Summarizing Data

This document describes how to find different descriptive statistics like mean, median, standard deviation and much more in R. We also show how to make different kinds of plots for quantitative and categorical variables.

1 Five Number Summary

In this part we work with the life expectancy data which shows for different countries the average life expectancy. First we need to input the text data file into R. After specifying your working directory [1], we can download the table to R using 'read.table' function. This data file has headers in the first line which we must indicate using 'header=TRUE' in the 'read.table' function: (blue print is the user's commands, while red one is the R output)

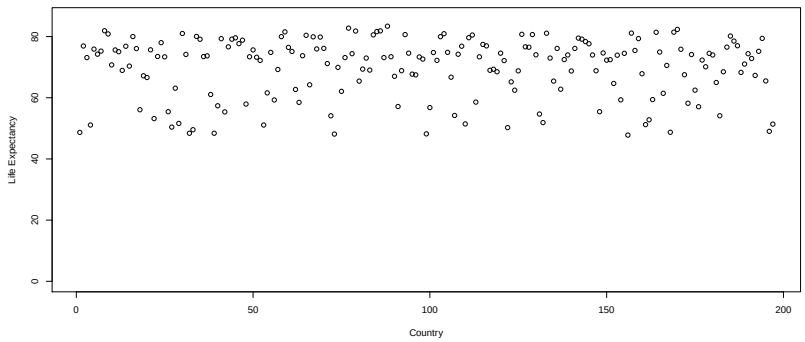
```
Life.Exp.data = read.table('LifeExpComplete.txt',header=TRUE)
```

Now 'Life.Exp.data' variable contains the whole table, to analyse each variable ('Region', 'LifeExp','GDP','HIV') individually we can separate the table into variables with the names of the variables correspond to the header names:

```
attach(Life.Exp.data)
```

Now for example the variable 'LifeExp' contains 197 observations of life expectancies.
To make a very simple plot of this variable use the 'plot' function:

```
plot(LifeExp, xlab='Country', ylab='Life Expectancy',ylim=range(0:86))
```

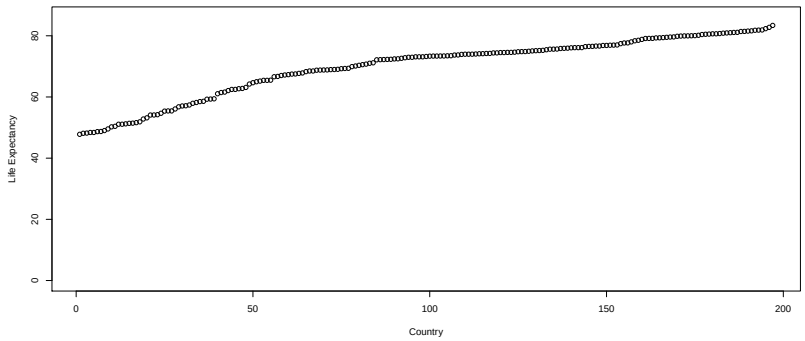


Note: 'xlab' and 'ylab' correspond to the titles of x and y axis, 'ylim' specifies the range for the y axis (in our case it is from 0 to 86) similarly you can use 'xlim' for the x axis.

To make the plot less scattered we can first sort the data from smallest to largest value and then plot the graph (here 'sort' function just sorts the data from smallest to largest):

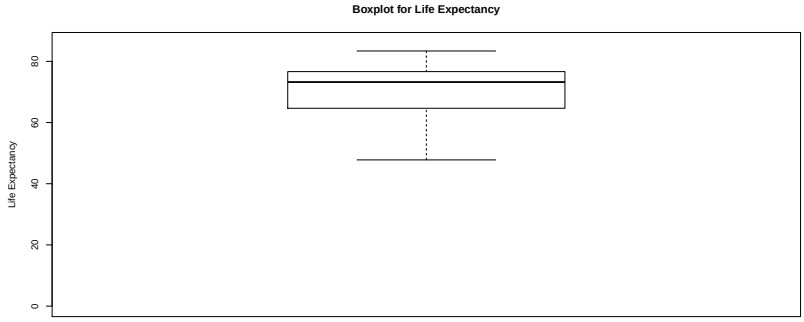
```
plot(sort(LifeExp), xlab='Country', ylab='Life Expectancy',ylim=range(0:86))
```

[1] See 'Introduction to basic R' paper



These were some very simple plots. Boxplots however produce more informative plots. From boxplot we can estimate the maximum, minimum, median, first quantile and third quantile. Use the following command ('range=0' indicates that it is a simple boxplot, 'main' argument corresponds to the title of the plot):

```
boxplot(LifeExp, ylab='Life Expectancy',ylim=range(0:86),range=0,
 main='Boxplot for Life Expectancy')
```



If you want to find exact quantities for these statistics, you can access them using 'summary' function:

```
summary(LifeExp)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  47.79   64.67   73.24   69.86   76.65   83.39
```

From left to right it is: minimum, first quantile, median, mean, third quantile and maximum. You can also find these quantities separately using the next commands:

```
min(LifeExp)
```

```
[1] 47.794
```

```
quantile(LifeExp,0.25)
```

```
   25%
64.666
```

```
median(LifeExp)
```

```
[1] 73.235
```

```
mean(LifeExp)
```

```
[1] 69.86282
```

```
quantile(LifeExp,0.75)
```

```
   75%
76.652
```

```
max(LifeExp)
```

```
[1] 83.394
```

2 Center of Data

In this section we are going to analyse the Skeleton data. First we input the data file into R as usual.

```
Skeleton.data = read.table('SkeletonDataComplete.txt', header=TRUE)
```

These data have 400 observations; to see how the data are organized we can print first several observations using 'head' function:

```
head(Skeleton.data)
```

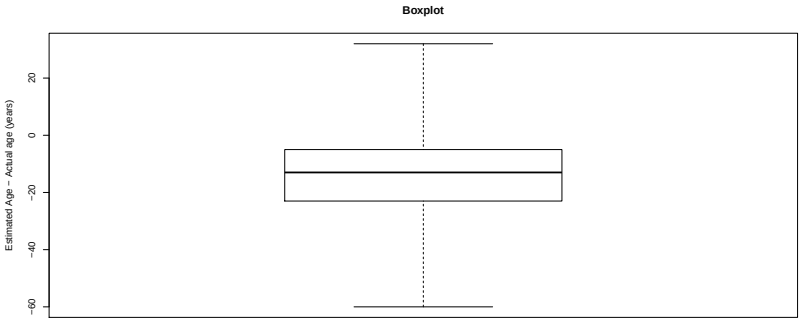
```
  Sex      BMIcat BMIquant Age DGestimate DGerror SBestimate SBerror
1   2 underweight    15.66  78         44     -34         60     -18
2   1      normal    23.03  44         32     -12         35      -9
3   1  overweight    27.92  72         32     -40         61     -11
4   1  overweight    27.83  59         44     -15         61       2
5   1      normal    21.41  60         32     -28         46     -14
6   1 underweight    13.65  34         25      -9         35       1
```

To analyse each variable individually we separate the table using 'attach':

```
attach(Skeleton.data)
```

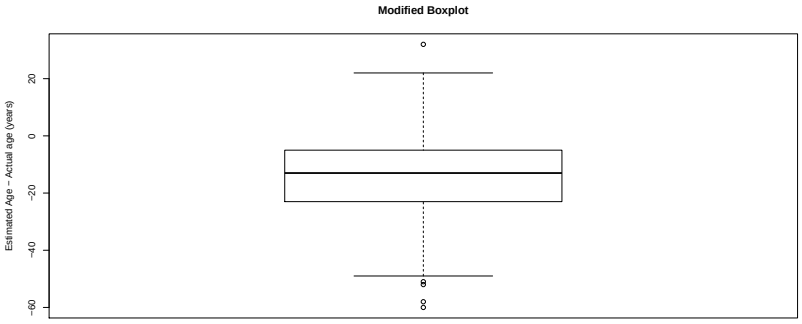
Next we want to plot the boxplot of the 'DGerror' variable (which is the difference between DGestimate and the actual age):

```
boxplot(DGerror, range=0, ylab='Estimated Age - Actual age
 (years)',main='Boxplot')
```



To make a modified boxplot (that shows observations beyond inner fences) just omit the 'range=0' argument:

```
boxplot(DGerror, ylab='Estimated Age - Actual age (years)',
 main='Modified Boxplot')
```



To find some basic statistics for these data, use 'summary' as before:

```
summary(DGerror)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -60.00  -23.00  -13.00  -14.15   -5.00   32.00
```

If we think that outliers may affect the mean we can "trim" the data by deleting largest x% and smallest x% values and then find the mean. R can do it very easily, for example to trim 10%, type:

```
mean(DGerror, trim=0.10)
```

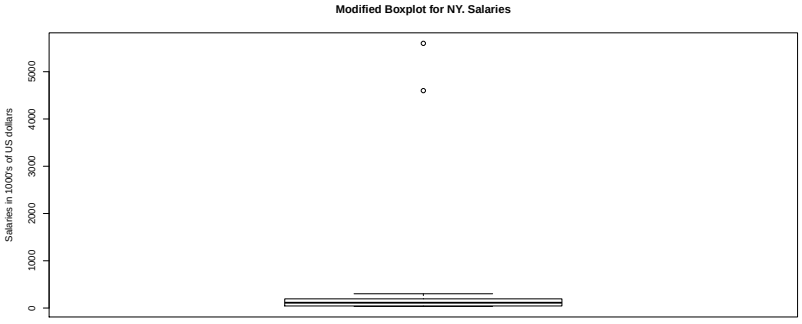
```
[1] -13.82188
```

Next we need to download another data set which consists of salaries for NY Red Bulls players. Since it is not a table but only one column of numbers we use 'scan' function instead of 'read.table':

```
ny.salaries = scan('NYRedBullsSalaries.txt')
```

To make a good visual representation of this data we use boxplot: (since we want to measure in thousands of dollars, we divide the 'ny.salaries' by 1000)

```
boxplot(ny.salaries/1000, ylab="Salaries in 1000's of US dollars",
 main='Modified Boxplot for NY. Salaries')
```



We immediately see two outliers that probably will affect the mean of this sample. So lets first find mean and median of the salaries:

```
median(ny.salaries)
```

```
[1] 112495.5
```

```
mean(ny.salaries)
```

```
[1] 518311.6
```

As expected there is a big difference between mean and median. Now lets remove 8% largest and 8% lowest salaries and find the mean of the 'trimmed' data:

```
mean(ny.salaries, trim=0.08)
```

```
[1] 128109.1
```

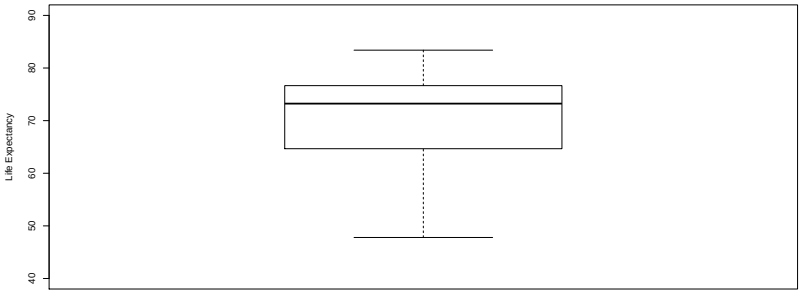
See how the mean changes when we trimmed the data.

3 Spread of the Data

We start with 'Life Expectancy' data set which we have already downloaded into R. Remember we have created the variable 'LifeExp' which contains just life expectancies values. Lets plot the boxplot and get some summary of these data again:

```
boxplot(LifeExp, ylab='Life Expectancy', ylim=c(40,90))
summary(LifeExp)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  47.79   64.67   73.24   69.86   76.65   83.39
```



We have several measures of the spread of quantitative data: range, IQR and standard deviation. We find all this statistics in R using the following commands:
Range:

```
range=max(LifeExp) - min(LifeExp);range
```

```
[1] 35.6
```

Interquartile Range:

```
IQR(LifeExp)
```

```
[1] 11.986
```

Standard Deviation:

```
sd(LifeExp)
```

```
[1] 9.668736
```

Square Root of Variance:

```
sqrt(var(LifeExp))
```

```
[1] 9.668736
```

Note that standard deviation is exactly the same as square root of the variance. The next part is a little more complicated compared to other subjects and therefore can be considered as optional.
We want to compare different measures of data (statistics) for robustness using NY Red Bull salaries. We already have the variable 'ny.salaries' which we will analyse. First lets find median, mean, range, IQR, standard deviation for original data and store all this information into variable 'stat':

```
median.sal = median(ny.salaries)
mean.sal = mean(ny.salaries)
range.sal = max(ny.salaries) - min(ny.salaries)
iqr.sal = IQR(ny.salaries)
sd.sal = sd(ny.salaries)
stat = c(median.sal, mean.sal, range.sal, iqr.sal, sd.sal)
```

Secondly we construct a new variable of salaries but without 2 largest and 2 smallest values and call this variable 'ny.salaries.trim': (note that since the sample size is 25 and we want to delete 2 largest and 2 smallest values, we can do that by first sorting the data and then store values on positions 3,4,5,,23 which is done by '[3:23]')

```
ny.salaries.trim = sort(ny.salaries)
ny.salaries.trim = ny.salaries.trim[3:23]
```

Next we find all the needed measures but for the trimmed data and store them in variable 'stat.trim':

```
median.sal.trim = median(ny.salaries.trim)
mean.sal.trim = mean(ny.salaries.trim)
range.sal.trim = max(ny.salaries.trim) - min(ny.salaries.trim)
iqr.sal.trim = IQR(ny.salaries.trim)
sd.sal.trim = sd(ny.salaries.trim)
stat.trim = c(median.sal.trim, mean.sal.trim, range.sal.trim, iqr.sal.trim, sd.sal.trim)
```

Then we put two variables side by side for comparison ('cbind' function binds two variables together):

```
comparison = cbind(stat, stat.trim)
rownames(comparison) = c('Median', 'Mean', 'Range', 'IQR', 'SD')
comparison
```

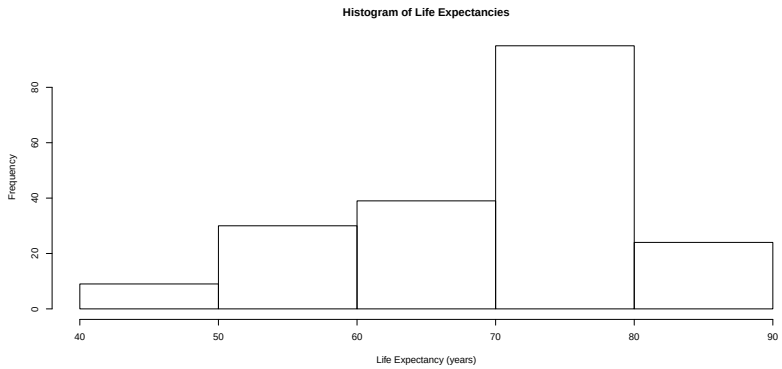
```
            stat stat.trim
Median  112495.5 112495.50
Mean    518311.6 128109.09
Range  5566250.0 268249.00
IQR     150375.0 145999.96
SD     1388822.1  83990.81
```

Based on this comparison we can conclude that median and IQR almost did not change after trimming the data and therefore are robust to outliers, on the other hand mean, range and standard deviation changed a lot by trimming and hence are not robust to outliers.

4 Shape of the Data

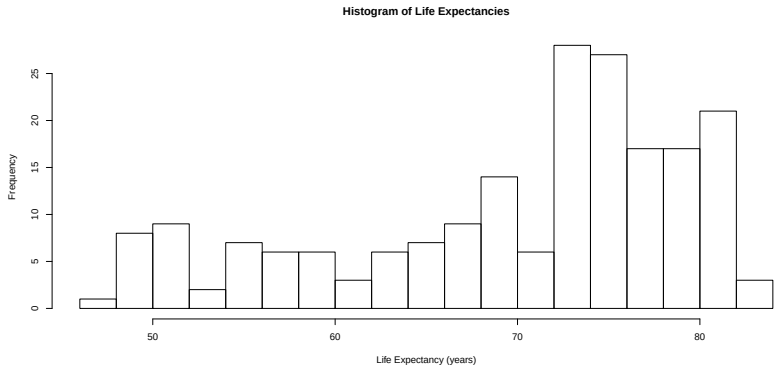
In this section we introduce histograms and investigate shapes of different variables. Lets start with plotting the histogram of the life expectancy data:

```
hist(LifeExp, breaks=5, xlab = 'Life Expectancy (years)',
 main = 'Histogram of Life Expectancies')
```



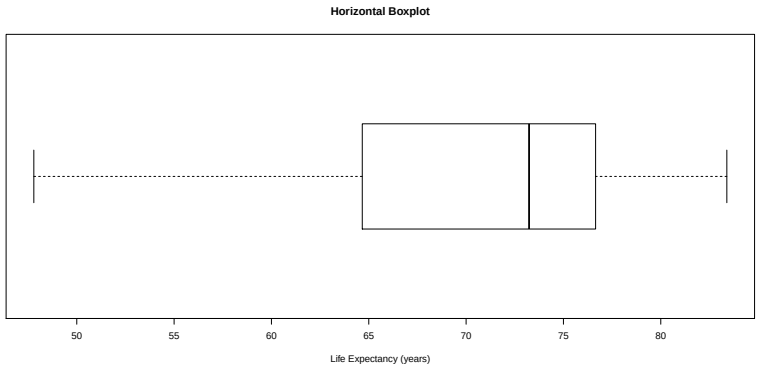
To make bins narrower (and thus noisier) we can use 'breaks' argument, for example 'breaks=20' produce approximately 20 bins:

```
hist(LifeExp, breaks=20, xlab = 'Life Expectancy (years)',
 main = 'Histogram of Life Expectancies')
```



To compare previous histograms with boxplot, we need to construct a horizontal boxplot. To do that just use 'horizontal=TRUE' in the boxplot function:

```
boxplot(LifeExp, horizontal=TRUE, range=0,
 xlab = 'Life Expectancy (years)',main='Horizontal Boxplot')
```



As before we can get basic statistics of the Life Expectancy data:

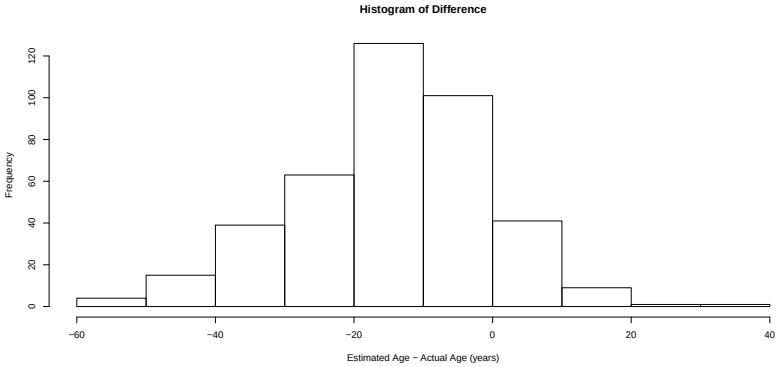
```
summary(LifeExp)
```

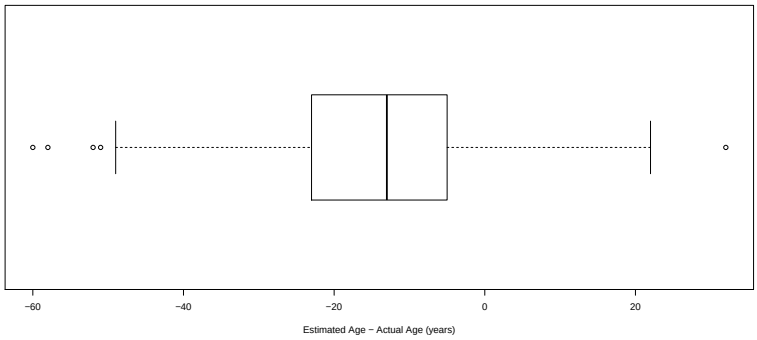
```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  47.79   64.67   73.24   69.86   76.65   83.39
```

Since the mean is smaller than median and based on the histogram we can conclude that this data is skewed to the left (has longer left tail). Now lets consider 'Skeleton' data set and make the histogram, horizontal modified boxplot and summary for the difference variable ('DGerror'):

```
hist(DGerror, xlab="Estimated Age - Actual Age (years)",
 main="Histogram of Difference")
boxplot(DGerror, horizontal=TRUE,
 xlab="Estimated Age - Actual Age (years)")
summary(DGerror)
```

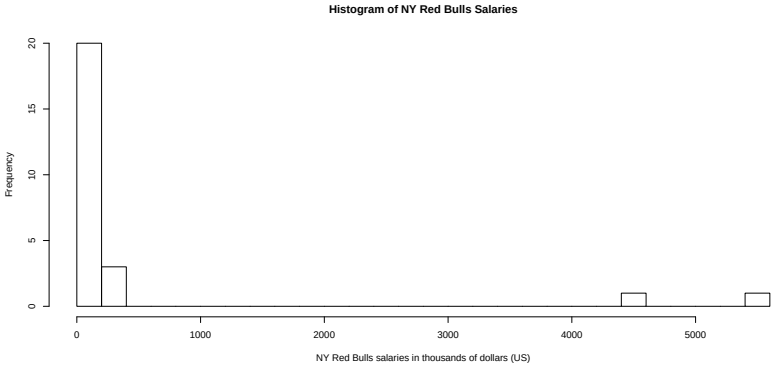
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -60.00  -23.00  -13.00  -14.15   -5.00   32.00
```





Based on the plots and since mean and median are almost the same we see that the distribution of differences is symmetric. The last data set that we will analyse in this section is 'NY Red Bull Salaries'. Plotting the histogram of this data we again notice two large outliers:

```
hist(ny.salaries/1000,breaks=24, xlab="NY Red Bulls salaries in
 thousands of dollars (US)", main="Histogram of NY Red Bulls Salaries")
```



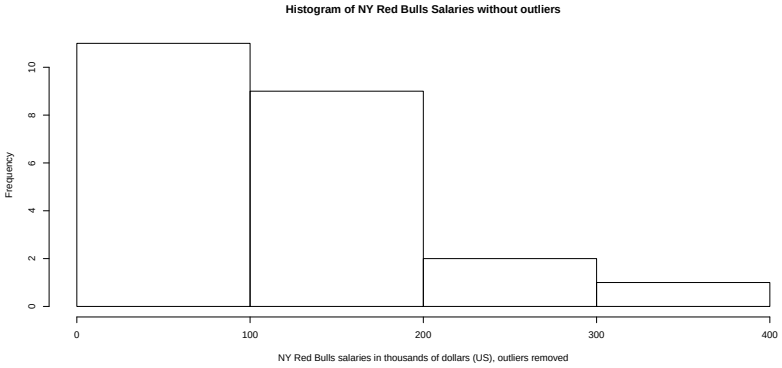
To make histogram without these two values, we first create a new variable 'ny.salaries.no.outliers' with these two outliers removed:

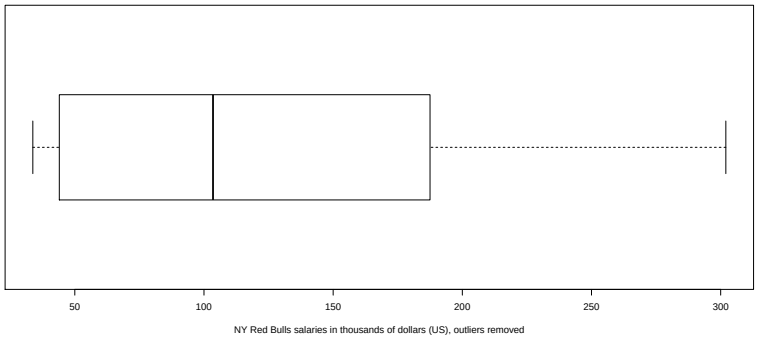
```
ny.salaries.no.outliers = sort(ny.salaries)[1:23]
```

And as we did before plot histogram, modified boxplot and summary for this new variable:

```
hist(ny.salaries.no.outliers/1000, breaks=3, xlab="NY Red Bulls salaries
 in thousands of dollars (US), outliers removed", main="Histogram of NY
 Red Bulls Salaries without outliers")
boxplot(ny.salaries.no.outliers/1000, horizontal=TRUE, xlab="NY Red Bulls
 salaries in thousands of dollars (US), outliers removed")
summary(ny.salaries.no.outliers)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  33750   44000  103500  119900  187500  302000
```





This data set has mean which is larger than median and looking on the plots we say that this data is skewed to the right (or has longer right tail).
Now lets return to the 'DGerror' variable, and we want to check that it follows the empirical rule. Since the distribution of this variable is unimodal and symmetric we expect to get close to theoretical results. First we find mean, standard deviation and number of observations of the differences variable:

```
mean.DGerror=mean(DGerror)
sd.DGerror=sd(DGerror)
n.DGerror=length(DGerror)
```

Next step is to count how many differences are between the mean minus one standard deviation and mean plus one standard deviation and then divide this count by the sample size. To see which observations are less than mean plus one standard deviation we can use '<=' logical operator which returns 'TRUE' or 'FALSE' for each observation, for example the first six values are:

```
head(DGerror <= mean.DGerror + sd.DGerror)
```

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

We see that it turns out that first 6 observations are less than mean plus one standard deviation. Similarly

```
head(DGerror >= mean.DGerror - sd.DGerror)
```

```
[1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE
```

This shows that second, fourth, fifth and sixth observations are larger than mean minus one standard deviation (first and third are smaller). To see which are in between, just multiply the two variables above, so that if two are 'TRUE' then product is 1 if one of them is 'FALSE' then product is 0:

```
head((DGerror <= mean.DGerror + sd.DGerror)*(DGerror >= mean.DGerror - sd.DGerror))
```

```
[1] 0 1 0 1 1 1
```

It means that second, fourth, fifth and sixth observations are in this interval. So to get proportion we just sum the last variable (get total number of observations between mean plus one standard deviation and mean minus one standard deviation) and divide by the total number of observations:

```
sum( (DGDifference<=mean.DGDiff+sd.DGDiff)*(DGDifference>=
 mean.DGDiff-sd.DGDiff) )/n.DGDiff
```

```
[1] 0.6825
```

Similarly we do it for two and three standard deviations.

```
sum( (DGDifference<=mean.DGDiff+2*sd.DGDiff)*(DGDifference>=
 mean.DGDiff-2*sd.DGDiff) )/n.DGDiff
```

```
[1] 0.95
```

```
sum( (DGDifference<=mean.DGDiff+3*sd.DGDiff)*(DGDifference>=
 mean.DGDiff-3*sd.DGDiff) )/n.DGDiff
```

```
[1] 0.9925
```

The proportions that we get are very close to theoretical empirical rule!

5 Categorical Variables

In this section we focus on visual representation of categorical variables. We start with the variable 'Region' from Life Expectancy data file that shows to which regions different countries belong to. The variable of regions is of course categorical with six classes. First we make a table of counts:

```
table(Region)
```

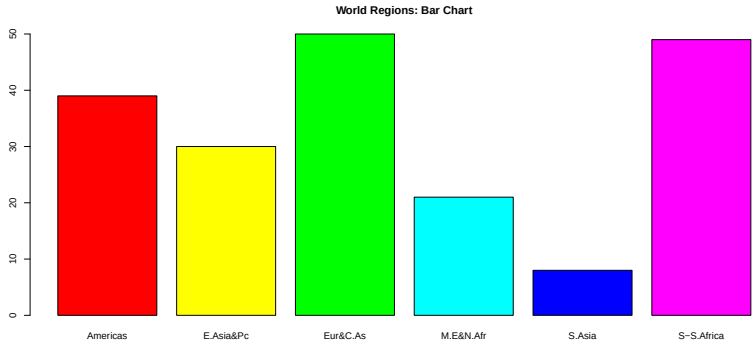
```
Region
Amer  EAP EuCA MENA  SAs  SSA
  39   30   50   21    8   49
```

This table just shows how many observations are in each category. It will be useful to construct a new variable of these counts and relative frequencies (which is just counts divided by total number of observations):

```
counts = table(Region)
rel.freq = counts/sum(counts)
```

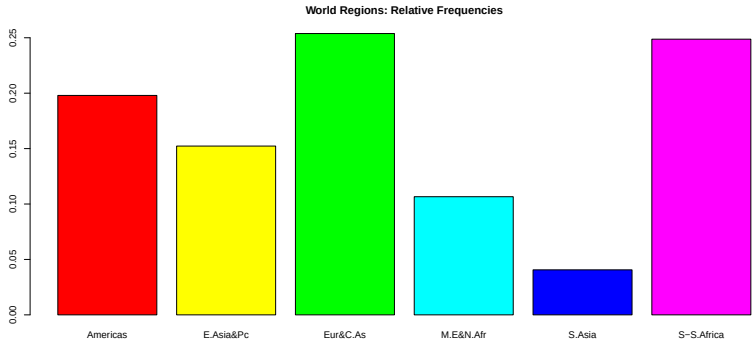
Now we are ready to make a bar-plot of counts which is appropriate for qualitative data. Note that 'col' and 'names.arg' in the bar-plot arguments correspond to colors and names for each bar. Also 'rainbow(x)' produce x random colors, we need 6 colors here because we have 6 categories:

```
region_names = c("Americas", "E.Asia&Pc", "Eur&C.As",
 "M.E&N.Afr","S.Asia", "S-S.Africa")
barplot(counts, col=rainbow(6),names.arg = region_names,
 main = "World Regions: Bar Chart")
```



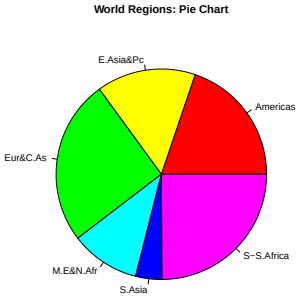
To make a bar-plot of relative frequencies, just use 'rel.freq' variable instead of 'counts':

```
barplot(rel.freq, col=rainbow(6),names.arg = region_names,
 main = "World Regions: Relative Frequencies")
```



Another useful plot for categorical variables is a pie chart:

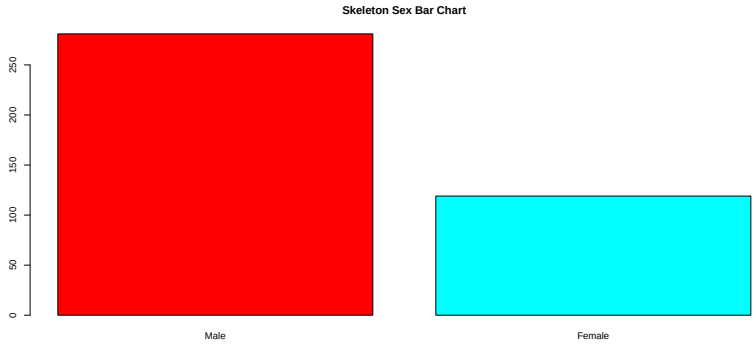
```
pie(counts, col=rainbow(6),label = region_names, main = "World Regions: Pie Chart")
```

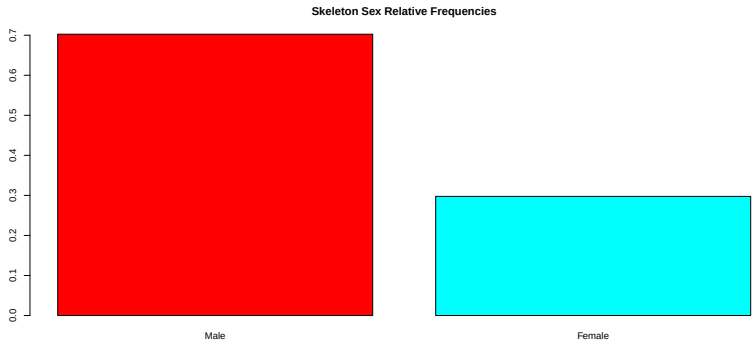


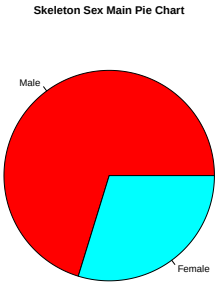
Now we return to the 'Skeleton' data and first analyse 'sex' variable. As before we make two variables of counts and relative frequencies and construct two bar charts and a pie chart. Note that 1 corresponds to male and 2 for female and therefore in the names variable 'Male' is first and 'Female' is second:

```
sex_counts = table(Sex)
sex_freq = sex_counts/sum(sex_counts)
sex_names = c('Male', 'Female')
barplot(sex_counts, col=rainbow(2), names.arg = sex_names,
```

```
 main='Skeleton Sex Bar Chart')
barplot(sex_freq, col=rainbow(2), names.arg = sex_names,
 main='Skeleton Sex Relative Frequencies')
pie(sex_counts, col=rainbow(2), label=sex_names,
 main='Skeleton Sex Main Pie Chart')
```







Similarly we do for the 'BMIcat' variable from 'Skeleton' file, first find counts:

```
BMI_counts = table(BMIcat)
BMI_counts
```

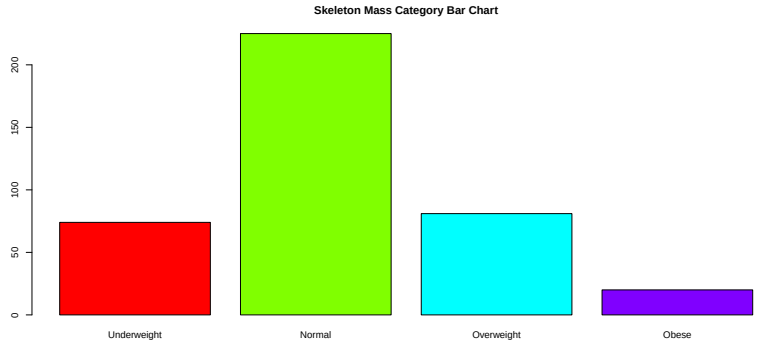
```
BMIcat
     normal       obese  overweight underweight
        225          20          81          74
```

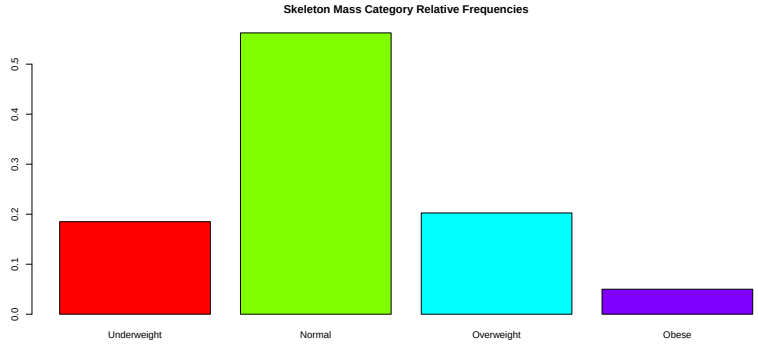
This table is not perfect because it is not in the logical order (underweight, normal, overweight and obese) but in alphabetical. So we just take these counts and put them in the right order:

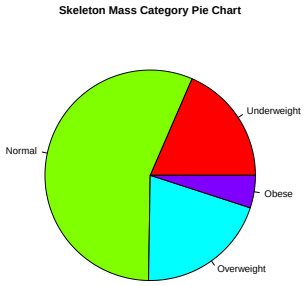
```
BMI_counts=c(74,225,81,20)
```

Note that 'c()' function makes a variable from numbers. Now as before we find relative frequencies, histograms and a pie chart:

```
BMI_freq = BMI_counts/sum(BMI_counts)
BMI_names = c('Underweight','Normal', 'Overweight', 'Obese' )
barplot(BMI_counts, col=rainbow(4), names.arg=BMI_names,
 main='Skeleton Mass Category Bar Chart')
barplot(BMI_freq, col=rainbow(4), names.arg=BMI_names,
 main='Skeleton Mass Category Relative Frequencies')
pie(BMI_counts, col=rainbow(4), label=BMI_names,
 main='Skeleton Mass Category Pie Chart')
```







6 Summary of R Functions

We give a short summary of all the R functions [and arguments] that we used in this Module:

Data Input

```
read.table() [header]
scan() [header]
```

Plots

```
barplot() [col,names.arg,main]
boxplot() [ylab,ylim,range,main,horizontal]
hist() [breaks,xlab,main]
pie() [col,label,main]
plot() [xlab,ylab,ylim,xlim,main]
```

Statistics

```
quantile()
IQR()
max()
mean() [trim]
median()
min()
sd()
summary()
var()
```

Miscellaneous

```
attach()
c()
cbind()
head()
length()
rownames()
sqrt()
sort()
sum()
table()
```