PCM with 'camb'
Chemistry **A**ware **M**odel **B**uilder
Cambridge. November 2013

Isidro Cortes-Ciriano*[1,3] and Daniel Murrell†[2,3]

[1] *Unite de Bioinformatique Structurale, Institut Pasteur and CNRS UMR 3825, Structural Biology and Chemistry Department, 25-28, rue Dr. Roux, 75 724 Paris, France.*
[2] *Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Cambridge, United Kingdom.*
[*] *Equal contributors*

November 21, 2013

Firstly, we load the package and set the working directory:

1 Compounds

1.1 Reading and Preprocessing

```
smiles <- read.table("smiles_COX.smi", header = FALSE)
StandardiseMolecules(structures.file = "smiles_COX.smi",
    standardised.file = "smiles_COX_processed.sdf",
    is.training = TRUE)
```

1.2 PaDEL Descriptors

*isidrolauscher@gmail.com
†dsmurrell@gmail.com

```
descriptors_COX <- GeneratePadelDescriptors(standardised.file = "smiles_COX.smi",
    threads = 1)
descriptors <- RemoveStandardisedPrefix(descriptors)
saveRDS(descriptors, file = "descriptors.rds")
descriptors <- readRDS("descriptors.rds")
```

1.3 Circular Morgan Fingerprints

```
Sys.setenv(RDBASE = "/usr/local/share/RDKit")
Sys.setenv(PYTHONPATH = "/usr/local/lib/python2.7/site-packages")
# fps_COX_512 <-
# MorganFPs(bits=512,radius=2,type='smi',mols='smiles_COX.smi',
# output='COX',keep='hashed_counts')
# saveRDS(fps_COX_512,file='fps_COX_512.rds')
fps_COX_512 <- readRDS("fps_COX_512.rds")
```

2 Targets

2.1 Read and Preprocessing

We read the read the amino acids from a .csv file:

```
amino_accompound_compound_IDs <- read.table("AAs_COX.csv",
    sep = ",", header = TRUE, colClasses = c("character"),
    row.names = 1)
amino_accompound_IDs <- amino_accompound_IDs[, 2:ncol(amino_accompound_IDs)]
```

```
amino_accompound_IDs_zscales <- AA_descs(Data = amino_accompound_IDs,
    type = "Z3")
```

Ensuingly, we save the descriptors in a .rds file.

```
# saveRDS(amino_accompound_IDs_zscales,file='Z3_COX.rds')
amino_accompound_IDs_zscales <- readRDS("Z3_COX.rds")
```

2.2 Reading the Data-set Information

Now, we are going to read the file with the information about the dataset, namely: target names, bioctivities, etc.. Be careful: when reading smiles from a .csv file into an R dataframe, the smils are clipped after a hash ('#') symbol. Good practice: also keep the smiles alone in a {.smi,.smiles} file.

```
dataset <- readRDS("COX_dataset_info.rds")
bioactivity <- dataset$standard_value
```

The bioactivity is in nM. We convert it to pIC50:

```
bioactivity <- bioactivity * 10^-9
bioactivity <- -log(bioactivity, base = 10)
```

3 Data-set Visualization

We can have a look at the response variable:

```
DensityResponse(bioactivity, xlab = "pIC50", main = "",
    ylab = "Densitiy", TitleSize = 26, XAxisSize = 20,
    YAxisSize = 20, TitleAxesSize = 24)
```

Plotting a PCA analysis of the target descriptors gives:

```
target_PCA <- PCAProt(amino_accompound_IDs_zscales,
    SeqsName = dataset$accession)
PCAProtPlot(target_PCA, PointSize = 8, main = "", TitleSize = 26,
    XAxisSize = 20, YAxisSize = 20, TitleAxesSize = 24)

## [1] "Sequence names provided in the third column of the data.frame"
```

Similarly, we can analyze the chemical space by calculating pairwise compound similarities based upon the compound descriptors. In this case, we use the Jaccard metric to calculate the distance between compounds.

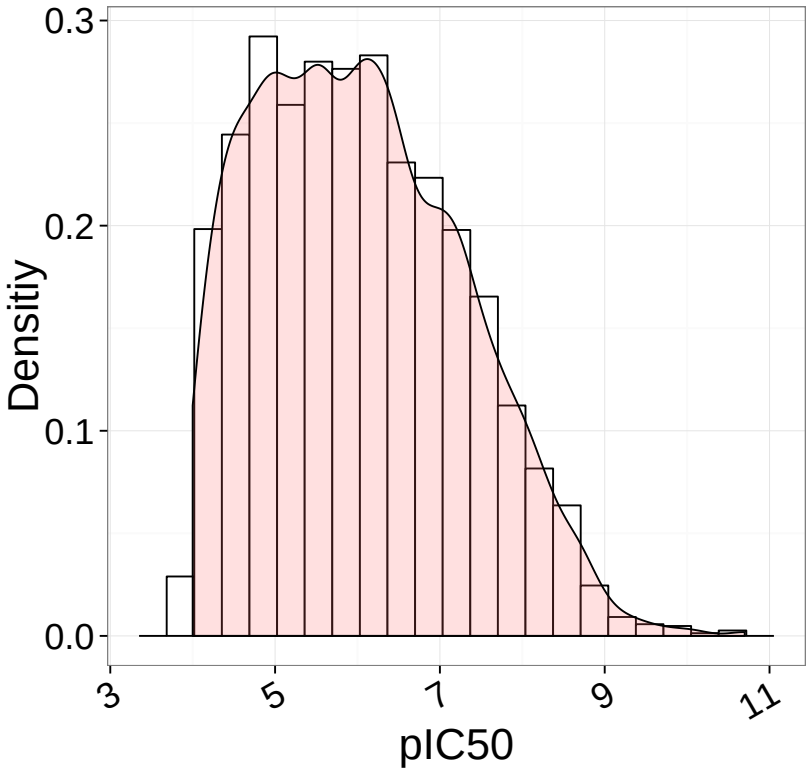

Figure 1: Bioactivity Distribution

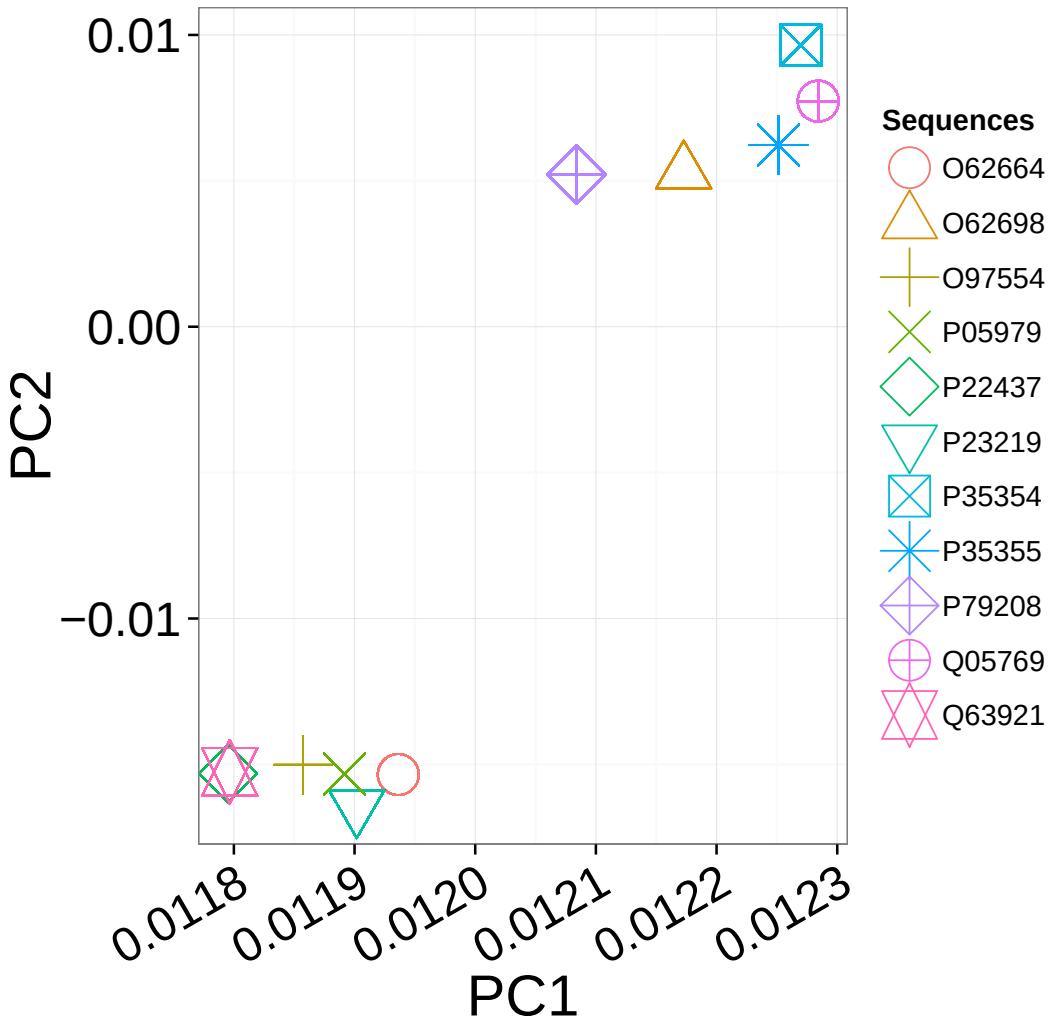

Figure 2: PCA Analysis on the Amino Acid Descriptors

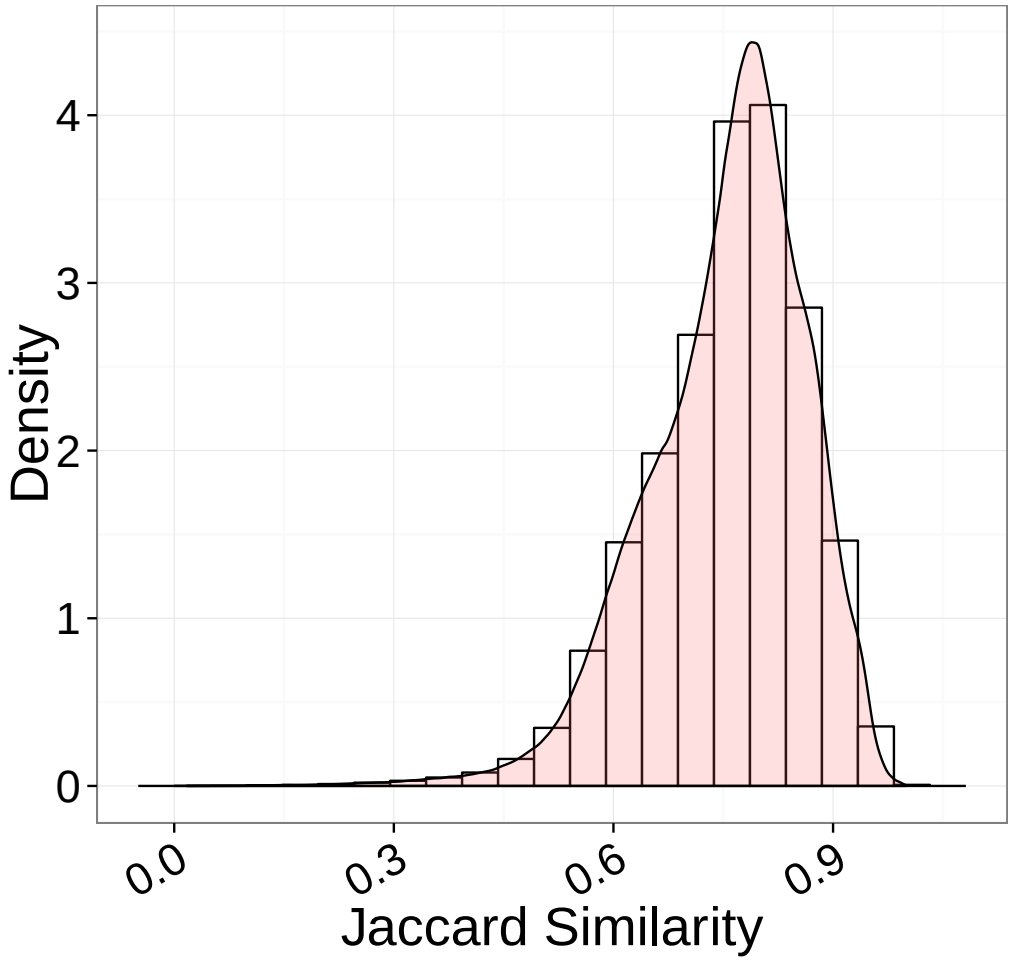

Figure 3: Pairwise Compound Jaccard Similarity

```
# pw_dist_comp_fps <-
# PairwiseDist(fps_COX_512,method='jaccard')
# saveRDS(pw_dist_comp_fps,file='pairwise_dist_COX.rds')
pw_dist_comp_fps <- readRDS("pairwise_dist_COX.rds")
PairwiseDistPlot(pw_dist_comp_fps, xlab = "Jaccard Similarity",
    ylab = "Density", TitleSize = 26, XAxisSize = 20,
    YAxisSize = 20, TitleAxesSize = 24)
```

Before any modeling attempt, it is interesting to know which is the maximum performance achievable *on the basis* of the available data.
By that, we consider the experimental uncertainty and the size of our data-set. In this

case, a Gaussian Process (GP) model was trained in Matlab (data not shown) where the experimental uncertainty was optimized as a hyperparameter. The obtained value was 0.60.
This value is in accordance with recently prublished value of 0.68 for public IC50 data. With the function 'MaxPerf', we can calculate the manximum achievable performance:

```
MaxPerf(meanNoise = 0, sdNoise = 0.6, meanResp = mean(bioactivity),
    sdResp = sd(bioactivity), lenPred = 800)
```

4 Statistical Pre-processing

Bioactivity annotations in ChEMBL are sometimes redundant, meaning that for a given target-compound combination there are more than one annotated values.
To avoid this issue, we will remove redundant pairs and will keep the mean bioactivity value for those compound-target combinations repeated.

```
source("remove_duplicates.R")
```

Now, we load the dataset without repetitions generated in the previous step. In addition, we remove those columns not containing descriptors (e.g. compound name):

```
dataset <- readRDS("Whole_dataset_NO_REP.rds")
killset <- expression(c(tid, pref_name, accession,
    organism, chembl_id, standard_value, standard_units,
    standard_type, chembl_id.1, Name, Name.1, Name.2,
    rows))
bioactivity <- dataset$standard_value
compound_IDs <- dataset$chembl_id.1
dataset <- subset(dataset, select = -eval(killset))
```

Subsequently, we split the dataset into a training (70%) and a hold-out (external; 30%) set that will be used to assess the predictive ability of the models. Furthermore, we remove the following descriptors: (i) those with a variance close to zero (near-zero variance), and (ii) those highly correlated:

```
# split the dataset into a training and holdout set
dataset <- SplitSet(compound_IDs, dataset, bioactivity,
    percentage = 30)

# remove the descriptors that are highly correlated
# or have low variance
dataset <- RemoveNearZeroVarianceFeatures(dataset,
    frequencyCutoff = 30/1)
dataset <- RemoveHighlyCorrelatedFeatures(dataset)
```

We convert the descriptors to z-scores by centering them to zero mean and scaling their values to unit variance:

```
dataset <- PreProcess(dataset)
```

Given that cross-validation (CV) will be used to optimize the hyperparameters of the models, we divide the training setin 5 folds:

```
dataset <- GetCVTrainControl(dataset)
saveRDS(dataset, file = "dataset_COX_preprocessed.rda")
```

5 Model Training

```
dataset <- readRDS("dataset_COX_preprocessed.rda")
# Set the number of cores for parallelization of
# the training
library(doMC)

## Loading required package:  iterators

registerDoMC(cores = 4)  #from the package 'doMC'
```

5.1 Support Vector Machines (SVM)

Firstly, a SVM will be trained. We define an exponential grid (base 2) to optimize the hyperparameters:

```
method <- "svmRadial"
exp_grid <- expGrid(ini = -8, end = -6, stride = 2,
    base = 2)
tune.grid <- expand.grid(.sigma = exp_grid)
```

Training:

```
# modelCoxSVMrad <- train(dataset£x.train,
# dataset£y.train, method, tuneGrid=tune.grid,
# trControl=dataset£trControl)
# saveRDS(modelCoxSVMrad, file='svm_model_COX.rds')
modelCoxSVMrad <- readRDS("COXsvm.rds")
```

5.2 Random Forest

We proceed similarly in the case of a random forest model.

```
method <- "rf"
tune.grid <- expand.grid(.mtry = seq(5, 100, 5))

# modelCoxRF<- train(dataset£x.train,
# dataset£y.train, method, tuneGrid=tune.grid,
# trControl=dataset£trControl) saveRDS(modelCoxRF,
# file='rf_model_COX.rds')
modelCoxRF <- readRDS("COXrf.rds")
```

6 Model Evaluation

Once the models are trained, the cross validated metrics can be calculated:

```
# Cross Validation Metrics.  We assume the metric
# used for the choice of the best combination of
# hyperparameters is 'RMSE'.  This can be checked
# by: 'my_model'£metric
RMSE_CV = signif(min(as.vector(na.omit(modelCoxRF$results$RMSE))),
    digits = 3)
```

```
Rsquared_CV = modelCoxRF$results$Rsquared[which(modelCoxRF$results$RMSE %in%
    min(modelCoxRF$results$RMSE, na.rm = TRUE))]
print(RMSE_CV)

## [1] 0.78

print(Rsquared_CV)

## [1] 0.5834
```

On the basis of the soundness of the obtained models, we predict the values for the hod-out set:

```
holdout.predictions <- as.vector(predict(modelCoxRF,
    newdata = dataset$x.holdout))
```

We evaluate the predictive ability of our models by calculation the following statistical metrics:

Internal validation:

$$q^2_{int} = 1 - \frac{\sum_{i=1}^{N}(y_i - \widetilde{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_{tr})^2} \tag{1}$$

$$RMSE_{int} = \frac{\sqrt{(y_i - \widetilde{y}_i)^2}}{N} \tag{2}$$

where N, y_i, $\widetilde{y}_i$ and $\bar{y}_{tr}$ represent the size of the training set, the observed, the predicted and the averaged values of the response variable for those datapoints included in the training set. The ith position within the training set is defined by i.

External validation:

$$q^2_{ext} = 1 - \frac{\sum_{j=1}^{N}(y_j - \widetilde{y}_j)^2}{\sum_{j=1}^{N}(y_j - \bar{y}_{ext})^2} \tag{3}$$

$$RMSE_{ext} = \frac{\sqrt{(y_i - \widetilde{y}_i)^2}}{N} \tag{4}$$

$$R^2_{ext} = \frac{\sum_{i=1}^{N}(y_i - \bar{y}_{ext})(\widetilde{y}_i - \bar{\widetilde{y}}_{ext})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y}_{ext})^2 \sum (\widetilde{y}_i - \bar{\widetilde{y}}_{ext})^2}} \tag{5}$$

$$R^2_{0\,ext} = 1 - \frac{\sum_{j=1}^{N}(y_j - \widetilde{y}_j^{r0})^2}{\sum_{j=1}^{N}(y_j - \bar{y}_{ext})^2} \tag{6}$$

where N, y_j, $\widetilde{y}_j$, $\bar{y}_{ext}$ and $\breve{y}_j$ represent the size of the training set, the observed, the predicted, the averaged values and the fitted values of the response variable for those datapoints comprising the external set. The jth position within the external set is defined by j. $R^2_{0\,ext}$ is the square of the coefficient of determination through the origin, being $\widetilde{y}_j^{r0} = k\widetilde{y}_j$ the regression through the origin (observed versus predicted) and k its slope.
For a detailed discussion of both the evaluation of the predictive ability through the external set and different formulations for q^2, see ref.[1]. To be considered as predictive, a model must satisfy the following criteria:[2, 3]

1. $q^2_{int} > 0.5$
2. $R^2_{ext} > 0.6$
3. $\frac{(R^2_{ext} - R^2_{0\,ext})}{R^2_{ext}} < 0.1$
4. $0.85 \leq k \leq 1.15$

The metrics for the external validatin are given by:

```
MetricsRf <- Validation(holdout.predictions, dataset$y.holdout)
MetricsRf

## $R2
## [1] 0.6073
##
## $R02
## [1] 0.602
##
## $Q2
## [1] 0.5998
##
## $RMSEP
```

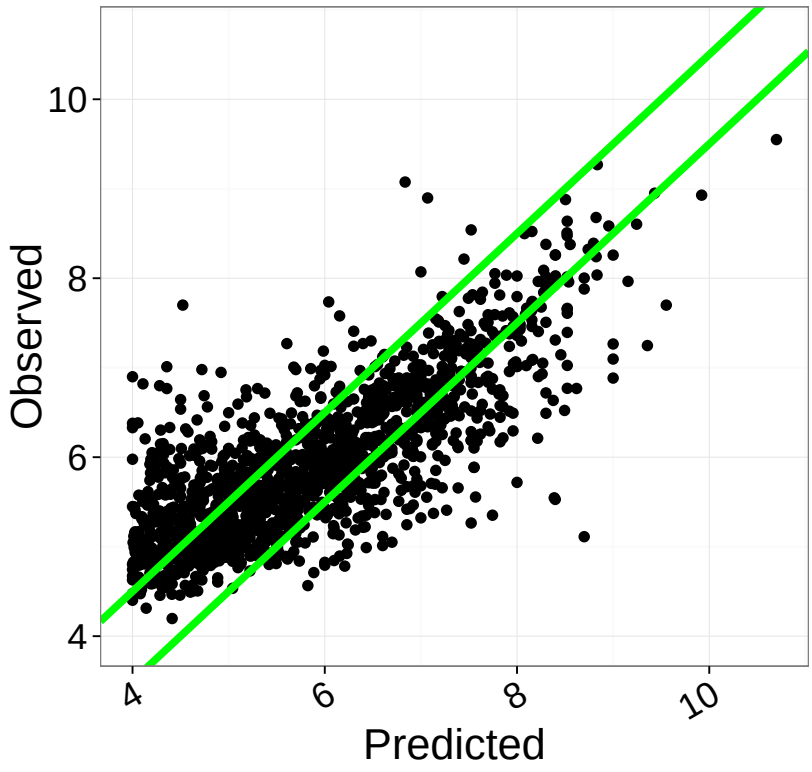

Figure 4: TRUE

```
## [1] 0.7675
##
## $Slope
## [1] 0.9906
```

To have a look at the correlation between predicted and observed values, we can use the 'ObsPred' function:

```
ObsPred(pred = holdout.predictions, obs = dataset$y.holdout,
    PointSize = 3, ColMargin = "green", TitleSize = 26,
    XAxisSize = 20, YAxisSize = 20, TitleAxesSize = 24,
    margin = 1, PointColor = "black", PointShape = 16,
    MarginWidth = 2)
```

7 Bibliography

References

[1] Viviana Consonni, Davide Ballabio, and Roberto Todeschini. "Evaluation of model predictive ability by external validation techniques". en. In: *Journal of Chemometrics* 24.3-4 (2010), 194201. ISSN: 1099-128X. DOI: 10.1002/cem.1290. URL: http://onlinelibrary.wiley.com/doi/10.1002/cem.1290/abstract (visited on 04/04/2013).

[2] Alexander Golbraikh and Alexander Tropsha. "Beware of q2!" eng. In: *Journal of molecular graphics & modelling* 20.4 (Jan. 2002). PMID: 11858635, pp. 269–276. ISSN: 1093-3263.

[3] Alexander Tropsha, Paola Gramatica, and Vijay K. Gombar. "The Importance of Being Earnest: Validation is the Absolute Essential for Successful Application and Interpretation of QSPR Models". en. In: *QSAR & Combinatorial Science* 22.1 (2003), 6977. ISSN: 1611-0218. DOI: 10.1002/qsar.200390007. URL: http://onlinelibrary.wiley.com/doi/10.1002/qsar.200390007/abstract (visited on 04/04/2013).