMatrix, y = drv, col = "coral", na.col = "grey", legend = FALSE
)
cplot(jacMatrix, y = drv, col = c("green", "yellow", "red"), na.col = "grey50")
```





The matrix containing the PCs, the eigenvalues and the projected data can be obtained using the `kpca` function of **kernlab**:

```
jacMatrix <- as.kernelMatrix(jacMatrix)
catPCA <- kpca(jacMatrix, kernel = matrix)
pcv(catPCA) ## ALL PCs
eig(catPCA) ## Eigenvalues
rotated(catPCA) ## Projection coordinates
```

Appendix

Package dependencies

catkern strongly relies in three R packages: **kernlab**, to perform the kernel-PCA and to compute the linear and RBF kernel; **ggplot2**, to plot the kernel-PCAs, and **randomForest**, to compute the variable importances.

'kernInt' vignette

Elies Ramon

2020-08-18

Purpose

kernInt uses the kernel framework to unify supervised and unsupervised microbiome analyses, while paying special attention to spatial and temporal-related samples integration.

Installation and loading

In R console:

```
if (!requireNamespace("devtools")) install.packages("devtools")
devtools::install_github("elies-ramon/kernInt")
```

If the metagenomeSeq package was not installed previously:

```
if (!requireNamespace("BiocManager")) install.packages("BiocManager")
BiocManager::install("metagenomeSeq")
```

Once the package is installed, it can be loaded anytime typing:

```
library(kernInt)
#> Loading required package: kernLab
#> Registered S3 method overwritten by 'GGally':
#> method from
#> +.gg ggplot2
#> sROC 0.1-2 Loaded
```

Package Overview

Main features

- Implementation of kernels for compositional data.
- Kernels derived from classical ecology distances, as Jaccard and Jensen-Shannon, are also available.
- Implementation of kernels specific for functional data.
- A previously unpublished longitudinal pig gut microbiome dataset
- Automatic training/test splitting of the input data, k-Cross Validation and SVM classification and regression.
- Microbial signatures of the classification and regression models.
- Outliers / Novelty detection via SVMs.
- Integration of data from different sources via Multiple Kernel Learning (MKL)

Available kernels

- Widely used kernels for real vectors: the linear (*lin*) and RBF (*rbf*) kernels.
- Kernels for compositional data: the compositional linear *clin* and Aitchison-RBF *crbf* kernels. Both kernels can be implemented directly from the raw counts from tables.
- Kernels derived from ecological distances: the Jaccard *jac* and Jensen-Shanon *jsk* kernels. Metagenomic data should be normalized before applying these kernels, for example by using CSS (cumulative sum scaling).
- Kernels specific for functional data: the functional linear *flin* and functional RBF *frbf* kernels. They need functions (for instance, representing time series) as an input.

Example data

We offer three metagenomic datasets with the package: a single point soil dataset, a human health dataset with an spatial component, and a novel longitudinal dataset concerning pig production. Also, to better illustrate the longitudinal treatment of data, we include the classical Berkeley Growth Dataset.

Soil data: Bacterial abundances (raw counts) in 88 soils from across North and South America. Metadata as soil pH, annual season precipitation and temperature, country, elevation, etc. is available. The dataset can be accessed typing *soil*:

Smokers: Microorganism abundances of oro and nasopharynx in 29 smokers and 33 nonsmokers. The dataset can be accessed typing *smoker*.

Pig data: Previously unpublished longitudinal gut microbiome dataset of 153 piglets during their first week of life. The dataset can be accessed typing *pig*.

Growth: Berkeley longitudinal height data of 54 girls and 39 boys (93 individuals in total) from ages 11 to 18. The dataset can be accessed typing *growth*.

Usage - Standard case

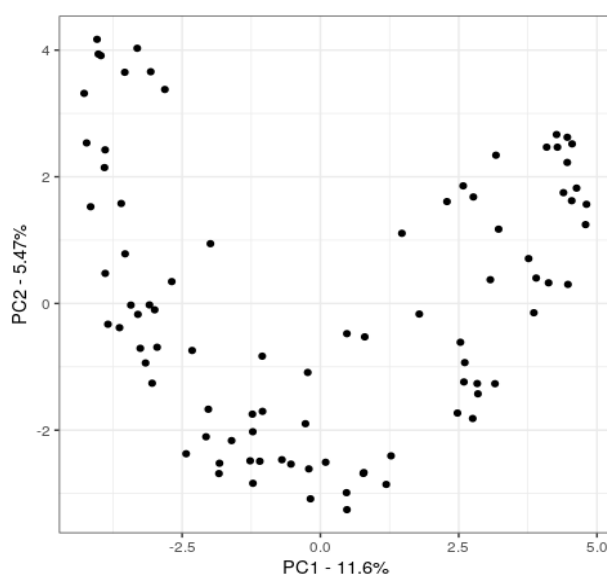
We refer data consisting in a sample of individuals (for example: human, animals, soil sites...) with no repeated measures as “the standard case”. The format of the input table must be with sample names as rows and taxon/OTUs as columns. The input of the ‘kernInt’ functions, thus, has class *data.frame* or *matrix*. We will use as example for this case the *soil* data.

kernel PCA

As the standard PCA, kernel PCA can be used to summarize, visualize and/or extract features of a dataset. Data can be projected in a linear or nonlinear way, depending on the kernel used. When the kernel is the standard linear (*kernel=lin*), kernel PCA is equivalent to standard PCA.

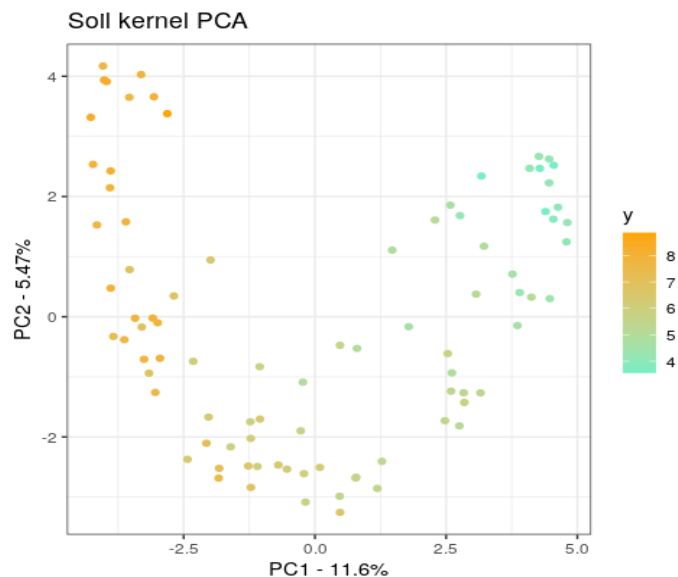
The *kernPCA()* function have two mandatory arguments - *data* and *kernel*:

```
kernPCA(data=soil$abund, kernel="clin")
```



The rest of arguments customize the plot. For example, the dots can be coloured according to a desired variable, which can be continuous or discrete. Here we show a kernel PCA of soil samples in which each sample is coloured according to its pH:

```
kernPCA(data=soil$abund, kernel="clin", y=soil$metadata$ph,
         col=c("aquamarine2", "orange"), title = "Soil kernel PCA", legend =
TRUE)
```

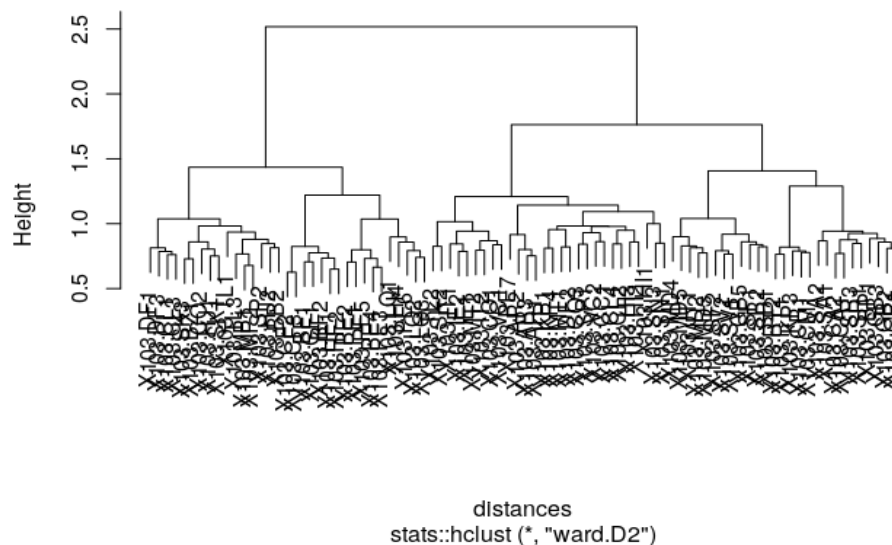


The projected data can be retrieved setting `pLot=FALSE`.

Clustering

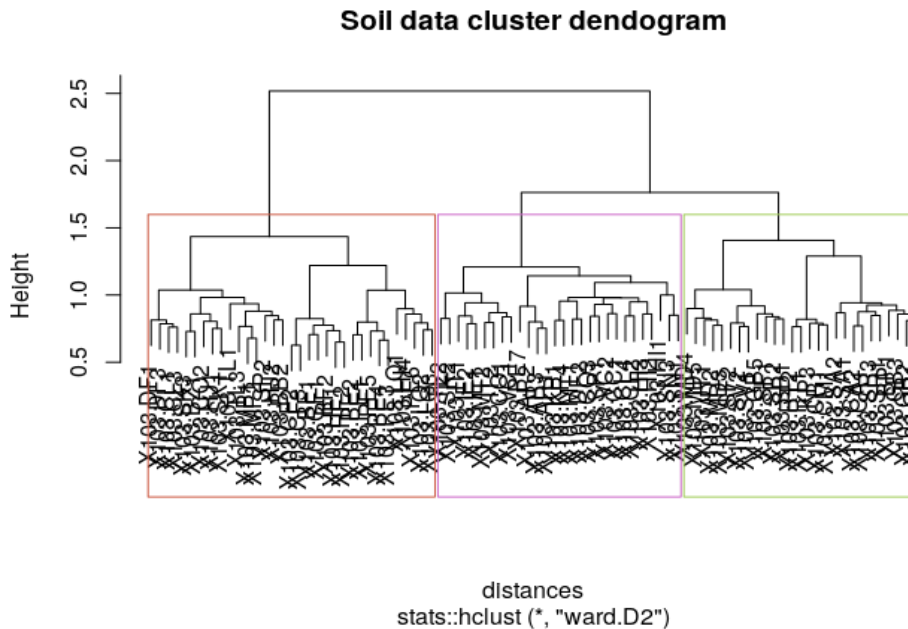
A dendrogram plot presenting a hierarchical clustering of data can be obtained with:

```
soilData <- CSSnorm(data=soil$abund) ### CSS normalization
hklust(data=soilData, kernel="jac")
```



Additional arguments allow changing the agglomeration method, outlining clusters or customizing the plot:

```
hclust(data=soilData, kernel="jac", title = "Soil data cluster dendrogram", cut=3, colors=c("coral3", "orchid3", "darkolivegreen3"))
```



The dendrogram object can be retrieved setting `plot=FALSE`.

SVM regression

SVM regression is performed with the `regress()` function. `regress()` performs automatic training/test splitting of the input data, k-cross validation if requested, and regression with optimal hyperparameters. To perform MKL, go to [MKL section](#).

The most basic call only needs three arguments: `data` (predictor variables; e.g. taxonomic abundances), `y` (target variable; e.g. a phenotype) and `kernel`.

For example, if we want to predict the pH of soil (`y`) from the abundances (`data`) using the compositional linear kernel:

```
modelreg <- regress(data=soil$abund, kernel="clin", y=soil$metadata$ph)
```

If the user has a pre-computed kernel matrix at hand, it can be passed as input to `data`. `kernel` should then be turned to `kernel="matrix"`.

The SVM hyperparameters Cost (C) and Epsilon (E) can be specified, thus tuning how the model will adjust to the data. Roughly speaking, C is the cost of doing errors. In SVM regression models, differences between predicted and actual values smaller than E are not considered errors. Both increasing C and decreasing E increases the complexity of the model and the danger of overfitting.

```
regress(data=soil$abund, kernel="clin", y=soil$metadata$ph, C=5, E=0.00)
```

In addition, a generic kernel hyperparameter (H) can be specified. For example, if the chosen kernel is RBF, H will be interpreted as γ : $\text{RBF}(x,y) = \exp(-\gamma \cdot \|x-y\|^2)$

```
regress(data=soil$abund, y=soil$metadata$ph, kernel="crbf", C=5, H=0.1)
```

k-Cross Validation can be performed to obtain the optimal hyperparameters. This is done providing an argument to `k`. Then, the best hyperparameter value (or the best combination of hyperparameter values) will be selected among the values provided by the user:

```
regress(data=soil$abund, y=soil$metadata$ph, kernel="clin", C=c(1,5,10), E=c(0.001,0.1), k=5)
```

Training/test splitting is controlled with the `p` argument and the rownames of `data`. If given a numeric value between 0 and 1, `regress()` will consider it the proportion of data instances for the test set, and will do a random splitting. Default is `p=0.3`. Otherwise, a vector containing the indexes or the names of the rows of the test set is also allowed.

If the input data has repeated rownames, `regress()` will consider that the rows that share id are repeated measures coming from the same individual. The function will ensure that all repeated measures are used either to train or to test the model, but not for both, thus preserving the independence of the training and test sets. However, users can enter the test partition, setting `p` to be a numeric (row indexes) or character (rownames) vector. The remaining data will be used as training.

Output

A list containing:

- `$nmse`: Normalized mean squared error over the test data. This permits evaluating how good the model is at predicting.
- `$hyperparam`: Hyperparameters' values used to build the model and their cross-validation error (if applicable).
- `$prediction`: Predicted and true values (test set). Rownames correspond to the indexes in the original data.
- `$var.imp`: The variable importance (*e.g.* microbial signature) if a linear or linear-like kernel is used. To present relative values of

SVM classification

SVM classification is performed via the `classify()` function. Both binary or multiclass classification (one-vs-one) are supported. One-class classification is available in the `outliers()` function.

The usage of `classify()` is for the most part identical to that of `regress()`. For example, to predict if a certain soil came from forest, tropical, shrubland or grassland environment:

```
modelclas <- classify(data=soil$abund ,y=soil$metadata[ ,"env_feature"],
kernel="clin")
```

Probabilistic classification is available setting `prob=TRUE`:

```
classify(data=soil$abund ,y=soil$metadata[ ,"env_feature"],kernel="clin,
prob=TRUE)
```

The available hyperparameters are H and C (where C is the cost of missclassification). Also, `classify()` supports several methods to deal with imbalanced data:

-Class weighting:

```
classify(data=soil$abund ,y=soil$metadata[ ,"env_feature"],kernel="clin,
classimb="weights",C=c(0.001,0.01),k=10)`
```

-Undersampling:

```
classify(data=soil$abund ,y=soil$metadata[ ,"env_feature"],kernel="clin,
classimb="ubUnder",C=c(0.001,0.01),k=5)
```

-Oversampling:

```
classify(data=soil$abund ,y=soil$metadata[ ,"env_feature"],kernel="clin,
classimb="ubOver",C=c(0.001,0.01),k=5)
```

Output

A list containing:

- `$conf.matrix`: Confusion matrix (true versus predicted) for the test data. This permits evaluating how good the model is at predicting.
- `$hyperparam`: Hyperparameters' values used to build the model and their cross-validation error (if applicable).
- `$prediction`: Predicted and true values (test set). Rownames correspond to the indexes in the original data. If `prob=TRUE`, the probability of each observation to belong to a given class.
- `$var.imp`: The variable importance (e.g. microbial signature) if a linear or linear-like kernel is used.

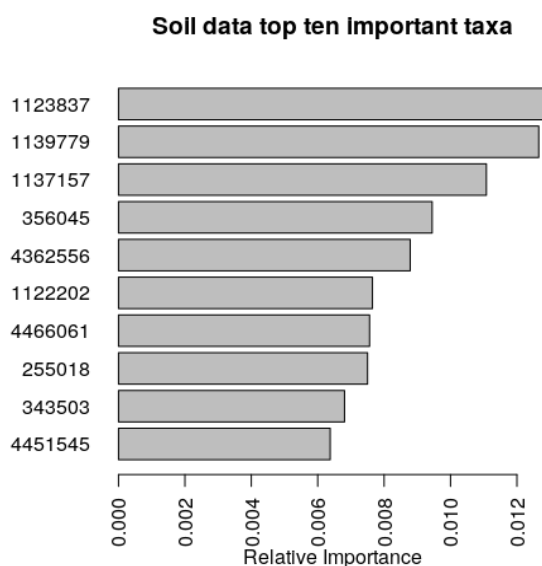
Variable importances

Following [Guyon et al. \(2002\)](#), we can obtain the importance of a variable in a SVM model as:

```
imp <- modelreg$var.imp^2
```

Then, to generate a plot of the 10 most important features:

```
imp <- imp/sum(imp) ## To give relative importances
imp10 <- sort(imp,decreasing = TRUE)[1:10]
par(mar=c(4,5,4,2))
barplot(sort(imp10),horiz = TRUE,las=2,
main="Soil data top ten important taxa",xlab = "Relative Importance")
```



```
## To know the taxonomic classification of the ten most important OTUs,
do:
soil$taxonomy[names(imp10),]
```


Outlier detection

The `outliers()` function can be used either in a supervised or in an unsupervised way.

In the latter approach, the most basic call to this function needs two arguments: `data` (predictor variables) and `kernel` (the kernel function used). Then, the function will return the data outliers:

```
outliers(data=soil$abund ,kernel="clin")
```

The nu hyperparameter (ν) and a gamma hyperparameter H can be entered:

```
outliers(data=soil$abund,kernel="crbf",nu=0.2,H=0.05)
```

If an argument for y is provided, `outliers()` functions as an one-class SVM. In that case, cross-validation will be performed if k has an argument. Also, p stands for the proportion (or indexes, or rownames) of data instances reserved for the test set.

```
outliers(data=soil$abund,y=soil$metadata[ , "env_feature"],kernel="clin",
nu=c(0.1,0.2),k=5)
```

Usage - Multiple data sources

The availability of multiple types of data for the same sample of individuals is becoming increasingly common. For instance, for several patients, we may have types as diverse as metagenomic data, metabolomics and blood analysis. Another example is when we have spatial-related samples, as in the *smoker* dataset. There, each individual is sampled in four body sites: left and right nasopharynx, and left and right oropharynx. Combining this different kind of data types can be tricky for most methods, but the kernel framework offers a straightforward solution: using specific kernels for each data source and then directly combining the kernel matrices. The process of obtaining an optimal convex combination is known as MKL (Multiple Kernel Learning).

MKL

MKL is available to `classify()`, `regress()` and `outliers()`. All features of these two functions are available when performing MKL. To do MKL, the `data` argument must be a list of length > 1 . Each element of the list should be a data.frame or matrix, and rows should coincide. If `kernel="matrix"` data may be a three-dimensional array. `kernel` argument may contain only one kernel name (thus implying that the kernel is the same for all datasets) or a vector of kernel names. That way a different kernel will be applied to each data type. For example, if we have a list of metagenomic abundances, as in `smoker$abund`:

```
css_data <- lapply(smoker$abund,CSSnorm) ## CSS normalization
smoking <- smoker$metadata$smoker[seq(from=1,to=62*4,by=4)] ## Target v
variable
classify(data=css_data, y=smoking, kernel="jac")
```

```
## This is equivalent to:
```

```
jacc_kern <- sapply(smoker$abund, qJacc, simplify = "array") ## 3D Jaccard
array
classify(data=jacc_kern, y=smoking, kernel="matrix")
```

The *coeff* argument is for the weight of each data type in the kernel combination. When absent, the mean across all kernel matrices is performed.

```
classify(data=jacc_kern, y=smoking, coeff = c(0.1,0.2,0.4,0.3) , kernel="
matrix")
```

The use of additional arguments as *C*, *E*, *p*... remains the same. Kernel(s)' generic hyperparameter in the MKL usage, *H* must be NULL or, either, a list so each element is the hyperparameter applied of each kernel function:

```
h <- list(nasL=0.001, nasR=0.1, oroL=0.01, oroR=0.01)
classify(data=smoker$abund, y=smoking, coeff = c(0.1,0.2,0.4,0.3), C=10,
H=h, kernel="crbf")
```

In the case of k-Cross-Validation:

```
h <- list(nasL=c(0.01,0.001), nasR=c(0.01,0.001), oroL=0.0001, oroR=0.0001)
classify(data=smoker$abund, y=smoking, coeff = c(0.1,0.2,0.4,0.3), C=c(1,
10), H=h, kernel="crbf", k=5)
```

Side functions for data fusion

KInt() and *fuseData()* return a fused kernel matrix, but the former uses kernel matrices as input while the latter needs a list with the different data sources.

```
KInt(jacc_kern)
```

We can use different kernel functions for each type of data; for example, the Jaccard kernel for the first data type and the compositional linear kernel for the second:

```
fuseData(DATA=css_data, kernel=c("jac", "clin"))
```

The former command consider the two sources equally important. If not, we can state the weights:

```
fuseData(DATA=css_data, kernel=c("jac", "lin"), coeff=c(0.9,0.1))
```

Usage - Longitudinal data

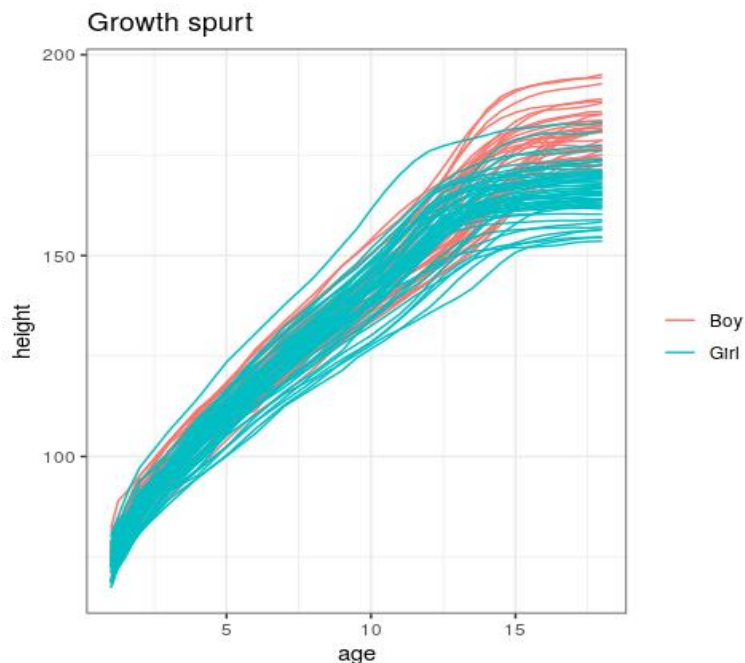
In this case, we have repeated samples for the same individuals indexed by time. Take as an example the *growth* dataset, which follow the growth over time of several girls and boys from birth until they turn 18-years-old:

```
library(ggplot2)
#>
#> Attaching package: 'ggplot2'
#> The following object is masked from 'package:kernlab':
#>
#> alpha
```

```

target <- rep("Girl", nrow(growth))
target[ grep("boy", rownames(growth))] <- "Boy"
target <- as.factor(target)
growplot <- data.frame(rownames(growth), growth, target=target)
ggplot(growplot, aes(x = age, y = height, group=rownames.growth., color =
target)) +
  geom_line()+ ggtitle("Growth spurt") + theme_bw()+ theme(legend.title
= element_blank())

```



This kind of data is called longitudinal and is typically represented as functions. The coefficients of a simple polynomial fitting can be obtained with `Lsq()` by least squares. For the `growth` dataset, we interpolate the growth curves with a polynomial of degree 2:

```

growth2 <- growth
colnames(growth2) <- c("time", "height")
growth_coeff <- lsq(data=growth2, degree=2)

```

The kernel framework allows direct handling of complex data types as functions using specific kernels. 'kernInt' implements the functional linear (*kernel=flin*) and functional RBF (*kernel=frbf*) kernel. When used, a *domain* argument to evaluate the functions (e.g. a time interval) has to be provided. For instance, we use `classify` to predict from the growth curve if the individual was a girl or a boy:

```

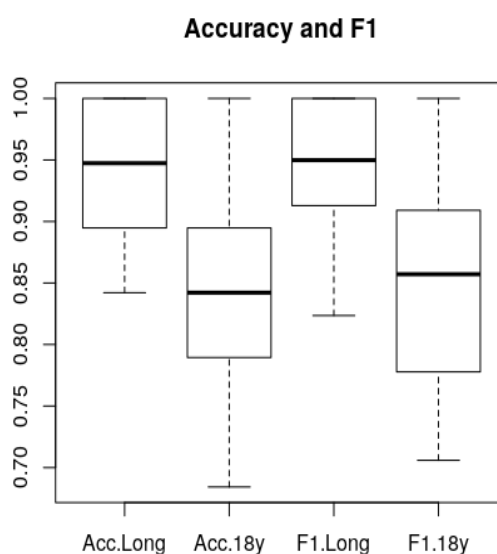
target <- rep("Girl", 93)
target[ grep("boy", rownames(growth_coeff$coef))] <- "Boy"
target <- as.factor(target)
cm <- array(0, dim=c(2,2,40))
for(i in 1:40) {
  model <- classify(data=growth_coeff, kernel="frbf", C=c(1,5,10,50), H=0.
0001, domain=c(1,18), y=target, k=5)
  cm[, , i] <- model$conf.matrix
}

```

We may compare this approach to perform a nonlongitudinal prediction using only the last time point (age 18):

```
cm2 <- array(0,dim=c(2,2,40))
for(i in 1:40) {
  model <- classify(data=matrix(growth[which(growth[,1]==18),2]),kernel=
"rbf",C=c(1,5,10,50), H=0.001, y=target,k=5)
  cm2[, ,i] <- model$conf.matrix
}

acc <- matrix(0,nrow=40,ncol=4)
colnames(acc) <- c("Acc.Long", "Acc.18y", "F1.Long", "F1.18y")
for(i in 1:40) {
  acc[i,1] <- Acc(cm[, ,i])
  acc[i,2] <- Acc(cm2[, ,i])
  acc[i,3] <- F1(cm[, ,i])
  acc[i,4] <- F1(cm2[, ,i])
}
boxplot(acc,main="Accuracy and F1")
```



It can be observed that taking account all time points delivers a much better prediction performance than keeping a single time point, even if it is that of maximum separation between the two groups.

Additional help

A thorough, argument-by-argument documentation is available for each function with:

```
help(regress) ## or the specific name of the function
?regress
```

The documentation of the example datasets is available in an analogous way, typing:

```
help(soil) ## or the specific name of the example dataset  
?soil
```

ACKNOWLEDGEMENTS

