Main functions

2022-06-15

Apply

```
#apply() is used on vectors and matrices (matrices have rows and cols that
#are simply vectors, so it makes sense)

m <- matrix(c(1:6), nrow = 2)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
#mean of the rows (MARGIN = 1):
mean_rows <- apply(X = m, MARGIN = 1, FUN = mean)
mean_rows
```

```
## [1] 3 4
```

```
#mean of the columns (MARGIN = 2):
mean_cols <- apply(X = m, MARGIN = 2, FUN = mean)
mean_cols
```

```
## [1] 1.5 3.5 5.5
```

```
#length of the rows:
length_rows <- apply(X = m, MARGIN = 1, FUN = length)
length_rows
```

```
## [1] 3 3
```

```
#class of the columns:
length_cols <- apply(X = m, MARGIN = 2, FUN = class)
length_cols
```

```
## [1] "integer" "integer" "integer"
```

Lapply

```r
#the apply() version for lists and data frames:
lapply(iris, class)
```

```
## $Sepal.Length
## [1] "numeric"
##
## $Sepal.Width
## [1] "numeric"
##
## $Petal.Length
## [1] "numeric"
##
## $Petal.Width
## [1] "numeric"
##
## $Species
## [1] "factor"
```

Named vectors

```r
x <- 1:3

names(x) <- c("One", "Two", "Three")
x
```

```
##   One   Two Three
##     1     2     3
```

Welch Two Sample t-test

```r
x <- c(0.4, 0.5, 0.3, 0.6)
y <- c(1.1, 1.0, 1.2, 0.9, 1.1)

test <- t.test(x, y)
test$statistic
```

```
##         t
## -7.415534
```

```r
test$p.value
```

```
## [1] 0.0002809965
```

RNA-seq counts normalization

```r
logtransf <- function(x, base = 2, a = 1) {
  log(x + a, base = base)
}
```

```r
load("C:/Users/seren/Desktop/test_2106.RData")
head(data)
```

```
##              ERBB2_expr SRCIN1_expr   her2_status
## TCGA-3C-AAAU   36.77080    3.033554      Negative
## TCGA-3C-AALI  972.58669   17.174349      Positive
## TCGA-3C-AALJ   54.89144    5.288192 Indeterminate
## TCGA-3C-AALK  165.80549   14.136236      Positive
## TCGA-4H-AAAK   60.51770    4.198549     Equivocal
## TCGA-5L-AAT0   46.80810    3.792047      Negative
```

```r
data$ERBB2_expr <- logtransf(data$ERBB2_expr, 2, 1)
data$SRCIN1_expr <- logtransf(data$SRCIN1_expr, 2, 1)
head(data)
```

```
##              ERBB2_expr SRCIN1_expr   her2_status
## TCGA-3C-AAAU   5.239199    2.012051      Negative
## TCGA-3C-AALI   9.927166    4.183832      Positive
## TCGA-3C-AALJ   5.804556    2.652645 Indeterminate
## TCGA-3C-AALK   7.382023    3.919935      Positive
## TCGA-4H-AAAK   5.942930    2.378109     Equivocal
## TCGA-5L-AAT0   5.579183    2.260642      Negative
```

Creating, subsetting and extracting from data frames

```r
#creating a data frame:
log_expr <- c(5.239199, 9.927166, 5.804556, 7.382023)
dose <- c(0.01, 0.05, 0.01, 0.05)

dataframe <- data.frame(log_expr, dose)
dataframe
```

```
##   log_expr dose
## 1 5.239199 0.01
## 2 9.927166 0.05
## 3 5.804556 0.01
## 4 7.382023 0.05
```

```r
#subsetting a data frame:
subset_data <- data[c(1, 2, 6), c(2,3)]
#equal to: data[c(1,2,6), c("SRCIN_expr", "her2_status")]
subset_data
```

```
##              SRCIN1_expr her2_status
## TCGA-3C-AAAU    2.012051    Negative
## TCGA-3C-AALI    4.183832    Positive
## TCGA-5L-AAT0    2.260642    Negative
```

```r
subset2_data <- data[1:4, 1]
subset2_data #output is a vector (numeric in this case)
```

```
## [1] 5.239199 9.927166 5.804556 7.382023
```

```r
subset3_data <- data[1:4, 1, drop = FALSE]
subset3_data #output is a data frame
```

```
##              ERBB2_expr
## TCGA-3C-AAAU   5.239199
## TCGA-3C-AALI   9.927166
## TCGA-3C-AALJ   5.804556
## TCGA-3C-AALK   7.382023
```

```r
#subsetting by selecting a specific content of rows:
iris_subset <- iris[iris$Sepal.Length >= 5 & iris$Petal.Length >= 5, ]
head(iris_subset)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 78           6.7         3.0          5.0         1.7 versicolor
## 84           6.0         2.7          5.1         1.6 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 102          5.8         2.7          5.1         1.9  virginica
## 103          7.1         3.0          5.9         2.1  virginica
## 104          6.3         2.9          5.6         1.8  virginica
```

```r
#by using | instead of & we'd have selected rows satisfying either one condition
#or the other, but not both
```

```r
#extracting from a data frame:
subset_data$her2_status
```

```
## [1] "Negative" "Positive" "Negative"
```

```r
subset_data[ , 2, drop = FALSE]
```

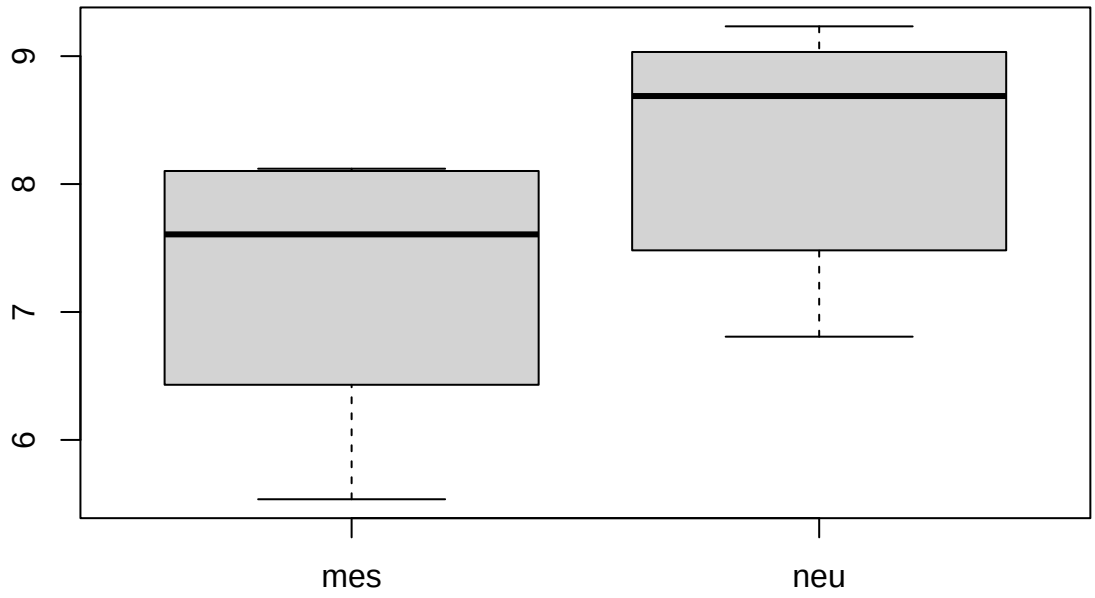
```
##              her2_status
## TCGA-3C-AAAU    Negative
## TCGA-3C-AALI    Positive
## TCGA-5L-AAT0    Negative
```

Plots

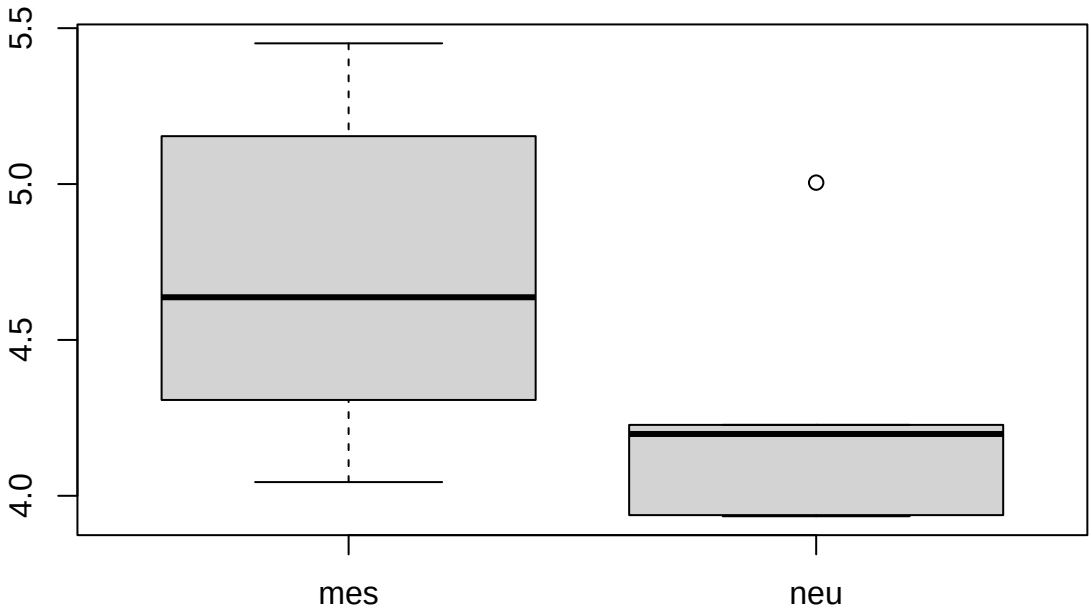
```r
load("C:/Users/seren/Desktop/expr.RData")
head(expr)
```

```
##            mes1     mes2     mes3     mes4     mes5     mes6     neu1     neu2
## RNF14  8.102109 6.431024 7.233297 5.536356 7.980071 8.120170 6.807137 7.481830
## UBE2Q1 9.535052 9.531678 6.666299 6.934365 9.379369 9.046204 9.466937 9.369915
## RNF17  4.149430 4.374377 5.037466 5.135278 4.172273 4.529581 4.609084 4.456974
## RNF10  7.142845 6.882857 5.524656 6.236647 7.379514 6.893025 6.624718 7.057433
## RNF11  9.563294 8.782727 8.522499 6.927014 8.938691 8.707519 8.289256 9.274853
## RNF13  9.552312 9.237758 9.556759 7.951833 9.873957 9.457725 9.294424 9.016314
##            neu3      neu4     neu5      neu6
## RNF14   9.232656  8.474975 9.032406  8.901127
## UBE2Q1  9.171364  9.165563 9.742695  9.236809
## RNF17   4.161883  4.132093 4.293262  4.159367
## RNF10   6.897234  7.229054 6.788821  6.940517
## RNF11   9.317841  9.505812 9.941937  9.478019
## RNF13  10.465560 10.228730 9.599728 10.633883
```

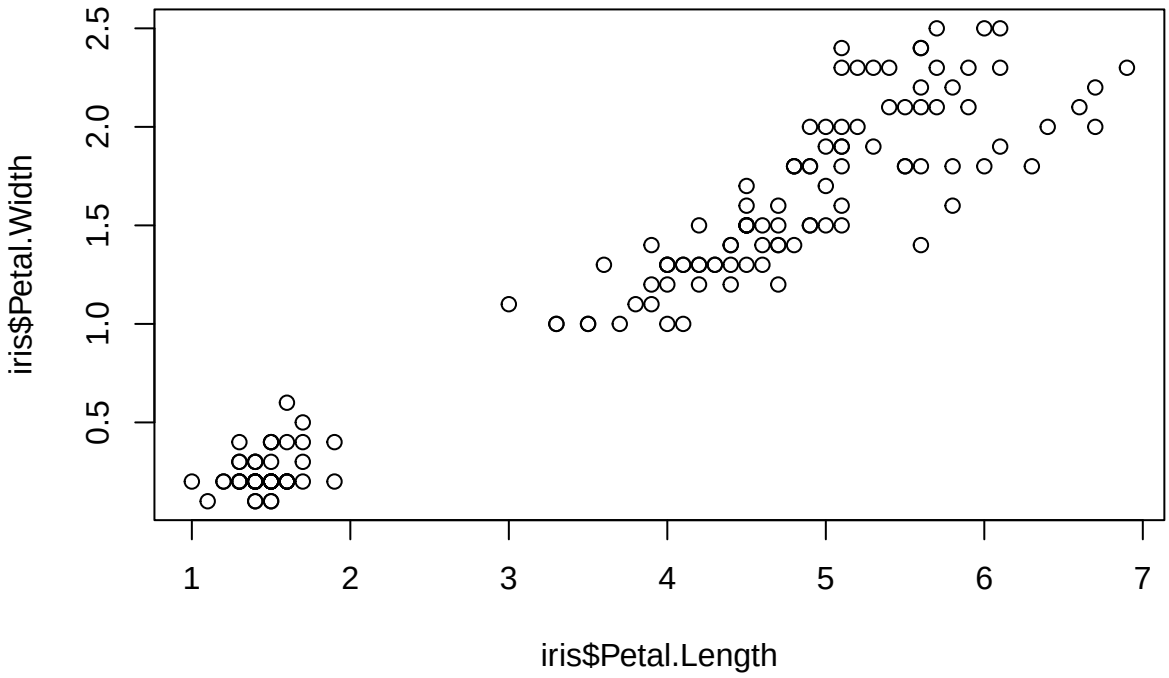
```
#comparing expresssion of RNF14 gene in "mes" vs "neu" samples:
boxplot(expr[1, 1:6], expr[1, 7:12], names = c("mes", "neu"))
```




```
#evaluating all genes' expression in "mes" vs "neu":
boxplot(expr[12042, 1:6], expr[12042, 7:12], names = c("mes", "neu"))
```

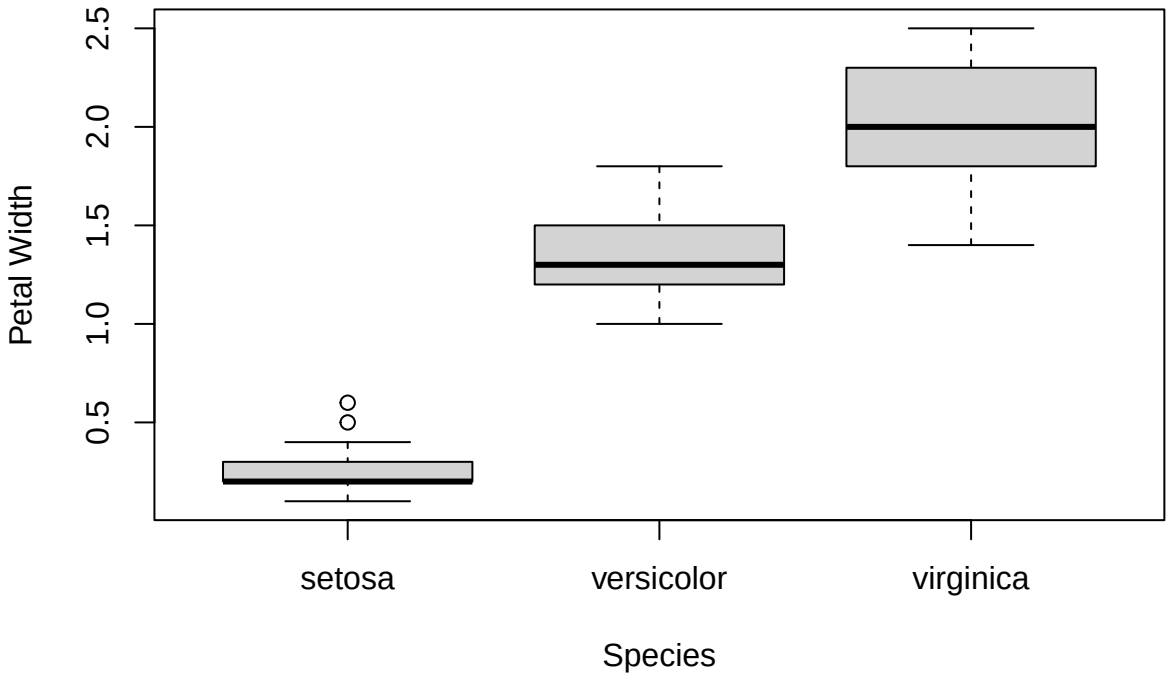



```
#plot(x, y)

plot(x = iris$Petal.Length, y = iris$Petal.Width)
```




```
#equal to: plot(iris$Petal.Width ~ iris$Petal.Length)

#both x and y are numerical, continuous variables.
```

```
plot(x = iris$Species, y = iris$Petal.Width, xlab = "Species", ylab = "Petal Width")
```




```
#the independent (x) variable is a categorical variable
#the dependent (y) variable is a numerical, continuous variable
```

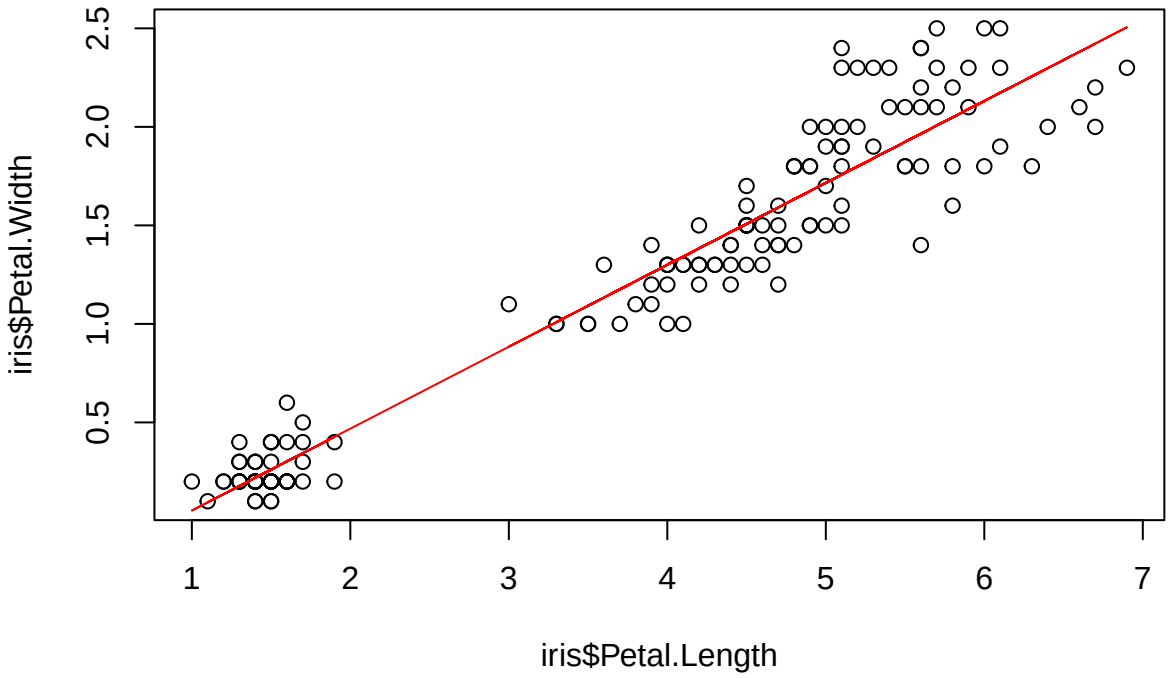
Univariable regression models

Linear regression

```
#lm() is used to fit linear models, including multivariate ones. Used to carry out regression.
#Used to predict a numerical variable from either a numerical (1) or categorical (2) variable:

#1:
lin_reg1 <- lm(iris$Petal.Width ~  iris$Petal.Length)
#where the width depends on the length (y ~  x)

summary(lin_reg1)$coefficients
```

```
##                     Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)       -0.3630755 0.039761990 -9.131221 4.699798e-16
## iris$Petal.Length  0.4157554 0.009582436 43.387237 4.675004e-86
```

```
plot(x = iris$Petal.Length, y = iris$Petal.Width)
lines(iris$Petal.Length, lin_reg1$fitted.values, col = "red")
```




```
#2:
lin_reg2 <- lm(iris$Petal.Width ~  iris$Species)

summary(lin_reg2)$coefficients
```

```
##                         Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)                0.246 0.02894188  8.499792 1.959455e-14
## iris$Speciesversicolor     1.080 0.04093000 26.386510 1.254978e-57
## iris$Speciesvirginica      1.780 0.04093000 43.488878 7.951748e-86
```

Logistic regression

```
#Based on the glm() function, which fits generalized linear models.
#Used to predict a categorical variable from either a numerical or categorical one.

#The categorical var. to predict must be provided in 0-1 values of T-F logicals:

data_logreg <- data
data_logreg[data_logreg$her2_status == "Positive", "her2_status"] <- 1
data_logreg[data_logreg$her2_status == "Negative" | data_logreg$her2_status == "Equivocal" | data_logreg

data_logreg$her2_status <- as.numeric(data_logreg$her2_status)

head(data_logreg)
```

```
##              ERBB2_expr SRCIN1_expr her2_status
## TCGA-3C-AAAU   5.239199    2.012051           0
## TCGA-3C-AALI   9.927166    4.183832           1
## TCGA-3C-AALJ   5.804556    2.652645           0
## TCGA-3C-AALK   7.382023    3.919935           1
## TCGA-4H-AAAK   5.942930    2.378109           0
## TCGA-5L-AAT0   5.579183    2.260642           0
```

```
#instead of transofmring existing columns, it's easier to initialize new ones
#with the desired values
```

```
class(data_logreg$her2_status)
```

```
## [1] "numeric"
```

```
#logistic regression can now be carried out:
log_reg <- glm(data_logreg$her2_status ~ data_logreg$ERBB2_expr, family = "binomial")
summary(log_reg)$coefficients
```

```
##                          Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)            -7.175388 0.48569493 -14.77345 2.173139e-49
## data_logreg$ERBB2_expr  0.977223 0.08032575  12.16575 4.730808e-34
```

Multivariable regression

```
#When there is more than a regressor.

multi_lin <- lm(iris$Sepal.Length ~ iris$Petal.Length + iris$Petal.Width)
summary(multi_lin)$coefficients
```

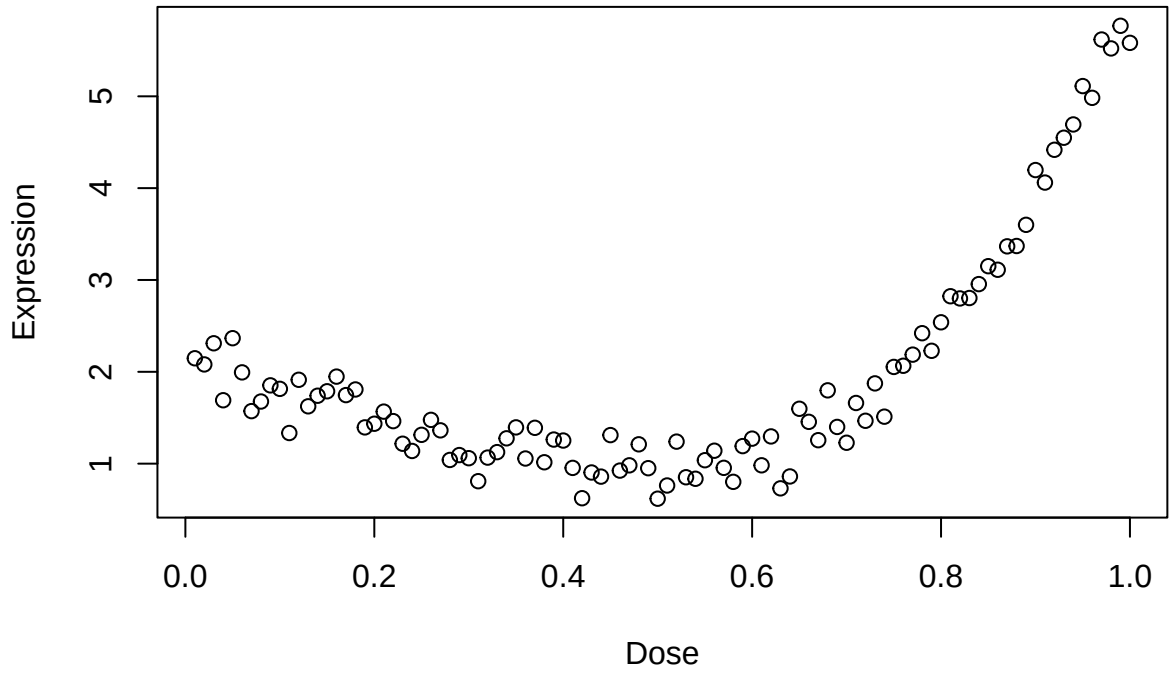
```
##                       Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)         4.1905824 0.09704587 43.181459 2.092645e-85
## iris$Petal.Length   0.5417772 0.06928179  7.819907 9.414477e-13
## iris$Petal.Width   -0.3195506 0.16045262 -1.991557 4.827246e-02
```

Linear regression for non-linear models (polynomial regression)

```
#It always require the lm() function.
load("C:/Users/seren/Desktop/test_2107.RData")
head(exp1)
```

```
##   dose expression
## 1 0.01   2.147618
## 2 0.02   2.080570
## 3 0.03   2.310649
## 4 0.04   1.690693
## 5 0.05   2.366353
## 6 0.06   1.993866
```

```r
plot(exp1$expression ~ exp1$dose, xlab = "Dose", ylab = "Expression")
```




```r
#Let's start with a simple univariable linear model:
lm1 <- lm(exp1$expression ~ exp1$dose)
summary(lm1)$coefficients
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 0.6404943  0.2033794 3.149258 2.170136e-03
## exp1$dose   2.5957978  0.3496423 7.424153 4.202027e-11
```

```r
summary(lm1)$r.squared
```

```
## [1] 0.3599709
```

```r
#Let's continue by adding terms (regressors):

#+x^2
lm2 <- lm(exp1$expression ~ poly(exp1$dose, degree = 2, raw = TRUE))
summary(lm2)$coefficients
```

```
##                                            Estimate Std. Error   t value
## (Intercept)                                2.804316  0.1060304  26.44822
## poly(exp1$dose, degree = 2, raw = TRUE)1 -10.132567  0.4845879 -20.90966
```

```
## poly(exp1$dose, degree = 2, raw = TRUE)2  12.602341  0.4648467  27.11075
##                                              Pr(>|t|)
## (Intercept)                                3.852369e-46
## poly(exp1$dose, degree = 2, raw = TRUE)1 1.033304e-37
## poly(exp1$dose, degree = 2, raw = TRUE)2 4.638879e-47
```

```
summary(lm2)$r.squared
```

```
## [1] 0.9253806
```

```
#+x^3
lm3 <- lm(exp1$expression ~ poly(exp1$dose, degree = 3, raw = TRUE))
summary(lm3)$coefficients
```

```
##                                          Estimate Std. Error   t value
## (Intercept)                               2.035663 0.08641486 23.556859
## poly(exp1$dose, degree = 3, raw = TRUE)1 -1.220988 0.73728616 -1.656057
## poly(exp1$dose, degree = 3, raw = TRUE)2 -9.346647 1.69170849 -5.524975
## poly(exp1$dose, degree = 3, raw = TRUE)3 14.487781 1.10133610 13.154732
##                                             Pr(>|t|)
## (Intercept)                               1.104264e-41
## poly(exp1$dose, degree = 3, raw = TRUE)1 1.009755e-01
## poly(exp1$dose, degree = 3, raw = TRUE)2 2.818034e-07
## poly(exp1$dose, degree = 3, raw = TRUE)3 3.313981e-23
```

```
summary(lm3)$r.squared
```

```
## [1] 0.9733747
```

```
#+x^4
lm4 <- lm(exp1$expression ~ poly(exp1$dose, degree = 4, raw = TRUE))
summary(lm4)$coefficients
```

```
##                                          Estimate Std. Error    t value
## (Intercept)                               2.121855  0.1102306 19.2492458
## poly(exp1$dose, degree = 4, raw = TRUE)1 -2.854714  1.4962597 -1.9078999
## poly(exp1$dose, degree = 4, raw = TRUE)2 -2.151868  5.9820383 -0.3597215
## poly(exp1$dose, degree = 4, raw = TRUE)3  3.443763  8.8780564  0.3878961
## poly(exp1$dose, degree = 4, raw = TRUE)4  5.467336  4.3613306  1.2535935
##                                             Pr(>|t|)
## (Intercept)                               1.494720e-34
## poly(exp1$dose, degree = 4, raw = TRUE)1 5.942414e-02
## poly(exp1$dose, degree = 4, raw = TRUE)2 7.198538e-01
## poly(exp1$dose, degree = 4, raw = TRUE)3 6.989607e-01
## poly(exp1$dose, degree = 4, raw = TRUE)4 2.130661e-01
```

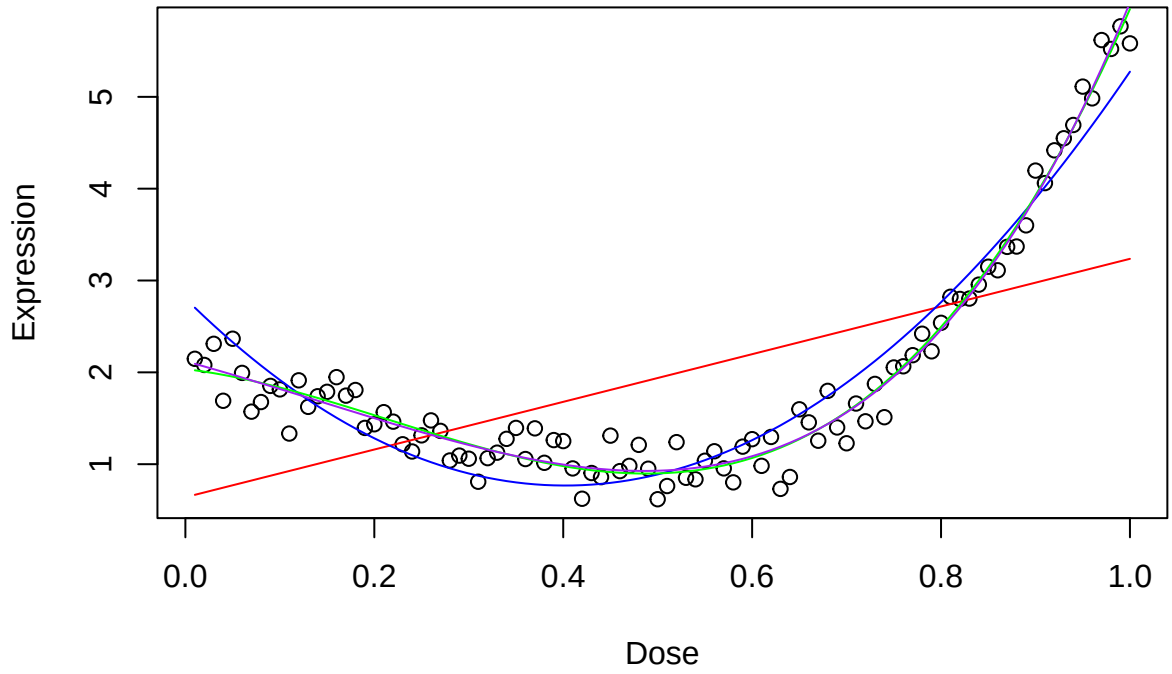
```
summary(lm4)$r.squared
```

```
## [1] 0.9738079
```

and this last model (in general all the others) corresponds to: x <- exp1$dose x2 <- x^2 x3 <- x^3 x4 <- x^4

lm_test <- lm(exp1$expression ~ x + x2 + x3 + x4)

```
#Let's plot all these models on the plot above:

plot(exp1$expression ~ exp1$dose, xlab = "Dose", ylab = "Expression")

lines(exp1$dose, lm1$fitted.values, col = "red")
lines(exp1$dose, lm2$fitted.values, col = "blue")
lines(exp1$dose, lm3$fitted.values, col = "green")
lines(exp1$dose, lm4$fitted.values, col = "purple")
```



ANOVA test

```
#ANOVA test needs nested models and tells which model between two nested models
#is the best describes the relationship between the given variables:

pl_sl <- lm(iris$Sepal.Length ~ iris$Petal.Length)
plpw_sl <- lm(iris$Sepal.Length ~ iris$Petal.Length + iris$Petal.Width)

an_test <- anova(plpw_sl, pl_sl)
an_test
```

```
## Analysis of Variance Table
##
```

```
## Model 1: iris$Sepal.Length ~ iris$Petal.Length + iris$Petal.Width
## Model 2: iris$Sepal.Length ~ iris$Petal.Length
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    147 23.881
## 2    148 24.525 -1  -0.64434 3.9663 0.04827 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#the multivariable model plpw_sl doesn't add much info, as it is not really significant
#(P-value = 0.04827)
```

```
#Let's test the best model between the above-described ones using ANOVA:
an_1 <- anova(lm2, lm1)
an_1
```

```
## Analysis of Variance Table
##
## Model 1: exp1$expression ~ poly(exp1$dose, degree = 2, raw = TRUE)
## Model 2: exp1$expression ~ exp1$dose
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     97 11.639
## 2     98 99.827 -1   -88.189 734.99 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
an_2 <- anova(lm3, lm2)
an_2
```

```
## Analysis of Variance Table
##
## Model 1: exp1$expression ~ poly(exp1$dose, degree = 3, raw = TRUE)
## Model 2: exp1$expression ~ poly(exp1$dose, degree = 2, raw = TRUE)
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     96  4.1528
## 2     97 11.6386 -1   -7.4858 173.05 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
an_3 <- anova(lm4, lm3)
an_3
```

```
## Analysis of Variance Table
##
## Model 1: exp1$expression ~ poly(exp1$dose, degree = 4, raw = TRUE)
## Model 2: exp1$expression ~ poly(exp1$dose, degree = 3, raw = TRUE)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     95 4.0853
## 2     96 4.1528 -1 -0.067579 1.5715 0.2131
```

```
#lm 2 is better than lm1 but worse than lm3, while lm3 is superior to all the others.
#lm3 is the model best describing the dependence of expression to dose.
```

Cross-validation

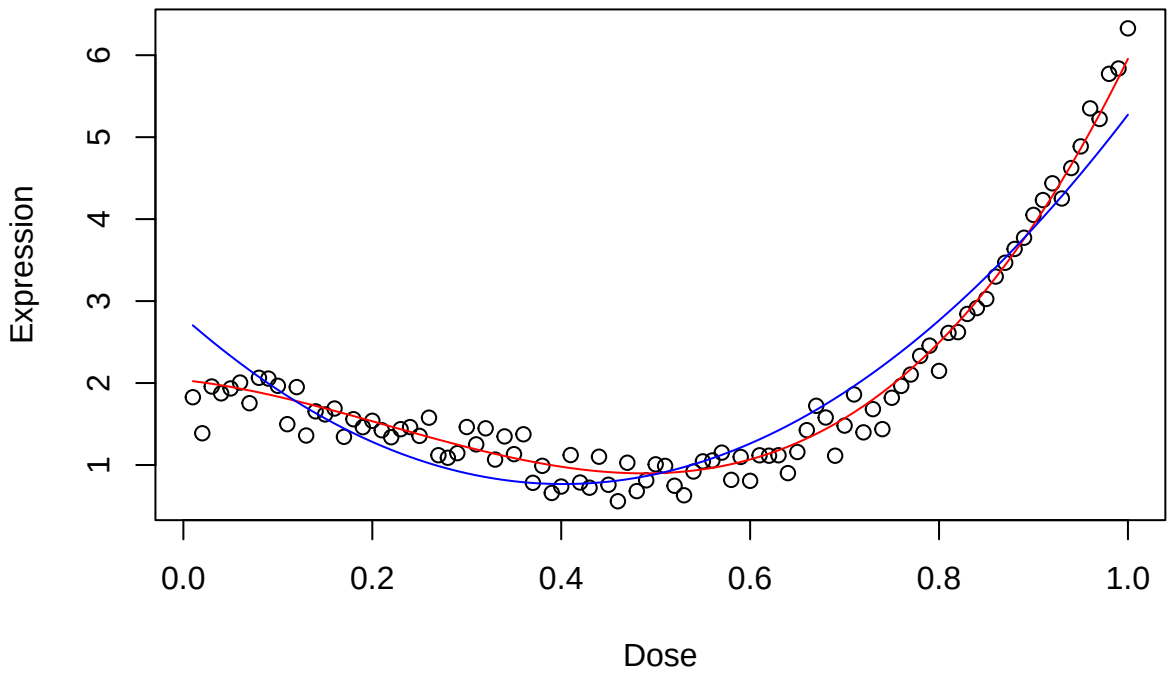
```
#We used ANOVA to test different regression models on the very same dataset.
#The overfitting problem is not overcome in this way though and we can't be sure
#that R^2 is really significant (as it will always increase by adding regressors).

#Cross-validation tests a regression model obtained on a dataset 1 on a new dataset.
#The two datasets can be different or be two parts of the same original dataset:
#in this case they are called "training set" and "testing set".

#The training set is 1/3 of the original dataset, the testing one is 2/3.

head(data)
```

```
##              ERBB2_expr SRCIN1_expr   her2_status
## TCGA-3C-AAAU   5.239199    2.012051      Negative
## TCGA-3C-AALI   9.927166    4.183832      Positive
## TCGA-3C-AALJ   5.804556    2.652645 Indeterminate
## TCGA-3C-AALK   7.382023    3.919935      Positive
## TCGA-4H-AAAK   5.942930    2.378109     Equivocal
## TCGA-5L-AAT0   5.579183    2.260642      Negative
```

```
#Let's start from CV on two different datasets.
```

(1)

```
#We evaluate the best regression model obtained on "exp1" dataset (lm3)
#on another similarly-obtained dataset called "exp2", comparing it to another
#model (less performative than lm3 in the exp1 dataset, let's take lm2):
express_pred_lm3 <- predict(lm3, newdata = exp2)
express_pred_lm2 <- predict(lm2, newdata = exp2)

plot(exp2$expression ~ exp2$dose, xlab = "Dose", ylab = "Expression")
lines(exp2$dose, express_pred_lm3, col = "red")
lines(exp2$dose, express_pred_lm2, col = "blue")
```

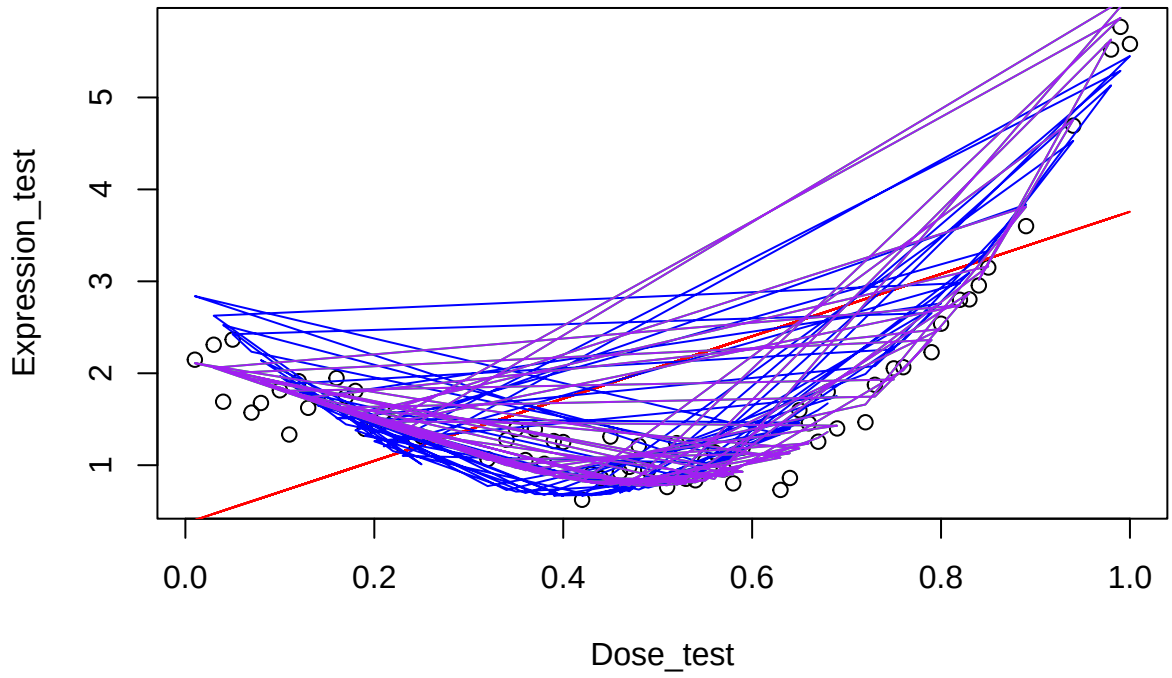



```
#Let's try now dividing a dataset into training and testing sets.
```

(2)

```
#Let's use the dataset "exp1". It has 100 rows, thus the division will be:
random <- sample(1:100, 100)

exp1_train <- exp1[random[1:33], ]
exp1_test <- exp1[random[34:100], ]

#the "random" vector was to select randomly values to create the two datasets.

#The model should be re-calculated on the training set only:
lm_1 <- lm(expression ~ dose, data = exp1_train)
lm_2 <- lm(expression ~ poly(dose, degree = 2, raw = TRUE), data = exp1_train)
lm_3 <- lm(expression ~ poly(dose, degree = 3, raw = TRUE), data = exp1_train)
lm_4 <- lm(expression ~ poly(dose, degree = 3, raw = TRUE), data = exp1_train)

express_pred_lm_1 <- predict(lm_1, newdata = exp1_test)
express_pred_lm_2 <- predict(lm_2, newdata = exp1_test)
express_pred_lm_3 <- predict(lm_3, newdata = exp1_test)
express_pred_lm_4 <- predict(lm_4, newdata = exp1_test)

plot(exp1_test$expression ~ exp1_test$dose, xlab = "Dose_test", ylab = "Expression_test")
lines(exp1_test$dose, express_pred_lm_1, col = "red")
```

```
lines(exp1_test$dose, express_pred_lm_2, col = "blue")
lines(exp1_test$dose, express_pred_lm_3, col = "green")
lines(exp1_test$dose, express_pred_lm_4, col = "purple")
```




```
#viene strano, venisse sensato continuerei con il test R^2 e avrei fatto anche ANOVA
```

R^2

```
#To compute R^2 when doing cross-validation, this is the function:

r2 <- function(y, y_pred) {
  1 - sum((y - y_pred)^2)/sum((y - mean(y))^2)
}
```

```
#Let's compute the R^2 of the two models (lm3, lm2) applied to the exp2 dataset:
r2_express_pred_lm3 <- r2(exp2$expression, express_pred_lm3)
r2_express_pred_lm3
```

```
## [1] 0.9777476
```

```
r2_express_pred_lm2 <- r2(exp2$expression, express_pred_lm2)
r2_express_pred_lm2
```

```
## [1] 0.9146092
```

```
#If this is true, then lm3 is really better than lm2 (not only in exp1):
r2_express_pred_lm3 > r2_express_pred_lm2
```

```
## [1] TRUE
```

```
#we can also compare lm3 to other models we generated before.
```