QSPR with ’camb’
Chemistry **A**ware **M**odel **B**uilder
Cambridge. November 2013

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

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

November 28, 2013

Install the camb package and it’s dependencies. Then load the package.

1 Molecules

1.1 Reading and Preprocessing

```
StandardiseMolecules(structures.file = "solubility_2007_ref2.sdf",
    standardised.file = "standardised.sdf", removed.file = "removed.sdf",
    output = "properties.csv", remove.inorganic = TRUE,
    fluorine.limit = 3, chlorine.limit = 3, bromine.limit = 3,
    iodine.limit = 3, min.mass.limit = 20, max.mass.limit = 900)
```

1.2 Calculating PaDEL Descriptors

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

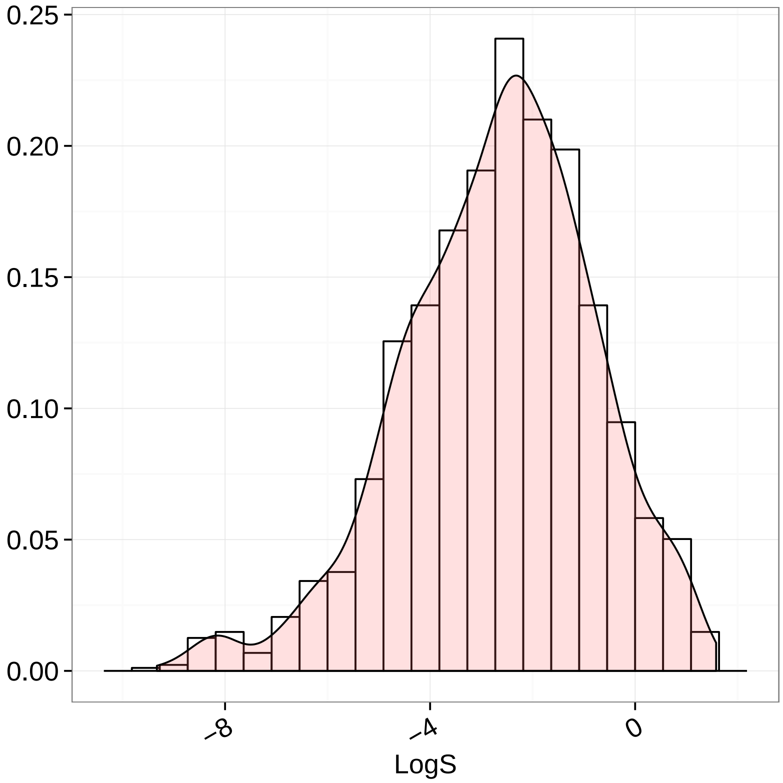


Figure 1: LogS Distribution

```
descriptors <- GeneratePadelDescriptors(standardised.file = "standardised.sdf",
    types = c("2D"), threads = 1)
descriptors <- RemoveStandardisedPrefix(descriptors)
```

2 Target Visualization

We can have a look at the response variable:

```
properties <- read.csv("properties.csv")
properties <- properties[properties$Kept == 1, ]
targets <- data.frame(Name = properties$Name, target = properties$EXPT)
p <- DensityResponse(targets$target) + xlab("LogS")
p
```

3 Statistical Pre-processing

Merge the calculated descriptors and the target values by name into a single data.frame. Check that the number of rows of the merged and original data.frames are the same. Split the data.frame into ids, x and y where ids are the molecule names, x are the descriptor values and y is the target values.

```r
all <- merge(x = targets, y = descriptors, by = "Name")
# check the number of rows are the same
dim(all)
dim(targets)
dim(descriptors)
ids <- all$Name
x <- all[3:ncol(all)]
y <- all$target
```

Split the dataset into a training (80%) and a holdout (20%) set that will be used to assess the predictive ability of the models. Remove the following descriptors: (i) those with a variance close to zero (near-zero variance), and (ii) those highly correlated:

```r
# replace the infinite values with NA and impute
# the remaining NA values
x.finite <- ReplaceInfinitesWithNA(x)
x.imputed <- ImputeFeatures(x.finite)

# split the dataset into a training and holdout set
dataset <- SplitSet(ids, x.imputed, y, percentage = 20)

# remove the descriptors that are highly correlated
# or have low variance
dataset <- RemoveNearZeroVarianceFeatures(dataset)
dataset <- RemoveHighlyCorrelatedFeatures(dataset)
```

Center and scale the descriptors:

```r
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.rds")
```

4 Model Training

```
# register the number of cores to use in training
registerDoMC(cores = 1)

# train and save a support vector machine model
method <- "svmRadial"
tune.grid <- expand.grid(.sigma = expGrid(-8, 4, 2,
    2), .C = c(1e-04, 0.001, 0.01, 0.1, 1, 10, 100))
model <- train(dataset$x.train, dataset$y.train, method,
    tuneGrid = tune.grid, trControl = dataset$trControl)
saveRDS(model, file = paste(method, ".rds", sep = ""))

model <- readRDS("svmRadial.rds")
plot(model, metric = "RMSE")

# train and save a random forest model
method <- "rf"
tune.grid <- expand.grid(.mtry = seq(5, 100, 5))
model <- train(dataset$x.train, dataset$y.train, method,
    tuneGrid = tune.grid, trControl = dataset$trControl)
saveRDS(model, file = paste(method, ".rds", sep = ""))

# train and save a generalised boosted regression
# model
method <- "gbm"
tune.grid <- expand.grid(.n.trees = c(500, 1000), .interaction.depth = c(25),
    .shrinkage = c(0.04, 0.08, 0.16))
model <- train(dataset$x.train, dataset$y.train, method,
    tuneGrid = tune.grid, trControl = dataset$trControl)
saveRDS(model, file = paste(method, ".rds", sep = ""))
```

Determine if your hyper-parameter search needs to be altered.

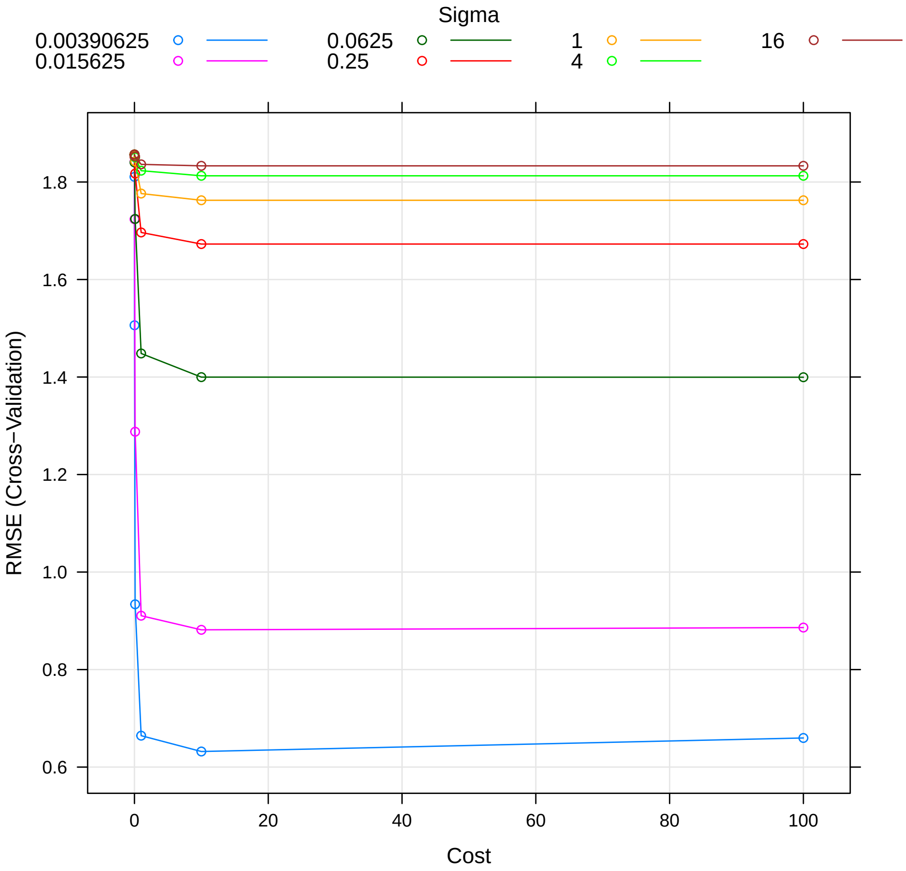


Figure 2: CV RMSE over the hyperparameters

```
model <- readRDS("svmRadial.rds")
plot(model, metric = "RMSE")
```

5 Model Evaluation

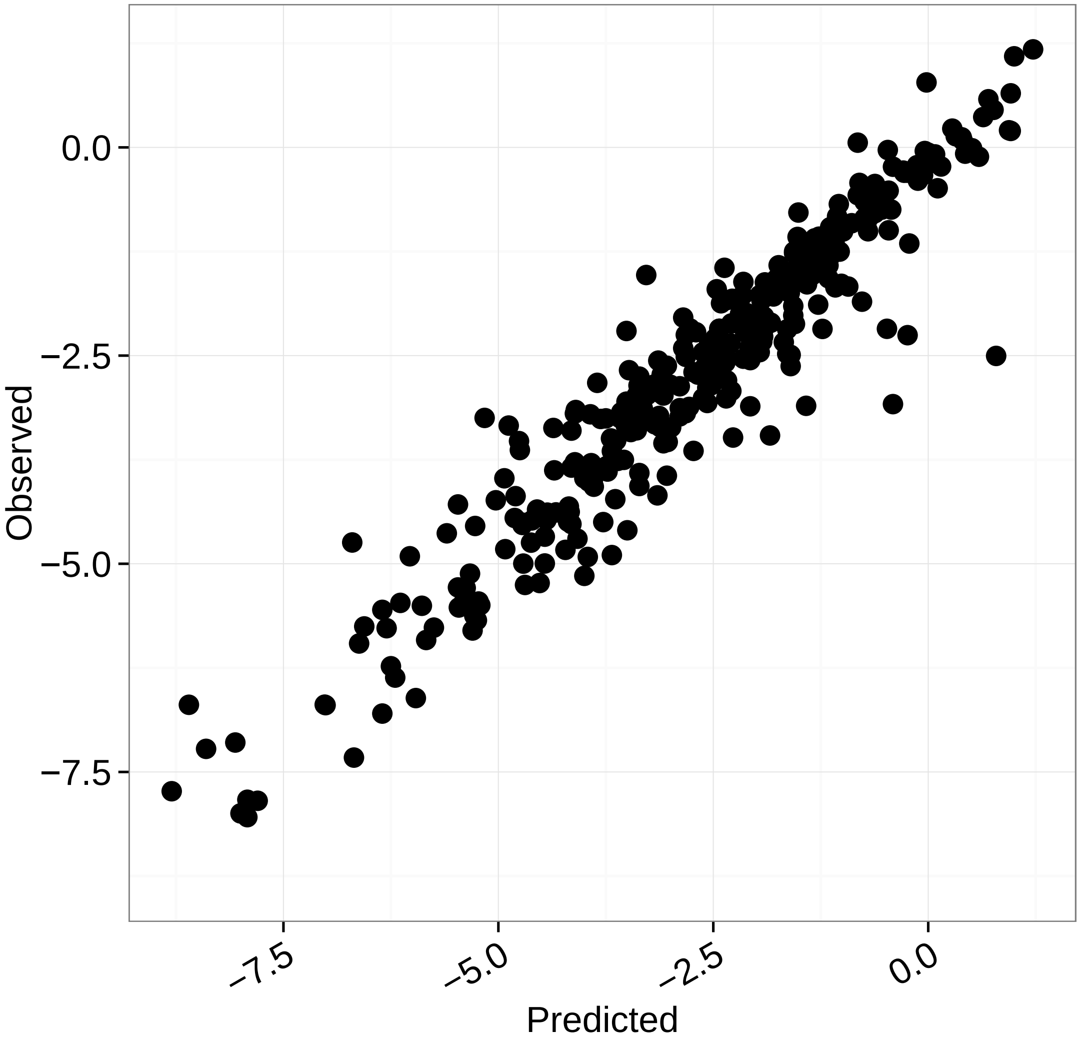
Once the models are trained, the cross validated metrics can be calculated:

```
dataset <- readRDS("dataset.rds")
model <- readRDS("svmRadial.rds")

# Cross Validation Metrics.
RMSE_CV(model)

## [1] 0.632
```

```
Rsquared_CV(model)

## [1] 0.8834
```

On the basis of the soundness of the obtained models, we predict the values for the holdout set:

```
holdout.predictions <- as.vector(predict(model, newdata = dataset$x.holdout))
CorrelationPlot(pred = holdout.predictions, obs = dataset$y.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.[**consonni**]. To be considered as predictive, a model must satisfy the following criteria:[**beware**, **earnest**]

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 validation are given by:

```r
# Statistics for Model Validation
Validation(holdout.predictions, dataset$y.holdout)

## $R2
## [1] 0.9078
##
## $R02
## [1] 0.907
##
## $Q2
## [1] 0.907
##
## $RMSE
## [1] 0.5984
##
## $Slope
## [1] 1.001
##
## $MAE
## [1] 0.4101
```