QSAR model by a PLS regression

Edward Francisco Mendez-Otalvaro, Daniel Alberto Barragan, Isaias Lans-Vargas

2021

Functions to calculate metrics of the model

The metrics that are going to be used are:

- Root mean square error of training set (RMSEC)
- Root mean square error of test set (RMSEP)
- Square of the correlation coefficient of training set (R2tr).
- Square of the correlation coefficient of test set (Also called Q2F2 coefficient).
- Q2F3 coefficient of Todeschini.
- Spearman correlation coefficient of test set (σ).
- Square of the Spearman correlation coefficient of test set ($\sigma 2$).
- Square of the correlation coefficient of cross validation leaving one outside (R2LOO).
- Root mean square error of cross validation leaving one outside (RMSELOO).

All the above metrics are rotinarious in QSAR modelling, in order to get some criteria about a good predictive or ranking model (our main objetive).

```
Metricas <- function(obs_tr, pred_tr, obs_out, pred_out) {

    ssr_tr <- sum((obs_tr - pred_tr)^2)
    sst_tr <- sum((obs_tr - mean(obs_tr))^2)

    ssr_out <- sum((obs_out - pred_out)^2)
    sst_out <- sum((obs_out - mean(obs_out))^2)

    n_tr <- 20
    n_out <- 7

    ## RMSEC

    RMSEC <- sqrt(ssr_tr/(n_tr))

    ## RMSEP

    RMSEP <- sqrt(ssr_out/(n_out))

    ## R2tr

    R2tr <- (1 - (ssr_tr/sst_tr))

    ## R2out/Q2F2/R2Tropsha

    R2out <- (1 - (ssr_out/sst_out))
```

```
    ## Q2F3

    Q2F3 <- (1 - ((ssr_out/n_out)/(sst_tr/n_tr)))

    ## Sigma

    rho <- cor(obs_out, pred_out, method = c("spearman"))

    ## Sigma2

    rho2 <- (rho)^2

    ## Salida

    output <- list(RMSEC = RMSEC, RMSEP = RMSEP, R2tr = R2tr, R2out = R2out, Q2F3 = Q2F3,
        rho = rho, rho2 = rho2)
    return(output)
}

## LOO Metrics

Metrica_LOO <- function(obs, pred) {

    ssr_LOO <- sum((obs - pred)^2)
    sst_LOO <- sum((obs - mean(obs))^2)
    n_LOO <- 20

    ## R2LOO

    R2LOO <- (1 - (ssr_LOO/sst_LOO))

    ## RMSELOO

    RMSELOO <- sqrt(ssr_LOO/n_LOO)

    output <- list(R2LOO = R2LOO, RMSELOO = RMSELOO)
    return(output)
}
```

Importing dataset

Now, the dataset previously prepared is upload (we concatenate the molecular descriptors with the biological activity):

```
Descriptores_I <- read.csv("Descriptores_Alvrunner.csv", stringsAsFactors = FALSE)

Descriptores_II <- read.csv("Descriptores_MOPAC.csv", stringsAsFactors = FALSE)

Descriptores <- cbind(Descriptores_I, Descriptores_II)


## Biological activity

Actividad <- read.csv2("Actividad.csv", stringsAsFactors = FALSE)
```

```r
## Joining molecular descriptors with activity

Dataset <- cbind(Descriptores, Actividad)

## Cleaning labels

Dataset$ID <- NULL
Dataset$IC50..uM. <- NULL

## Renaming biological activity

names(Dataset)[names(Dataset) == "IC50..M."] <- "pIC50"
```

Converting biological activity (EC50) into logarithmic scale, and then, calculating dimension of the dataframe

```r
Dataset[, 2284] <- -log10(Dataset[, 2284])

dim(Dataset)
```

```
## [1]   27 2284
```

So, there are 27 molecules with 2283 molecular descriptors and a biological activity response.

Pretreatment of the dataset (cleaning)

Let's remove NA columns; columns with variance of zero (constant columns) and columns with more than half filled with zeros (Refer to the Manuscript to the criteria selected).

```r
## Removing NA's
Dataset <- Dataset[, !apply(Dataset, 2, function(x) any(is.na(x)))]

## Removing columns with variance equal to zero.
Dataset <- Dataset[, apply(Dataset, 2, var, na.rm = TRUE) != 0]

## Removing columns with constant values
Dataset <- Dataset[, !apply(Dataset, 2, function(x) length(unique(x)) == 1)]

## Removing columns with more than half filled with zeros
Dataset <- Dataset[, colSums(Dataset != 0) > nrow(Dataset)/2]

dim(Dataset)
```

```
## [1]  27 365
```

Since the dataset is very high dimensional `27 X 365`. Let's calculate a correlation matrix for all the descriptors, and then, let's remove the high correlated molecular descriptors (R2 of Pearson >0.99), with this, the multicolinearity between columns could be improved (a problem that could generate bias in our QSAR model)

```r
## Removing high correlated descriptors
library(caret)

## Correlation matrix calculation
Dataset_Cor = cor(Dataset)

## Removing high correlated descriptors with a R2>0.99
```

```r
hc = findCorrelation(Dataset_Cor, cutoff = 0.99)
hc = sort(hc)

## Non correlated descriptor matrix

Dataset_hc = Dataset[, -c(hc)]

## Dimension of the dataset
dim(Dataset_hc)
```

```
## [1]  27 275
```

So now, the dataset has a dimension of `27 X 275`. The improvement of the descriptors is good, but the high dimensionality still appears (n<p, ie; more descriptors than observations)

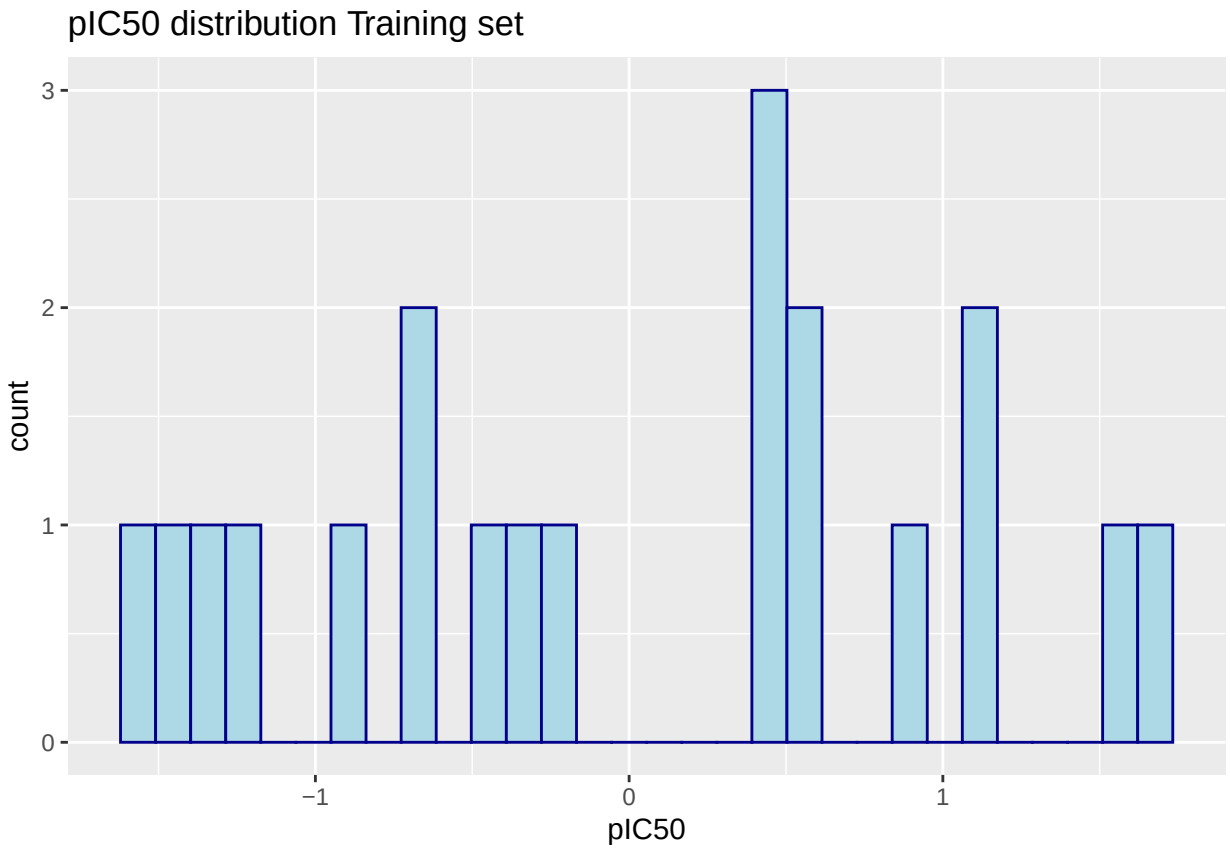
Splitting dataset into training a test set

Since the dataset is very asymmetrical, and there are few observations, let's try to split the dataset in order to get a good balance between both groups (avoiding artifacts by asymetrical splitting). The **caret** function from R allows to carry out this task. Also, the dataset will be scaled subtracting the mean and dividing by standard deviation of data. The ratio will be 70% training and 30% testing.

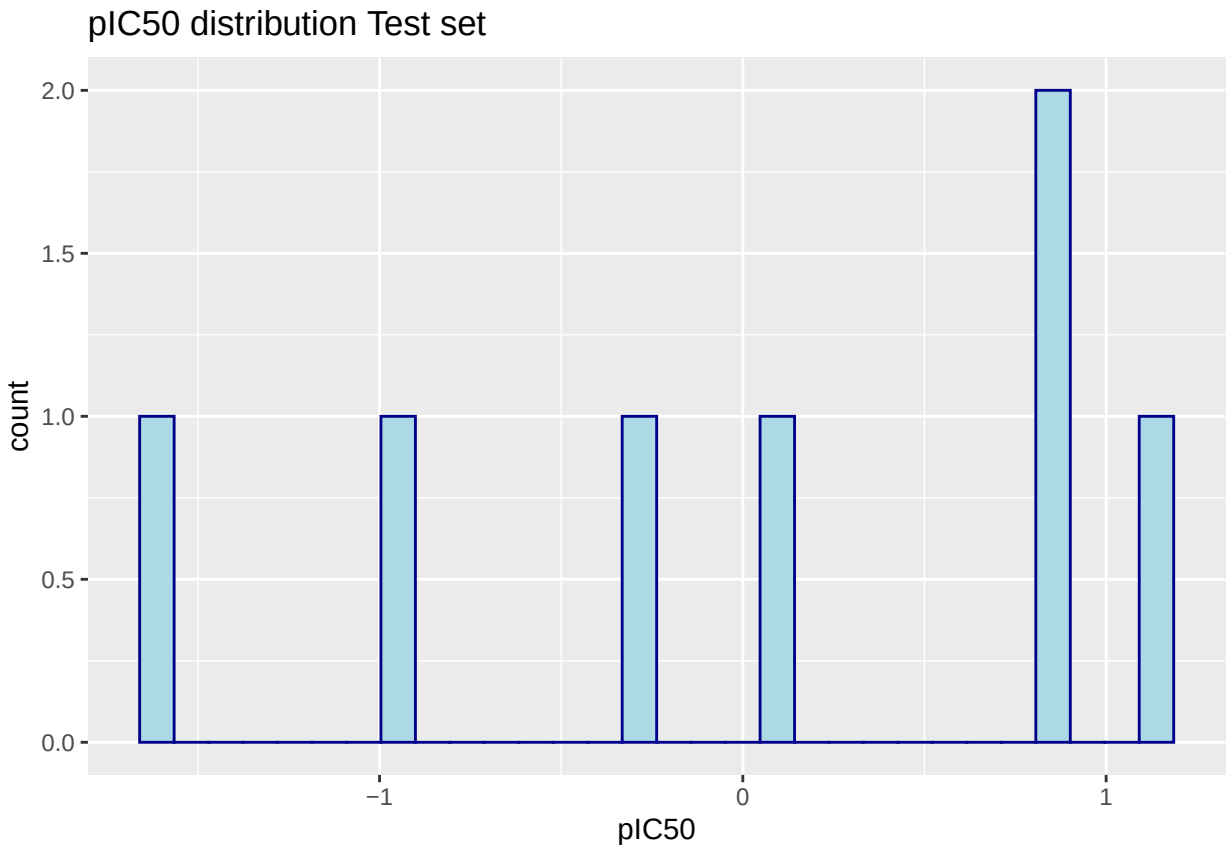
```r
## Scaling
Mean <- apply(Dataset_hc, 2, mean)
SD <- apply(Dataset_hc, 2, sd)

Dataset_S <- as.matrix(scale(Dataset_hc, center = Mean, scale = SD))

centrado <- t(attr(Dataset_S, "scaled:center"))
escalado <- t(attr(Dataset_S, "scaled:scale"))


## Splitting the data in training and test (70% training and 30% test)

set.seed(101)
Muestra <- createDataPartition(Dataset_S[, 275], times = 1, p = 0.7, list = FALSE)

Modelamiento <- as.matrix(Dataset_S[Muestra, ])
Prueba <- as.matrix(Dataset_S[-Muestra, ])
```

Let's apreciate the distribution of the data in the training and test set

```r
library(ggplot2)

## Basic histogram

ggplot(as.data.frame(Modelamiento), aes(x = pIC50)) + geom_histogram(color = "darkblue",
    fill = "lightblue") + ggtitle("pIC50 distribution Training set")
```




```
ggplot(as.data.frame(Prueba), aes(x = pIC50)) + geom_histogram(color = "darkblue",
    fill = "lightblue") + ggtitle("pIC50 distribution Test set")
```



A Student-t statistical test will be carried out to get insight about the differences between groups. The null hypothesis states that the data comes from the same statistical distribution. The alternative hypothesis states that the data does not come from the same statistical distribution. The test will be carried out with a significance level of 95% (p-value = 0.05)

```
t.test(Modelamiento[, 275], Prueba[, 275])
```

```
##
##  Welch Two Sample t-test
##
## data:  Modelamiento[, 275] and Prueba[, 275]
## t = -0.019338, df = 10.475, p-value = 0.9849
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.0032746  0.9859042
## sample estimates:
##    mean of x    mean of y
## -0.002251715  0.006433471
```

Since `p-value>0.05`, then the null hypothesis is accepted, so both data distributions come from the same statistical distribution with a statistical significance level of 95%.

Fitting a partial least squares model (PLS):

Partial least squares is an algorithm very useful in cases when the matrix of predictors has more variables than observations (n<p) and also, when there is multicollinearity among regressors. The algorithm tries to find a linear regression model projection the predicted and observable variables to a new space determined by the principal components (PC) of the data, specifically the predictors and outcome are projected into the

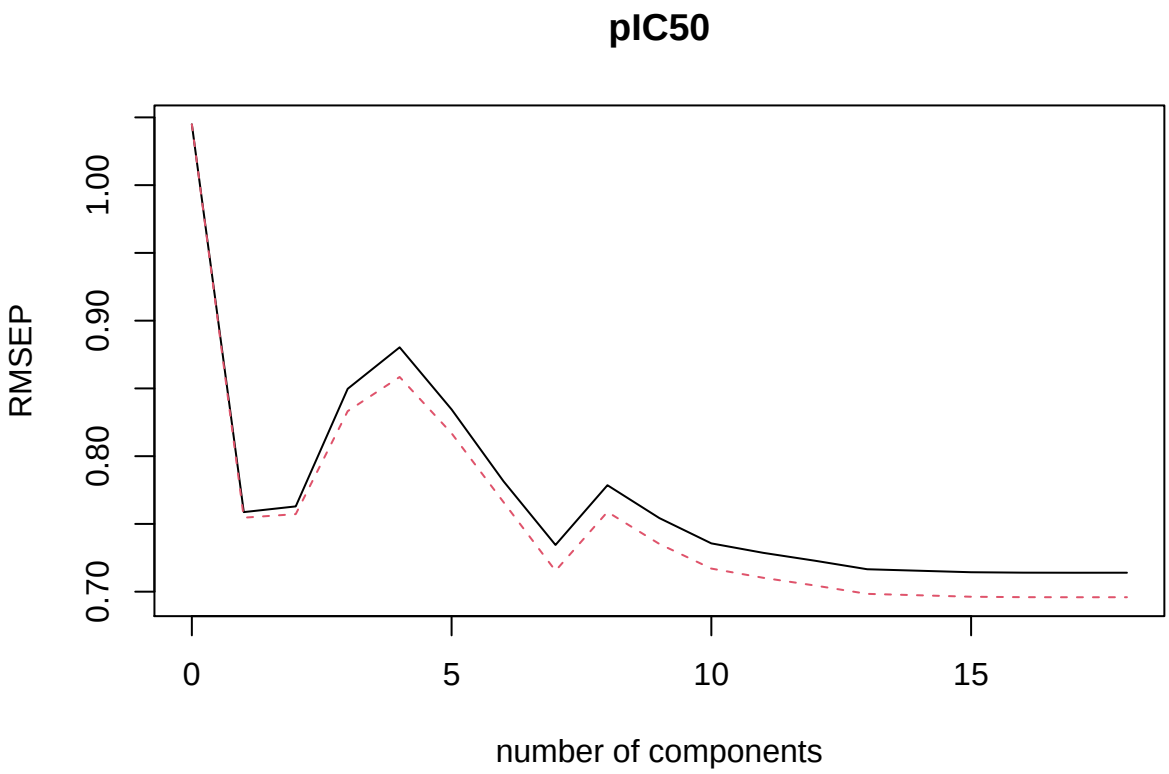
scores matrix of the PC. Since the dataset is projected in a new space (or axis of coordenates), there is not an explicit expression for the correlation (like in the OLS-MLR), because the linear fitting is carry out over the PC.

```
## Setting training and test sets

Modelamiento_pls <- as.data.frame(Modelamiento)
Prueba_pls <- as.data.frame(Prueba)

## Fitting PLS regression model
library(pls)

set.seed(1)
Modelo_PLS = plsr(pIC50 ~ ., data = Modelamiento_pls, scale = FALSE, validation = "LOO")
summary(Modelo_PLS)
```

```
## Data:    X dimension: 20 274
##  Y dimension: 20 1
## Fit method: kernelpls
## Number of components considered: 18
##
## VALIDATION: RMSEP
## Cross-validated using 20 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
## CV           1.045   0.7587   0.7629   0.8497   0.8803   0.8345
## adjCV        1.045   0.7546   0.7572   0.8332   0.8584   0.8169
##        6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
## CV      0.7814   0.7345   0.7786   0.7543    0.7356    0.7287
## adjCV   0.7660   0.7154   0.7589   0.7352    0.7170    0.7102
##        12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
## CV       0.7228    0.7165    0.7155    0.7143     0.714    0.7140
## adjCV    0.7045    0.6984    0.6973    0.6963     0.696    0.6959
##        18 comps
## CV       0.7140
## adjCV    0.6959
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        26.99    46.77    52.62    64.81    81.23    86.10    87.92
## pIC50    65.34    74.82    85.96    91.85    95.04    96.88    99.49
##        8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X        89.98    91.59     93.55     94.90     95.75     96.33
## pIC50    99.78    99.90     99.95     99.98    100.00    100.00
##        14 comps  15 comps  16 comps  17 comps  18 comps
## X         97.46     98.35     99.03     99.39     99.79
## pIC50    100.00    100.00    100.00    100.00    100.00
```

So, apparently with 15 PC the model can explain almost 100% of the variance of the data. Let's plot all the PC respect to the error (RMSE) in order to get the true value of the optimal PC.

```
## Number of principal components (PC) that minimize the RMSE
validationplot(Modelo_PLS, val.type = "RMSEP")
```



Since the optimal number of principal components that minimize the error are 15, let's refit the model with these parameters.

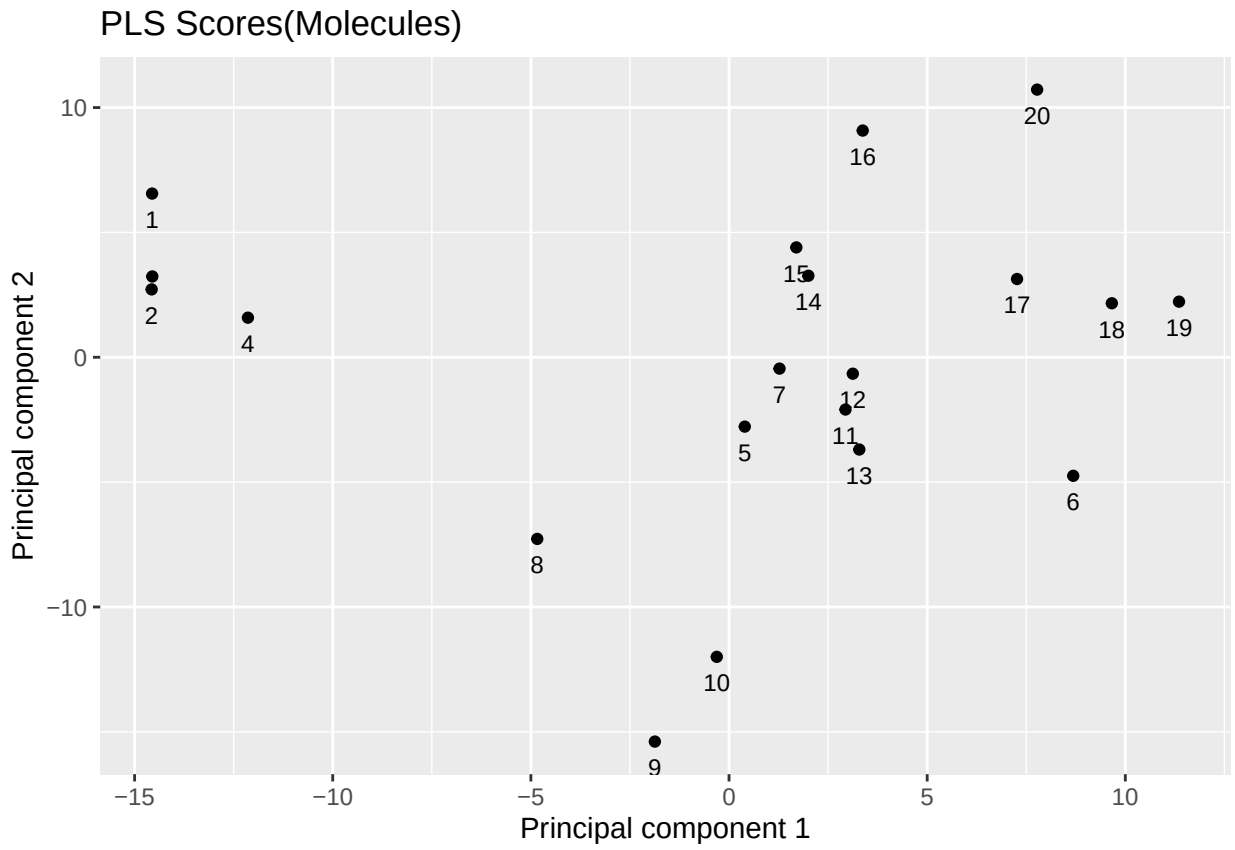
```
## Refitting the PLS model with 15 PC
Modelo_PLS_Final = plsr(pIC50 ~ ., data = Modelamiento_pls, scale = FALSE, ncomp = 15)
summary(Modelo_PLS_Final)
```

```
## Data:    X dimension: 20 274
##  Y dimension: 20 1
## Fit method: kernelpls
## Number of components considered: 15
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        26.99    46.77    52.62    64.81    81.23    86.10    87.92
## pIC50    65.34    74.82    85.96    91.85    95.04    96.88    99.49
##        8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X        89.98    91.59     93.55     94.90     95.75     96.33
## pIC50    99.78    99.90     99.95     99.98    100.00    100.00
##        14 comps  15 comps
## X         97.46     98.35
## pIC50    100.00    100.00
```

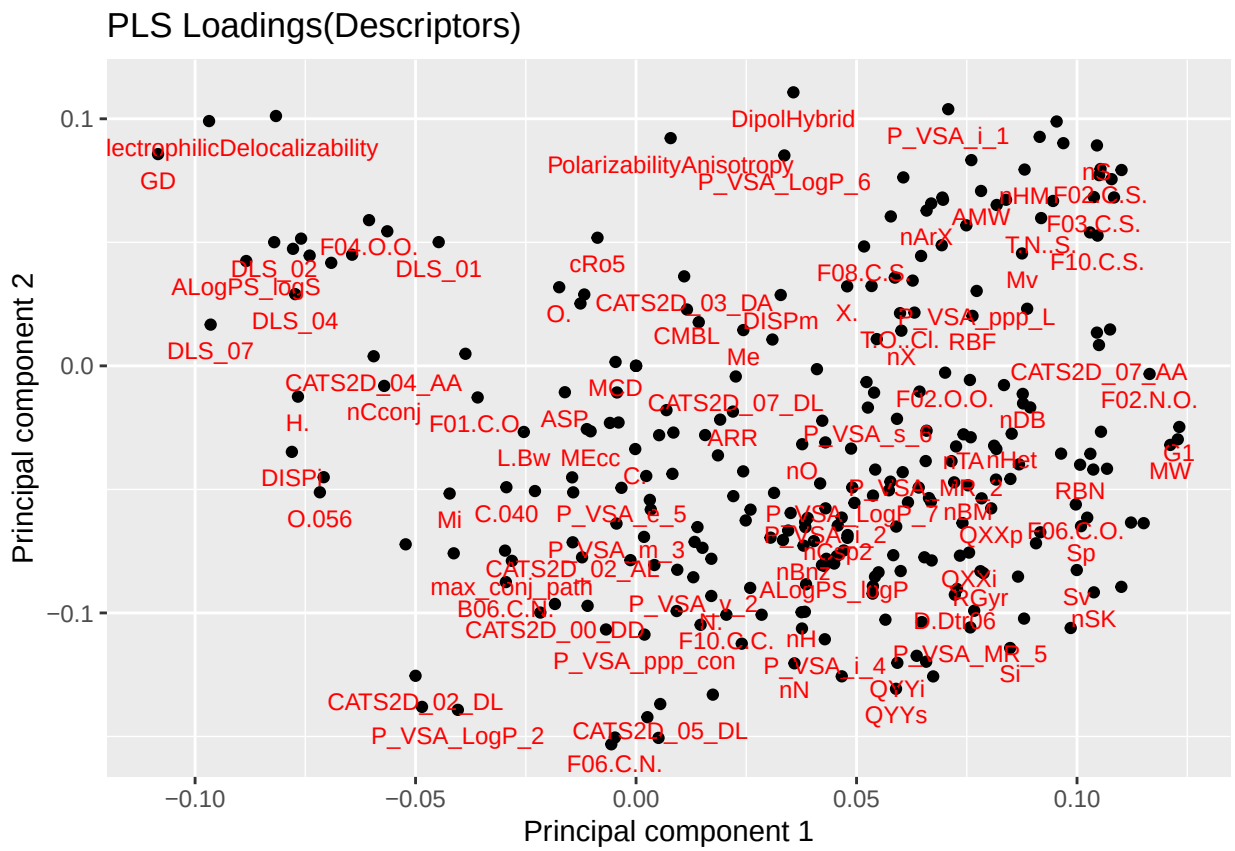
Scores and loadings predictors plot (requirement when a PLS model is fitted):

```
library(ggplot2)
scores_modelo <- Modelo_PLS_Final$scores
loadings_modelo <- Modelo_PLS_Final$loadings
```

```
sc1 <- scores_modelo[, 1]
sc2 <- scores_modelo[, 2]
scor_plot <- as.data.frame(cbind(sc1, sc2))
rownames(scor_plot) <- Muestra

ld1 <- loadings_modelo[, 1]
ld2 <- loadings_modelo[, 2]
loa_plot <- as.data.frame(cbind(ld1, ld2))
desc_names <- colnames(Modelamiento_pls)
rownames(loa_plot) <- desc_names[1:274]

p <- ggplot(scor_plot, aes(x = sc1, y = sc2)) + geom_point() + geom_text(label = rownames(scores_modelo)
    check_overlap = TRUE, size = 3, vjust = 2) + geom_point(shape = ".") + labs(title = "PLS Scores(Mol
    xlab("Principal component 1") + ylab("Principal component 2")
p
```




```
m <- ggplot(loa_plot, aes(x = ld1, y = ld2)) + geom_point() + geom_text(label = rownames(loa_plot),
    check_overlap = TRUE, size = 3, vjust = 2, colour = "red") + geom_point(shape = ".") +
    labs(title = "PLS Loadings(Descriptors)") + xlab("Principal component 1") + ylab("Principal componer
m
```



Predicting the training dataset with the model (Internal):

```
Predicho_tr <- predict(Modelo_PLS_Final, Modelamiento_pls, ncomp = 15)
Observado_tr <- Modelamiento_pls$pIC50
```

Predicting the test dataset with the model (External):

```
Predicho_out <- predict(Modelo_PLS_Final, Prueba_pls, ncomp = 15)
Observado_out <- Prueba_pls$pIC50
```

Unscaling data

```
## Function to unscaling

Desescalar <- function(X) {
    X_desescalada <- (X * escalado[, 275] + centrado[, 275])
}

## Internal prediction

Predicho_tr <- Desescalar(Predicho_tr)
Observado_tr <- Desescalar(Observado_tr)

Obs_df <- as.data.frame(cbind(Observado_tr, Predicho_tr))
colnames(Obs_df) <- c("Observed", "Predicted")
rownames(Obs_df) <- Muestra[1:20]
```

```
## External prediction

Predicho_out <- Desescalar(Predicho_out)
Observado_out <- Desescalar(Observado_out)

Pred_df <- as.data.frame(cbind(Observado_out, Predicho_out))
colnames(Pred_df) <- c("Observed", "Predicted")
rownames(Pred_df) <- c("4", "5", "10", "15", "16", "19", "26")
```

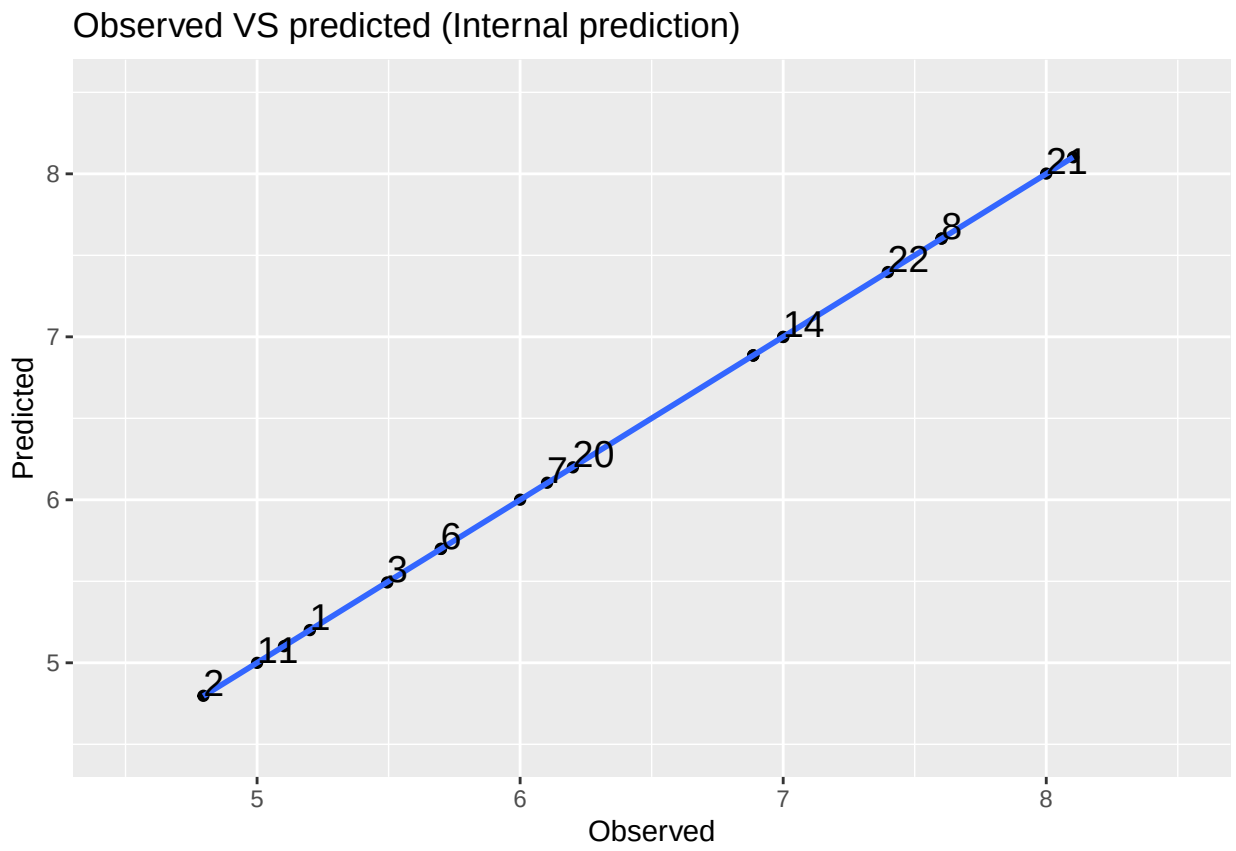
Statistical metrics and plotting of the model

```
Metricas_PLS <- Metricas(Observado_tr, Predicho_tr, Observado_out, Predicho_out)

Metricas_PLS <- (do.call(rbind, Metricas_PLS))
rownames(Metricas_PLS) <- c("RMSEC", "RMSEP", "R2tr", "R2out", "Q2F3", "rho", "rho2")
Metricas_PLS <- round(Metricas_PLS, 1)
print(Metricas_PLS)
```

```
##       [,1]
## RMSEC  0.0
## RMSEP  0.6
## R2tr   1.0
## R2out  0.6
## Q2F3   0.6
## rho    0.8
## rho2   0.7
```

```
library(ggplot2)

# Basic scatter plot Training
ggplot(Obs_df, aes(x = Observed, y = Predicted)) + geom_point() + geom_smooth(method = lm) +
    ggtitle("Observed VS predicted (Internal prediction)") + geom_text(size = 5,
    aes(label = rownames(Obs_df)), hjust = 0, vjust = 0, check_overlap = TRUE) +
    coord_cartesian(xlim = c(4.5, 8.5), ylim = c(4.5, 8.5))
```

```
## `geom_smooth()` using formula 'y ~ x'
```

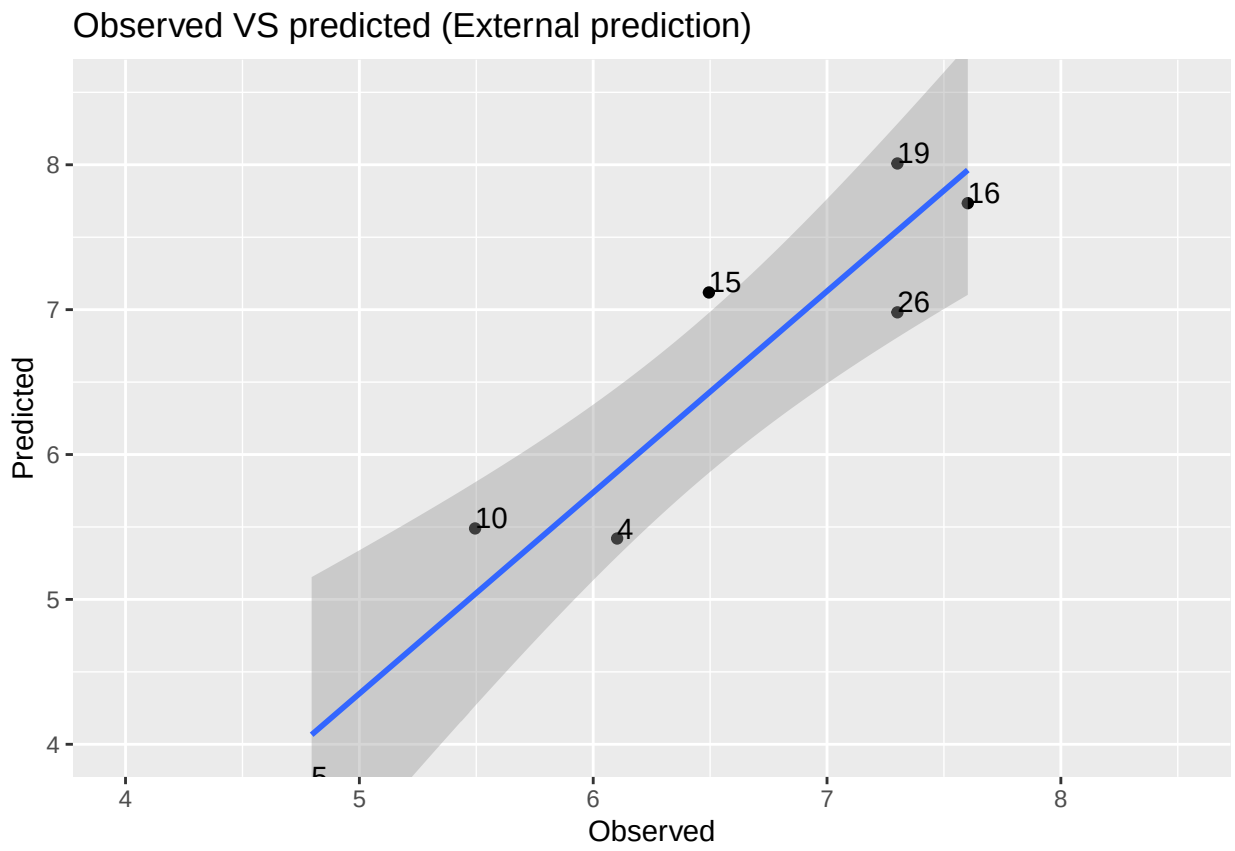



```
ggsave("plsexternal.png")
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using formula 'y ~ x'
```

```
# Basic scatter plot Test
ggplot(Pred_df, aes(x = Observed, y = Predicted)) + geom_point() + geom_smooth(method = lm) +
    ggtitle("Observed VS predicted (External prediction)") + geom_text(aes(label = rownames(Pred_df)),
    check_overlap = TRUE, hjust = 0, vjust = 0) + coord_cartesian(xlim = c(4, 8.5),
    ylim = c(4, 8.5))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggsave("plsinternal.png")
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using formula 'y ~ x'
```

Leaving one out cross-validation (LOO-CV) of the data

A statistical metric that is common in QSAR modeling is the RMSE and R2 of the LOO-CV. Let's estimate these parameters

```
## Storing matrix for the cycle

Pred_CV = matrix()
Modelo_i = list()

## Training set

Modelamiento_CV <- as.data.frame(Modelamiento_pls)
set.seed(1e+05)

for (i in 1:nrow(Modelamiento_CV)) {

    ## Removing i row from 1 to the number of rows of the dataset (each cycle left one
    ## out)

    Calibracion_CV <- Modelamiento_CV[-i, ]
    Prueba_CV <- Modelamiento_CV[i, ]
```

```
    ## Training the model
    Modelo_PLS_CV = plsr(pIC50 ~ ., data = Calibracion_CV, scale = FALSE, ncomp = 15)

    ## Predicting the i-row
    Predicho_CV = predict(Modelo_PLS_CV, Prueba_CV, ncomp = 15)

    ## Predicted values for each i-row
    Pred_CV[i] <- Predicho_CV

    ## Model for each i-row
    Modelo_i[[i]] <- Modelo_PLS_CV

}
```

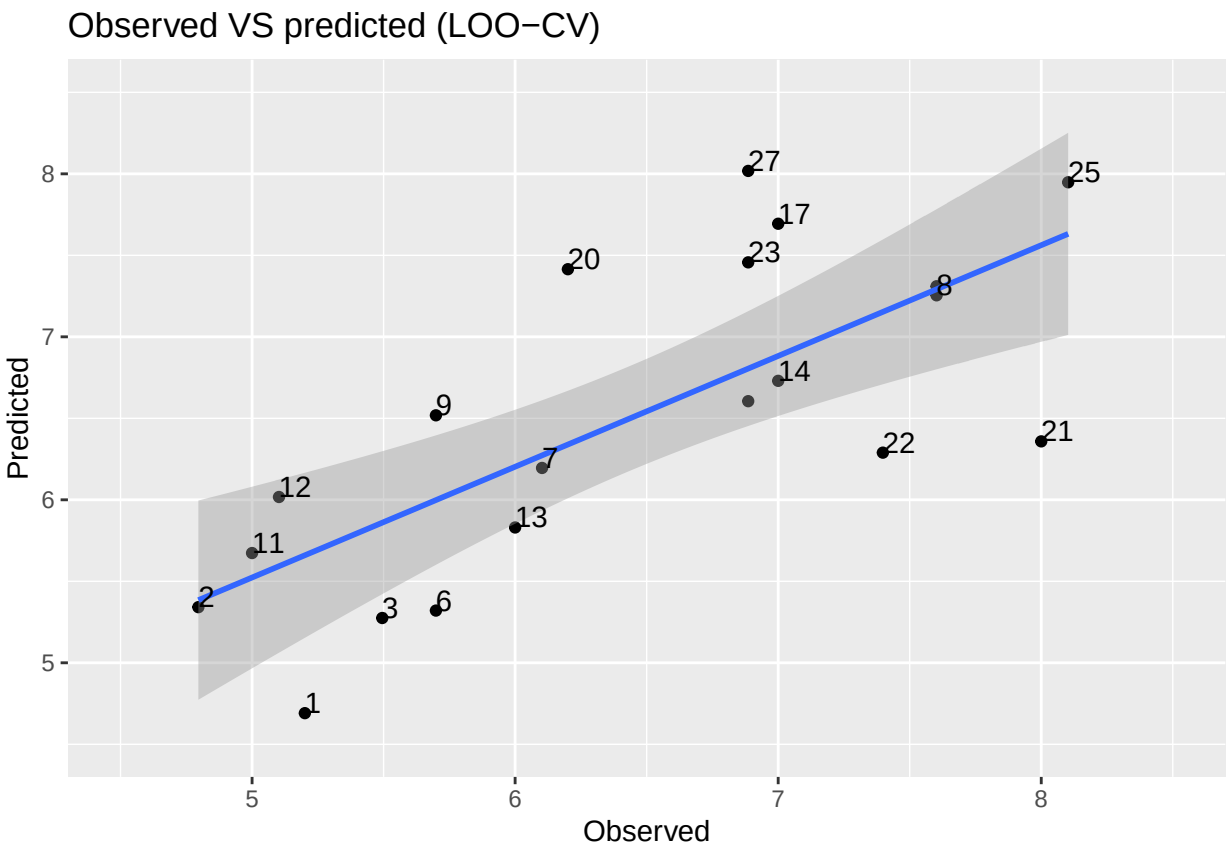
Calculating the metrics and plotting the results

```
## Unscaling observed and predicted data

Observado_LOO <- Observado_tr
Predicho_LOO <- Desescalar(Pred_CV)

LOO_df <- as.data.frame(cbind(Observado_LOO, Predicho_LOO))
rownames(LOO_df) <- Muestra[1:20]

## Metric calculation
LOO_M <- Metrica_LOO(Observado_LOO, Predicho_LOO)

LOO_M <- (do.call(rbind, LOO_M))
LOO_M <- round(LOO_M, 1)
print(LOO_M)
```

```
##         [,1]
## R2LOO    0.5
## RMSELOO  0.7
```

```
# Basic scatter plot LOO-CV
ggplot(LOO_df, aes(x = Observado_LOO, y = Predicho_LOO)) + geom_point() + geom_smooth(method = lm) +
    ggtitle("Observed VS predicted (LOO-CV)") + xlab("Observed") + ylab("Predicted") +
    geom_text(aes(label = rownames(LOO_df)), check_overlap = TRUE, hjust = 0, vjust = 0) +
    coord_cartesian(xlim = c(4.5, 8.5), ylim = c(4.5, 8.5))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggsave("plsloo.png")
```

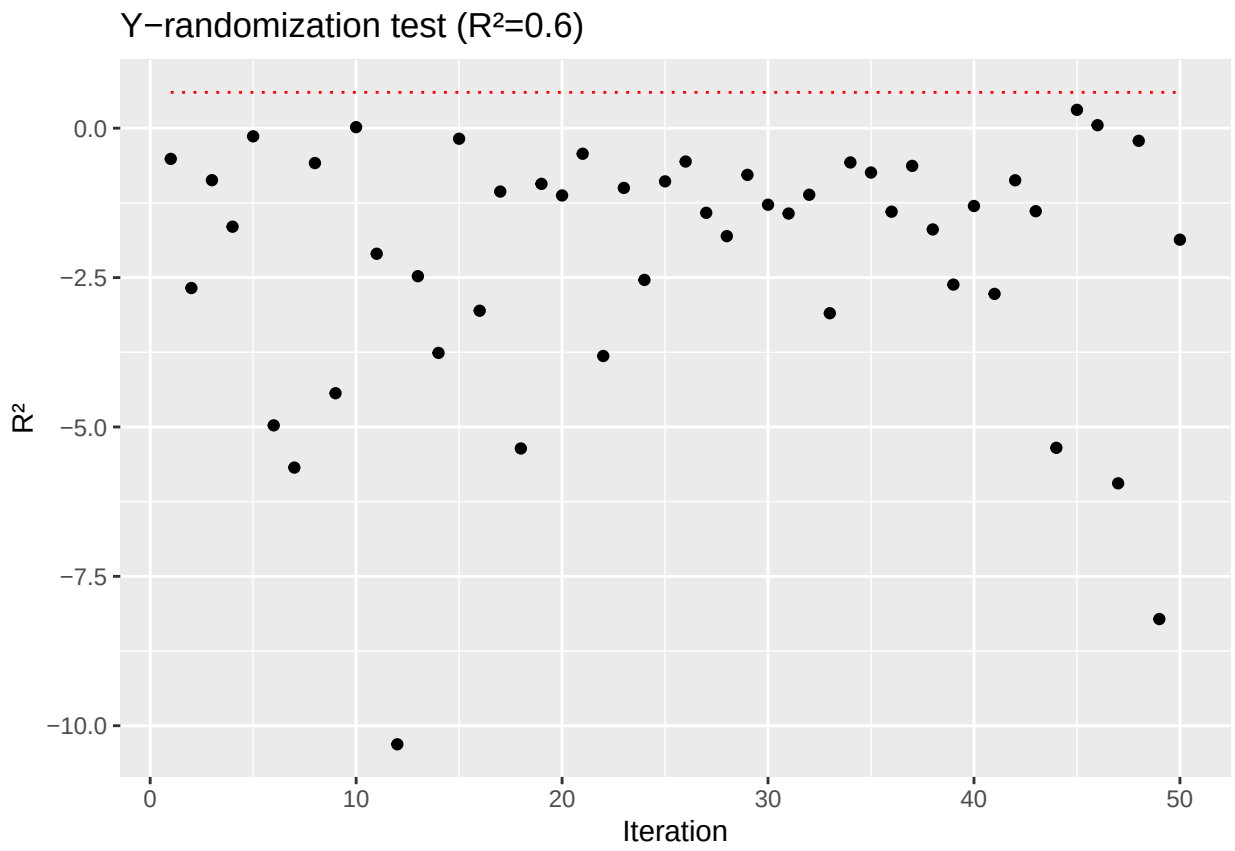
```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using formula 'y ~ x'
```

Y-randomization test for PLS model:

Another statistical test to get insight about the reliability of the QSAR model is the Y-randomization. In this algorithm, the outcome variable (Y=pIC50) is randomized, and in each iteration a new model is build. The R2 for all the random models should be worse than the R2 from the constructed model. This test tries to prove that the model build does not come from random chance.

```
## 50-fold loop for Y-randomization

## Empty list for store the data that comes from the loop

Score_Y <- list()
set.seed(1e+05)

for (i in 1:50) {

    ## Shuffling the outcome (pIC50), and building training and test sets with this
    ## randomized dataset
    Dataset_Y_shuffle <- as.data.frame(Dataset_S)
    Y_shuffle <- sample(Dataset_S[, 275], replace = FALSE)
    Dataset_Y_shuffle[, 275] <- Y_shuffle
```

```r
    Modelado_shuffle <- Dataset_Y_shuffle[Muestra, ]
    Prueba_shuffle <- Dataset_Y_shuffle[-Muestra, ]

    ## Training the model with randomized outcome data

    Modelo_Lineal_Y <- plsr(pIC50 ~ ., data = Modelado_shuffle, scale = FALSE, ncomp = 15)

    ## Predicting test set with randomized outcome data

    Pred_Y = predict(Modelo_Lineal_Y, Prueba_shuffle, ncomp = 15)

    ## Unscaling data

    Observado_Y <- Desescalar(Prueba_shuffle$pIC50)
    Predicho_Y <- Desescalar(Pred_Y)

    ## Model metrics
    Score_Y[[i]] <- Metrica_LOO(Observado_Y, Predicho_Y)
}

## Extracting metrics from the loop
Resul_Y_Random <- as.data.frame(t(unlist(Score_Y)))

par_indexes <- seq(2, 100, 2)
impar_indexes <- seq(1, 99, 2)

RMSEY <- Resul_Y_Random[, par_indexes]
Q2Y <- Resul_Y_Random[, impar_indexes]
Q2Y <- as.data.frame(t(Q2Y))
colnames(Q2Y) <- "R2"
rownames(Q2Y) <- c(1:50)
Q2Y$Iteration <- c(1:50)


library(ggplot2)

# Basic scatter plot Y-randomization
ggplot(Q2Y, aes(x = Iteration, y = R2)) + geom_point() + geom_line(aes(y = 0.6),
    linetype = "dotted", color = "red") + ggtitle("Y-randomization test (R²=0.6)") +
    ylab("R²")
```




```
ggsave("plseyrandom.png")
```

```
## Saving 6.5 x 4.5 in image
```

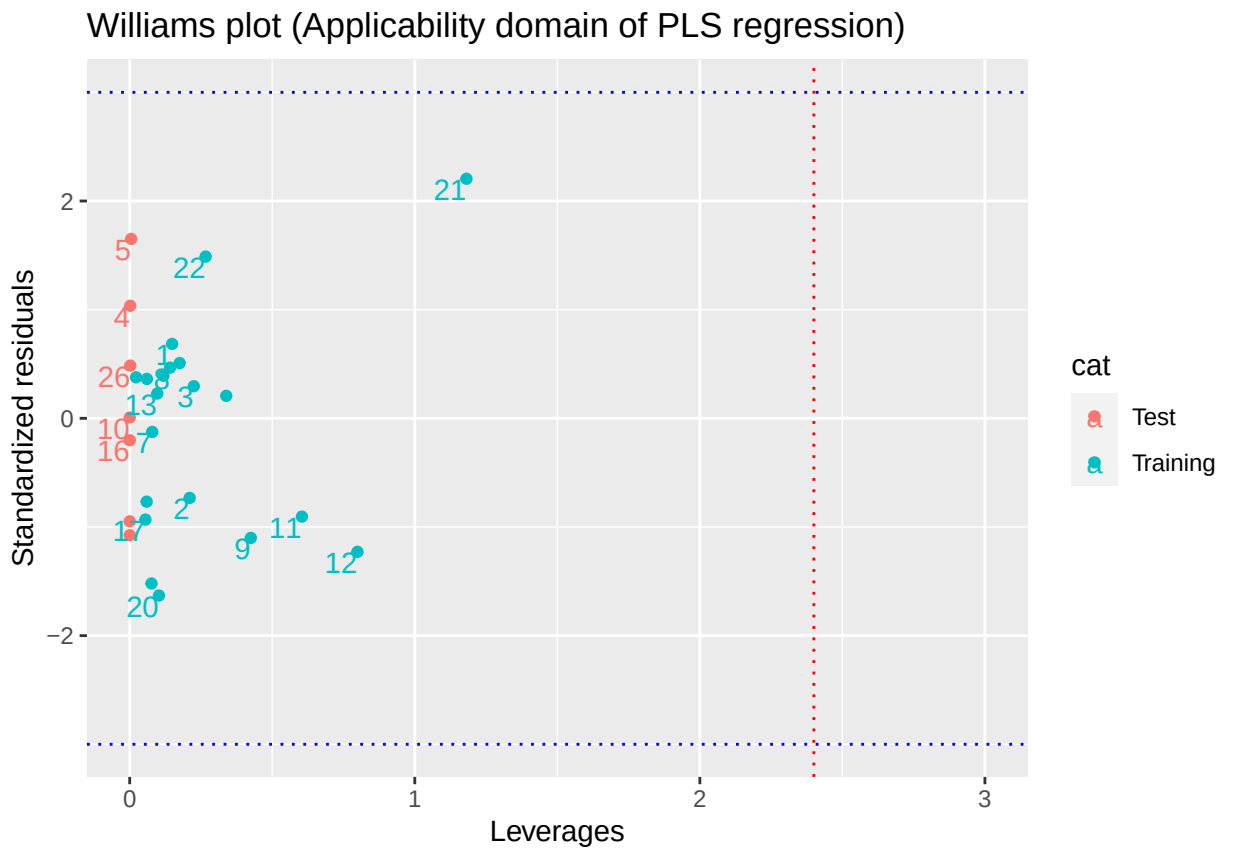
Since none of the randomized models had a better performance than the constructed model (R2=0.6), the QSAR is reliable.

Applicability domain (AD) of the QSAR model

The last step for the development of this QSAR model is the determination of the chemical space gives by the molecular descriptors used to construct the regression. The AD permits to interpolate in a chemical region with reliable results. Finally, the AD also detects possible outliers that are being extrapolated from the model. For more information about AD theory, please refer to the Manuscript.

```
### For training set

## Getting leverages from the model
X_T <- as.matrix(scor_plot)
X_t <- (as.matrix(Modelo_PLS$Yscores))[, 1:2]

H_pls <- X_t %*% solve(t(X_T) %*% X_T) %*% t(X_t)
hi <- diag(H_pls)  ##Leverages

## Getting standardized residuals from the model
obs <- Observado_LOO
pred <- Predicho_LOO

Residuales <- (obs - pred)
```

```r
K <- 20
Stand_Residuales <- Residuales/sd(Residuales)  ##Residuales

## Estimating warning leverage (every molecule that relies outside this limit,
## can't be interpolated and its predicted value is not trustworthy)

h_asterisco <- (3 * (15 + 1))/K  ##Fifteen regressors (PC) and 20 observations h*=3(p+1)/n

### For test set

## Leverages from test set

H_1 <- predict(Modelo_PLS_Final, Prueba_pls, ncomp = 1)
H_2 <- predict(Modelo_PLS_Final, Prueba_pls, ncomp = 2)
X_prueba <- (cbind(H_1, H_2))

H_test <- X_prueba %*% solve(t(X_T) %*% X_T) %*% t(X_prueba)  ##Hat matrix for test
hi_test <- diag(H_test)  ##Leverages


## Mathematical calculation of standardized residuals from test prediction

obs_test <- Prueba_pls$pIC50
pred_test <- predict(Modelo_PLS_Final, Prueba_pls, ncomp = 15)

Residuales_test <- (obs_test - pred_test)
Stand_Residuales_test <- Residuales_test/sd(Residuales_test)


## Dataframes for training and test set of leverages and standardized residuals

Will_training <- as.data.frame(cbind(hi, Stand_Residuales))
colnames(Will_training) <- c("Leverages", "Standardized_residuals")
rownames(Will_training) <- Muestra[1:20]

Will_test <- as.data.frame(cbind(hi_test, Stand_Residuales_test))
colnames(Will_test) <- c("Leverages", "Standardized_residuals")
rownames(Will_test) <- c("4", "5", "10", "15", "16", "19", "26")

library(ggplot2)

Will_training$cat <- "Training"
Will_test$cat <- "Test"

ggplot_William <- rbind(Will_training, Will_test)

# Basic scatter plot Williams plot

ggplot(ggplot_William, aes(x = Leverages, y = Standardized_residuals, color = cat)) +
    geom_point() + xlim(0, 1) + geom_text(label = rownames(ggplot_William), check_overlap = TRUE,
    hjust = 1, vjust = 1) + geom_vline(xintercept = 2.4, linetype = "dotted", color = "red") +
    geom_hline(yintercept = c(-3, 3), linetype = "dotted", color = "blue") + ggtitle("Williams plot (Ap
    ylab("Standardized residuals") + xlim(0, 3)
```




```
ggsave("plsad.png")
```

With the Williams plot, the AD of the model is determined, and apparently all the training and test molecules are correctly interpolated into the chemical space defined by the molecular descriptors selected.

Importance of each descriptor on the PLS model

In order to get some mechanistic insight about the PLS model, let's plot the weight of each descriptor on the PC coefficient used for the PLS regression

```
## Importance of each molecular descriptor into the PLS regression

png("plscoef.png")
coefplot(Modelo_PLS_Final, xlab = "Descriptor", ylab = "Coefficient", main = "Descriptor coefficients o
    comps = 15)
dev.off()
```

```
## pdf
##   2
```

```
## Getting the most important features
PLS_Coeficientes <- as.data.frame(Modelo_PLS_Final$coefficients)

Final_import_Desc <- PLS_Coeficientes[order(PLS_Coeficientes$`pIC50.15 comps`, decreasing = TRUE),
    ]
head(Final_import_Desc)
```

```
##              pIC50.1 comps pIC50.2 comps pIC50.3 comps pIC50.4 comps
```

```
## CATS2D_04_DA   0.006171235  1.466374e-02    0.05282127    0.07805453
## DISPs          0.002375367  6.153599e-03    0.03400446    0.05352277
## B07.C.N.      -0.001310399  9.831898e-05    0.01872885    0.03468890
## DISPe          0.004721987  9.897267e-03    0.03917338    0.05879927
## DipolHybrid    0.007124098  1.381934e-02    0.02287914    0.02906092
## Mv             0.010044638  1.348044e-02    0.01654870    0.02502452
##               pIC50.5 comps pIC50.6 comps pIC50.7 comps pIC50.8 comps
## CATS2D_04_DA    0.09228303    0.10339884    0.11187120    0.11262486
## DISPs           0.06527649    0.07327326    0.08901201    0.09038916
## B07.C.N.        0.04480024    0.05421532    0.07126498    0.07767080
## DISPe           0.07065033    0.07292398    0.07462595    0.07193646
## DipolHybrid     0.03306636    0.03879357    0.04354278    0.04512809
## Mv              0.03035670    0.03606857    0.04391382    0.04418908
##               pIC50.9 comps pIC50.10 comps pIC50.11 comps
## CATS2D_04_DA    0.11251081     0.11206522     0.11213863
## DISPs           0.09170510     0.09018397     0.08892946
## B07.C.N.        0.08066379     0.08273837     0.08477687
## DISPe           0.07038313     0.06893050     0.06872966
## DipolHybrid     0.04960974     0.04989728     0.05129797
## Mv              0.04511565     0.04579271     0.04623238
##               pIC50.12 comps pIC50.13 comps pIC50.14 comps
## CATS2D_04_DA     0.11089179     0.11085060     0.11102692
## DISPs            0.08866683     0.08963071     0.08962992
## B07.C.N.         0.08647523     0.08626405     0.08626518
## DISPe            0.06956826     0.07102805     0.07138766
## DipolHybrid      0.05321482     0.05409537     0.05424894
## Mv               0.04596231     0.04577521     0.04567802
##               pIC50.15 comps
## CATS2D_04_DA     0.11155015
## DISPs            0.08980167
## B07.C.N.         0.08640160
## DISPe            0.07172378
## DipolHybrid      0.05449581
## Mv               0.04559548
```

Now, the QSAR model is finished.

Virtual screening

After the development of the QSAR model; two databases of molecules were taken from **PubChem** and **ZINC**. This molecules were filter by two parameters: **Glucosamine-like molecules** and **Lipinski rule of five**.

The *Lipinski rule of five* is a empirical rule for druglikeness of molecules, the physicochemical parameters for this rule are:

- No more than 5 hydrogen bond donors.
- No more than 10 hydrogen bond acceptors.
- A molecular mass less than 500 g/mol.
- An octanol-water partition coefficient (Log P) that does not exceed 5.

The *glucosamine-like molecules* are mandatory to kept into the databases since the QSAR was development under glucosamine derivative molecules, and the chemical space is defined or interpolated under this scaffold.

After the filtering we got 155 unique molecules in SMILES format. The correct protonation state for each molecule was carried out by **Gypsum-DL software** with a pH between 7.0 and 7.2. The tautomer and

isomer distribution for each molecule was carried out by the same software, and for each molecule was generated two (310) and three (465) possible tautomer/conformer and protonated states in order to get a better insight of all the chemical states on the dataset.

Finally, the datasets were uploaded into OCHEM web platform in order to calculate mechanistic interpretable 2D and 3D molecular descriptors (the same molecular descriptors used for the QSAR development).

Two variants

```
## Invoking datasets
library(dplyr)

## Alvadescriptors
alvdesc_2 <- read.csv("alvdesc-2.csv", stringsAsFactors = FALSE)

## MOPAC
mopac_2 <- read.csv("mopac-2.csv", stringsAsFactors = FALSE)

## Joining both molecular descriptor datasets

mopac_2$SMILES <- NULL

VS_db <- cbind(alvdesc_2, mopac_2)
SMILES <- VS_db$SMILES
```

Cleaning datasets

```
## Converting to numeric
VS_db <- apply(VS_db, 2, as.numeric)
VS_db <- as.data.frame(VS_db)
VS_db$SMILES <- NULL

## Scaling data
Mean_VS <- as.data.frame(t(Mean))
SD_VS <- as.data.frame(t(SD))
Mean_VS$pIC50 <- NULL
SD_VS$pIC50 <- NULL

VS_db <- select(VS_db, names(Mean_VS))

VS_db <- as.matrix(VS_db)
VS_db_S <- scale(VS_db, center = Mean_VS, scale = SD_VS)
```

Predicting activites

```
Predicho_out_VS <- as.data.frame(predict(Modelo_PLS_Final, as.data.frame(VS_db_S),
    ncomp = 15))

Predicho_out_VS <- Desescalar(Predicho_out_VS)
ID <- c(1:310)

Predicho_out_VS <- cbind(ID, Predicho_out_VS)

colnames(Predicho_out_VS) <- c("Index", "pIC50")
```

```r
## Prediction
Predicho_out_VS <- as.data.frame(Predicho_out_VS)
class(Predicho_out_VS)
```

```
## [1] "data.frame"
```

```r
write.csv(Predicho_out_VS[order(Predicho_out_VS$pIC50, decreasing = TRUE), ], file = "VS_PLS-2.csv",
    row.names = FALSE)
head(Predicho_out_VS)
```

```
##   Index    pIC50
## 1     1 5.213963
## 2     2 5.078698
## 3     3 4.283867
## 4     4 4.056027
## 5     5 3.991421
## 6     6 3.869327
```

Three variants

```r
## Invoking datasets
library(dplyr)

## Alvadescriptors
alvdesc_3 <- read.csv("alvdesc-3.csv", stringsAsFactors = FALSE)

## MOPAC
mopac_3 <- read.csv("mopac-3.csv", stringsAsFactors = FALSE)

## Joining both molecular descriptor datasets

mopac_3$SMILES <- NULL

VS_db3 <- cbind(alvdesc_3, mopac_3)
SMILES3 <- VS_db3$SMILES
```

Cleaning datasets

```r
## Converting to numeric
VS_db3 <- apply(VS_db3, 2, as.numeric)
VS_db3 <- as.data.frame(VS_db3)
VS_db3$SMILES <- NULL

## Scaling data
Mean_VS <- as.data.frame(t(Mean))
SD_VS <- as.data.frame(t(SD))
Mean_VS$pIC50 <- NULL
SD_VS$pIC50 <- NULL

VS_db3 <- select(VS_db3, names(Mean_VS))

VS_db3 <- as.matrix(VS_db3)
VS_db3_S <- scale(VS_db3, center = Mean_VS, scale = SD_VS)
```

Predicting activites

```
Predicho_out_VS3 <- as.data.frame(predict(Modelo_PLS_Final, as.data.frame(VS_db3_S),
    ncomp = 15))
Predicho_out_VS3 <- Desescalar(Predicho_out_VS3)
ID <- c(1:465)

Predicho_out_VS3 <- cbind(ID, Predicho_out_VS3)

colnames(Predicho_out_VS3) <- c("Index", "pIC50")

## Prediction
Predicho_out_VS3 <- as.data.frame(Predicho_out_VS3)
class(Predicho_out_VS3)
```

```
## [1] "data.frame"
```

```
write.csv(Predicho_out_VS3[order(Predicho_out_VS3$pIC50, decreasing = TRUE), ], file = "VS_PLS-3.csv",
    row.names = FALSE)
head(Predicho_out_VS3)
```

```
##   Index    pIC50
## 1     1 5.195210
## 2     2 4.189668
## 3     3 4.076271
## 4     4 3.265262
## 5     5 4.056027
## 6     6 3.308148
```