# R Basics

*Francisco Guzmán*

*May 12, 2016*

## 1 R Basics

R is a *programming language.*

Let's review some of its basic characteristics

- Basic operations
- Common functions
- Data types

### 1.1 Basic operations: Assignment

One of the main differences is the operator "<-"

```r
#Let's create variables of different types
#numeric
my_num1<- 15
print (my_num1)
```

```
>>> [1] 15
```

```r
#character
my_string<- "Hello world"
print (my_string)
```

```
>>> [1] "Hello world"
```

```r
#logical
my_bool<- TRUE
print (my_bool)
```

```
>>> [1] TRUE
```

### 1.2 Basic operations: Arithmetics

```r
# We can do operations with the variables
my_num1<-15
my_num2<-5
#substraction
my_num1-my_num2
```

```
>>> [1] 10
```

```
#addition
my_num1+my_num2
```

```
>>> [1] 20
```

```
#multiplicatoin
my_num1*my_num2
```

```
>>> [1] 75
```

```
#division
my_num1/my_num2
```

```
>>> [1] 3
```

```
# power
my_num1^2
```

```
>>> [1] 225
```

```
#square root
sqrt(my_num1 *my_num2)
```

```
>>> [1] 8.660254
```

## 1.3 Data types

```
# Let's review the internal representation of the variables

# Numeric: default number (float)
class(my_num1)
```

```
>>> [1] "numeric"
```

```
#Character: strings are characters
class(my_string)
```

```
>>> [1] "character"
```

```
#Logicals: booleans
class(my_bool)
```

```
>>> [1] "logical"
```

## 1.4 Vectors

```r
#You can create a list of variables with the combine **c()** command
ranks<-c(1,2,3,4)
print(ranks)
```

```
>>> [1] 1 2 3 4
```

```r
# Vectors preserve the class of its elements
class(ranks)
```

```
>>> [1] "numeric"
```

```r
#You can access the specific elements **INDEX 1**
print(ranks[1] )
```

```
>>> [1] 1
```

```r
# You can perform element-wise ops in vectors
weights<-c(78.0,66.0,90.0,55.5)
heights <-c(181.0,160.0,190.0,170.0)

#e.g. division
ratios<-weights/heights

# You can have vectors of characters
names<-c("John","Mary","Bob","Anna")
affiliations<-c("QF","QP","QP","QF")
# Or logicals
is_fit<-c(TRUE,FALSE,TRUE,FALSE)
```

## 1.5 Common functions

```r
# **str**: Structure: tells you about the variable(s) class, size and values
str(my_num1)
```

```
>>>  num 15
```

```r
str(my_string)
```

```
>>>  chr "Hello world"
```

```r
str(weights)
```

```
>>>  num [1:4] 78 66 90 55.5
```

## 1.6 Summary

```
# **summary**: Gives you a summary of a variable.
# Specially useful for lists/vectors

summary(weights)
```

```
>>>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
>>>   55.50   63.38   72.00   72.38   81.00   90.00
```

## 1.7 Factors

Factors are a useful data type in R.

They represent categorical variables, with a numerical correspondence (level), and a string representation (label)

```
#Let's factorize the affiliation vector
f_affiliations<-factor(affiliations)
print(f_affiliations)
```

```
>>> [1] QF QP QP QF
>>> Levels: QF QP
```

```
#Now let's compare
str(f_affiliations)
```

```
>>>  Factor w/ 2 levels "QF","QP": 1 2 2 1
```

```
str(affiliations)
```

```
>>>  chr [1:4] "QF" "QP" "QP" "QF"
```

```
#Even summaries are different
summary(f_affiliations)
```

```
>>> QF QP 
>>>  2  2
```

```
summary(affiliations)
```

```
>>>    Length     Class      Mode 
>>>         4 character character
```

## 1.8 Data Frames (DF)

They are collections of variables (columns) with different measurements(rows)

Let’s create a list of people!

```
my_df <- data.frame(name=names,affiliation=affiliations,weight=weights,height=heights,is_fit=is_fit)
# The command head shows us a preview of the dataset
head(my_df)
```

```
>>>   name affiliation weight height is_fit
>>> 1 John          QF   78.0    181   TRUE
>>> 2 Mary          QP   66.0    160  FALSE
>>> 3  Bob          QP   90.0    190   TRUE
>>> 4 Anna          QF   55.5    170  FALSE
```

```
# We can see its structure
str(my_df)
```

```
>>> 'data.frame':   4 obs. of  5 variables:
>>>  $ name       : Factor w/ 4 levels "Anna","Bob","John",..: 3 4 2 1
>>>  $ affiliation: Factor w/ 2 levels "QF","QP": 1 2 2 1
>>>  $ weight     : num  78 66 90 55.5
>>>  $ height     : num  181 160 190 170
>>>  $ is_fit     : logi  TRUE FALSE TRUE FALSE
```

```
# Or a complete summary
summary(my_df)
```

```
>>>    name   affiliation     weight          height         is_fit
>>>  Anna:1   QF:2        Min.   :55.50   Min.   :160.0   Mode :logical
>>>  Bob :1   QP:2        1st Qu.:63.38   1st Qu.:167.5   FALSE:2
>>>  John:1               Median :72.00   Median :175.5   TRUE :2
>>>  Mary:1               Mean   :72.38   Mean   :175.2   NA's :0
>>>                       3rd Qu.:81.00   3rd Qu.:183.2
>>>                       Max.   :90.00   Max.   :190.0
```

## 1.9 Variables in a DF

```
#print the column name
my_df$name
```

```
>>> [1] John Mary Bob  Anna
>>> Levels: Anna Bob John Mary
```

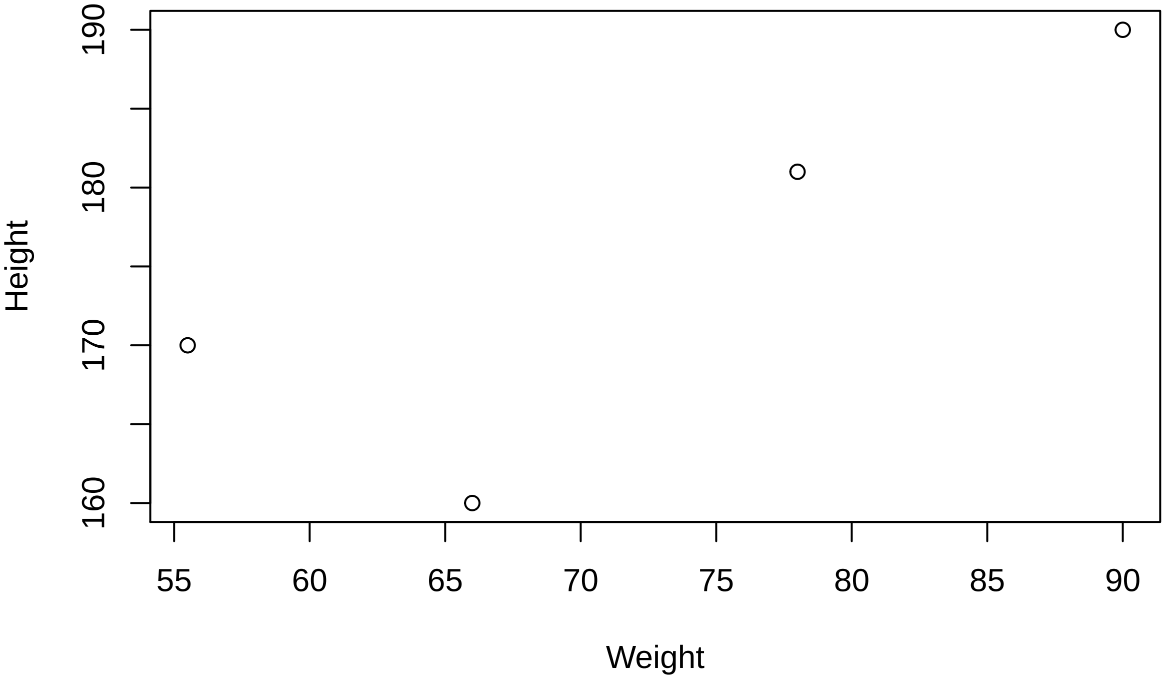
```
my_df$weight
```

```
>>> [1] 78.0 66.0 90.0 55.5
```

## 1.10 Basic plots

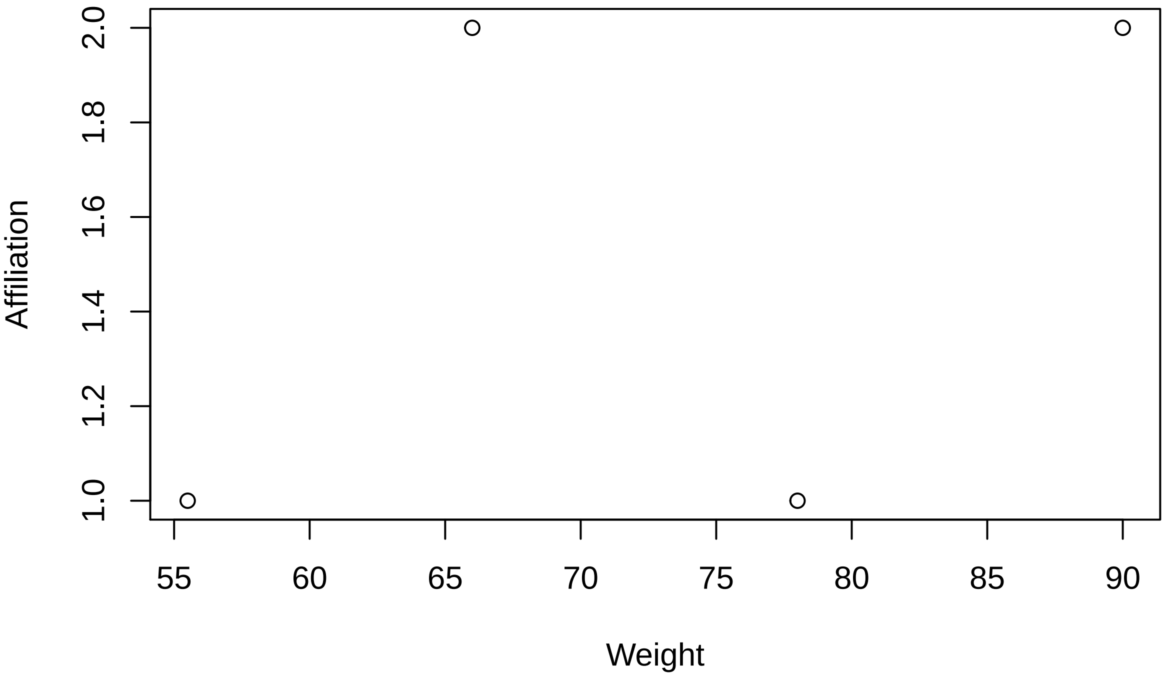
- Two numerics

```
plot(my_df$weight,my_df$height,xlab="Weight",ylab="Height")
```

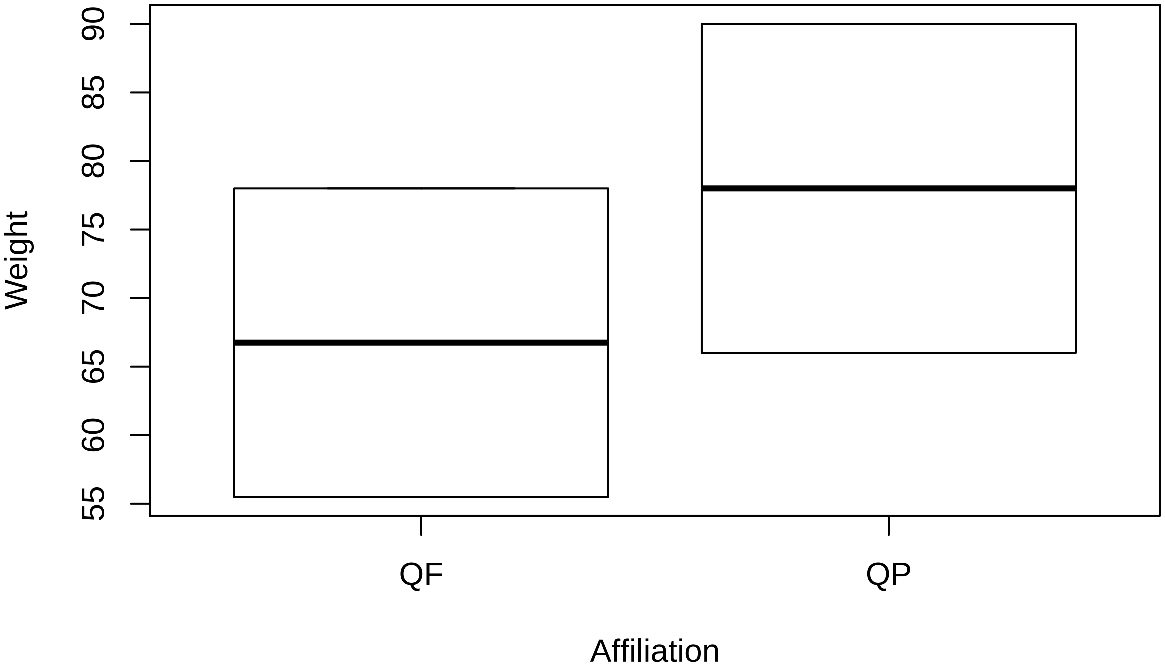


- One numeric, one factor

```
#This is the wrong way because affiliation is taken as a numeric
plot(my_df$weight,my_df$affiliation,xlab="Weight", ylab="Affiliation")
```



```
# Instead, we use the "by" operator "~".
#Note that this operator works as y~x. So now the axes are reversed
plot(my_df$weight~my_df$affiliation,ylab="Weight",xlab="Affiliation")
```



## 2 Exercise

### 2.1 Add more people to your data frame

Add “Salma” (weight=61, height=166, is_fit=TRUE, affiliation=QSTP ) and Add “Khaled” (weight=75, height=175, is_fit=FALSE, affiliation=QSTP )

to the data frame Hint: you can use *c()* to grow each of your vectors. E.g. `weights<-c(weights,61,75)`

then reconstruct the data frame

### 2.2 Plot height vs. weight

Re-plot height vs weight with the new data