Stat 210
Applied Statistics and Data Analysis
Two Sample Problems

Joaquín Ortega
KAUST

August, 2020

Contents

1 Introduction

In the previous chapter we considered problems that arise when we want to study questions linked to a single sample. In the examples we studied, the problem was to see whether the result of an experiment, or the data obtained from a single sample, was compatible with a distributional model that was accepted as valid for the whole population.

Now we want to compare two populations. We start by looking at graphical methods for representing and comparing samples, and we introduce this through examples. Later in this chapter we will look at different statistical methods for comparing two populations.

2 Descriptive statistics

2.1 Example 1; Comparing two populations

Let's start with an example comparing two experimental fish groups: The fish are the same species and age but the groups were fed with different diets and we want to compare the effect this has on the populations. The outcome is a composite measure of weight and length.

We read the data of the two samples into R, and store each data set in a vector (matrix/table of one column). To be able to red the data we need to have the data sets in the working directory, or else give the path to the data files in the `scan` command below. To find out which is our working directory we use the function `getwd()`. To change the working directory use `setwd()`.

```
getwd()
```

```
## [1] "/Users/ortegaj/Library/Mobile Documents/com~apple~CloudDocs/Mis Documentos/Cursos /2-Applied Sta
```

The following commands read the data set and store them in R.

```
fish1 = matrix(scan("data/diet1.1"), ncol=1, byrow=T)
fish2 = matrix(scan("data/diet1.2"), ncol=1, byrow=T)
```

Compute the mean (average) of each of the samples:

```
mean(fish1)
mean(fish2)
```

```
## [1] 49.63333
## [1] 49.05
```

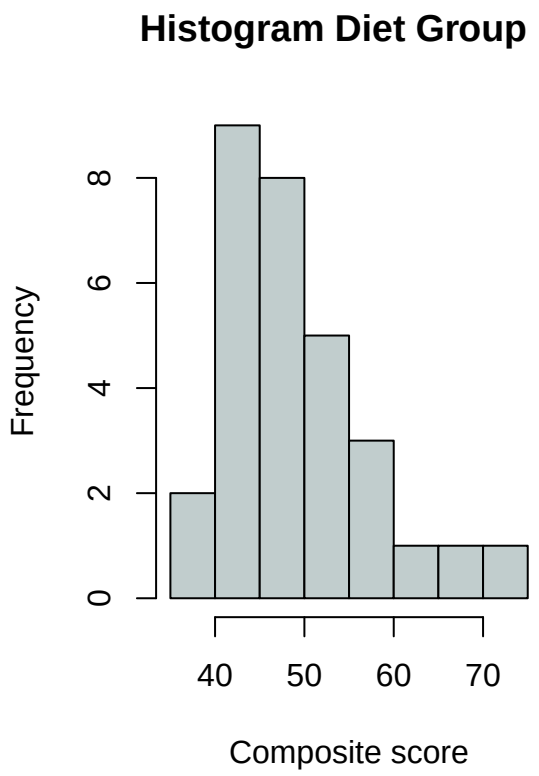
The means are almost equal, but means do not tell the whole story.

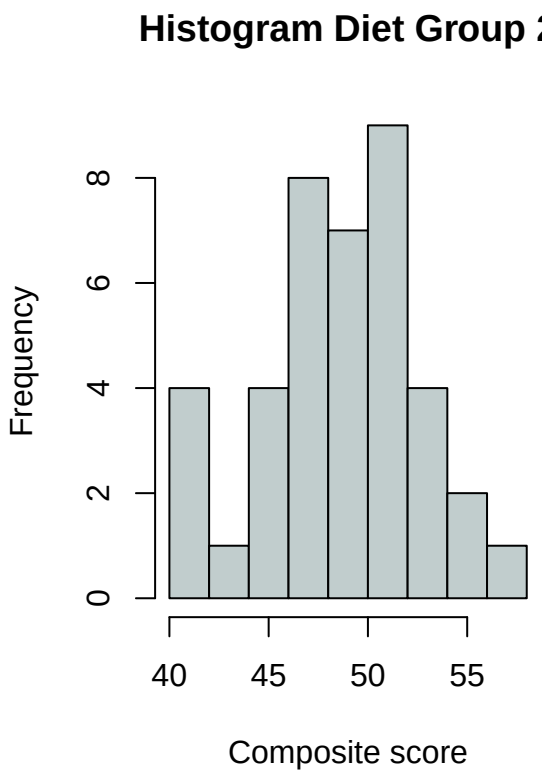
We have various ways of visualizing the data. We review some of the more frequently used.

2.1.1 Histograms

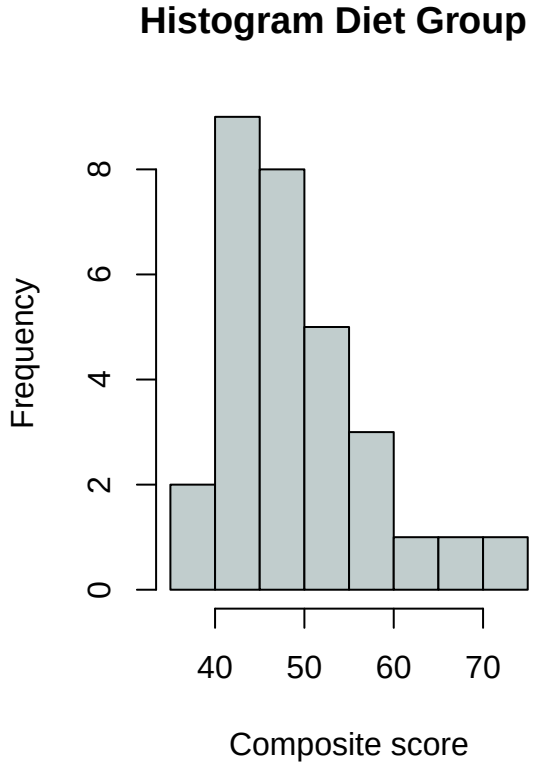
To make a histogram we divide the range of values into 'bins', which are usually all the same size. Then we count how many observations fall in each bin and draw a bar graph with this data. Histograms give an idea of the distribution of the data but their appearance depends on the number of bins and the value of the *anchor* (starting point of the division).

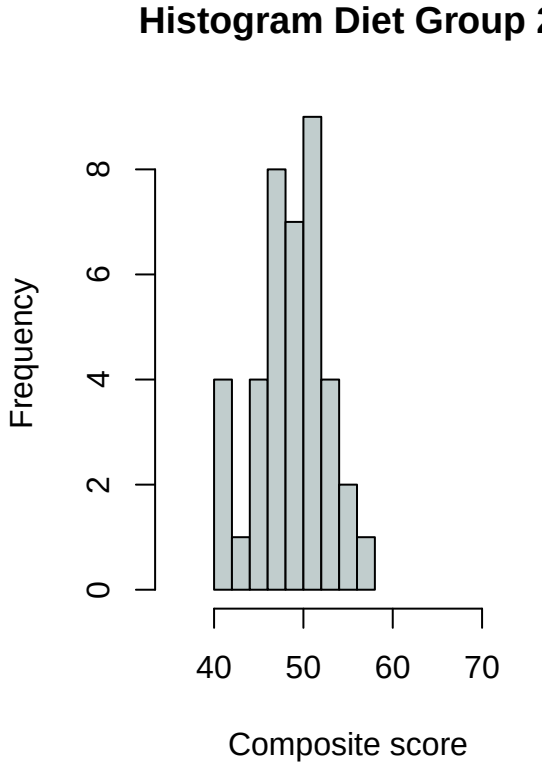
```
par(mfrow=c(1,2))
hist(fish1, xlab="Composite score",
     main="Histogram Diet Group 1", col='azure3')
hist(fish2, xlab="Composite score",
     main="Histogram Diet Group 2", col='azure3')
```





A problem with the previous graph if we want to compare the two samples is that the ranges in the x axes are not the same. This gives a wrong idea about their spread when doing the comparison.





We can see now that diet group 1 has more variability in the composite score values than group 2. To further improve the comparison we plot the graphs in a single column.

Here we compute the limit references for the x axis. The minimum of all points multiplied by .8 is the lower reference:

```
(minval = 0.80*min(fish1, fish2))
```

```
## [1] 30.4
```

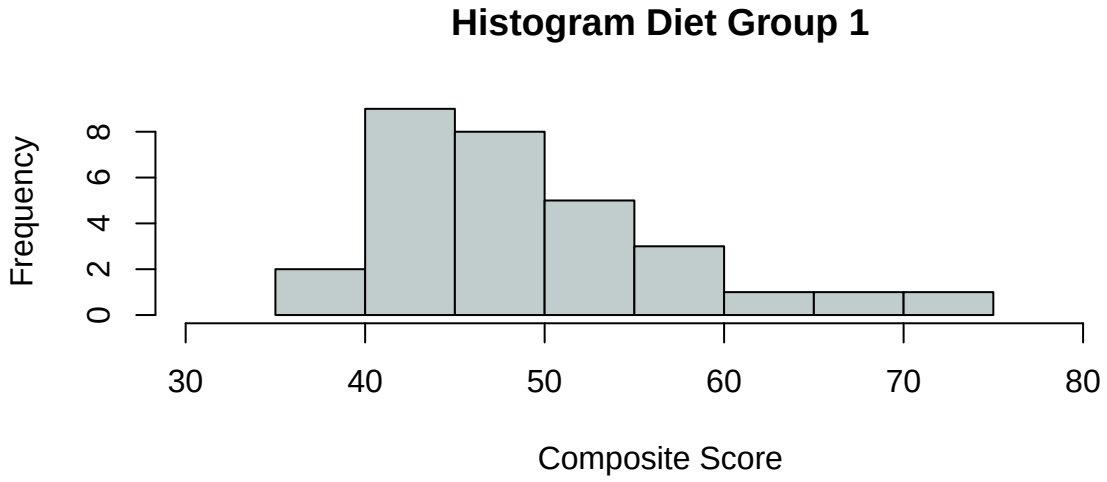
while the maximum of all points by 1.1 is the higher reference:

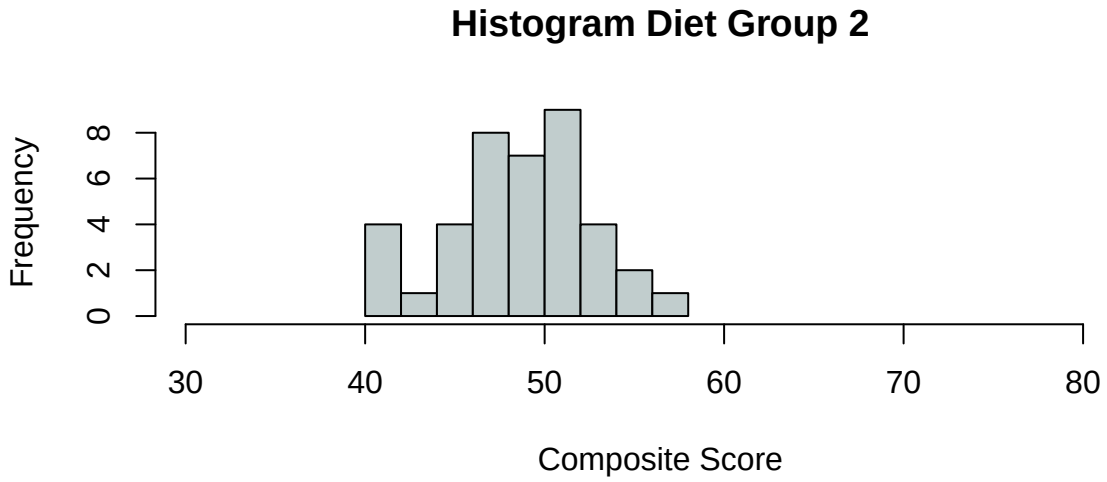
```
(maxval = 1.10*max(fish1, fish2))
```

```
## [1] 82.5
```

Let's visualize the histograms setting the new references with `xlim`

```
par(mfrow=c(2,1)) # Graphs in two rows and one column
hist(fish1, xlab="Composite Score", col='azure3', main=
       "Histogram Diet Group 1", xlim=c(minval, maxval))
hist(fish2, xlab="Composite Score", col='azure3', main=
       "Histogram Diet Group 2", xlim=c(minval, maxval))
```

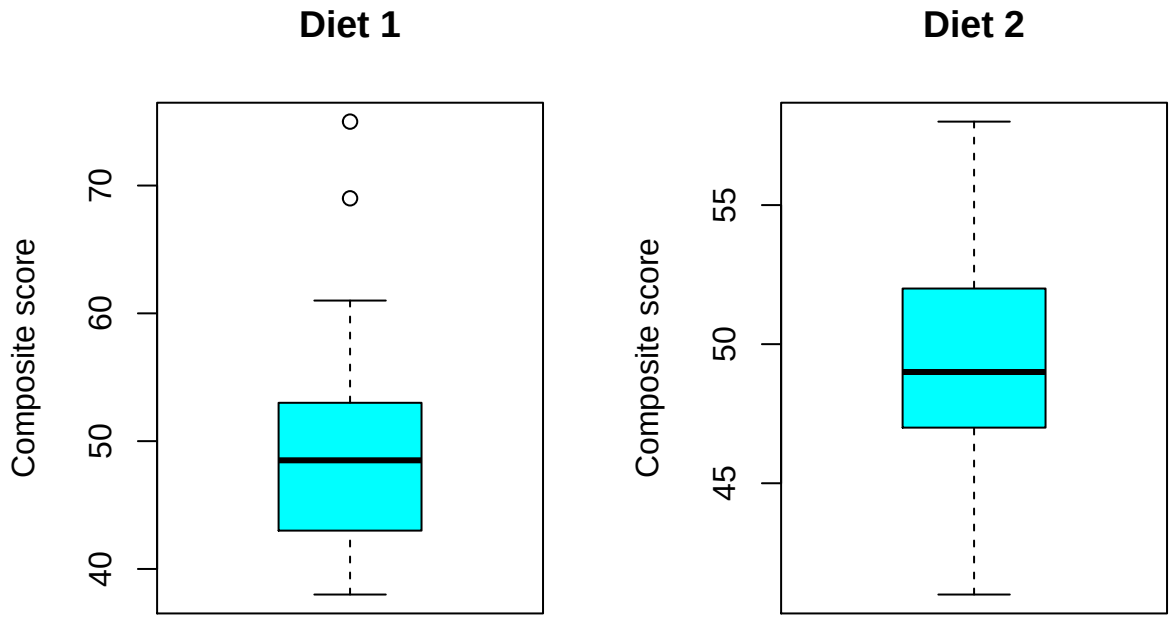




2.1.2 Boxplots

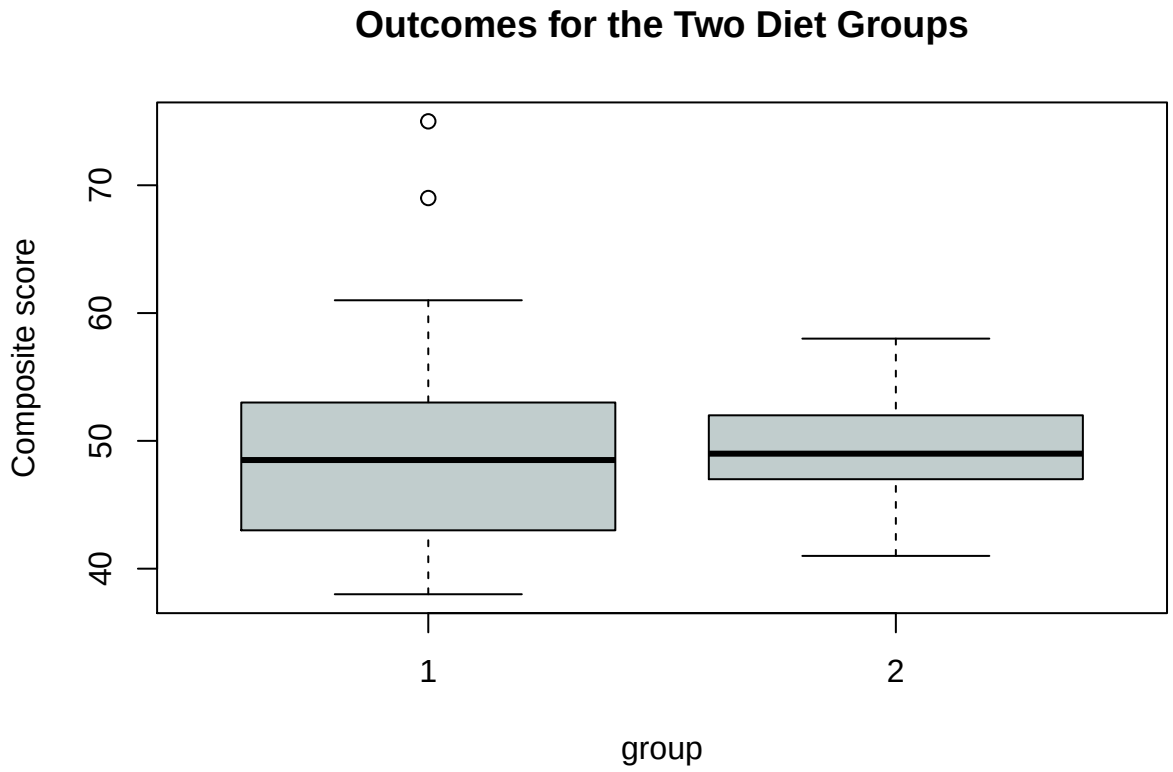
Another way to compare distributions is the use of boxplots.

```
par(mfrow=c(1,2))
boxplot(fish1,main='Diet 1',col=5, ylab = 'Composite score')
boxplot(fish2,main="Diet 2",col=5, ylab = 'Composite score')
```



In this case the y scales are different and comparisons can be misleading. To avoid this, we display the two boxplots is a single graph. For this, we need to build a data frame with the values for the composite score in one column and the group in a second column.

```
par(mfrow=c(1,1))
fish.vec = c(fish1, fish2)
n1 = length(fish1); n2 = length(fish2)
group = c(rep(1, n1), rep(2, n2)) # this is to identify each sample
    # in the new data frame
fish.df <- data.frame(comp.score=fish.vec, group = group)
boxplot(fish.vec ~ group, data = fish.df, col = 'azure3',
        main = "Outcomes for the Two Diet Groups", ylab = 'Composite score')
```

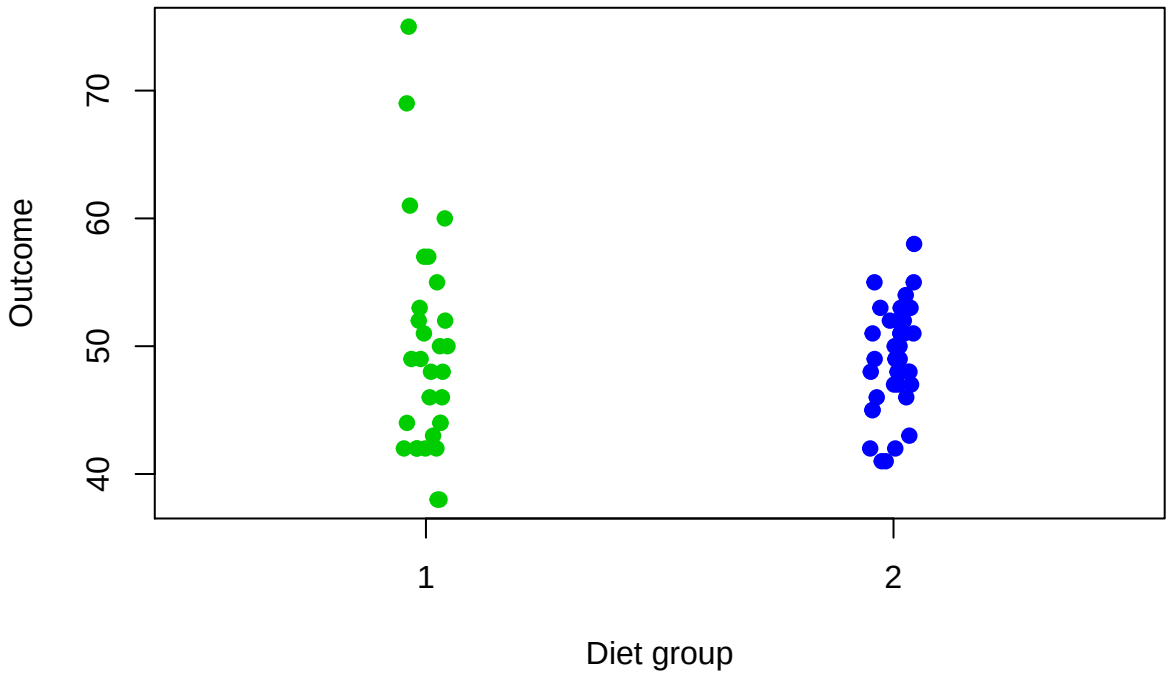


Now both plots are on the same figure and have the same scale, and we can see the difference in the spread of the two populations.

2.1.3 Dotplots

A third way to plot the data is using dotplots or dotcharts. In this type of chart, each result is plotted in a different row, so that the different values can be easily visualized. The values on the y axis have no meaning. In the plot below, values for diet 1 are represented on top, in green, and values for diet 2 are in blue, in the lower part of the graphs.

```
fsh2 <- c(fish1,fish2)
plot(jitter(group, factor=0.25),as.vector(fsh2),
     ylab='Outcome', xlab='Diet group', xlim = c(0.5,2.5),
     pch=19, col = group +2, xaxp = c(1,2,1))
```



The command `jitter` adds a little noise to the points in the horizontal direction, so that different points having the same value can be distinguished.

2.1.4 Numerical summaries

There are several ways of getting numerical summaries of data set in `R`. The standard way is to use the function `summary` from the base package. For example,

```
summary(fish1)
```

```
##        V1
##  Min.   :38.00
##  1st Qu.:43.25
##  Median :48.50
##  Mean   :49.63
##  3rd Qu.:52.75
##  Max.   :75.00
```

The function gives six statistics for the data: minimum and maximum value, mean, median and first and third quartiles. Neither variance nor standard deviation are included in this function. We can get these parameters using the functions `var` and `sd`:

```
var(fish1); sd(fish1)
```

```
##          [,1]
## [1,] 74.17126
```

```
## [1] 8.612274
```

To get a more complete set of descriptive statistics we can use the function `stat.desc` from the package `pastecs`:

```
library(pastecs)
stat.desc(fish1)
```

```
##                      V1
```

```
## nbr.val        30.0000000
## nbr.null        0.0000000
## nbr.na          0.0000000
## min            38.0000000
## max            75.0000000
## range          37.0000000
## sum          1489.0000000
## median         48.5000000
## mean           49.6333333
## SE.mean         1.5723789
## CI.mean.0.95    3.2158760
## var            74.1712644
## std.dev         8.6122741
## coef.var        0.1735179
```

The parameters included in this function are sample size, number of null values, number of NAs, Minimum, Maximum, range, sum, median, mean, standard error of the mean, confidence interval for the mean (default confidence level is .95) variance, standard deviation and variation coeficient, which is defined as the standard deviation divided by the mean.

We can combine either of these functions with commands from the `apply` family to obtain summaries for the subset of elements in each category within a data set. Recall the `fish.df` is a data frame with two columns, the first has the composite score for all the fish and the second has the group. If we use `summary` on these set we get

```
summary(fish.df)
```

```
##    comp.score        group
##  Min.   :38.00   Min.   :1.000
##  1st Qu.:45.25   1st Qu.:1.000
##  Median :49.00   Median :2.000
##  Mean   :49.30   Mean   :1.571
##  3rd Qu.:52.00   3rd Qu.:2.000
##  Max.   :75.00   Max.   :2.000
```

The first column is the summary for the composite score of all the fish, while the second is corresponds to the `group` variable, that has two values, and the summary is not interesting. Now, suppose that we want a summary for each group of fish. We can do this in several ways. For instance

```
with(fish.df,tapply(comp.score, group, summary))
```

```
## $`1`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   38.00   43.25   48.50   49.63   52.75   75.00
##
## $`2`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   41.00   47.00   49.00   49.05   52.00   58.00
```

An alternative way is

```
by(fish.df,group,summary)
```

```
## group: 1
##    comp.score        group
##  Min.   :38.00   Min.   :1
##  1st Qu.:43.25   1st Qu.:1
##  Median :48.50   Median :1
```

```
##  Mean   :49.63   Mean   :1
##  3rd Qu.:52.75   3rd Qu.:1
##  Max.   :75.00   Max.   :1
## ------------------------------------------------
## group: 2
##    comp.score        group
##  Min.   :41.00   Min.   :2
##  1st Qu.:47.00   1st Qu.:2
##  Median :49.00   Median :2
##  Mean   :49.05   Mean   :2
##  3rd Qu.:52.00   3rd Qu.:2
##  Max.   :58.00   Max.   :2
```

Observe that the output format is different.

2.1.5 Outliers

Outliers or atypical values are data points that differ significantly from the other points in the sample. Outliers can occur simply by chance but they frequently indicate either measurement error or a heavy-tailed distribution. An outlier can cause problems in statistical calculations and usually deserve careful consideration. *Robust* statistical methods seek to mitigate the effect of outliers in statistical analyses. In the following examples we consider the effect of (artificial) outliers in the analysis of our samples.

Order the sample from the minimum to the maximum

```
fish2.1 = sort(fish1); fish2.2 = sort(fish2)
```

The means do not change

```
mean(fish2.1); mean(fish2.2)
```

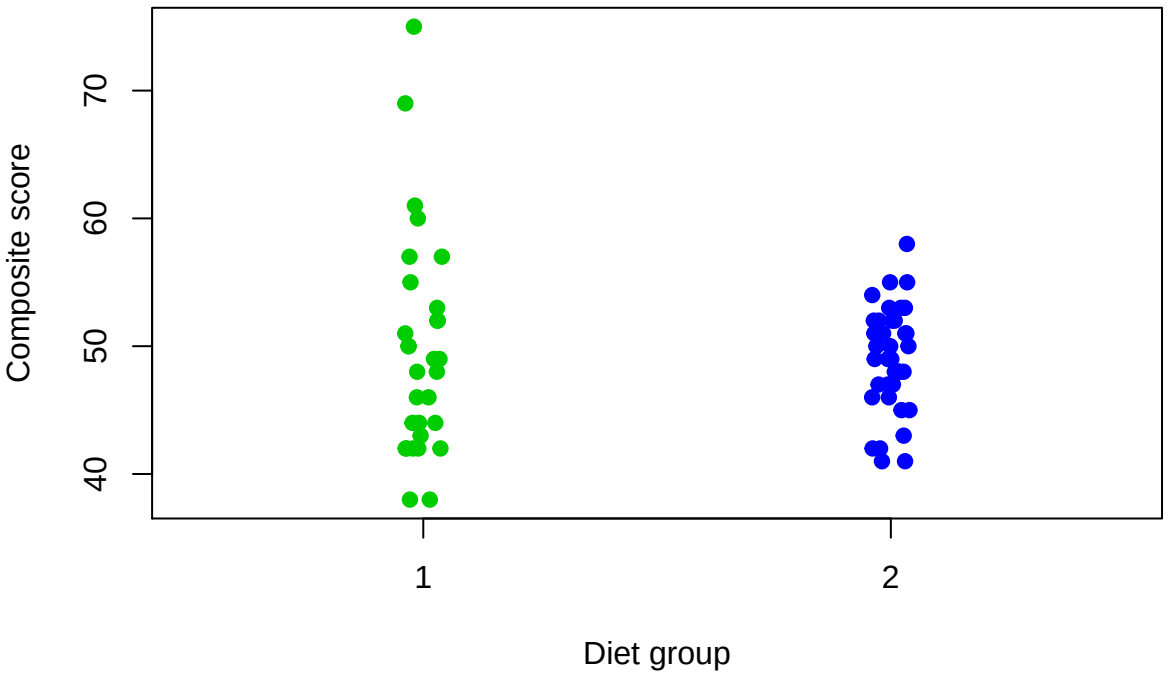
```
## [1] 49.63333
## [1] 49.05
```

Here we observe the data

```
##  [1] 38 38 42 42 42 42 42 43 44 44 44 46 46 48 48 49 49 50
## [19] 50 51 52 52 53 55 57 57 60 61 69 75
```

```
##  [1] 41 41 42 42 43 45 45 46 46 47 47 47 48 48 48 48 48 49
## [19] 49 49 49 50 50 50 51 51 51 51 51 52 52 52 52 53 53 53
## [37] 54 55 55 58
```

```
with(fish.df, plot(jitter(group, factor=0.2), comp.score,
     ylab='Composite score', xlab='Diet group', xlim = c(0.5,2.5),
     pch=19, col = group +2, xaxp = c(1,2,1)))
```




```
fish1x = fish2.1
fish2x = fish2.2
fish2x[n2] = 1000  # artificial outlier set at the end of the second sample
```

```
mean(fish1x)
mean(fish2x)
```

```
## [1] 49.63333
## [1] 72.6
```

The original values were

```
mean(fish1)
mean(fish2)
```

```
## [1] 49.63333
## [1] 49.05
```

The mean for the second sample has changed considerably. An alternative location parameter that is robust in the presence of outliers is the median, which is defined as the points that divides the ordered sample in two equal halves, in the sense that there are the same number of data points above and below it. The median may not be unique, i.e. there may be more than one point that divides the oredered sample in two equal halves, as the examples below will show.

Consider the ordered sample 12, 15, 15, 18, 20, 21, 27. For this sample the median is 18, as there are 3 points below and three above 18. On the other hand, for the sample 12, 15, 15, 17, 18, 20, 21, 27, any number (strictly) between 17 and 18, e.g. 17.34 or 17.7, has the property of dividing the data into two equal sets. In cases like this, the convention is to take the average value of the two middle points as the median:

$$\frac{17+18}{2} = 17.5.$$

Let's calculate the medians for the two samples.

```
median(fish1x)
median(fish2x)
```

```
## [1] 48.5
## [1] 49
```

The original values were

```
median(fish1)
median(fish2)
```

```
## [1] 48.5
## [1] 49
```

We see that nothing has changed. The value we altered in the sample has no effect on the central data points, which determine the value of the median.

The variance can also be considerable affected by outliers, as the calculations below show.

```
c(var(fish2x), sd(fish2x))
c(var(fish2), sd(fish2))
```

```
## [1] 22632.19   150.44
## [1] 15.741026  3.967496
```

One atypical value can radically change these statistics, as we can see form the values above. There are several robust dispersion statistics. One of them is the interquartile range (iqr), defined as the difference between the third and first quartiles of the sample. Recall that the first quartile is the median of the lower half of the sample while the third quartile is the median of the upper half. In other words, the quartiles divide the ordered sample into 4 groups having approximately the same number of data points. The second quartile is the median. In the first example above, for the sample summary 12, 15, 15, 18, 20, 21, 27, the first quartile is the media of the lower half, i.e., of 12, 15, 15, 18, which is 15, while the third quartile is the median of the upper half, i.e., 18, 20, 21, and 27, which is 20.5. We can get these values using `summary` in `R`:

```
summary(c(12, 15, 15, 18, 20, 21, 27))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.00   15.00   18.00   18.29   20.50   27.00
```

The interquartile range is the difference between the third and first quartiles. By definition, it is the width of the central 50% of the data, and thus is a measure of the spread of the sample. This statistic can be obtained in `R` with the function `IQR`:

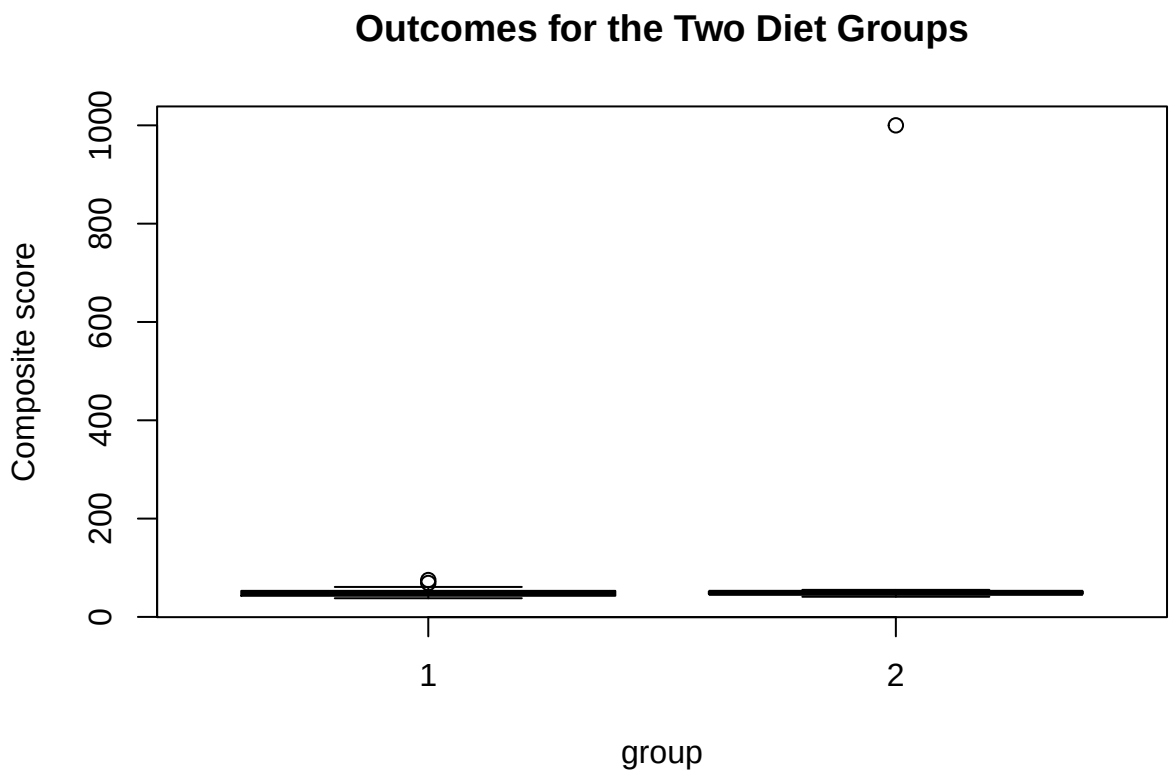
```
IQR(fish2x)
IQR(fish2)
```

```
## [1] 5
## [1] 5
```

The presence of the outlier has not changed the valur of the iqr for the fish sample.

Let's see what happens to the boxplot when there is an outlier

```
par(mfrow=c(1,1))
fish.vecx = c(fish1x, fish2x)
boxplot(fish.vecx ~ group, main="Outcomes for the Two Diet Groups",
        ylab='Composite score')
```



2.2 Example 2: The importance of the baseline

In this example we have also two experimental groups that now correspond to different methods for improving reading comprehension. The outcome is the score on a reading test. We have scores before and after the methods are applied.

We start by reading the data and storing it in 2 matrices of 2 columns each

```
group1 = matrix(scan("data/comp.1"), ncol=2, byrow=T)
group2 = matrix(scan("data/comp.2"), ncol=2, byrow=T)
```

Build up vectors for each variable

```
group1.pre = group1[,1];
group1.post = group1[,2];
group2.pre = group2[,1];
group2.post = group2[,2]
```

Examine the data

```
group1.post
```

```
##   [1] 71 81 77 87 86 74 79 82 77 77 77 85 77 82 76 84 79 91
##  [19] 73 81 77 79 80 78 87 77 86 85 85 65 83 74 73 78 87 78
##  [37] 77 86 82 90 76 83 75 77 84 80 81 79 81 81 88 64 81 74
##  [55] 91 81 80 75 84 89 75 77 76 75 78 83 80 73 83 89 91 74
##  [73] 84 80 82 83 79 88 70 86 75 81 87 94 87 75 83 76 71 84
##  [91] 74 69 79 67 79 84 81 78 87 81
```

```
sort(group1.post);sort(group2.post)
```

```
##   [1] 64 65 67 69 70 71 71 73 73 73 74 74 74 74 74 75 75 75 75 75 75
##  [22] 76 76 76 76 77 77 77 77 77 77 77 77 77 77 77 78 78 78 78 78 79 79
```

```
##  [43] 79 79 79 79 79 80 80 80 80 80 81 81 81 81 81 81 81 81 81 81 82
##  [64] 82 82 82 83 83 83 83 83 83 84 84 84 84 84 84 85 85 85 86 86 86
##  [85] 86 87 87 87 87 87 87 88 88 89 89 90 91 91 91 94
```

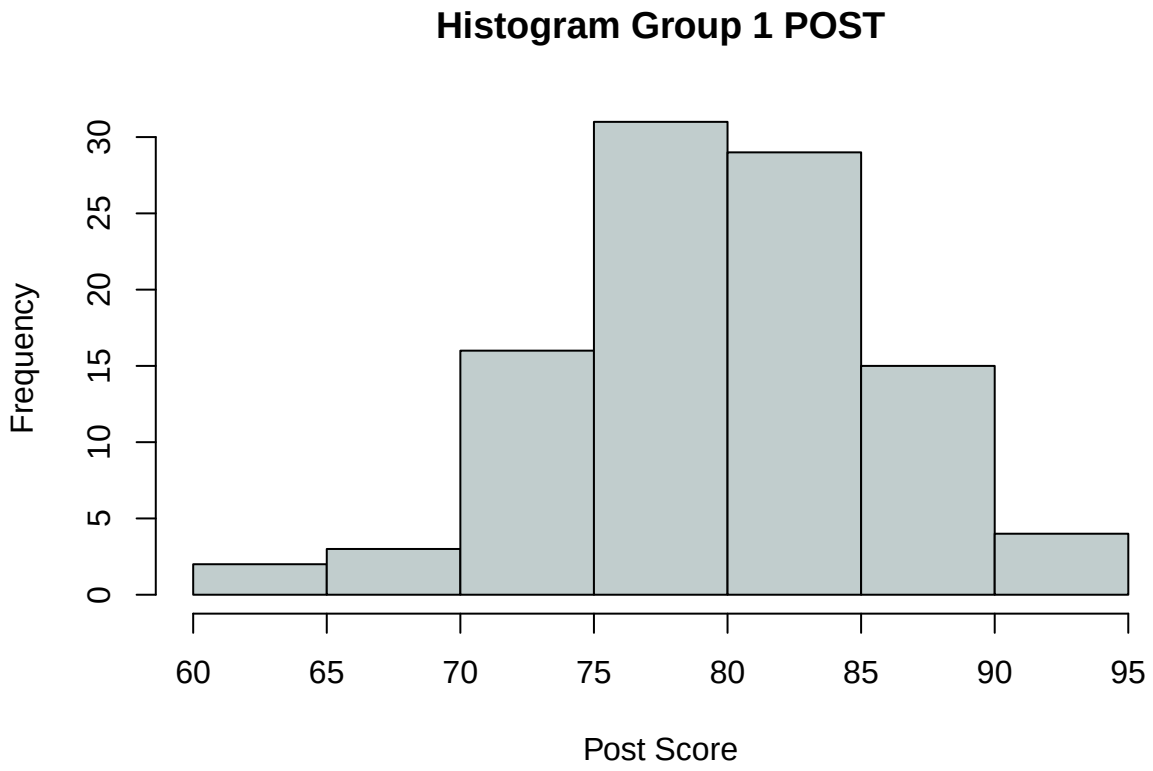
```
##   [1] 67 68 69 70 71 71 72 73 73 73 74 74 74 74 74 74 75 75 75 75 75
##  [22] 75 75 75 76 76 76 76 76 77 77 77 77 77 77 77 78 78 78 78 78 79
##  [43] 79 79 79 79 79 79 79 79 79 79 79 79 79 80 80 80 80 80 80 80
##  [64] 80 81 81 81 81 82 82 82 82 82 82 82 83 83 83 83 83 83 83 83 83
##  [85] 83 84 84 84 85 85 85 85 86 86 87 87 87 88 88 90
```

```
mean(group1.post);mean(group2.post)
```

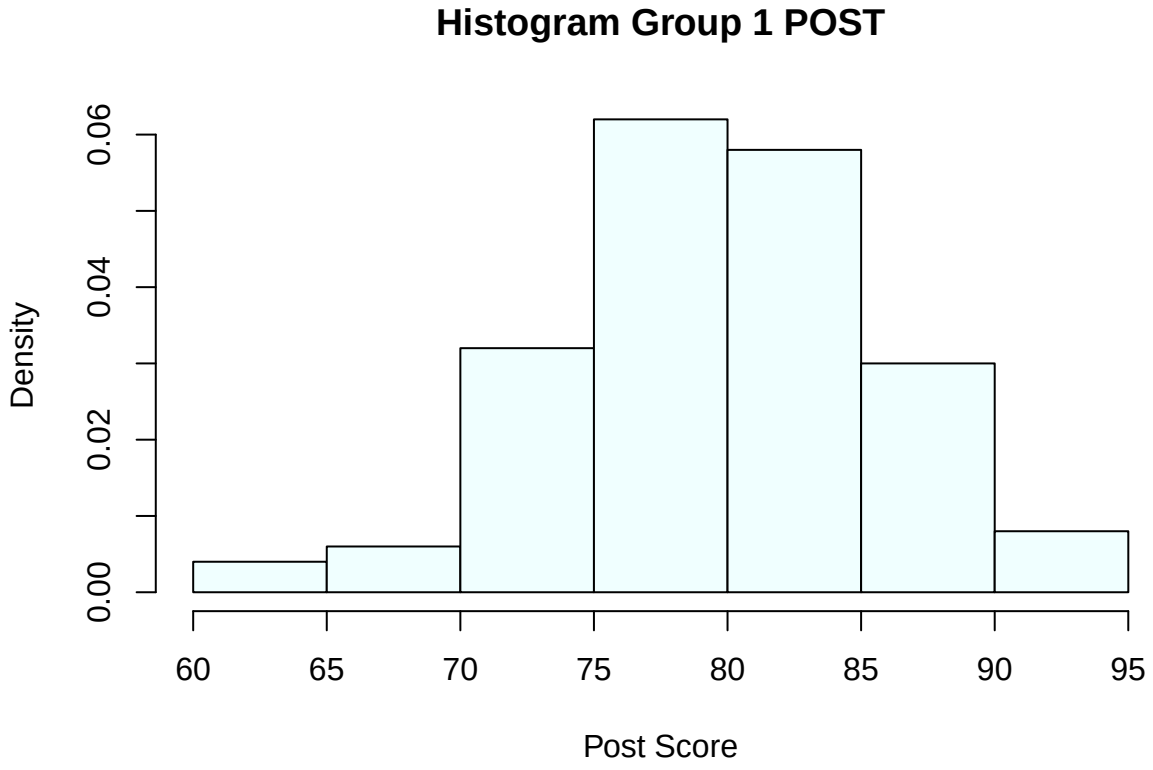
```
## [1] 80.05
```

```
## [1] 79.04
```

To explore the data graphically we plot a histogram. There are two flavors. In the first plot, the y axis represents the number of values that fall in each bin, i.e., the (absolute) frequency.

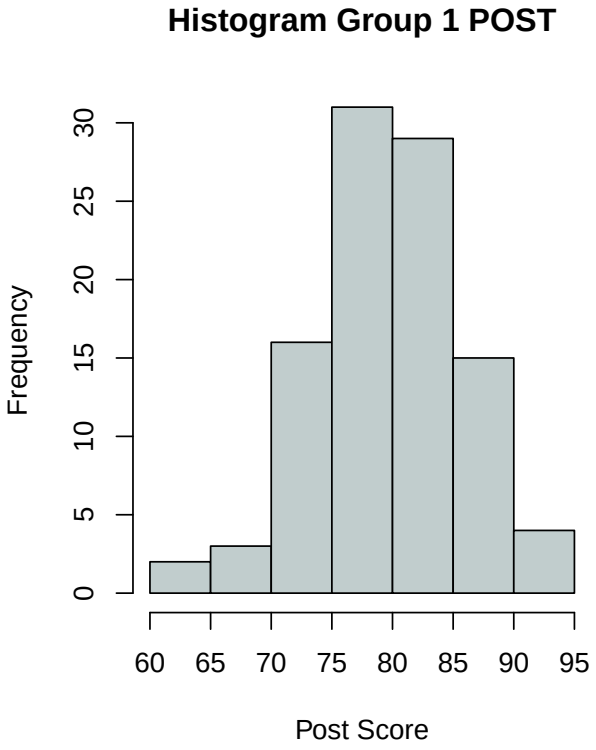


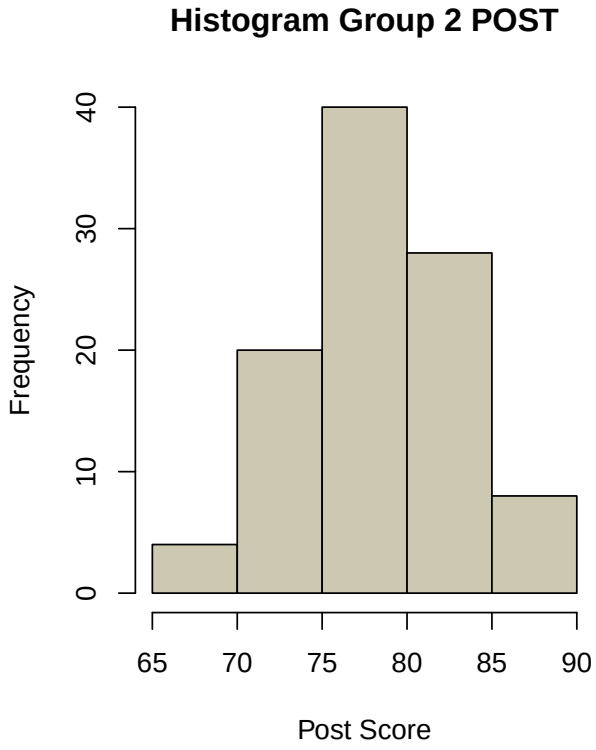
In the second version, the y axis represents the relative frequency for each bin, i.e. the number of values that fall in each bin divided by the total sample size.



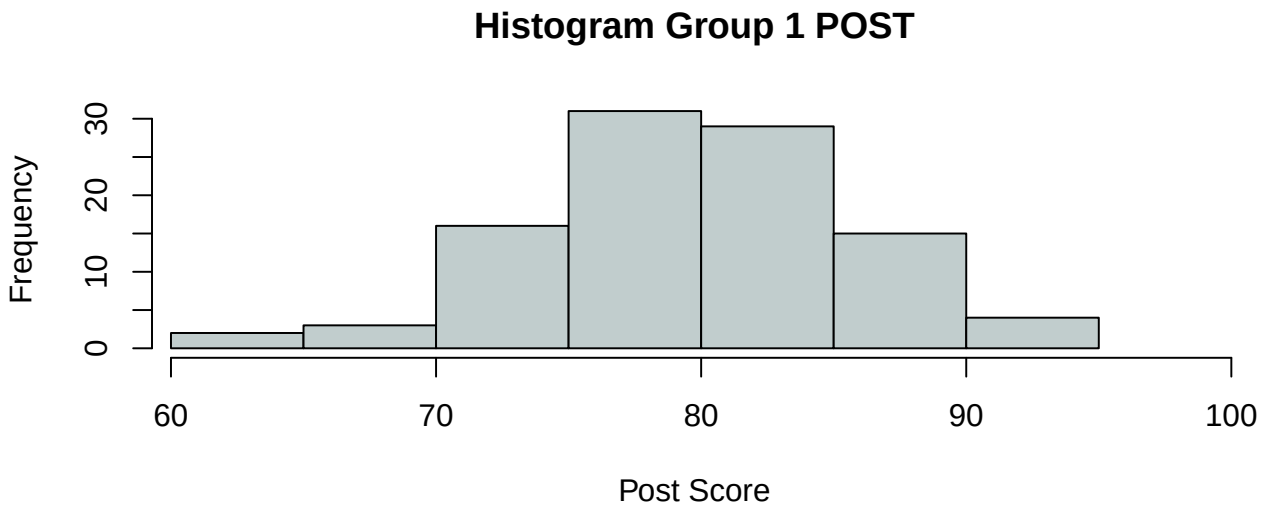
We see that the shape of the histogram is exactly the same, the only thing that changes is the scale in the y axis.

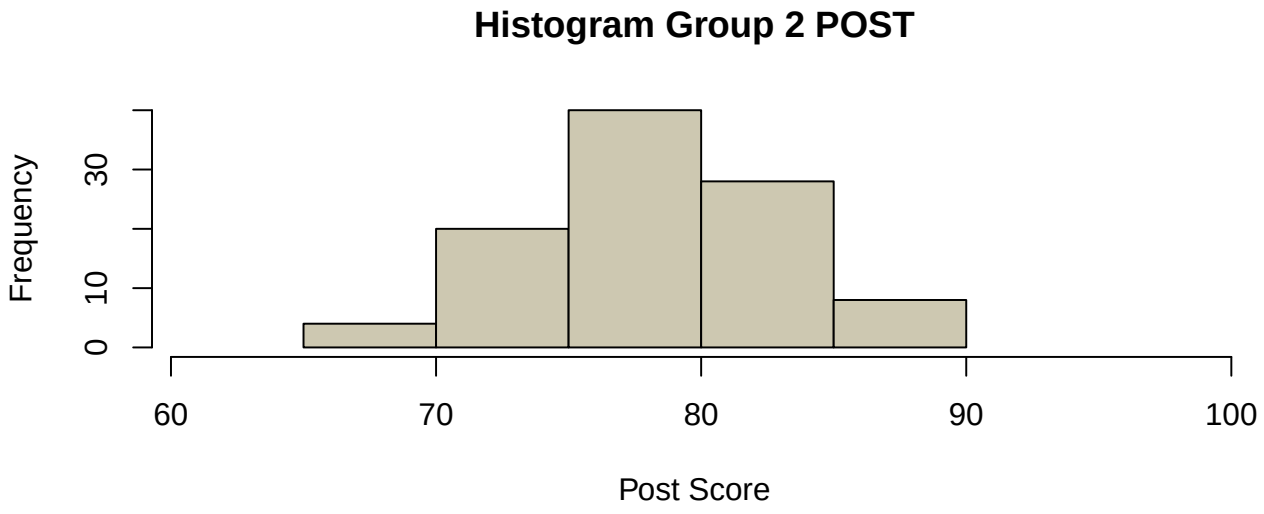
To compare the two populations we can plot the histograms together. There are two ways of doing this, but remember that the goal is to make comparisons.





Plotting the histograms side by side is not very efficient. It is better to put one on top of the other, taking care of using the same scale in the x axis for both plots.





2.2.1 Numerical Summaries

Let's look at the numerical summaries for the post scores.

```
summary(group1.post)
summary(group2.post)
```

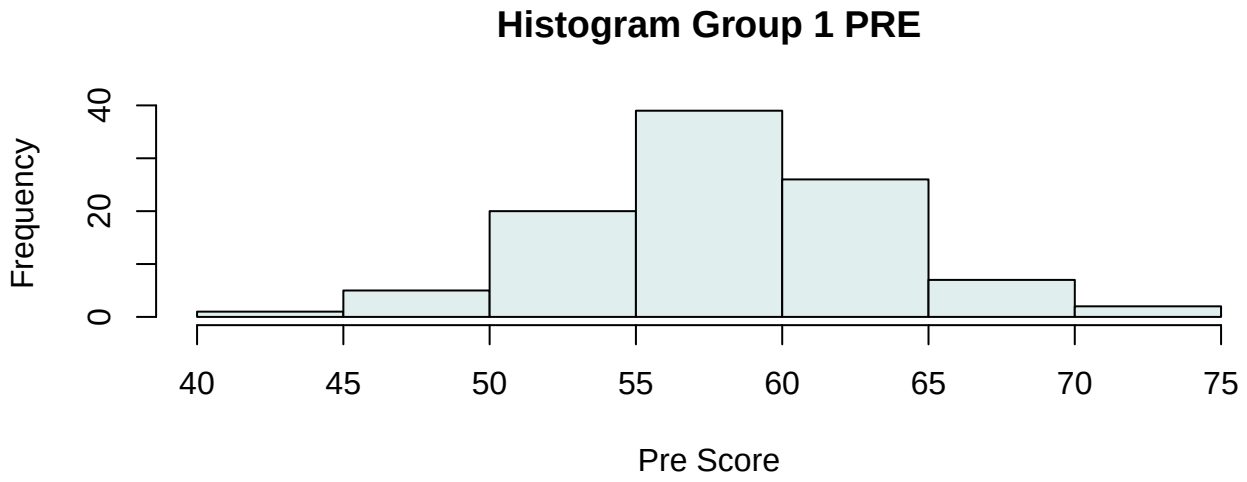
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   64.00   76.75   80.00   80.05   84.00   94.00
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   67.00   76.00   79.00   79.04   82.25   90.00
```

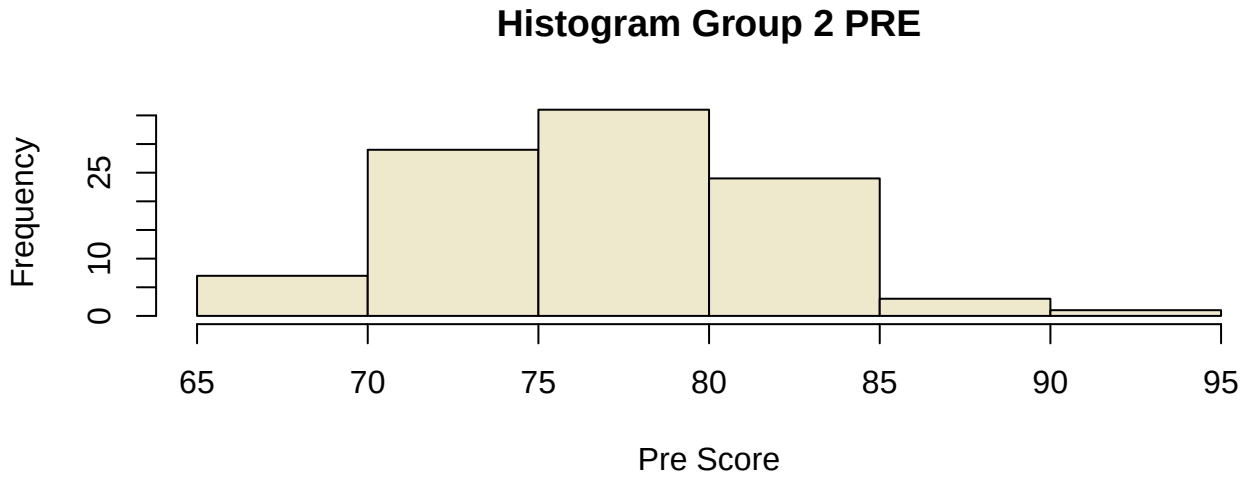
- Do we conclude that the post scores of the two methods are similar?

They certainly are very close, as was also evident from the last figure.

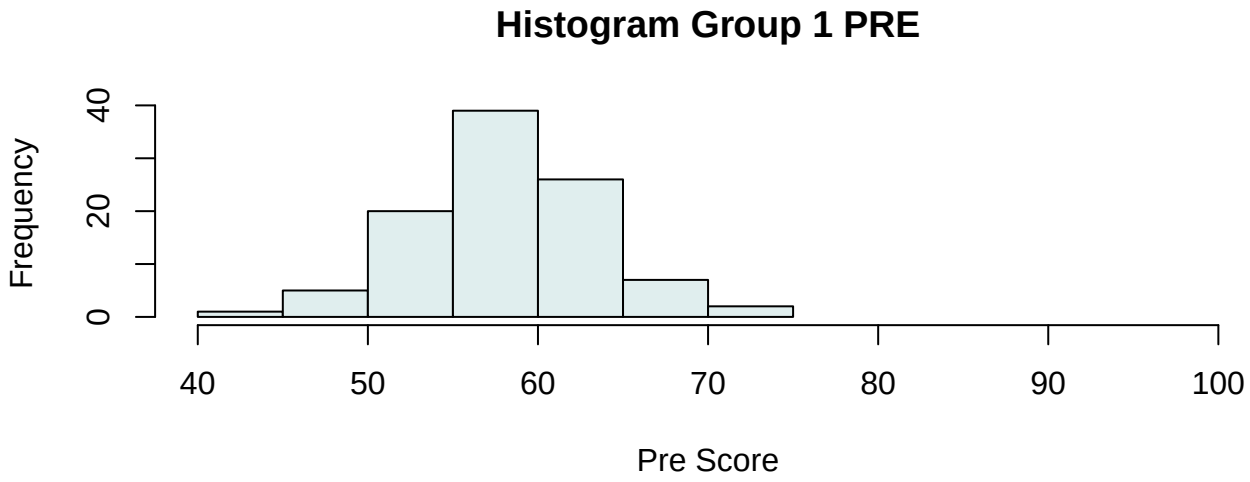
- Can we conclude that the effect of the two methods are the same?

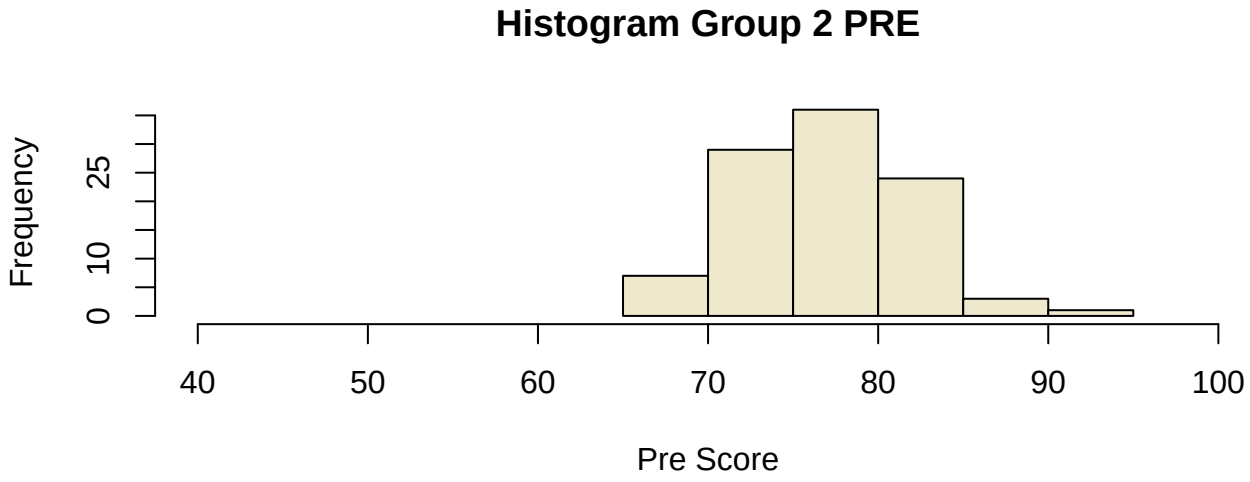
To measure the impact of a method we also need to look at the baseline.





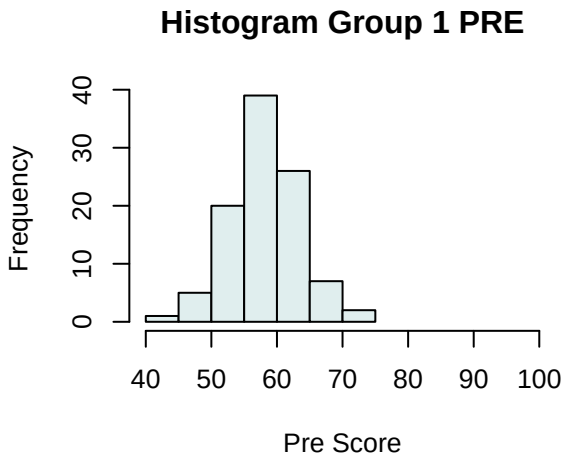
It looks as if there is not much difference between the two but the plot is quite misleading! The fact that the x-axis scales are different gives a false impression of similar distributions. This is how people "lie" with Statistics! We need to use the same scale in the x-axis (the values) in both histograms to make a fair comparison.

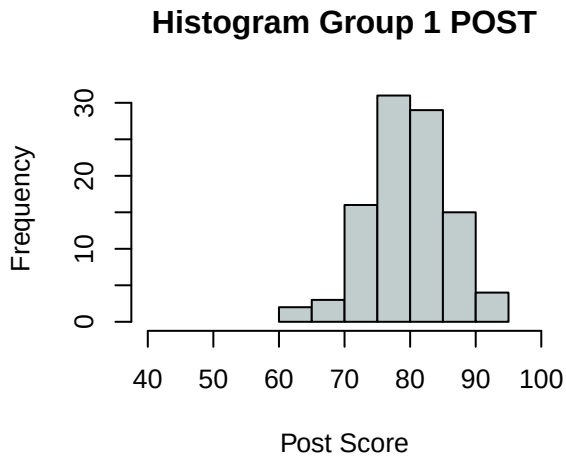


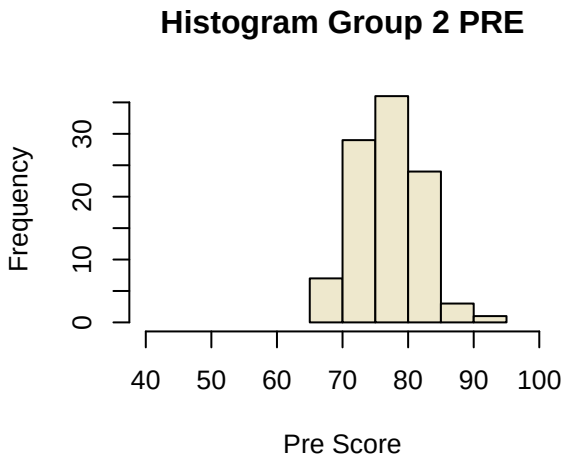


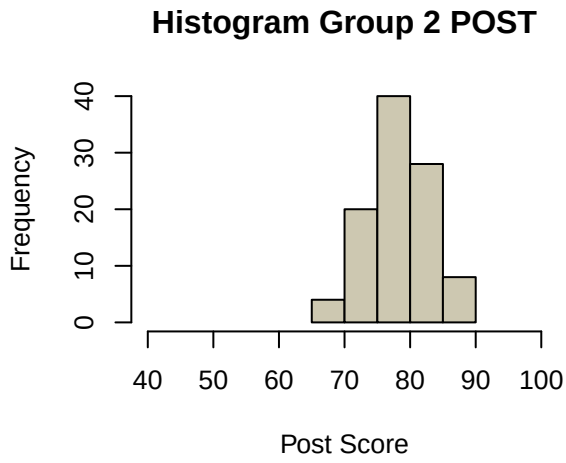
Now we see clearly that the starting scores for group 2 were much higher.

The next two figures combine all histograms in a single graph. We do this in two different ways. In the first, rows correspond to groups while in the second, columns correspond to groups.

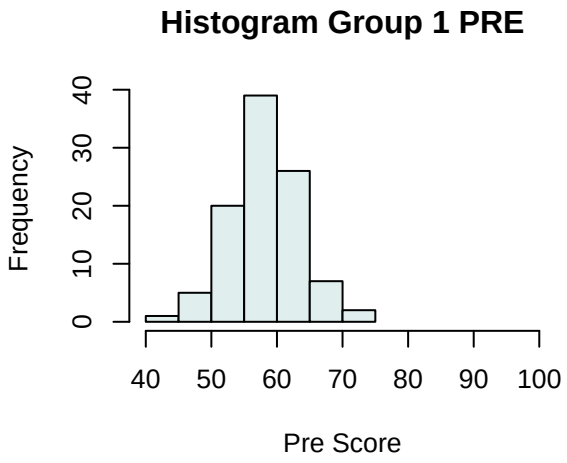


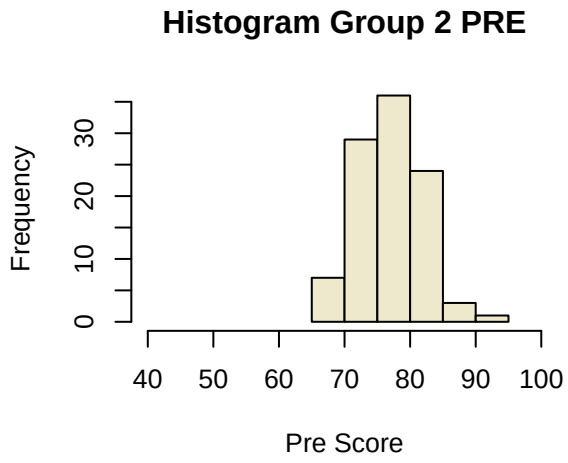


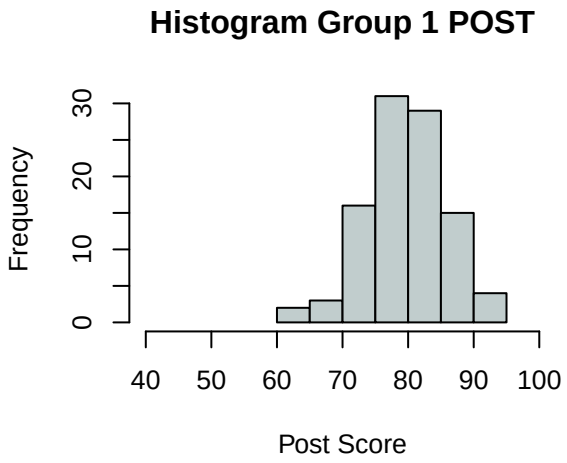


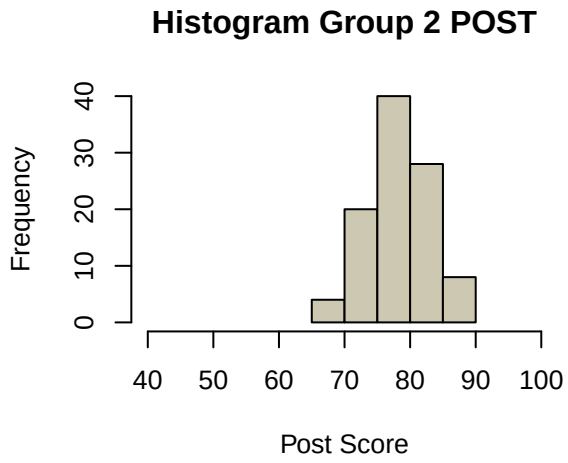


Another perspective: Groups as columns.





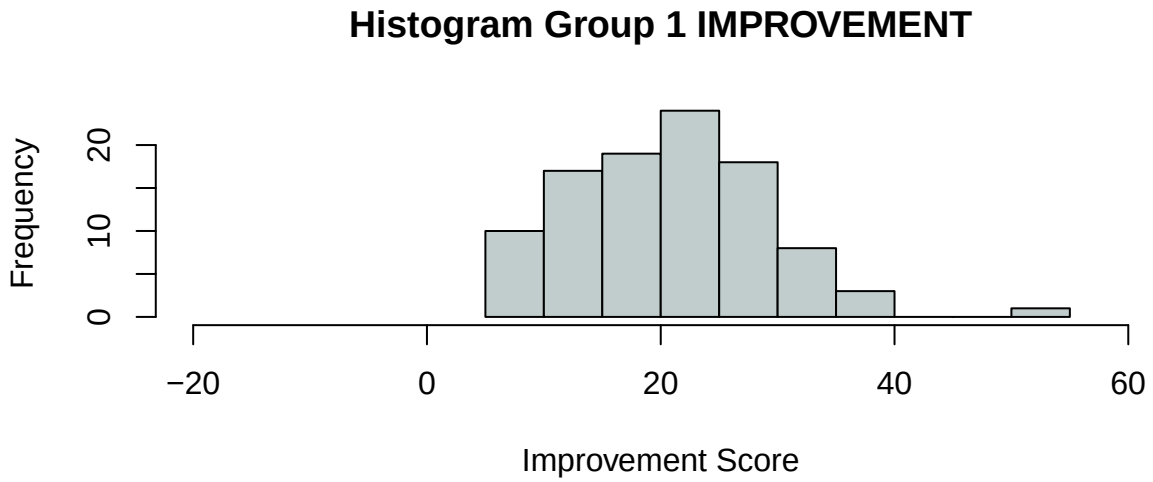


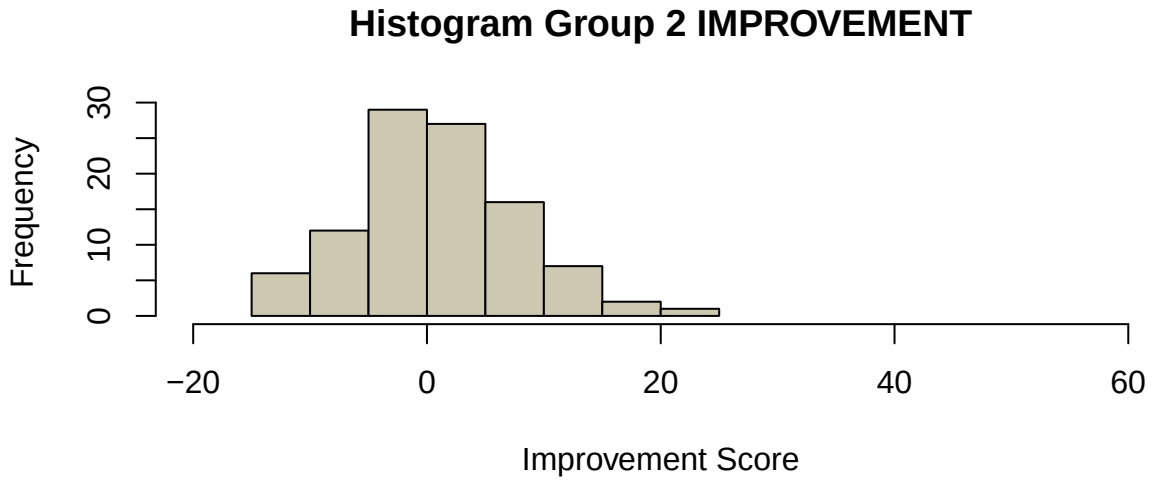


The graphs show clearly that using the post-score to measure the impact of the methods is misleading. The end result is similar but the first group had lower initial scores, so group 1 has improved more.

We collect the improvement score, which is the difference between post and pre scores, in the vectors `imp1` and `imp2`.

```
imp1 = group1.post - group1.pre
imp2 = group2.post - group2.pre
```





These graphs tell a different story. The improvement scores for group 1 are all positive while for group 2 we have both positive and negative values.

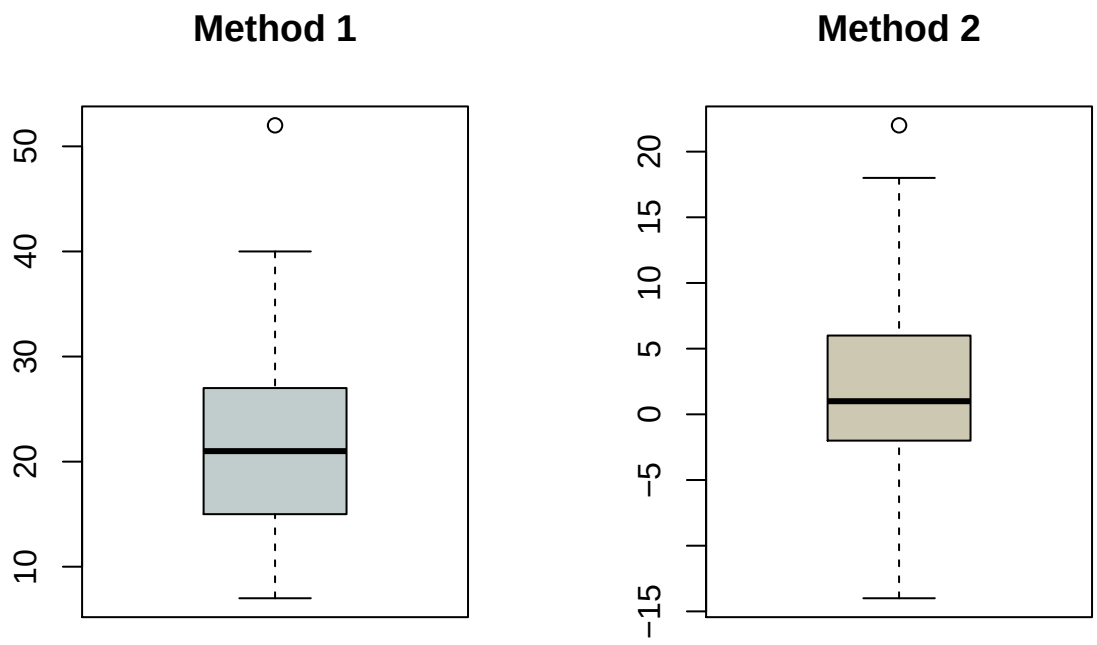
2.2.2 Summary statistics

```
summary(imp1)
summary(imp2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.00   15.00   21.00   21.35   27.00   52.00
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -14.00   -2.00    1.00    1.61    6.00   22.00
```

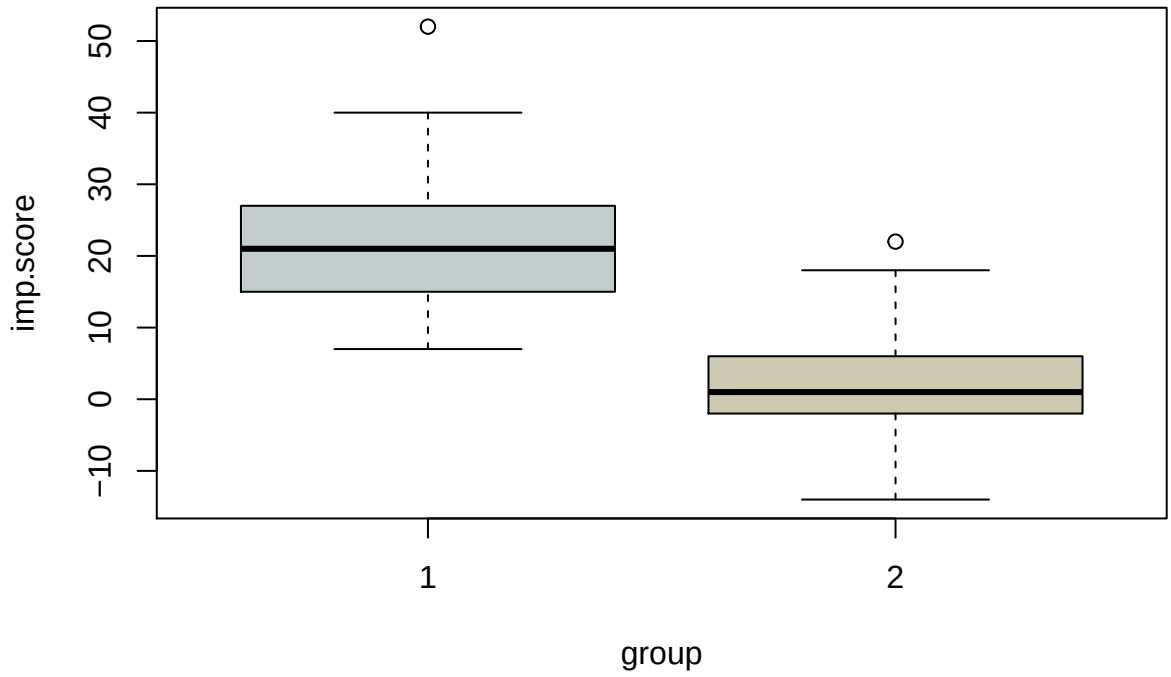
The summary statistics confirm the impresion we got from the graphs. Both median and mean for group 1 have considerably higher values.

Let's look at another graphical representation for the improvement scores, using boxplots



This is also a misleading graph! the scales on the y-axiz are different and the graphs suggest that the distribution are similar. To avoid this we will draw both boxplots together

```
improvement.df <- data.frame(imp.score = c(imp1, imp2),
                             group = c(rep(1,length(imp1)),rep(2,length(imp2))))
boxplot(imp.score ~ group, data = improvement.df, col = c('azure3','cornsilk3'))
```



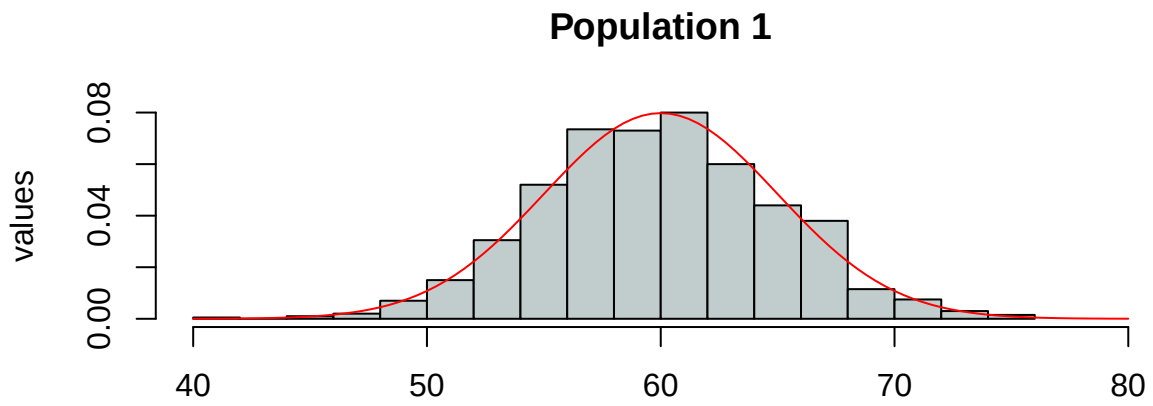
2.3 Example 3: Understanding uncertainty in statistical results arising from random sampling

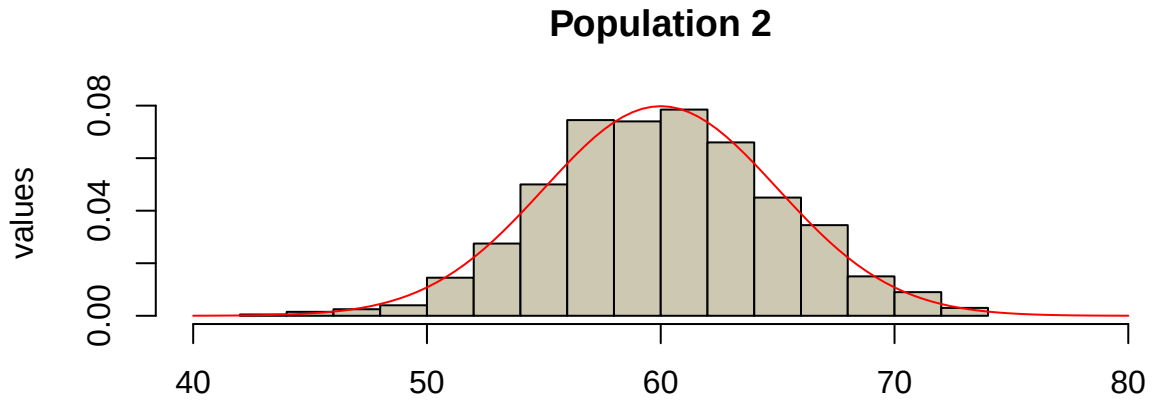
We consider a hypothetical population that has a normal distribution with parameters mean `mu = 60` and standard deviation `sd = 5`.

In real life, we hardly ever have access the whole population. We can only sample the population, i.e., we observe a random subset of the population, which is usually small. In this example we look at the effect the sample size on the variability of the sample.

If the sample is large, we expect that its statistical characteristics will be close to those of the population. As an example let's look at two samples of size 1000 and plot their histograms.

```
mu = 60; sd = 5; n1 = n2 = 1000
full1 = rnorm(n1, mu, sd)
full2 = rnorm(n2, mu, sd)
```





We see that the two samples have very similar histograms and are both close to the population distribution. Let's also calculate the summary statistics, this time using the function `describe` from the package `psych`:

```
library(psych)
describe(full1)
describe(full2)
```

```
##    vars    n  mean sd median trimmed  mad   min   max range skew
## X1    1 1000 59.89  5  59.81   59.87 5.05 41.47 74.85 33.38 0.05
##    kurtosis   se
## X1    -0.04 0.16
##    vars    n  mean   sd median trimmed  mad   min   max range skew
## X1    1 1000 60.09 4.88  60.02   60.07 4.98 42.92 73.77 30.85    0
##    kurtosis   se
```

```
## X1    -0.03 0.15
```

All the parameters calculated by the function are very similar for the two samples. However, these are very large samples and smaller ones will show more variability. To explore how sample variability depends on size, let's look at samples of size 5, 50 and 500, and focus on the difference of the means.

```
n1 = n2 = 5
samp1 = rnorm(n1, mu, sd)
samp2 = rnorm(n2, mu, sd)
mean(samp1)
```

```
## [1] 60.24559
```

```
mean(samp2)
```

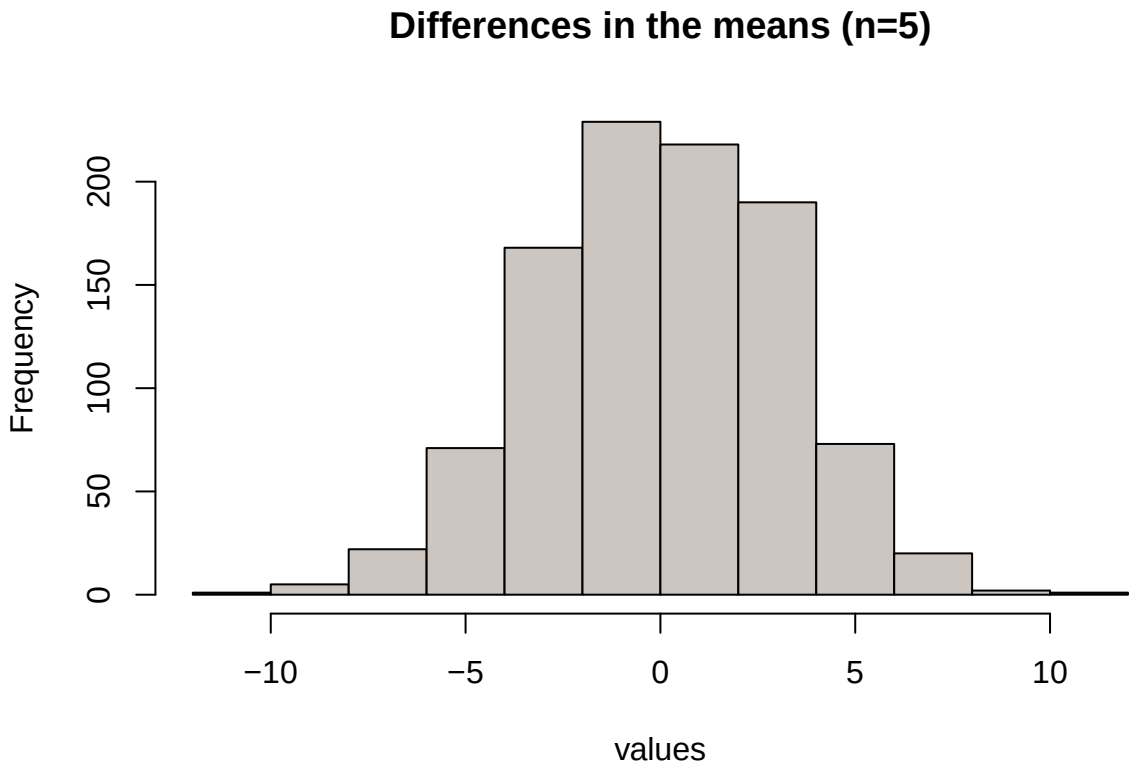
```
## [1] 62.65253
```

```
mean(samp1) - mean(samp2)
```

```
## [1] -2.406947
```

To illustrate the sample behavior of the difference in the sample means we will repeat this simulation 1,000 times and plot the results.

```
B = 1000
sim.norm5 <- rnorm(B*10,mu,sd)  # Simulate all the samples
sim.mat5 <- matrix(sim.norm5,ncol = 5) # Store in matrix. Each row is a sample
sim.mean5 <- apply(sim.mat5, 1, mean) # Calculate mean for each row
sim.diff5 <- sim.mean5[1:1000] - sim.mean5[1001:2000] # Calculate differences
hist(sim.diff5, col='seashell3', main='Differences in the means (n=5)', xlab='values', breaks=10)
```



Let's get some summary statistics for the 1,000 simulations

```
describe(sim.diff5)
```

```
##    vars    n mean   sd median trimmed  mad    min   max range  skew
## X1    1 1000 0.01 3.12   0.04    0.04 3.28 -11.08 10.05 21.13 -0.09
##    kurtosis  se
## X1    -0.06 0.1
```

We see that the mean and median are close to 0, so, on average, the two sample means are close, but there are very large values since the minimum value is -11.08 and the maximum is 10.05. The 10 and 90 percentile points are

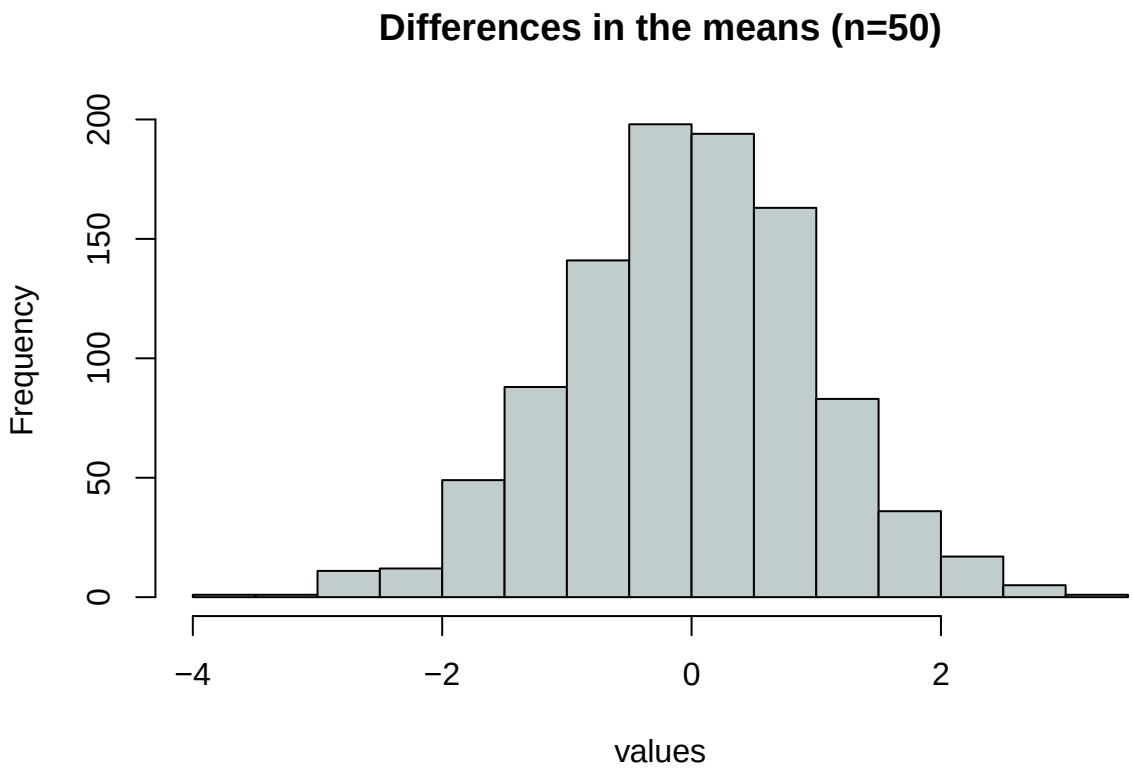
```
 sort(sim.diff5)[c(100,900)]
```

```
## [1] -3.989426  3.940958
```

This means that 20% of the samples give differences that are above 3.94 in absolute value.

To see how this changes with sample size, we repeat the simulation with samples of size 50 and 500.

```
sim.norm50 <- rnorm(100000,mu,sd)  # Simulate all the samples
sim.mat50 <- matrix(sim.norm50,ncol = 50) # Store in matrix. Each row is a sample
sim.mean50 <- apply(sim.mat50, 1, mean) # Calculate mean for each row
sim.diff50 <- sim.mean50[1:1000] - sim.mean50[1001:2000] # Calculate differences
hist(sim.diff50, col='azure3', main='Differences in the means (n=50)', xlab='values', breaks=10)
```



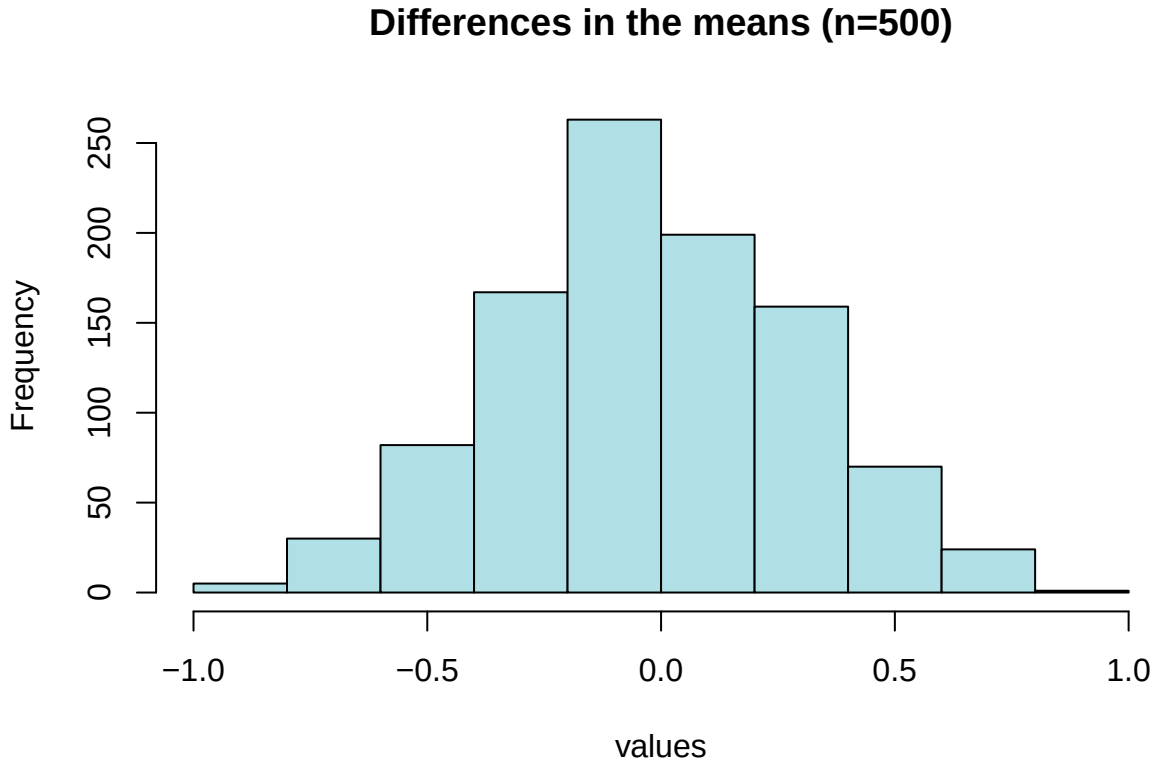

```
 sort(sim.diff50)[c(100,900)]
```

```
## [1] -1.328865  1.197667
```

The percentiles are now smaller, less than half of what they were before. If we increase the sample size to 500 the histogram is

```
sim.norm500 <- rnorm(1000000,mu,sd)
sim.mat500 <- matrix(sim.norm500,ncol = 500)
```

```
sim.mean500 <- apply(sim.mat500, 1, mean)
sim.diff500 <- sim.mean500[1:1000] - sim.mean500[1001:2000]
hist(sim.diff500, col='powderblue', main='Differences in the means (n=500)', xlab='values', breaks=10)
```

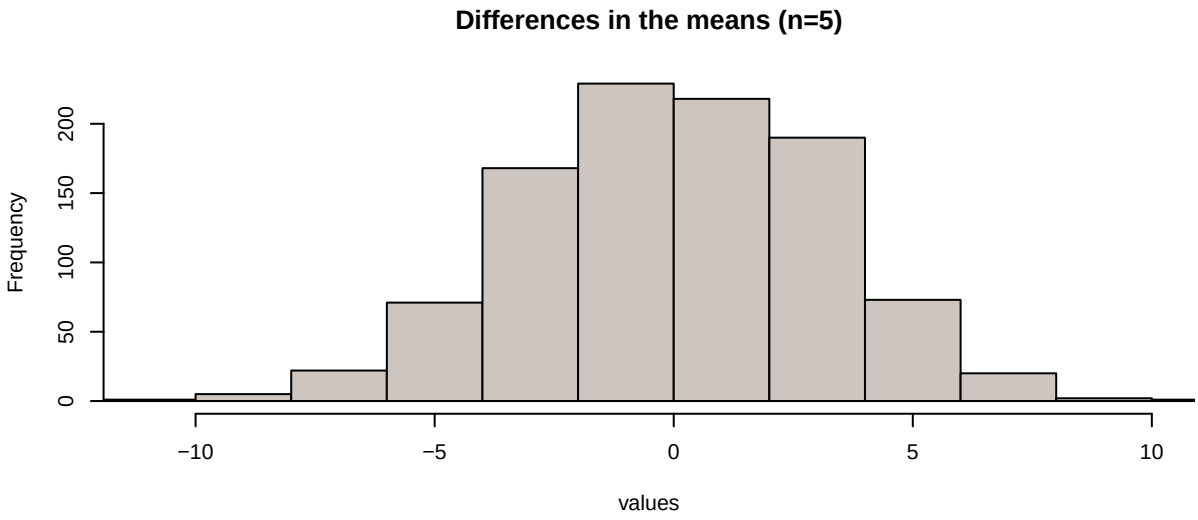


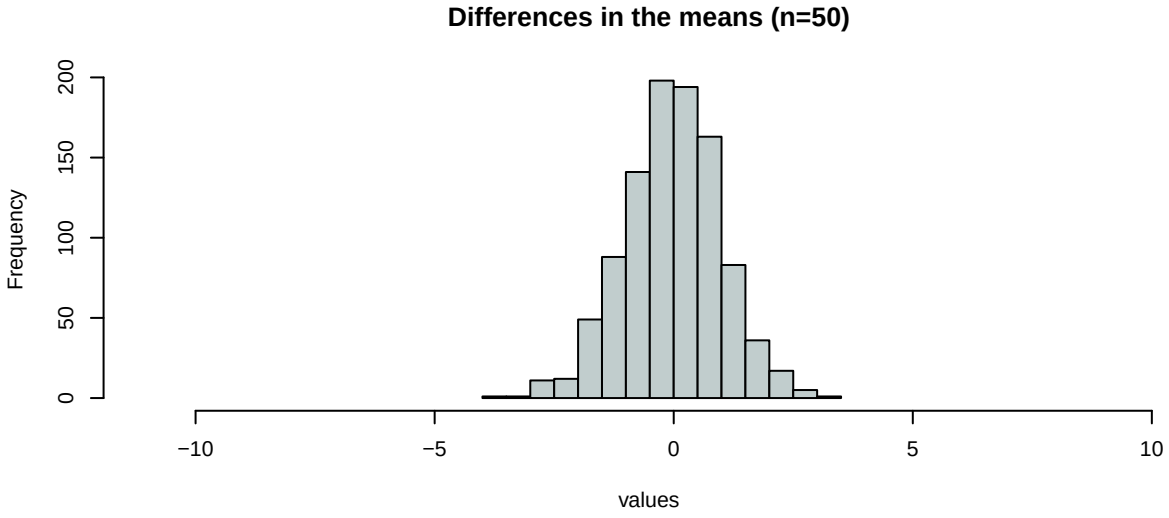
and the percentiles are a fraction of what they were before

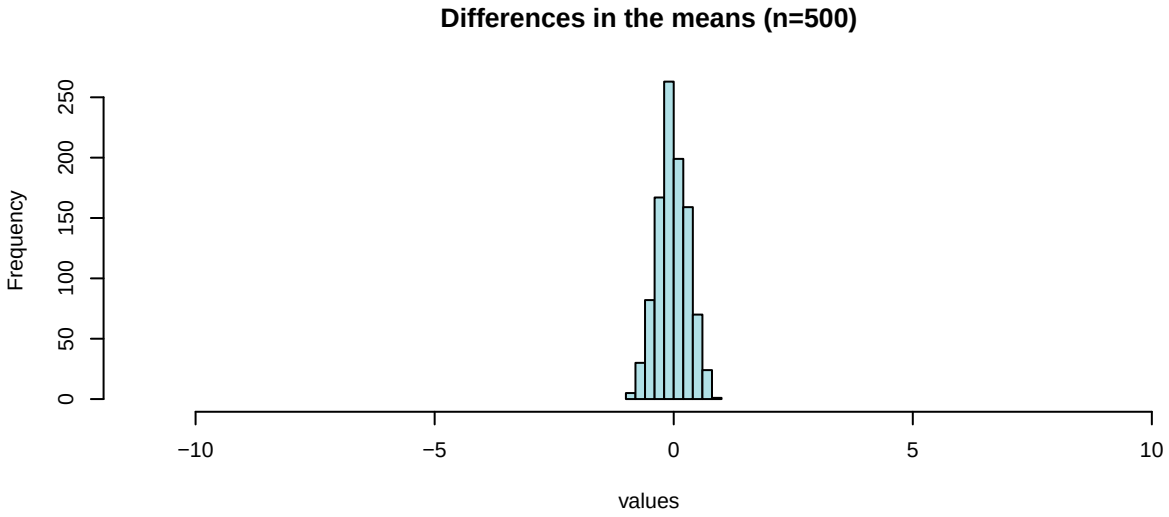
```
sort(sim.diff500)[c(100,900)]
```

```
## [1] -0.4184228  0.3917511
```

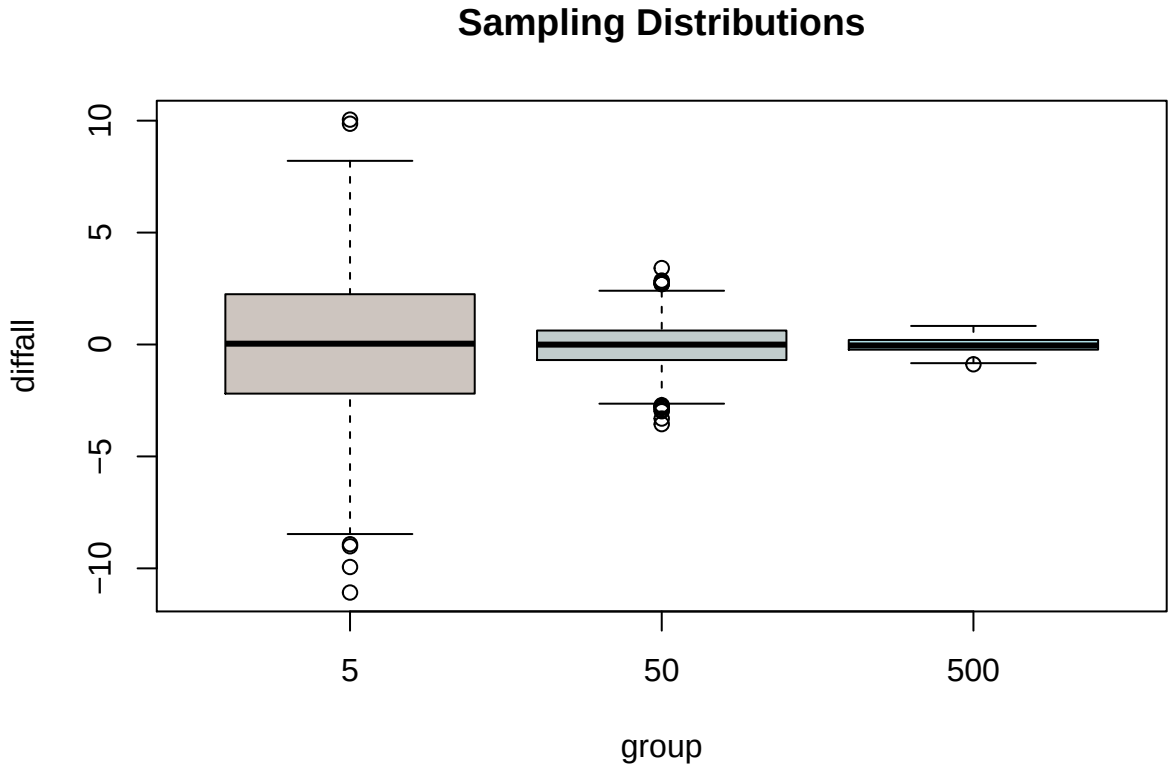
The next two figures show the three simulation results together to facilitate comparison of the results.


Differences in the means (n=5)
Frequency
0
50
100
150
200
−10
−5
0
5
10
values


Differences in the means (n=50)
Frequency
0
50
100
150
200
−10
−5
0
5
10
values


Differences in the means (n=500)
Frequency
0
50
100
150
200
250
−10
−5
0
5
10
values

```r
diffall = c(sim.diff5, sim.diff50, sim.diff500)
group = c(rep(5, B), rep(50,B), rep(500,B))
boxplot(diffall~group, col=c('seashell3','azure3','powderblue'), main="Sampling Distributions")
```



We see from the previous graphs that the dispersion strongly diminshes as the samples size grows.

2.4 Sampling distribution for the difference of the means

We have seen that if $X \sim N(\mu_1, \sigma_1^2)$, $Y \sim N(\mu_2, \sigma_2^2)$, and they are independent random variables then their sum

$$X + Y \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

When it comes to their differences we have that

$$X - Y \sim N(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2)$$

and the variances are added even if the variables are subtracted. We can use this in the analysis of the previous simulations. The samples we simulated come from a $N(60, 25)$ distribution. If the sample size is n we know the sample mean $\bar{X}_n$ has $N(60, 25/\sqrt{n})$ distribution. In our case we have two samples that come from the same distribution, let's denote by $\bar{X}_n^1$ and $\bar{X}_n^2$ the corresponding sample means, then their difference

$$\bar{X}_n^1 - \bar{X}_n^2 \sim N\Big(60 - 60, \frac{25}{n_1} + \frac{25}{n_2}\Big) = N\Big(0, 25\Big(\frac{1}{n_1} + \frac{1}{n_2}\Big)\Big)$$

where n_1 and n_2 are the sample sizes. In the example both samples have the same size n so we have

$$\bar{X}_n^1 - \bar{X}_n^2 \sim N\big(0, \frac{50}{n}\big)$$

with $n = 5, 50$ or 500. This formula says that the variance decreases in inverse proportion to the sample size. This is roughly what we observe in the variances we calculate below: Each value is roughly equal to ten times the next.

```r
var(sim.diff5);var(sim.diff50);var(sim.diff500)
```

```
## [1] 9.719822
```

```
## [1] 1.007213
```

```
## [1] 0.09814293
```

3 Comparing two population means

The problem we want to consider now is as follows. We have two populations and we want to compare certain characteristic of these populations, which are represented by two (possibly different) distributions. As examples consider the weight of fish that have been fed different diets, the yield of plants with different fertilizers, the lifetime of car tyres made with the same compound but with different threads or the improvement scores for two different methods for enhancing reading comprehension.

To compare the populations we take independent samples from each one. We denote by $x_1, x_2, \ldots, x_{n_1}$ the sample from population 1 and $y_1, y_2, \ldots, y_{n_2}$ the sample from population 2. We will focus on comparing the means of the two populations using the sample means.

Assume the populations have normal distributions $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$ and we want to test the null hypothesis

$$H_0 : \mu_1 = \mu_2$$

Let $\bar{x}$ and $\bar{y}$ be the sample means for each sample and s_x^2 and s_y^2 be the corresponding variances.

Recall that the *standard error* for the sample mean is the standard deviation of the sampling distribution for the mean, given by $\sigma_1/\sqrt{n_1}$ for the first population and similarly for the second. These unknown values are estimated by

$$\frac{s_i}{\sqrt{n_i}}$$

for $i = 1, 2$, where

$$s_1^2 = \frac{1}{n_1 - 1} \sum_1^{n_1} (x_i - \bar{x})^2$$

and similarly for s_2^2. The test statistic in this case is the difference between the sample means divided by the standard error of the difference of the means:

$$\frac{\bar{x} - \bar{y}}{sedm}$$

The standard error of the difference of the means is calculated by using the fact that for two independent random variables, the variance of their difference is the sum of the variances. Let X and Y be two random variables, let's calculate the variance of their difference. Denote by μ_X and μ_Y their means.

$$\begin{aligned} Var(X - Y) &= E[(X - Y - (\mu_X - \mu_Y))^2] \\ &= E[((X - \mu_X) - (Y - \mu_Y))^2] \\ &= E[(X - \mu_X)^2 + (Y - \mu_Y)^2 - 2(X - \mu_X)(Y - \mu_Y)] \\ &= Var(X) + Var(Y) - 2E[(X - \mu_X)(Y - \mu_Y)] \end{aligned} \tag{1}$$

The expected value in the last term is known as the covariance between X and Y:

$$Cov(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$$

If the variables are independent, the covariance is zero and

$$Var(X - Y) = Var(X) + Var(Y)$$

Therefore

$$sedm = \frac{s_1}{\sqrt{n_1}} + \frac{s_2}{\sqrt{n_2}}$$

3.1 Case 1: Equal variances

If we assume that the two variances are equal, which corresponds to the 'classical' test, the standard error is calculated using a 'pooled' estimator for the variance based on the standard deviations from the two groups. The pooled estimator s_p^2 is given by

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

The test is implemented in `R` with the function `t.test()` as in the one sample case. The option `var.equal` must be set to `TRUE`. Let's use this test on the fish diet data.

```
t.test(fish1,fish2, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  fish1 and fish2
## t = 0.37877, df = 68, p-value = 0.706
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.489833  3.656499
## sample estimates:
## mean of x mean of y
##  49.63333  49.05000
```

The output indicates that we are doing a two sample t-test with data sets `fish1` and `fish2`. The t statistic is 0.37877 with 68 degrees of freedom, giving a p-value of 0.706. This is too large to reject the null hypothesis of equal means. The 95% confidence interval for the difference between the means includes 0, which means that a value of zero for the difference is plausible at this confidence level. Finally, the means for each sample are presented.

Let us compare now the comprehension score for the two methods we considered in a previous section. In thi ocassion we will use the function `with`. The first input for this function is the data set we will using for the calculation we want to make. This avoids having to use the '$' notation to specify the variables within the data set.

```
with(improvement.df,t.test(imp.score ~ group, var.equal = TRUE))
```

```
##
##  Two Sample t-test
##
## data:  imp.score by group
## t = 18.65, df = 198, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  17.65277 21.82723
## sample estimates:
## mean in group 1 mean in group 2
##           21.35            1.61
```

In this case the p-value is very small, 2.2×10^{-16} and we have evidence to reject the null hypothesis of equal means.

3.2 Case 2: Unequal variances

The classical t-test requires equal variances but there is an extension due to Welch to the case of unequal variances. In this case variances are estimated separately for each sample. The distribution for the statistic is

not the t distribution but it may be approximated by a t distribution with a number of degrees of freedom that is calculated from the sample standard deviations and the group sizes. This parameter is usually not an integer. This is the default test in `t.test()`.

```
t.test(fish1,fish2)
```

```
##
##  Welch Two Sample t-test
##
## data:  fish1 and fish2
## t = 0.34458, df = 38.246, p-value = 0.7323
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.843034  4.009701
## sample estimates:
## mean of x mean of y
##  49.63333  49.05000
```

The result is usually not very different unless the group sizes and standard deviations are very different.

3.3 Comparison of Variances

One may also wish to compare variances from different samples. The test for this is known as Fisher's F test and is based on the quotient between the two variances, with the restriction that the bigger variance must be on top. The distribution for this quotient is known as Fisher's F distribution. It has two parameters which correspond to the degrees of freedom of the numerator and the denominator.

```
c(var(fish1), var(fish2))
```

```
## [1] 74.17126 15.74103
```

```
(F.ratio <- var(fish1)/var(fish2))
```

```
##          [,1]
## [1,] 4.711972
```

So the group 2 variance is about five times the variance for the first group. Now we calculate the probability of getting a value at least as big as this one and multiply this by 2.

```
2*(1-pf(F.ratio, length(fish1)-1, length(fish2)-1))
```

```
##              [,1]
## [1,] 1.033866e-05
```

The result is the p-value for the test. The function `var.test` carries out this test in `R`, as shown below.

```
var.test(fish1,fish2)
```

```
##
##  F test to compare two variances
##
## data:  fish1 and fish2
## F = 4.712, num df = 29, denom df = 39, p-value = 1.034e-05
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  2.401777 9.577864
## sample estimates:
## ratio of variances
##           4.711972
```

3.4 Wilcoxon's test

This non-parametric test is useful when the assumption of normality is not justified. The test statistic W is calculated as follows:

- Mix the two samples and order them.
- Calculate the ranks for the elements of each sample and sum the ranks for one of them.

If both samples come from the same distribution, the order of the elements of the samples is simply a random permutation of all possible orders and therefore the ranks are chosen at random. The problem is thus reduced to sampling n_1 values from the numbers 1 to $n_1 + n_2$.

As an exsample, let us use this test for the fish example.

```
wilcox.test(fish1,fish2)
```

```
## Warning in wilcox.test.default(fish1, fish2): cannot compute exact p-
## value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  fish1 and fish2
## W = 561.5, p-value = 0.6512
## alternative hypothesis: true location shift is not equal to 0
```

3.5 Tests on paired samples.

In the second example we considered results for a reading test before and after the subjects go through a reading comprehension improvement program. Instead of comparing the two methods, let us focus on one of the groups, the group corresponding to method 2, for instance. We would like to know if the treatment (method) had some effect on the outcome (reading comprehension). This is an example of paired data: we have scores before and after a treatment for the same subjects. In this case the two samples are clearly not independent: the score after treatment will be related to the score before treatment, since we are talking about the same person.

Analyzing this data as if the samples were independent could lead to serious errors. Remember that we showed that

$$Var(X - Y) = Var(X) + Var(Y) - 2Cov(X, Y)$$

If the covariance between X and Y is positive, by the equation above the variance of the difference will be smaller than the sum of the individual variances. This is an advantage worth exploiting because it makes easier to detect significant differences between the means.

In `R`, the same `t.test` function with the option `paired` set to `TRUE` will do the test for paired data. We present below the usual t-test and the paired t-test for comparison.

```
t.test(group2.pre, group2.post)
```

```
##
##  Welch Two Sample t-test
##
## data:  group2.pre and group2.post
## t = -2.3618, df = 196.87, p-value = 0.01916
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.954324 -0.265676
## sample estimates:
## mean of x mean of y
```

```
##     77.43     79.04
```

```
t.test(group2.pre, group2.post,paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  group2.pre and group2.post
## t = -2.4167, df = 99, p-value = 0.01749
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.9318775 -0.2881225
## sample estimates:
## mean of the differences
##                   -1.61
```

```
cor(group2.pre,group2.post)
```

```
## [1] 0.04503046
```

3.6 A more extreme example

This example comes from Box, Hunter & Hunter, *Statistics for Experimenters*. In the experiment two types of material for shoe soles are compared, A and B. These were randomly assigned to the left and right shoe of 10 boys, so that each boy had one shoe with each material. The data are stored in the file `shoes` in package `MASS`.

```
library(MASS)
shoes
```

```
## $A
##  [1] 13.2  8.2 10.9 14.3 10.7  6.6  9.5 10.8  8.8 13.3
##
## $B
##  [1] 14.0  8.8 11.2 14.2 11.8  6.4  9.8 11.3  9.3 13.6
```

For illustration purposes, let's do a standard t test without pairing. We will consider the two alternatives available, equal and unequal variances.

```
with(shoes,t.test(A, B, var.equal = TRUE))
```

```
##
##  Two Sample t-test
##
## data:  A and B
## t = -0.36891, df = 18, p-value = 0.7165
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.744924  1.924924
## sample estimates:
## mean of x mean of y
##     10.63     11.04
```

```
with(shoes,t.test(A, B))
```

```
##
##  Welch Two Sample t-test
##
```

```
## data:  A and B
## t = -0.36891, df = 17.987, p-value = 0.7165
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.745046  1.925046
## sample estimates:
## mean of x mean of y
##     10.63     11.04
```

Both tests give large p-values, so we cannot reject the null hypothesis, but we are not using the fact that the samples are correlated. If we do a paired test:

```
with(shoes,t.test(A, B, paired = TRUE))
```

```
##
##  Paired t-test
##
## data:  A and B
## t = -3.3489, df = 9, p-value = 0.008539
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.6869539 -0.1330461
## sample estimates:
## mean of the differences
##                   -0.41
```

```
with(shoes,cor(A,B))
```

```
## [1] 0.9882255
```

The p value is small and at the usual levels we would reject the null hypothesis of equality of the means. We see that there is a very large correlation in this case, that accounts for the difference in teh result.

3.7 The Wilcoxon test for matched-pairs.

There is also a non-parametric test for paired samples, which is the one-sample Wilcoxon signed-rank test applied to the differences.

```
with(shoes, wilcox.test(A, B, paired = TRUE))
```

```
## Warning in wilcox.test.default(A, B, paired = TRUE): cannot compute
## exact p-value with ties
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  A and B
## V = 3, p-value = 0.01431
## alternative hypothesis: true location shift is not equal to 0
```