Data Analysis in Software Engineering

Author: Javier Dolado. Collaborator: Daniel Rodriguez

This is the R Markdown document for the talk **"Data Analysis for Software Engineering"**, given at the *UCL-Software System Engineering (SSE) Reading Group (RG)* on Wednesday, 12th of August, 2015. This document requires the R system and the Rstudio installed. This document executes all chunks of R code and generates an html document.

The purpose of the talk is to give a hands-on experience of a data analysis in software engineering. We use basic statistical procedures for analysing some public software engineering datasets.

R Markdown This is an R Markdown presentation. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Important: for compiling this document you need to change the paths to the images used below, i.e., change */home/javier/DocProjects/PRESI2013/london2015/prediction.png* to the folder where you have downloaded the content of https://github.com/javierdolado/uclsserg2015

Outline of the talk

- Setting the environment
- The purpose of analysing data in software engineering
- Getting data
- Exploratory Data Analysis
- Model Building for Prediction
  - Linear Regression
  - Machine Learning Techniques
- Model Evaluation
  - Descriptive statistics
- Confidence Intervals
- Hypothesis Testing
  - p-values
  - Equivalence hypothesis testing
- Frequentism vs Bayesianism
- Bayesian Networks

Setting up the programming environment

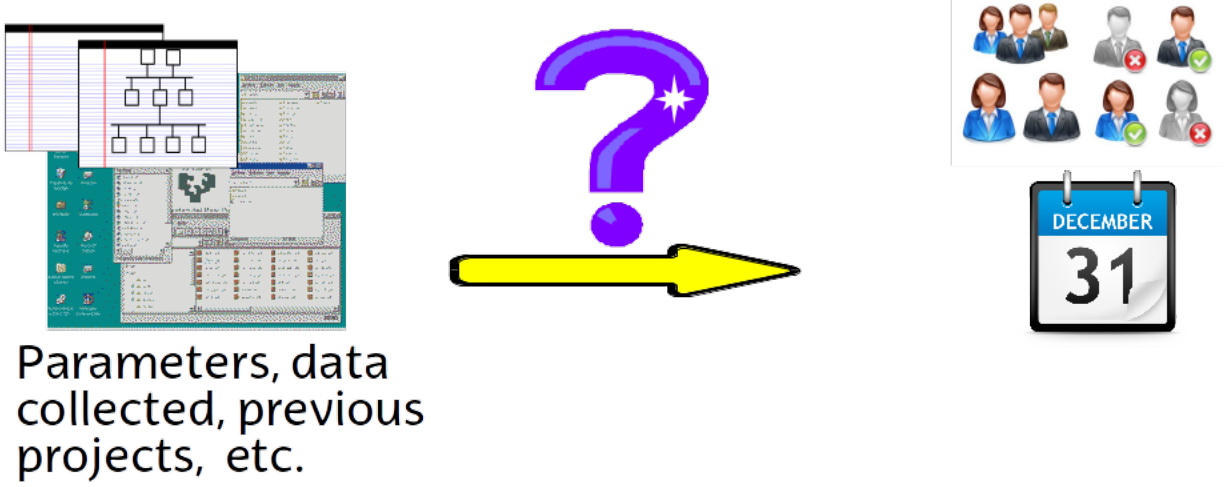
- Install R and Rstudio
- Install packages needed with the sentence install.packages("here-the-package-name")
- Example:

```
#  install.packages("rmarkdown")
#  install.packages("knitr")
```

will install the package rmarkdown on your R system - Load packages, when needed, with -> library(here-the-package-name) - Download the Rmarkdown document and other documents that can be found in https://github.com/javierdolado/uclsserg2015

The Aim of Data Analysis and Statistical Learning

- The aim of any data analysis is to **understand the data**
- and to build models for making predictions and estimating future events based on past data.
- We may want to test different hypothesis on the data
- Most probably we are interested in building a model for time and effort prediction



- Basic References:
  - John Verzani, simpleR - Using R for Introductory Statistics
  - Peter Dalgaard, *Introductory Statistics with R*, 2nd ed., Springer, 2008
  - Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning with Applications in R*, Springer 2014
  - Geoff Cumming, *Understanding the New Statistics: Effect Sizes, Confidence Intervals, and Meta_Analysis*, Routledge, New York, 2012

Getting the Data

- We may have collected and recorded some data about effort, defects, quality and other variables
- We may have performed some experiments controlling different variables

- or we use some databases already available
  - Public dataset: PROMISE repository
  - Private datasets: ISBSG, Tukutuku
- The datasets are used *only* for illustration purposes
- The datasets **should not** be used reliably in an industrial setting
- We will use the China and Telecom1 datasets
- The Telecom1 dataset is used in M. Shepperd and S. MacDonell, *Evaluating prediction systems in software project estimation*, Information and Software Technology, 2012

Cleaning the Data

- Most probably the hardest part of the analysis!!
- None of the datasets is ready for analysis
- Missing data, confusing figures, inconsistent rows
- Data is provided "as is", no guarantee on its accuracy
- ISBSG: we had to remove both attributes (variables) and instances
- Transformations needed in almost all cases

Descriptive statistics

- The first task with any dataset is to characterise it in terms of summary statistics and graphics
- Displaying information graphically will help us to identify the main characteristics of the data. To describe a distribution we often want to know where it is centered and and what the spread is (mean, median, quantiles)
- Basic Plots:
  - Histogram: The histogram defines a sequence of breaks and then counts the number of observation in the bins formed by the breaks.
  - Boxplot: The boxplot is used to summarize data succinctly, quickly displaying if the data is symmetric or has suspected outliers.
  - Q-Q plot: This plot is used to determine if the data is close to being normally distributed. The quantiles of the standard normal distribution is represented by a straight line. The normality of the data can be evaluated by observing the extent in which the points appear on the line. When the data is normally distributed around the mean, then the mean and the median should be equal.
  - Scatterplot: A scatter plot provides a graphical view of the relationship between two sets of numbers: one numerical variable against another.

Normality

- A normal distribution is an arrangement of a data set in which most values cluster in the middle of the range
- A graphical representation of a normal distribution is sometimes called a bell curve because of its shape.
- Many procedures in statistics are based on this property. Parametric procedures require the normality property.

Getting the Data. Descriptive statistics.

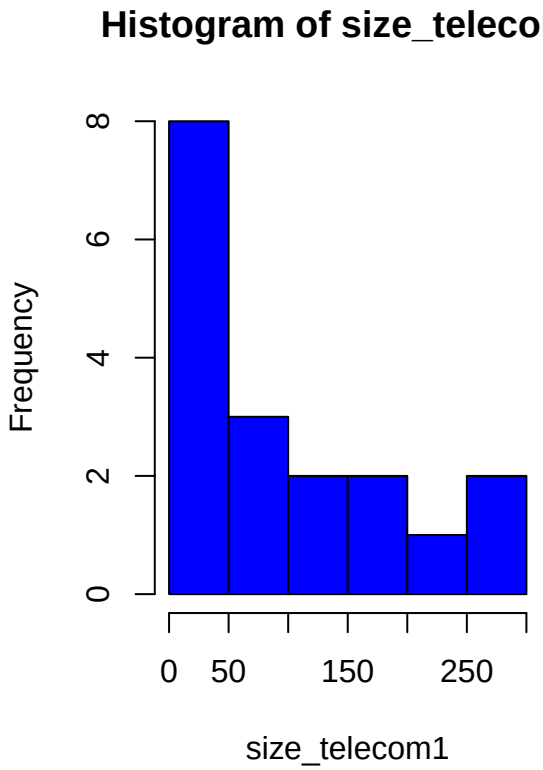
- Set the path to your files
- Read the Telecom1 dataset and print out the summary statistics with the command *summary*

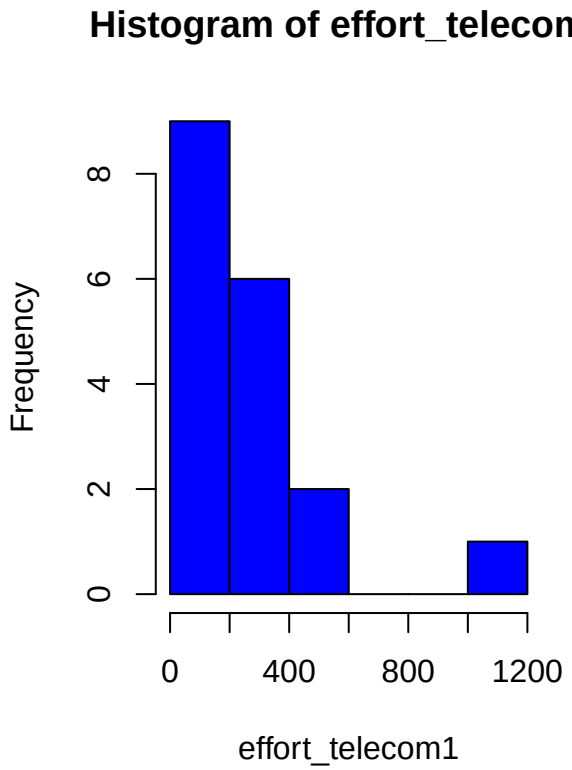
```
path2files <- "~/DocProjects/PRESI2013/london2015" # Set here the path to the folder where you have dow
setwd(path2files) #this command sets the default directory
telecom1 <- read.table("Telecom1.csv", sep=",",header=TRUE, stringsAsFactors=FALSE, dec = ".") #read da
summary(telecom1)
```

```
##       size            effort           EstTotal
##  Min.   :  3.00   Min.   :  23.54   Min.   : 30.0
##  1st Qu.: 37.25   1st Qu.: 119.18   1st Qu.:141.8
##  Median : 68.50   Median : 222.03   Median :289.0
##  Mean   :100.33   Mean   : 284.28   Mean   :320.3
##  3rd Qu.:164.00   3rd Qu.: 352.27   3rd Qu.:471.5
##  Max.   :284.00   Max.   :1115.54   Max.   :777.0
```

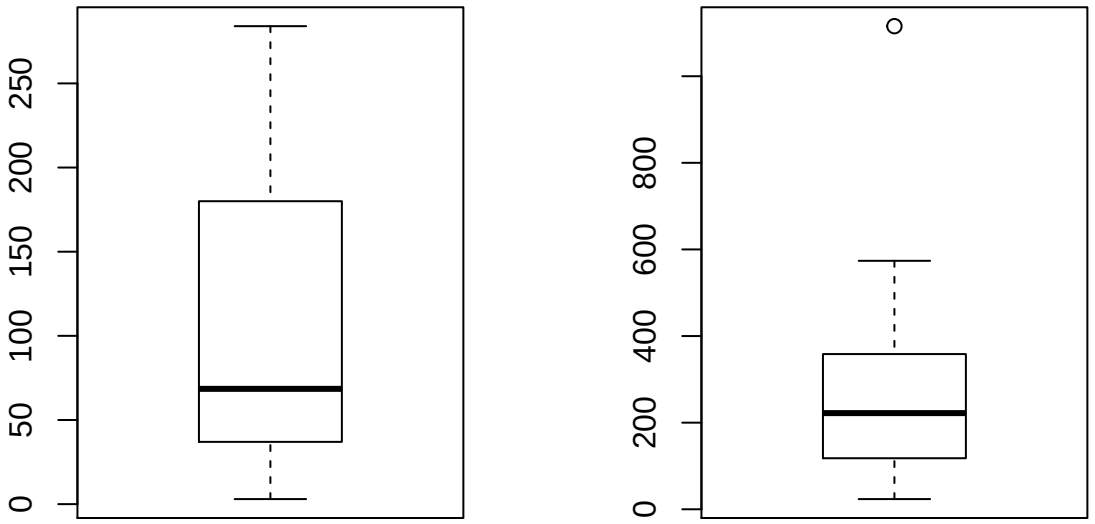
- We see that this dataset has three variables (or parameters) and few data points (18)
  - size: the independent variable
  - effort: the dependent variable
  - EstTotal: the estimates coming from an estimation method
- Basic Plots

```
par(mfrow=c(1,2)) #two figures per row
size_telecom1 <- telecom1$size
effort_telecom1 <- telecom1$effort
hist(size_telecom1, col="blue")
hist(effort_telecom1, col="blue")
```

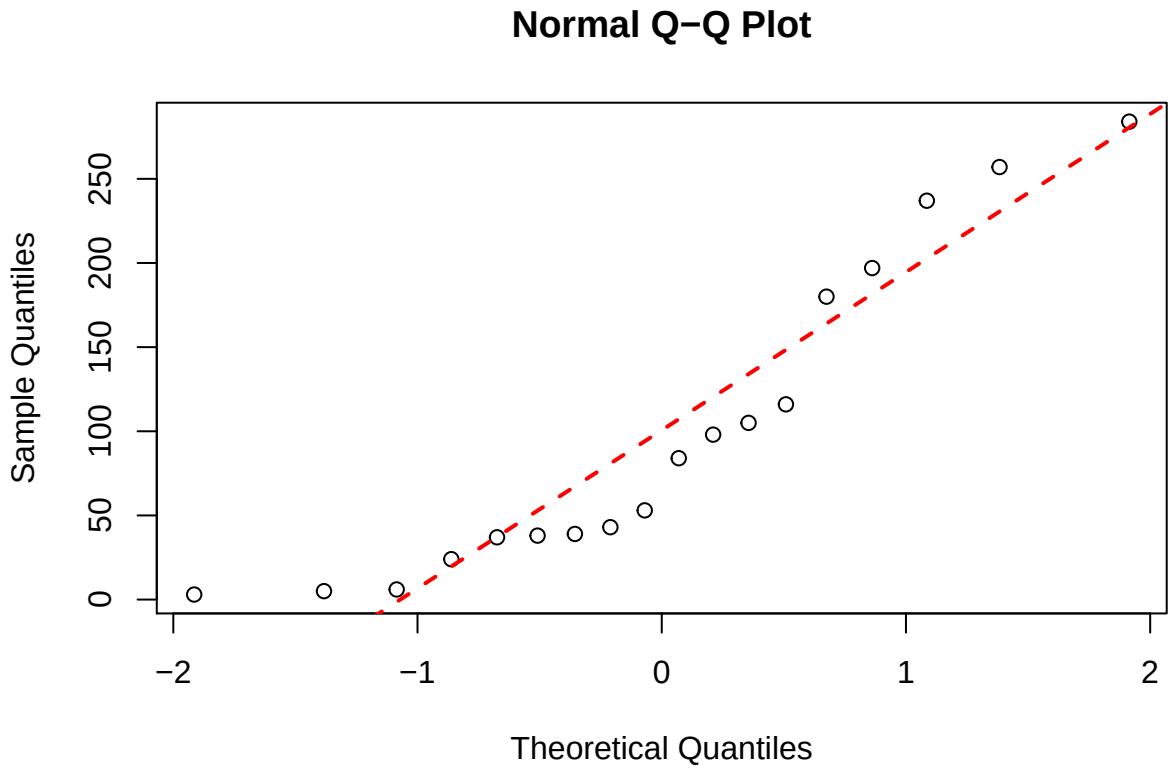




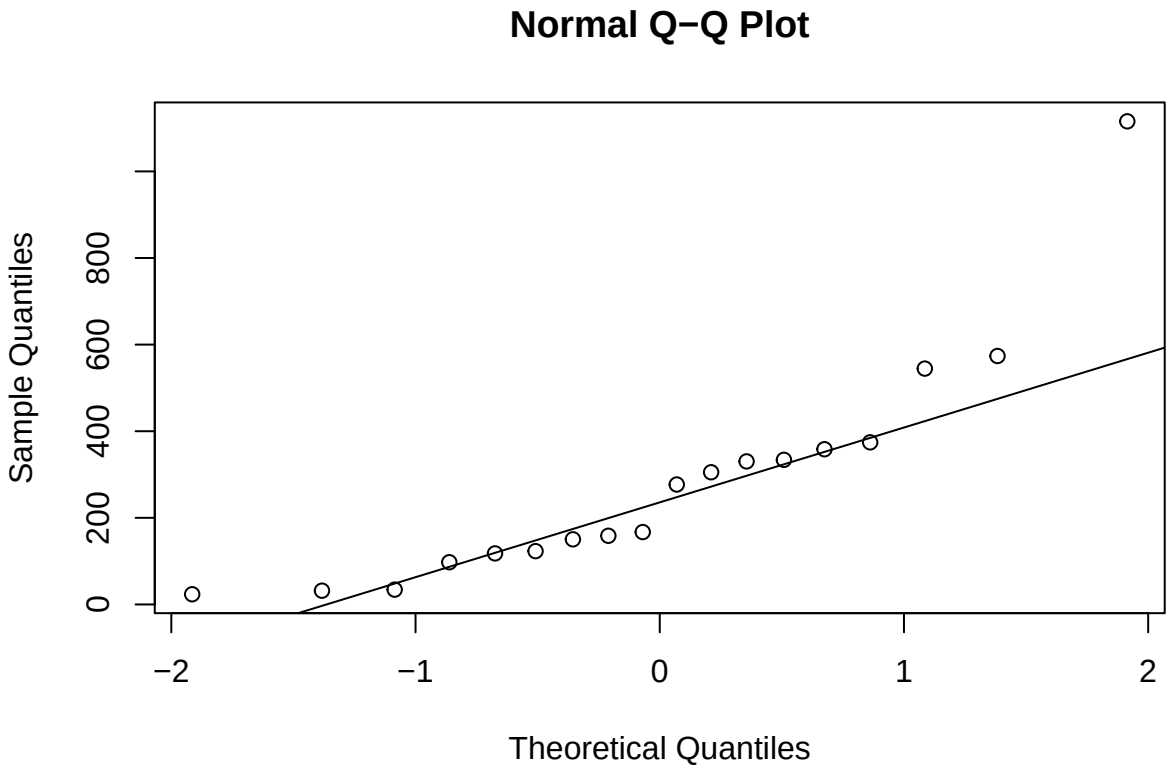
```r
boxplot(size_telecom1)
boxplot(effort_telecom1)
```




```r
par(mfrow=c(1,1))
qqnorm(size_telecom1)
qqline(size_telecom1, col=2, lwd=2, lty=2) #draws a line through the first and third quartiles
```




```r
qqnorm(effort_telecom1)
qqline(effort_telecom1)
```



- We observe the non-normality of the data.

China dataset

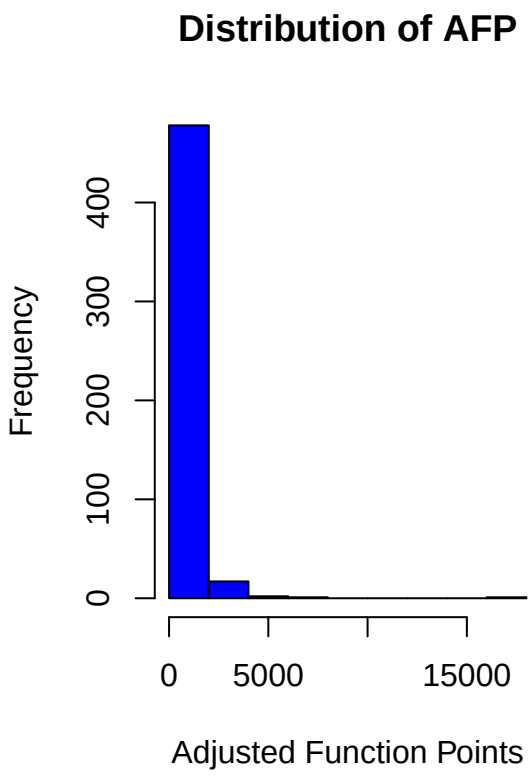
```
library(foreign)
china <- read.arff("china.arff")
china_size <- china$AFP
summary(china_size)
```

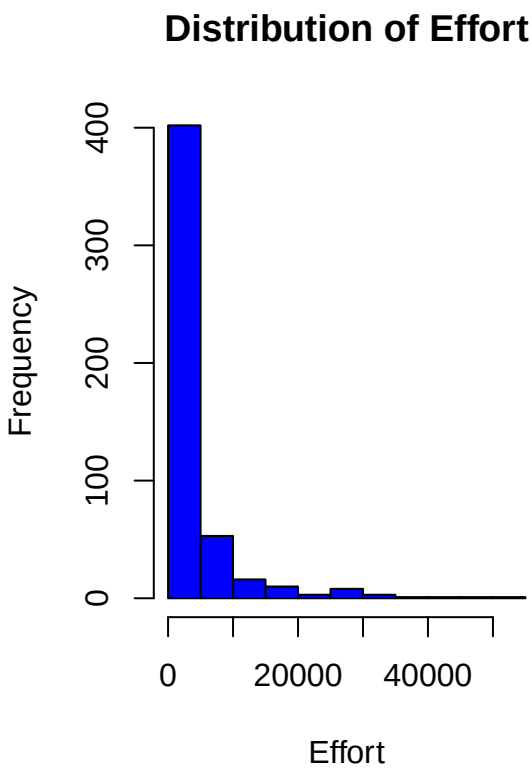
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     9.0   100.5   215.0   486.9   437.5 17520.0
```

```
china_effort <- china$Effort
summary(china_effort)
```

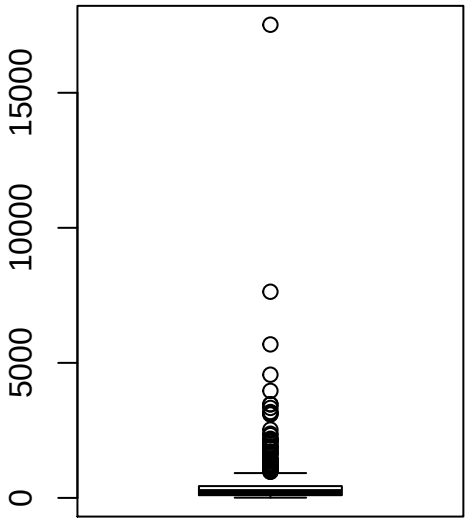
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    26.0   703.5  1829.0  3921.0  3826.0 54620.0
```

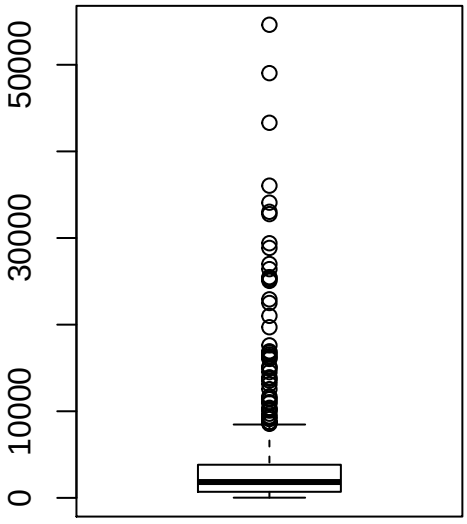
```
par(mfrow=c(1,2))
hist(china_size, col="blue", xlab="Adjusted Function Points", main="Distribution of AFP")
hist(china_effort, col="blue",xlab="Effort", main="Distribution of Effort")
```



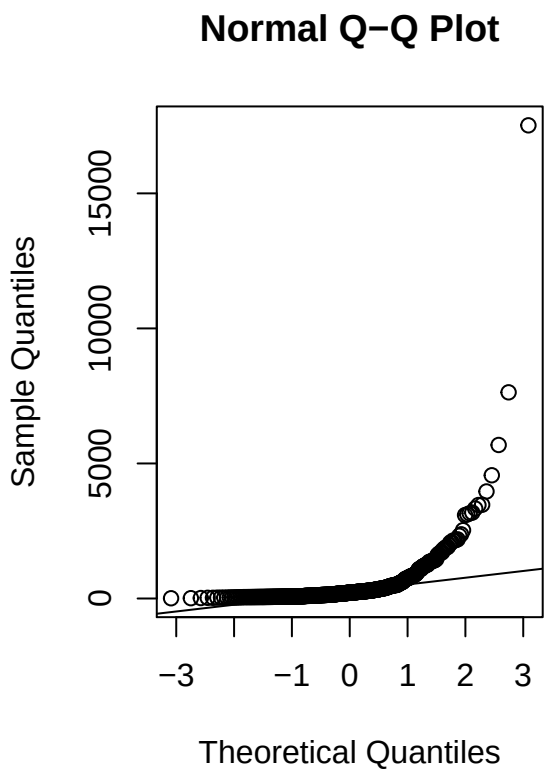


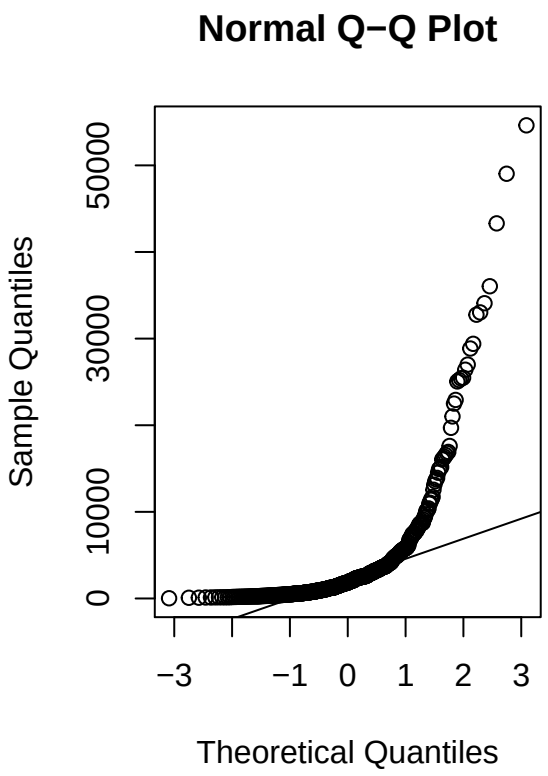

```
boxplot(china_size)
boxplot(china_effort)
```





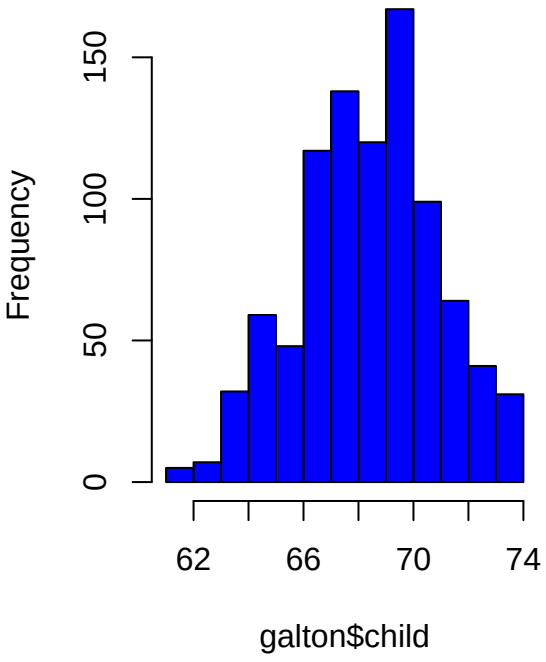

```
qqnorm(china_size)
qqline(china_size)
qqnorm(china_effort)
qqline(china_effort)
```

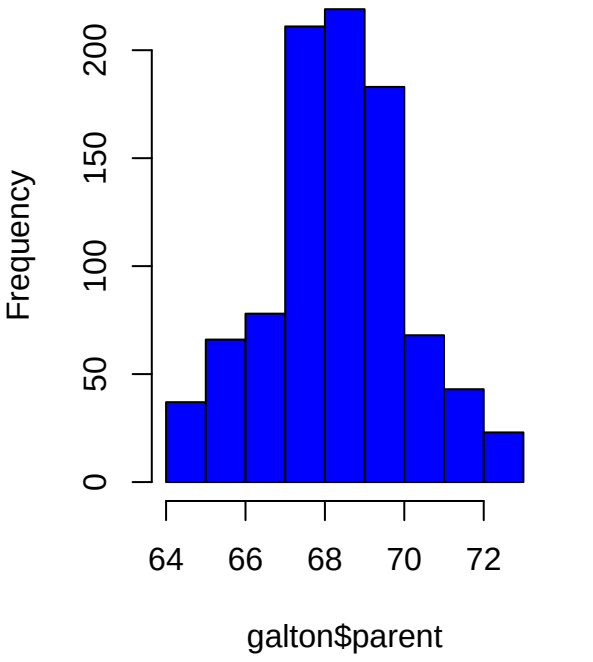




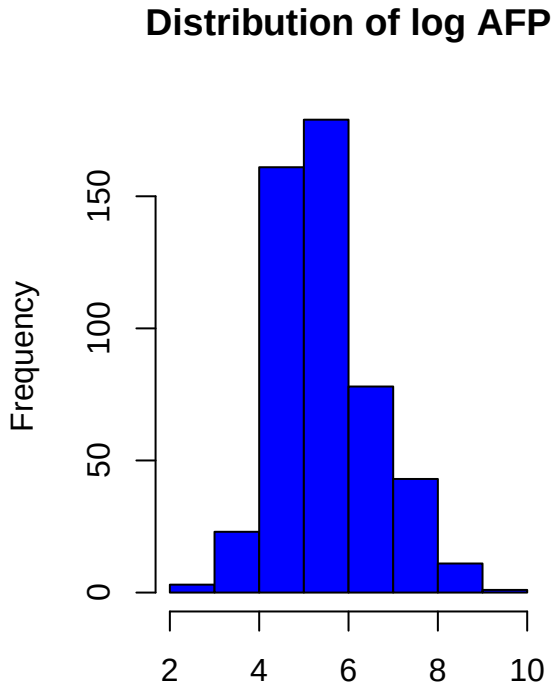
- We observe the non-normality of the data.

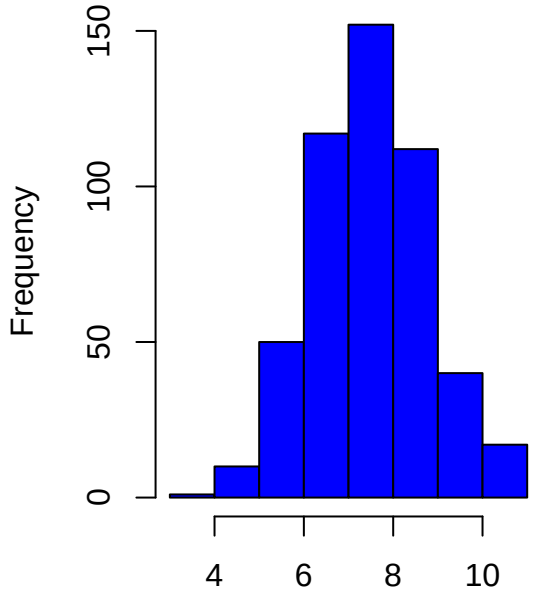
Normality. Galton data



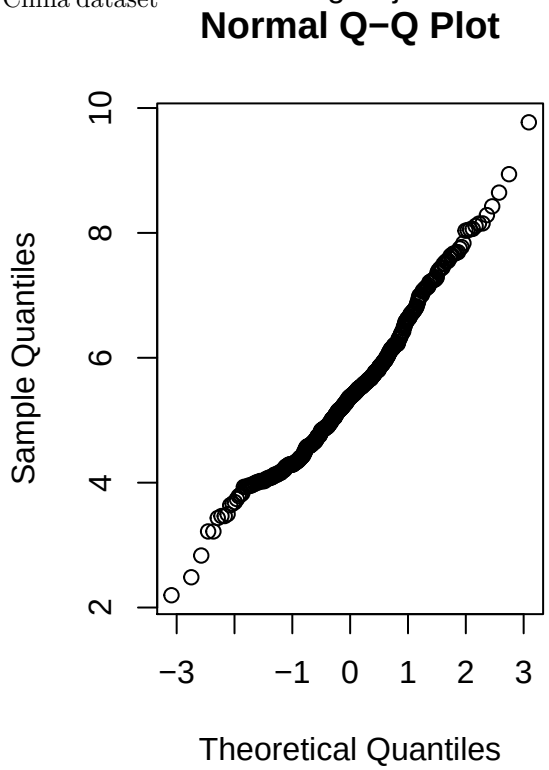


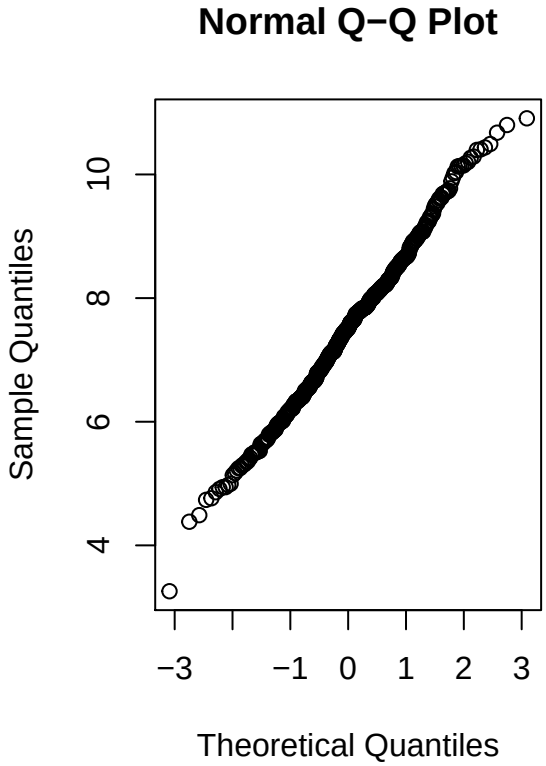
Normalization





- China dataset



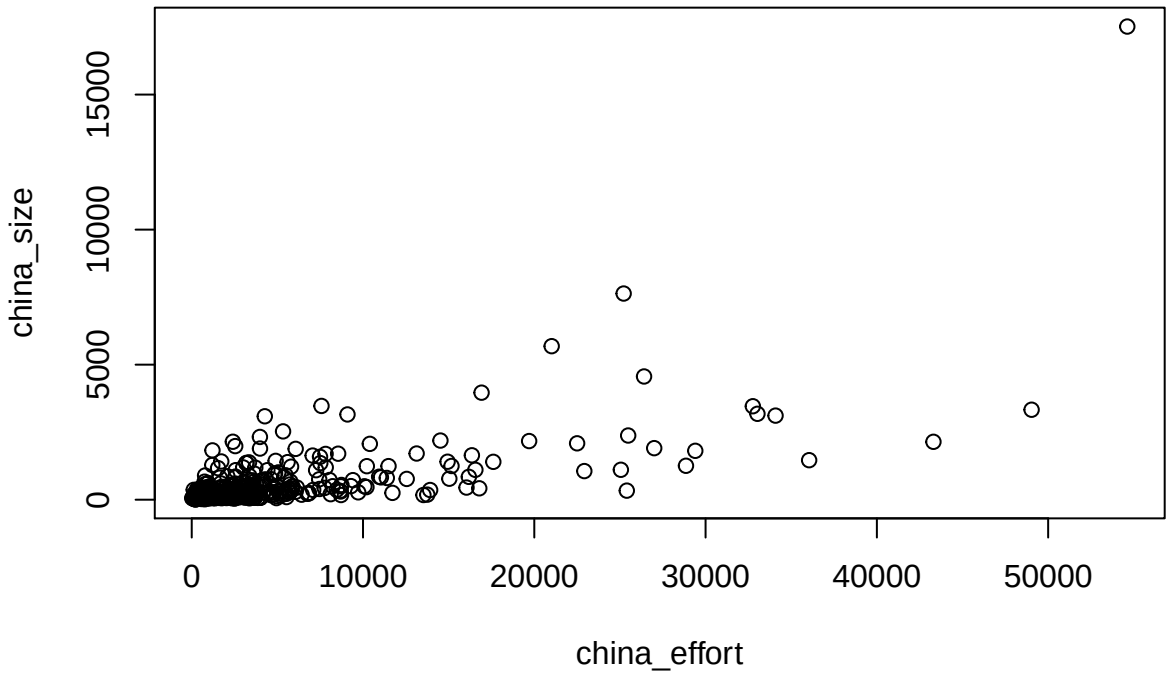


- If the log-log transformation is used the estimation equation is

$$y = e^{b_0 + b_1 log(x)}$$

China dataset. Correlation.

- With the whole dataset we may check for the Correlation of the variables we are interested in




```
## [1] 0.684644
```

```
##
##  Pearson's product-moment correlation
##
## data:  china_size and china_effort
## t = 20.9406, df = 497, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6350288 0.7286338
## sample estimates:
##      cor
## 0.684644
```

```
## [1] 0.6491668
```

```
## [1] 0.4684293
```

Linear Regression

- This procedure fits a straight line to the data. The idea is that the independent variable x is something the experimenter controls and the dependent variable y is something that the experimenter measures. The line is used to predict the value of y for a known value of x. The variable x is the predictor variable and y the response variable.

- First proposed many, many years ago. But still very useful ...

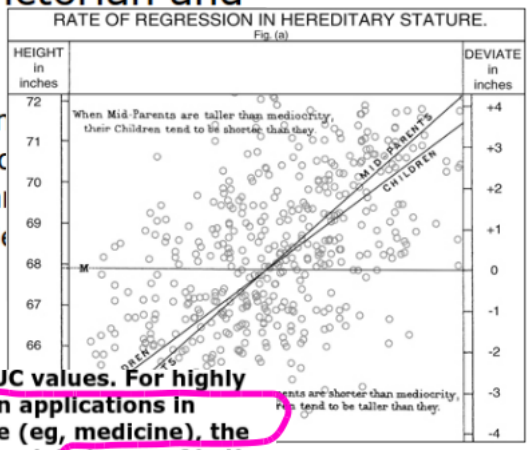
Article

European Journal of Human Genetics (2009) **17,** 1070–1075;
doi:10.1038/ejhg.2009.5; published online 18 February 2009

Predicting human height by Victorian and genomic methods

Yurii S Aulchenko[1,2,7], Maksim V Struchalir
M Belonogova[2,4], Tatiana I Axenovich[2], Mic
Albert Hofman[1], Andre G Uitterlinden[6], Ma
Ben A Oostra[1], Cornelia M van Duijn[1], A Ce
W Janssens[1] and Pavel M Borodin[2,4]

Sir Francis Galton, 1886

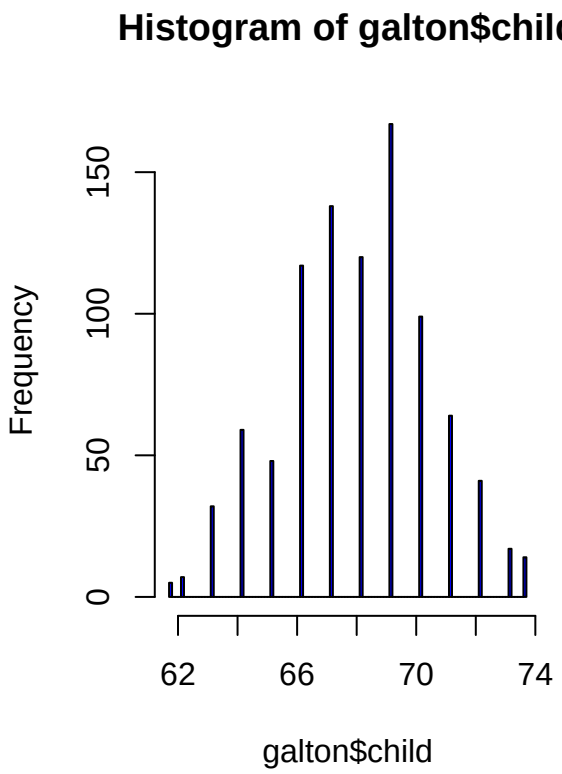


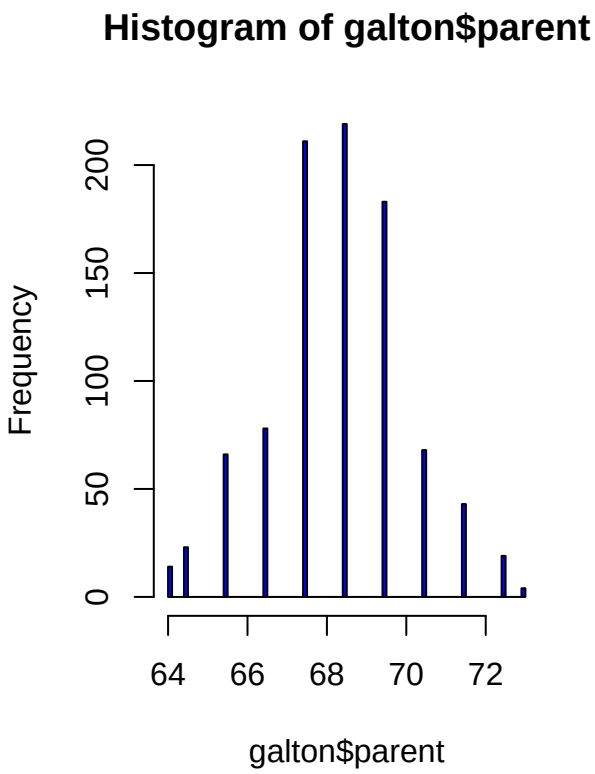
genomic profile should explain to reach certain AUC values. For highly heritable traits such as height, we conclude that in applications in which parental phenotypic information is available (eg, medicine), the Victorian Galton's method will long stay unsurpassed, in terms of both discriminative accuracy and costs. For less heritable traits, and in situations in which parental information is not available (eg, forensics), genomic methods may provide an alternative, given that

- The equation takes the form *haty*.....
- The method used to choose the values b_0 and b_1 is to minimize the sum of the squares of the residual errors.

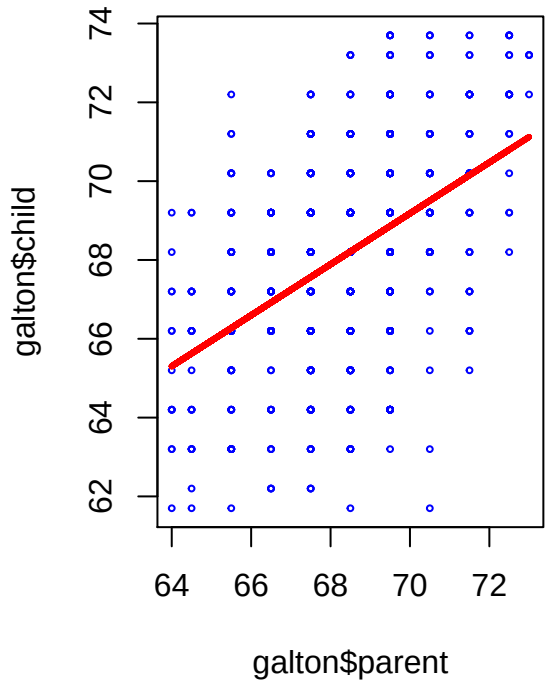
Galton Data

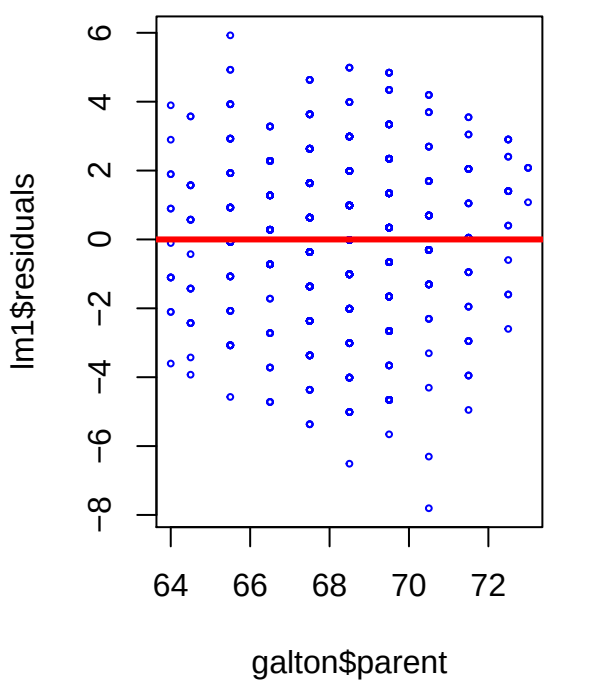
```
# library(UsingR); data(galton)
par(mfrow=c(1,2))
hist(galton$child,col="blue",breaks=100)
hist(galton$parent,col="blue",breaks=100)
```



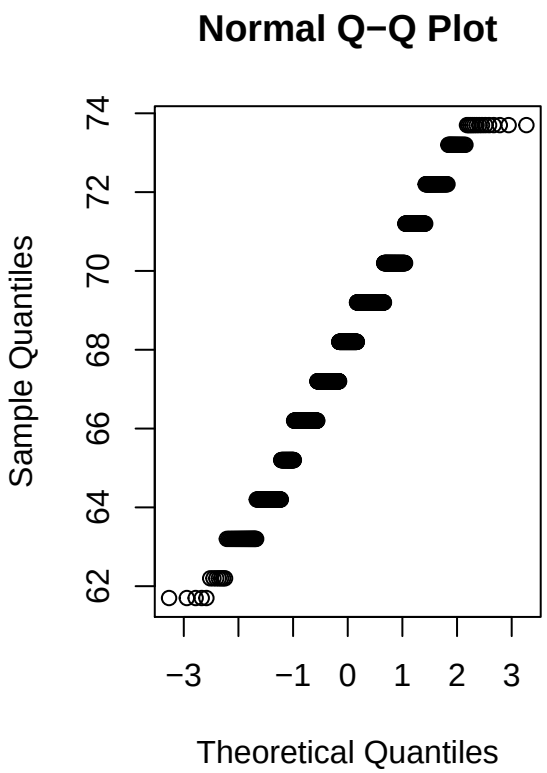



```
plot(galton$parent,galton$child,pch=1,col="blue", cex=0.4)
lm1 <- lm(galton$child ~ galton$parent)
lines(galton$parent,lm1$fitted,col="red",lwd=3)
plot(galton$parent,lm1$residuals,col="blue",pch=1, cex=0.4)
abline(c(0,0),col="red",lwd=3)
```





```r
qqnorm(galton$child)
```



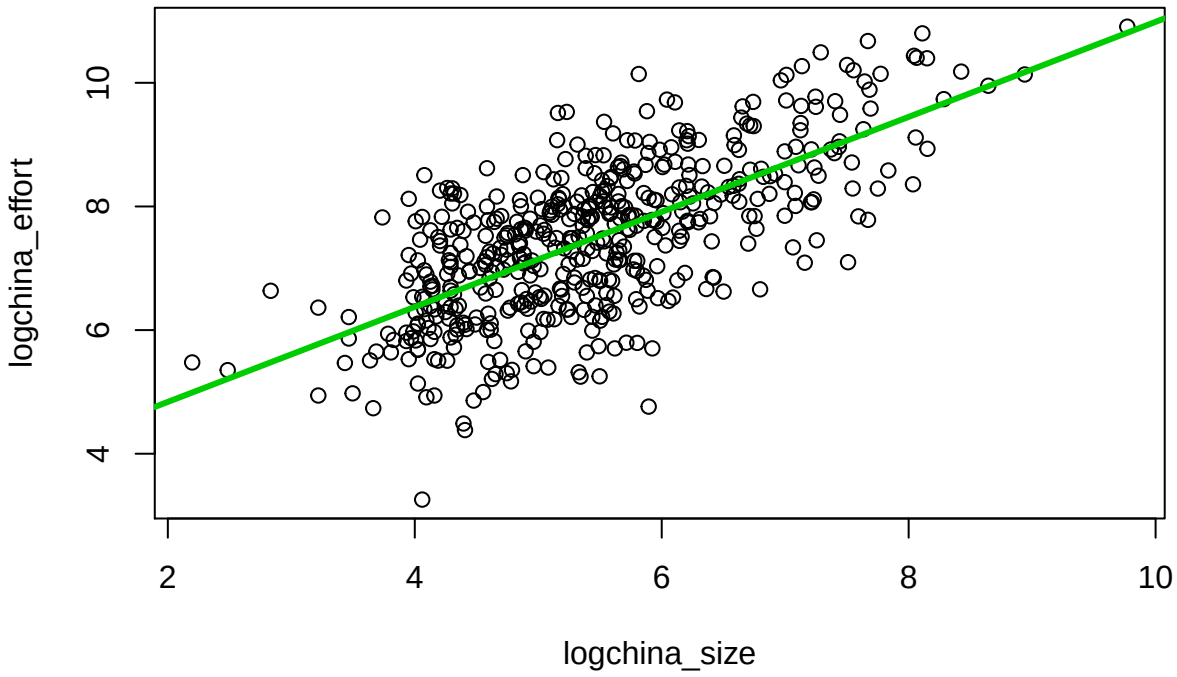
Linear Regression Diagnostics

- Several plots help to evaluate the suitability of the linear regression
  - Residuals vs fitted: The residuals should be randomly distributed around the horizontal line representing a residual error of zero; that is, there should not be a distinct trend in the distribution of points.
  - Standard Q-Q plot: residual errors are normally distributed
  - Square root of the standardized residuals vs the fitted values: there should be no obvious trend.
  - Leverage: measures the importance of each point in determining the regression result. Smaller values means that removing the observation has little effect on the regression result.

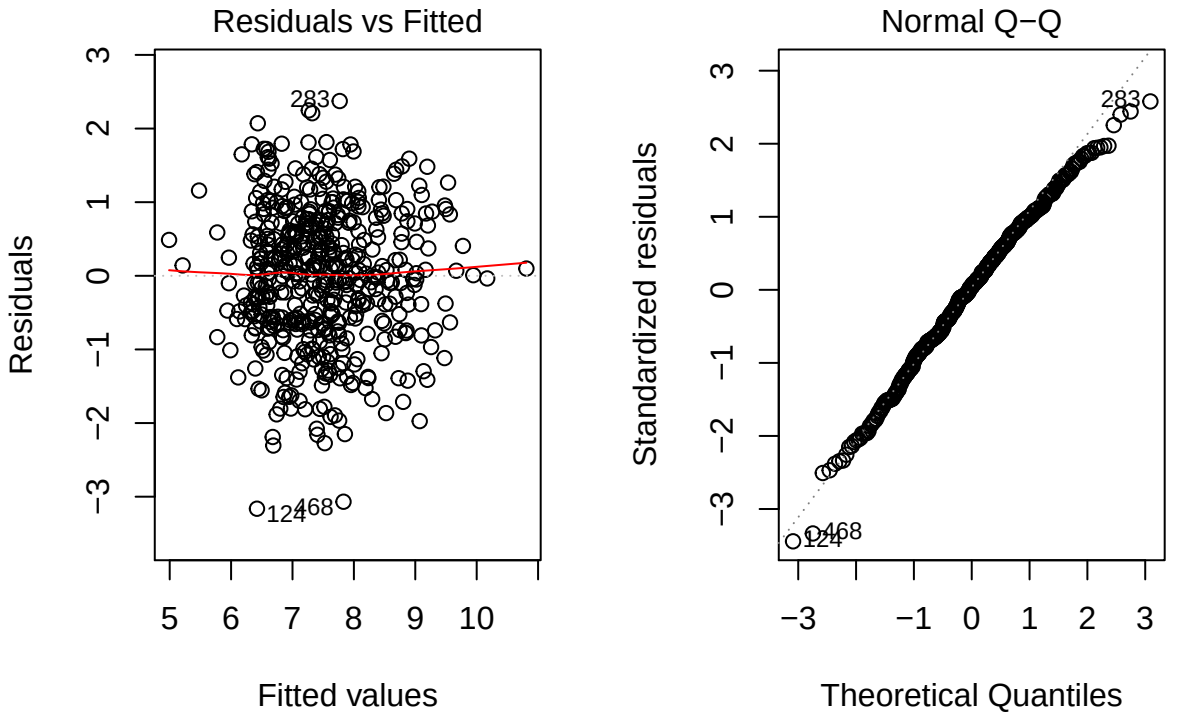
China dataset. Linear regression. Fitting a linear model to log-log

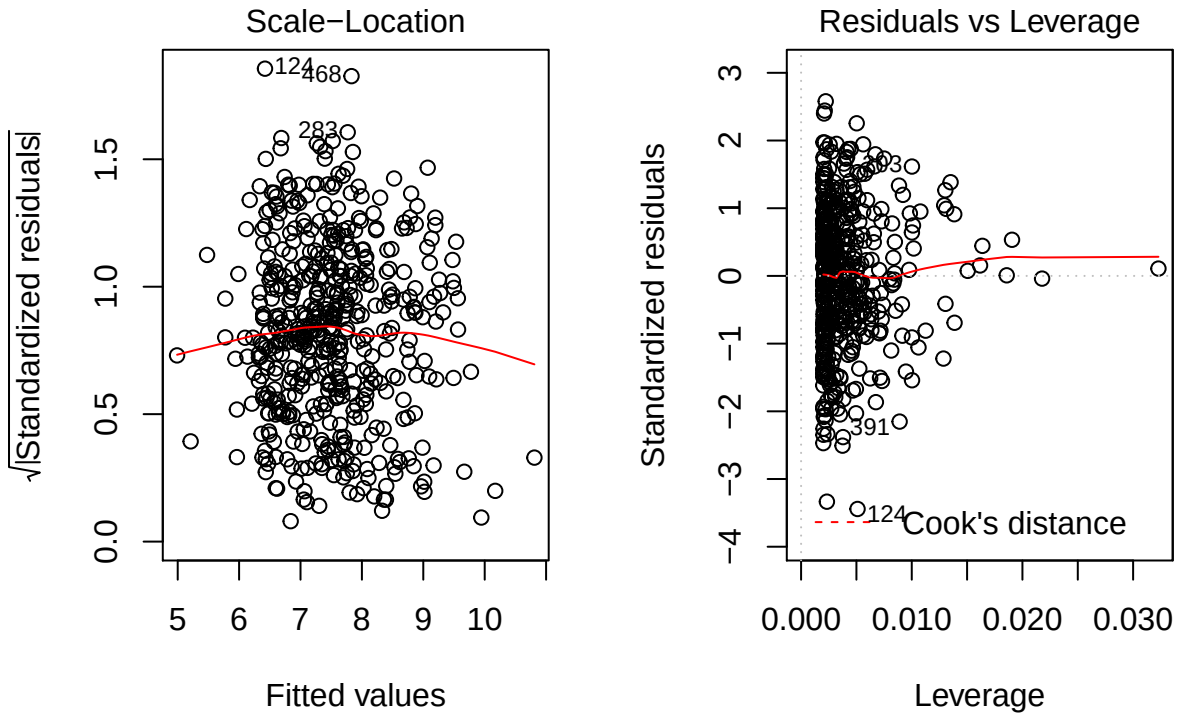
- the predictive power equation is $y = e^{b_0 + b_1 log(x)}$, ignoring the bias corrections
- First, we are fitting the model to the whole dataset. But it is not the right way to do it, because of overfitting

```r
linmodel <- lm(logchina_effort ~ logchina_size)
par(mfrow=c(1,1))
plot(logchina_size, logchina_effort)
abline(linmodel, lwd=3, col=3)
```




```
par(mfrow=c(1,2))
plot(linmodel, ask = FALSE)
```



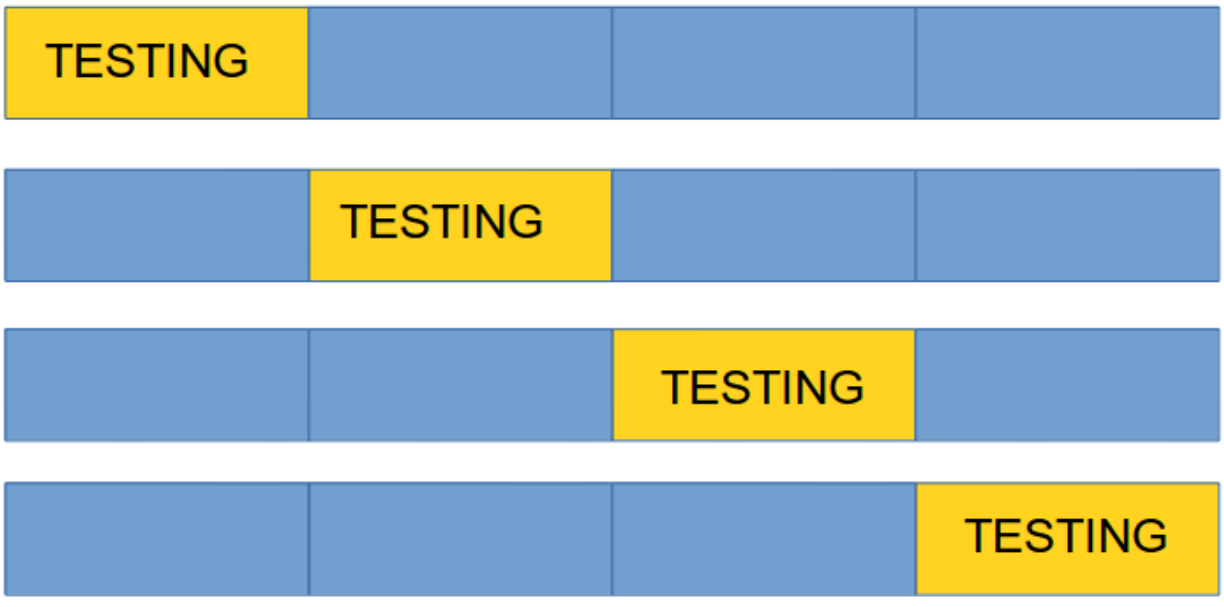


Building and Validating a Model

- Usually, part of the data points are used for building the model and the remaining points are used for validating the model. There are several approaches to this process.
- *Validation Set approach*: it is the simplest method. It consists of randomly dividing the available set of oservations into two parts, a *training set* and a *validation set* of hold-out set. Usually 2/3 of the data points are used for training and 1/3 is used for testing purposes.
- *Leave-One-Out Cross-Validation*: instead of creating two subsets for training and testing, a single observation is used for the validation set, and the remaining observations make up the training set. This approach is repeated n times (the total number of observations) and the estimate for the test mean squared error is the average of the n test estimates.

Leave one out

- *k-Fold Cross-Validation*: it involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, the the methods is fit on the remaining k-1 folds. This procedure is repeated k times. If k is equal to n we are in the previous method.

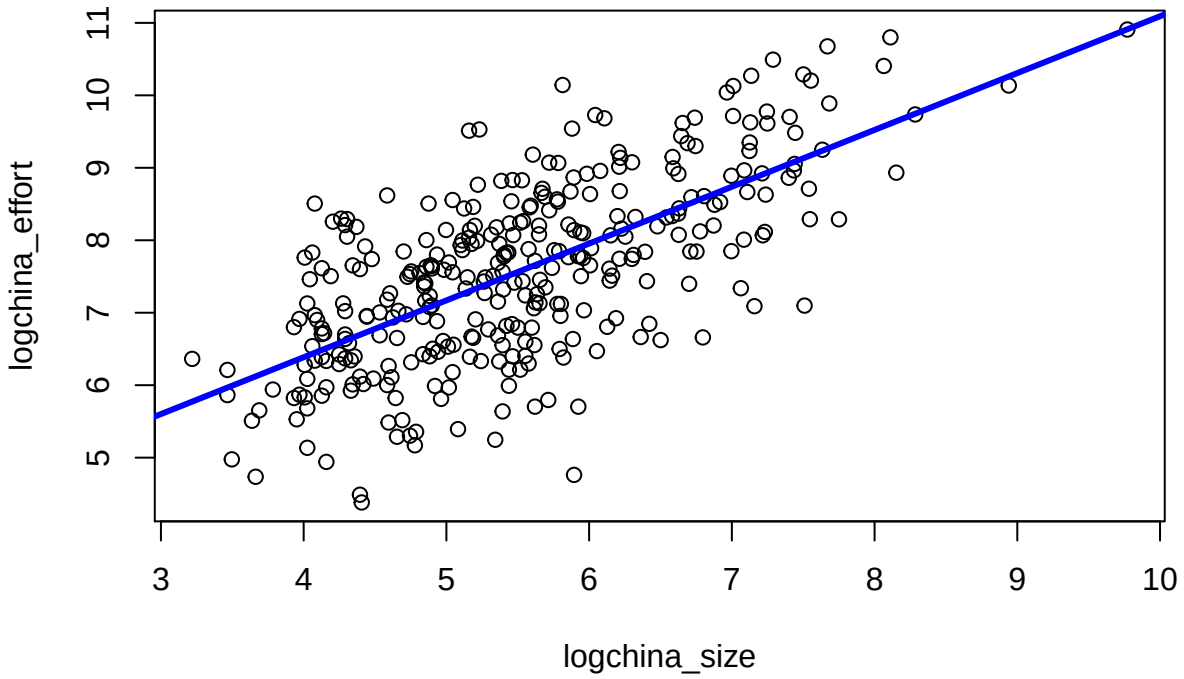


K-fold

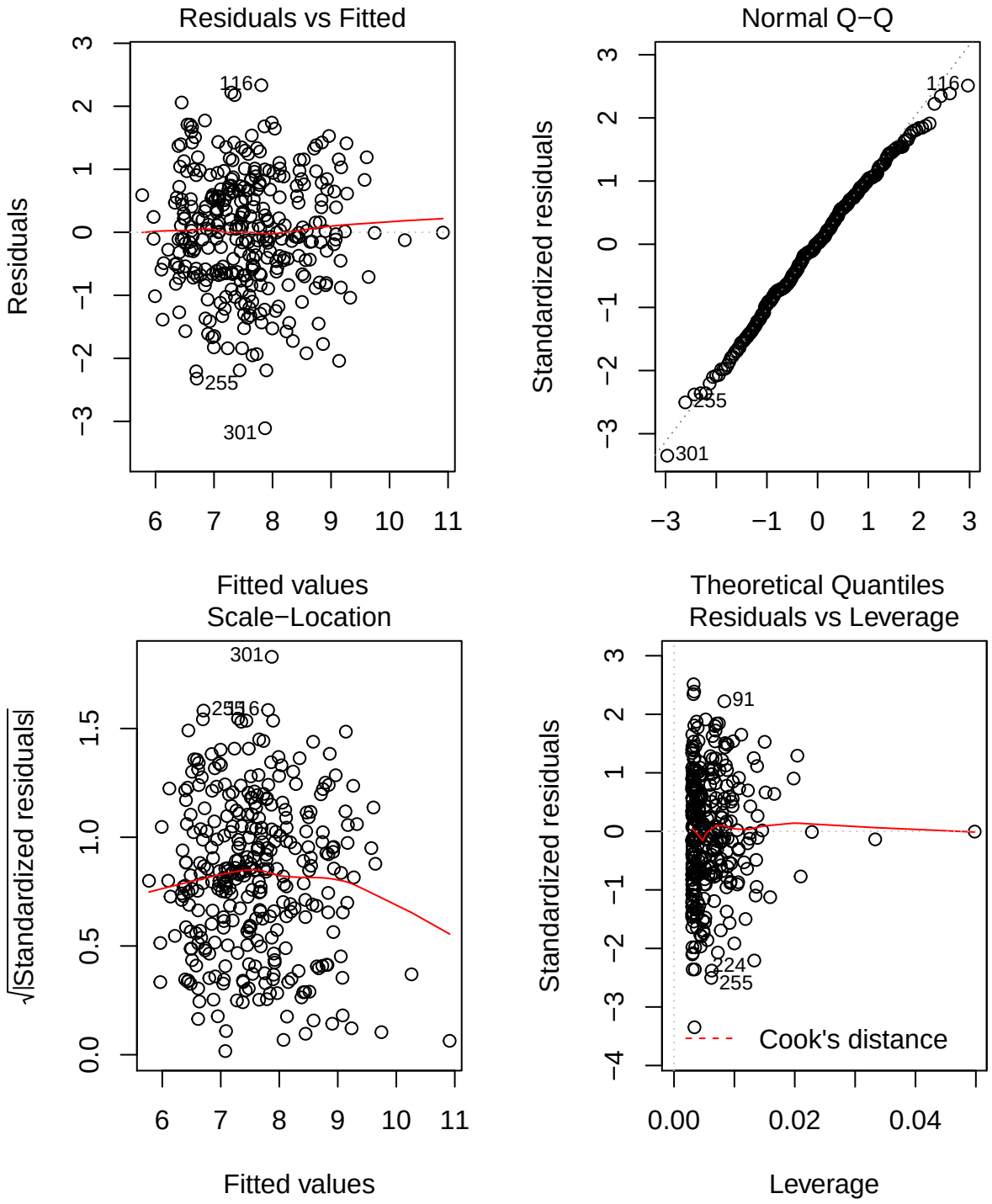
China dataset. Split data into Training and Testing

-

```
chinaTrain <- read.arff("china3AttSelectedAFPTrain.arff")
logchina_size <- log(chinaTrain$AFP)
logchina_effort <- log(chinaTrain$Effort)
linmodel <- lm(logchina_effort ~ logchina_size)
par(mfrow=c(1,1))
plot(logchina_size, logchina_effort)
abline(linmodel, lwd=3, col=4)
```




```
par(mfrow=c(1,2))
plot(linmodel, ask = FALSE)
```



Measures of Evaluation used in Software Engineering

- There are several measures usually used:
- *Mean Magnitude of the Relative Error (MMRE)*: this measure has been critisized many times.

$$\frac{\sum_{n=1}^{n} |\hat{y}_i|/y_i}{n}$$

- *Median Magnitude of the Relative Error (MdMRE)*: using the median insted of the mean

- *Level of Prediction (LPred(l))*: defined as the percentage of estimates that are within the level l% of the actual values. Usually the level of prediction is set at 25% below and above the actual value.
- *Standardised Accuracy (SA)*: this measure overcomes all the problems of the MMRE. It is defined as the MAR relative to random guessing:

$$SA = 1 - \frac{MAR}{\overline{MAR}_{P_0}} \times 100.$$

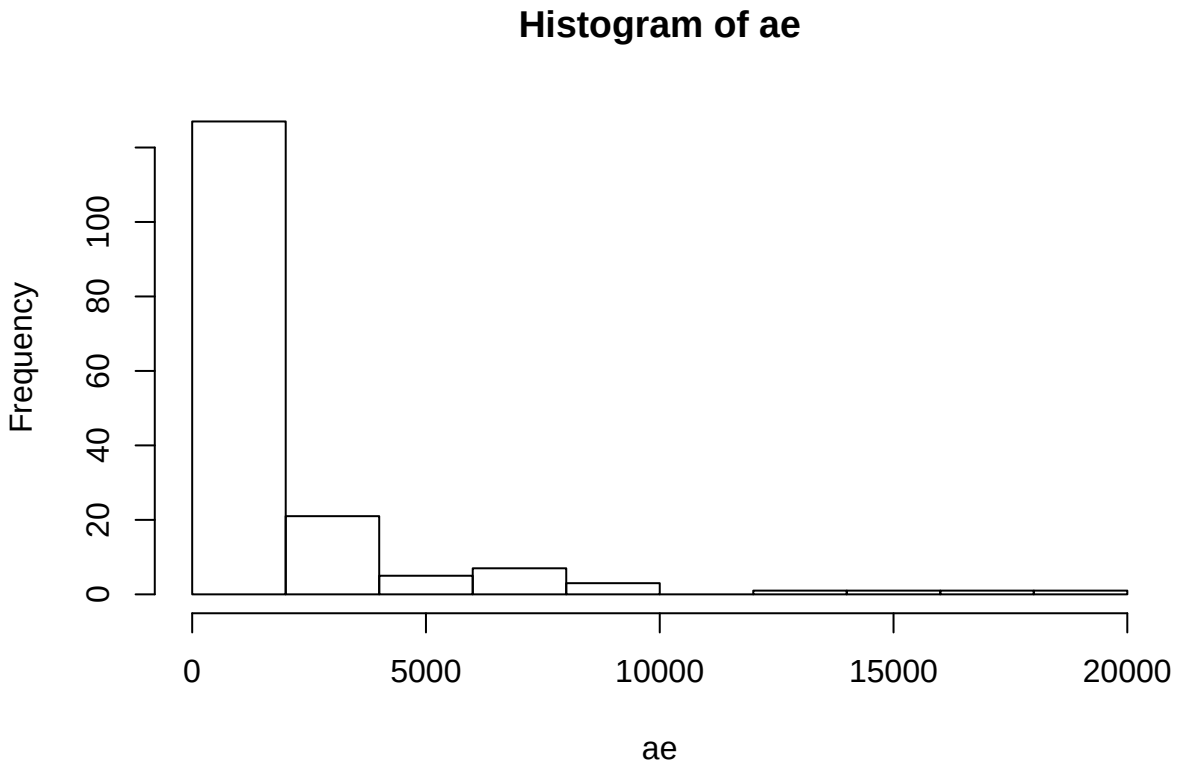
- "random guessing" is defined as: predict a $\hat{y}_t$ for the target case t by randomly sampling (with equal probability) over all the remaining n-1 cases and take $\hat{y}_t = y_r$ where r is drawn randomly from 1,n and $r \neq t$.

Evaluation of the model in the Testing data (2)

```
gm_mean = function(x, na.rm=TRUE){
  exp(sum(log(x[x > 0]), na.rm=na.rm) / length(x))}

chinaTest <- read.arff("china3AttSelectedAFPTest.arff")
b0 <- linmodel$coefficients[1]
b1 <- linmodel$coefficients[2]
china_size_test <- chinaTest$AFP
actualEffort <- chinaTest$Effort
predEffort <- exp(b0+b1*log(china_size_test))

err <- actualEffort - predEffort  #error or residual
ae <- abs(err)
hist(ae)
```



```r
mar <- mean(ae)
mre <- ae/actualEffort
mmre <- mean(mre)
mdmre <- median(mre)
gmar <- gm_mean(ae)
mar
```

```
## [1] 1866.819
```

```r
mmre
```

```
## [1] 1.15106
```

```r
mdmre
```

```
## [1] 0.5510077
```

```r
gmar
```

```
## [1] 832.5504
```

```r
level_pred <- 0.25 #below and above (both)
lowpred <- actualEffort*(1-level_pred)
uppred <-  actualEffort*(1+level_pred)
pred  <-  predEffort <= uppred & predEffort >= lowpred  #pred is a vector with logical values
Lpred <- sum(pred)/length(pred)
Lpred
```

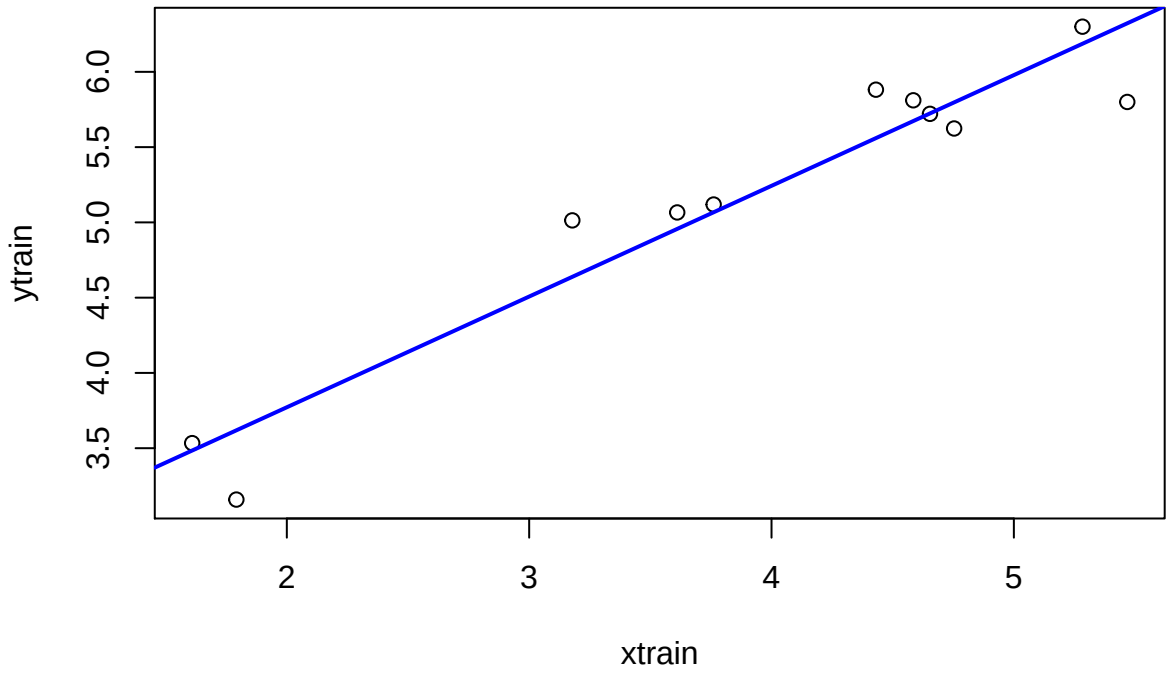
```
## [1] 0.1856287
```

Building a Linear Model on the Telecom1 dataset

- Although there are few datapoints we split the file into Train (2/3) and Test (1/3)

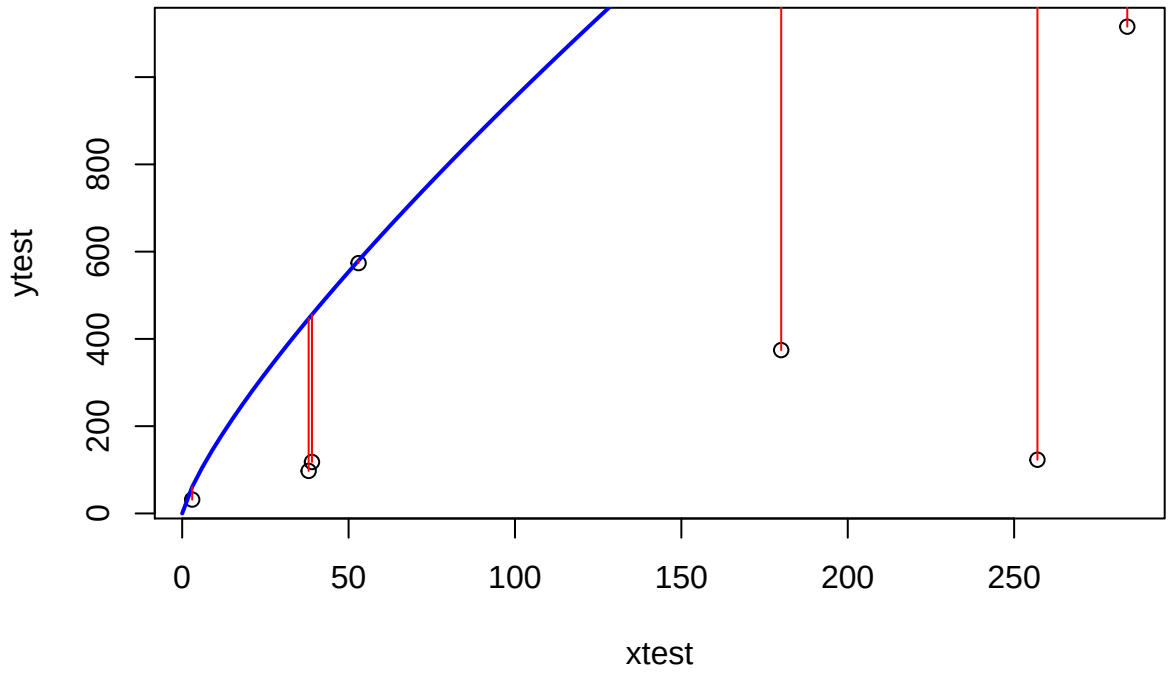
```r
samplesize <- floor(0.66*nrow(telecom1))
set.seed(012) # to make the partition reproducible
train_idx <- sample(seq_len(nrow(telecom1)), size = samplesize)
telecom1_train <- telecom1[train_idx, ]
telecom1_test <- telecom1[-train_idx, ]

par(mfrow=c(1,1))
# transformation of variables to log-log
xtrain <- log(telecom1_train$size)
ytrain <- log(telecom1_train$effort)
lmtelecom1 <- lm( ytrain ~ xtrain)
plot(xtrain, ytrain)
abline(lmtelecom1, lwd=2, col="blue")
```




```
b0_tel1 <- lmtelecom1$coefficients[1]
b1_tel1 <- lmtelecom1$coefficients[2]
# calculate residuals and predicted values
res <- signif(residuals(lmtelecom1), 5)

xtest <- telecom1_test$size
ytest <- telecom1_test$effort
pre_tel1 <- exp(b0+b1*log(xtest))
# plot distances between points and the regression line
plot(xtest, ytest)
curve(exp(b0+b1*log(x)), from=0, to=300, add=TRUE, col="blue", lwd=2)
segments(xtest, ytest, xtest, pre_tel1, col="red")
```



Building a Linear Model on the Telecom1 dataset with all observations

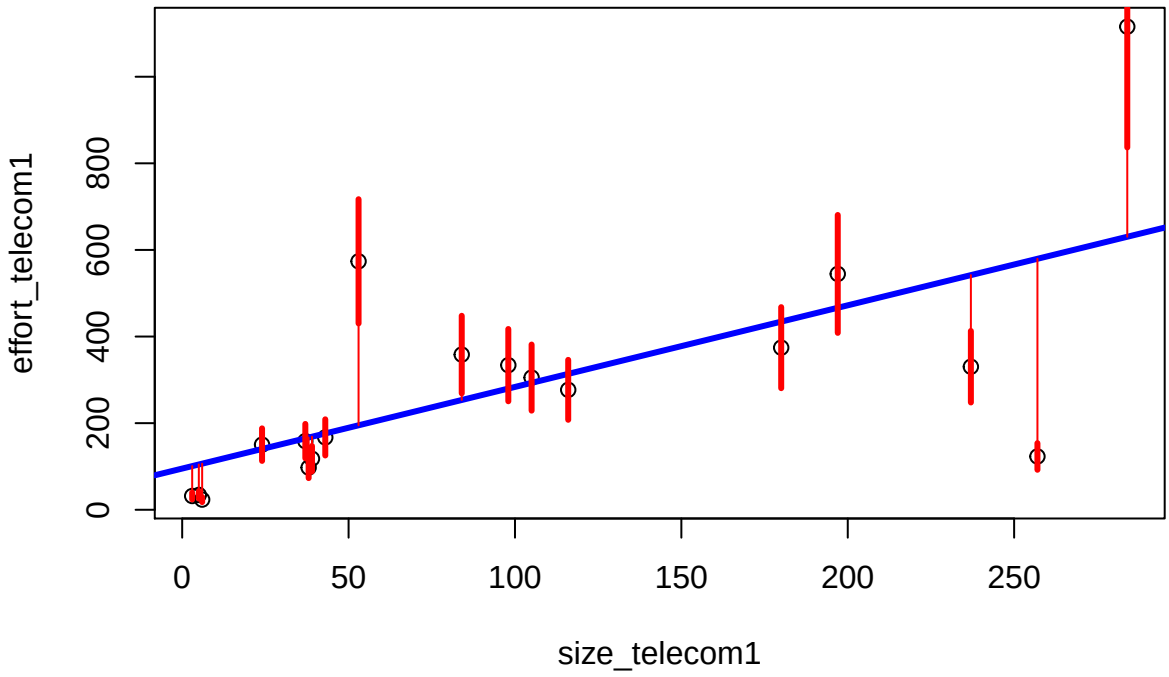
- Just to visualize results

```
par(mfrow=c(1,1))
lmtelecom <- lm(effort_telecom1 ~ size_telecom1)
plot(size_telecom1, effort_telecom1)
abline(lmtelecom, lwd=3, col="blue")
# calculate residuals and predicted values
res <- signif(residuals(lmtelecom), 5)
predicted <- predict(lmtelecom)
# plot distances between points and the regression line
segments(size_telecom1, effort_telecom1, size_telecom1, predicted, col="red")

level_pred <- 0.25 #below and above (both)
lowpred <- effort_telecom1*(1-level_pred)
uppred <-  effort_telecom1*(1+level_pred)
predict_inrange  <-  predicted <= uppred & predicted >= lowpred  #pred is a vector with logical values
Lpred <- sum(predict_inrange)/length(predict_inrange)
Lpred
```

```
## [1] 0.4444444
```

```
#Visually plot lpred
segments(size_telecom1, lowpred, size_telecom1, uppred, col="red", lwd=3)
```




```
err_telecom1 <- abs(effort_telecom1 - predicted)
mar_tel1 <- mean(err_telecom1)
mar_tel1
```

```
## [1] 124.7711
```

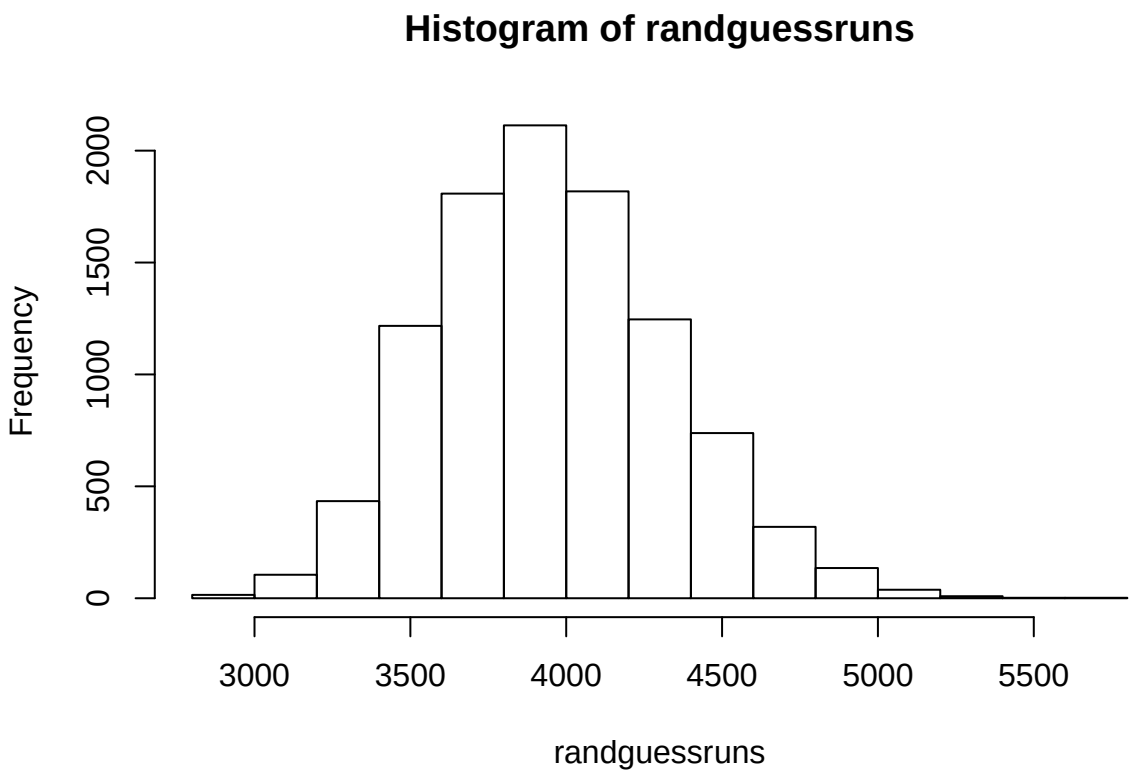
Standardised Accuracy. MARP0. ChinaTest

- Computing $MARP_0$ in the China Test data

```
estimEffChinaTest <- predEffort  # This will be overwritten, no problem
numruns <- 9999
randguessruns <- rep(0, numruns)
for (i in 1:numruns) {
  for (j in 1:length(estimEffChinaTest)) {
    estimEffChinaTest[j] <- sample(actualEffort[-j],1)}#replacement with random guessingt
  randguessruns[i] <- mean(abs(estimEffChinaTest-actualEffort))
  }
marp0Chinatest <- mean(randguessruns)
marp0Chinatest
```

```
## [1] 3955.782
```

```
hist(randguessruns)
```




```
saChina = (1- mar/marp0Chinatest)*100
saChina
```

```
## [1] 52.80784
```

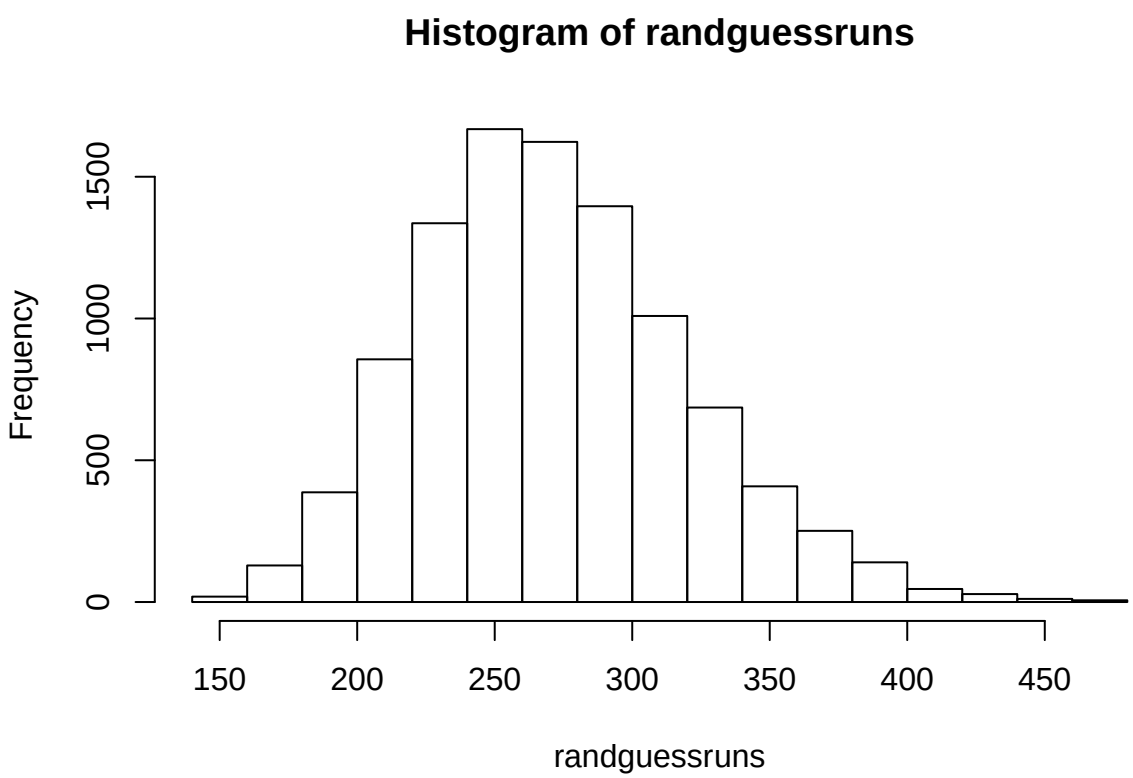
Standardised Accuracy. MARP0. Telecom1

- Computing $MARP_0$

```
path2files <- "~/DocProjects/PRESI2013/london2015"
setwd(path2files)
telecom1 <- read.table("Telecom1.csv", sep=",",header=TRUE, stringsAsFactors=FALSE, dec = ".") #read da
#par(mfrow=c(1,2))
#size <- telecom1[1]$size   not needed now
actualEffTelecom1 <- telecom1[2]$effort
estimEffTelecom1 <- telecom1[3]$EstTotal # this will be overwritten
numruns <- 9999
randguessruns <- rep(0, numruns)
for (i in 1:numruns) {
  for (j in 1:length(estimEffTelecom1)) {
    estimEffTelecom1[j] <- sample(actualEffTelecom1[-j],1)}#replacement with random guessingt
  randguessruns[i] <- mean(abs(estimEffTelecom1-actualEffTelecom1))
  }
marp0telecom1 <- mean(randguessruns)
marp0telecom1
```

```
## [1] 271.4162
```

```
hist(randguessruns)
```




```
saTelecom1 <- (1- mar_tel1/marp0telecom1)*100
saTelecom1
```

```
## [1] 54.02961
```

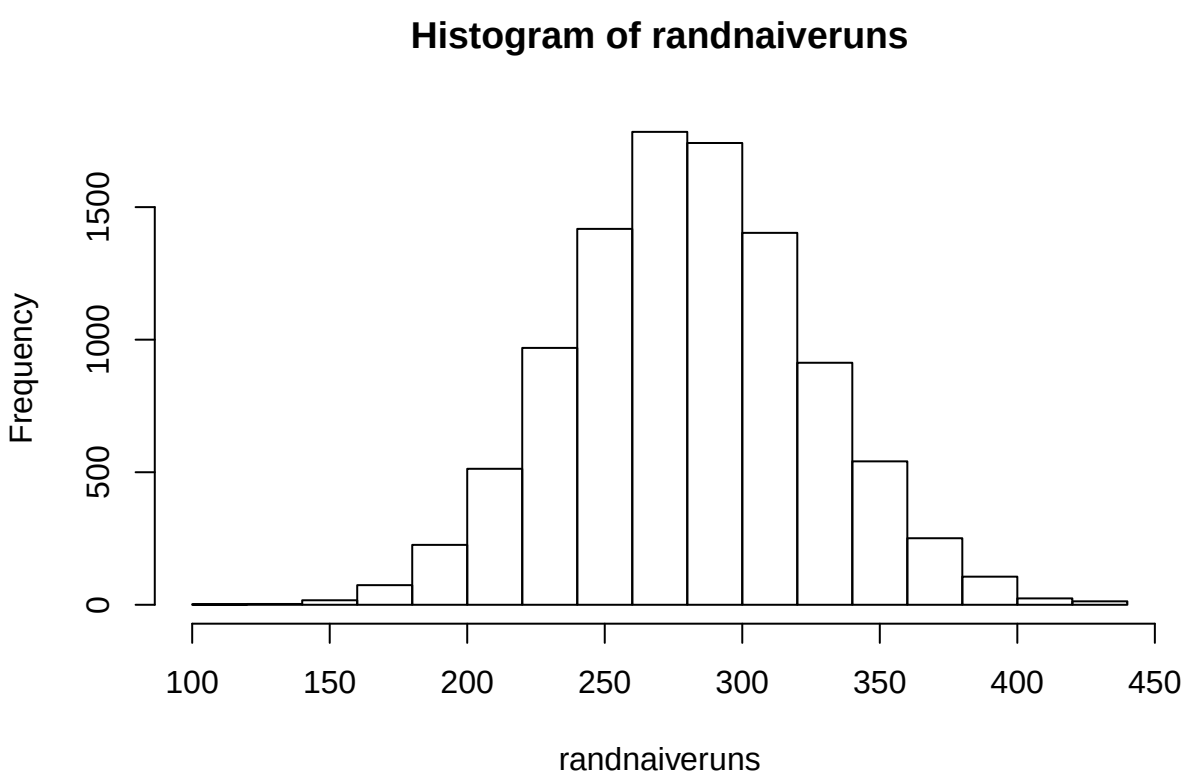
MARP0 in the Atkinson dataset

- For checking results you may use figure Atkinson in Shepperd&MacDonnell

```
act_effort <- c(670,912,218,595,267,344,229,190,869,109,289,616,557,416,578,438)
estim_effort <- rep(0, length(act_effort))
numruns <- 9999
randnaiveruns <- rep(0, numruns)
for (i in 1:numruns) {
  for (j in 1:length(act_effort)) {
    estim_effort[j] <- sample(act_effort[-j],1)}#replacement with random guessingt
  randnaiveruns[i] <- mean(abs(estim_effort-act_effort))
  }
marp0atkinson <- mean(randnaiveruns)
marp0atkinson
```

```
## [1] 280.644
```

```r
hist(randnaiveruns)
```



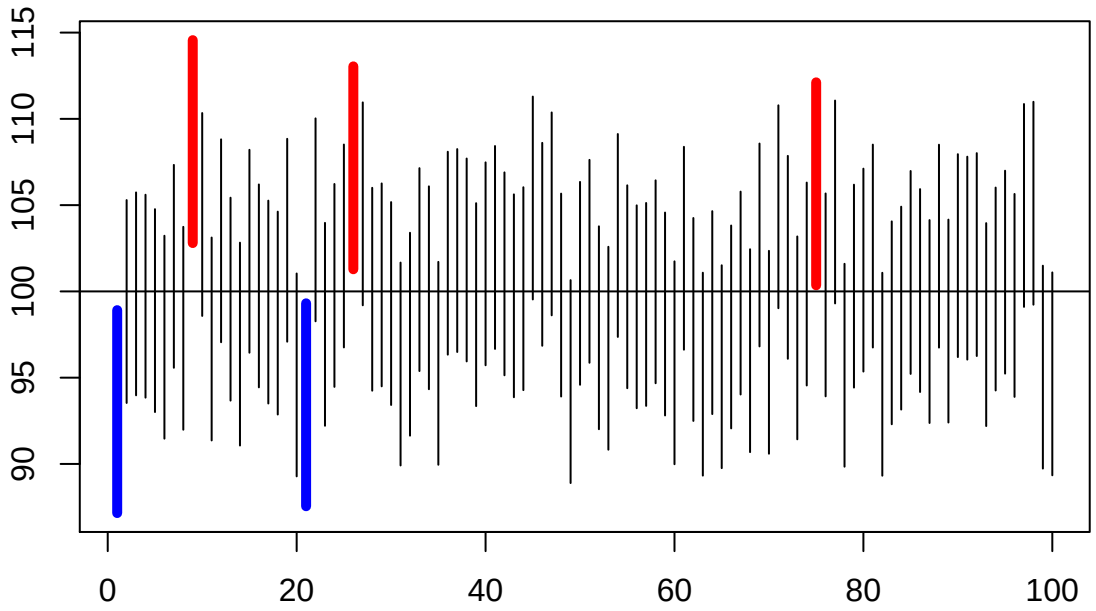
Confidence Intervals. Bootstrap

- A *confidence interval* (CI) is an interval estimate of a population parameter. The parameter can be the mean, the median or other. The frequentist CI is an observed interval that is different from sample to sample. It frequently includes the value of the unobservable parameter of interes if the experiment is repeated. The *confidence level* is the value that measures the frequency that the constructed intervals contain the true value of the parameter.
- The construction of a confidence interval with an exact value of confidence level for a distribution requires some statistical properties. Usually, *normality* is one of the properties required for computing confidence intervals.
  - Not all confidence intervals contain the true value of the parameter.
  - Simulation of confidence intervals

```r
# code from the book by Ugerte et al. Probability and statistics with R
  norsim <- function(sims = 100, n = 36, mu = 100, sigma = 18,
                     conf.level = 0.95){
  alpha <- 1 - conf.level
  CL <- conf.level * 100
  ll <- numeric(sims)
  ul <- numeric(sims)
  for (i in 1:sims){
    xbar <- mean(rnorm(n , mu, sigma))
    ll[i] <- xbar - qnorm(1 - alpha/2)*sigma/sqrt(n)
    ul[i] <- xbar + qnorm(1 - alpha/2)*sigma/sqrt(n)
  }
```

```
  notin <- sum((ll > mu) + (ul < mu))
  percentage <- round((notin/sims) * 100, 2)
  SCL <- 100 - percentage
  plot(ll, type = "n", ylim = c(min(ll), max(ul)), xlab = " ",
       ylab = " ")
  for (i in 1:sims) {
    low <- ll[i]
    high <- ul[i]
    if (low < mu & high > mu) {
      segments(i, low, i, high)
    }
    else if (low > mu & high > mu) {
      segments(i, low, i, high, col = "red", lwd = 5)
    }
    else {
      segments(i, low, i, high, col = "blue", lwd = 5)
    }
  }
  abline(h = mu)
#   cat(SCL, "\b% of the random confidence intervals contain Mu =", mu, "\b.", "\n")
}
```

```
set.seed(10)
norsim(sims = 100, n = 36, mu = 100, sigma = 18, conf.level = 0.95)
```

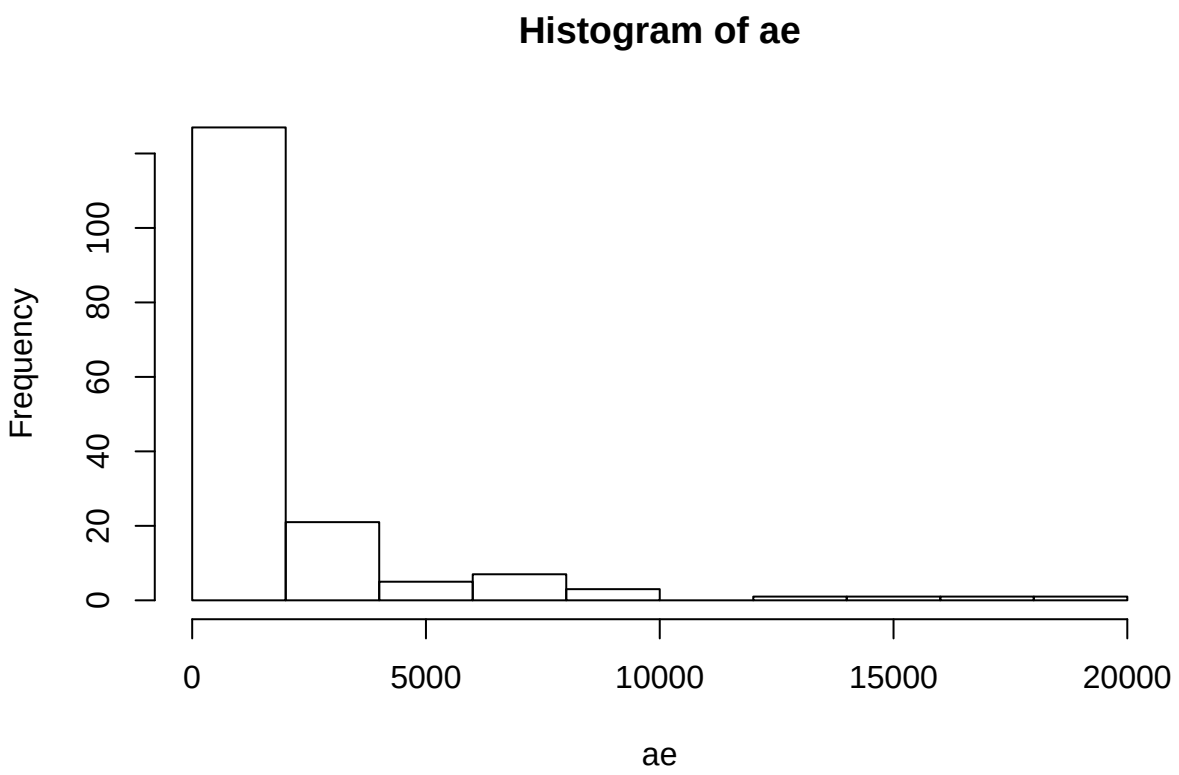


Nonparametric Bootstrap

- For computing CIs the important thing is to know the assumptions that are made to "know" the distribution of the statistic.
- There is a way to compute confidence intervals without meeting the requirements of parametric methods.

- **Resampling** or *bootstraping* is a method

```
library(boot)
hist(ae)
```




```
level_confidence <- 0.95
repetitionsboot <- 9999
samplemean <- function(x, d){return(mean(x[d]))}
b_mean <- boot(ae, samplemean, R=repetitionsboot)
confint_mean <- boot.ci(b_mean)
```

```
## Warning in boot.ci(b_mean): bootstrap variances needed for studentized
## intervals
```

```
confint_mean
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 9999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b_mean)
##
## Intervals :
## Level      Normal              Basic
## 95%   (1415, 2309 )   (1389, 2282 )
##
## Level     Percentile            BCa
## 95%   (1451, 2345 )   (1488, 2411 )
## Calculations and Intervals on Original Scale
```

```r
boot_geom_mean <- function(error_vec){
  log_error <- log(error_vec[error_vec > 0])
  log_error <-log_error[is.finite(log_error)] #remove the -Inf value before calculating the mean, just
  samplemean <- function(x, d){return(mean(x[d]))}
  b <- boot(log_error, samplemean, R=repetitionsboot) # with package boot
  # this is a boot for the logs
  return(b)
}
# BCAconfidence interval for the geometric mean
BCAciboot4geommean <- function(b){
  conf_int <- boot.ci(b, conf=level_confidence, type="bca")$bca #following 10.9 of Ugarte et al.'s book
  conf_int[5] <- exp(conf_int[5]) # the boot was computed with log. Now take the measure back to its pr
  conf_int[4] <- exp(conf_int[4])
  return (conf_int)
}
# this is a boot object
b_gm <- boot_geom_mean(ae)
exp(b_gm$t0)
```

```
## [1] 832.5504
```

```r
b_ci_gm <- BCAciboot4geommean(b_gm)
b_ci_gm
```

```
##      conf
## [1,] 0.95 227.64 9726.31 675.783 1012.201
```

```r
# Make a % confidence interval bca
# BCAciboot <- function(b){
#   conf_int <- boot.ci(b, conf=level_confidence, type="bca")$bca #following 10.9 of Ugarte et al.'s bo
#   return (conf_int)
# }
```

Genetic Programming for Symbolic Regression