Lecture 4: Continuous data

Matti Pirinen

14.8.2023

During the previous lectures we got familiar with basic statistics by using the binomial distribution that models the number of "successes" in a sample of n trials from a population where the success probability is p. It is a discrete distribution where the outcome variable can only take discrete values $0, 1, 2, \ldots, n$.

Now we move to outcome variables, such as BMI or LDL cholesterol levels, that can take any value in a continuous interval, such as, for example, in the interval (10.0, 150.0) for BMI measured in kg/m^2.

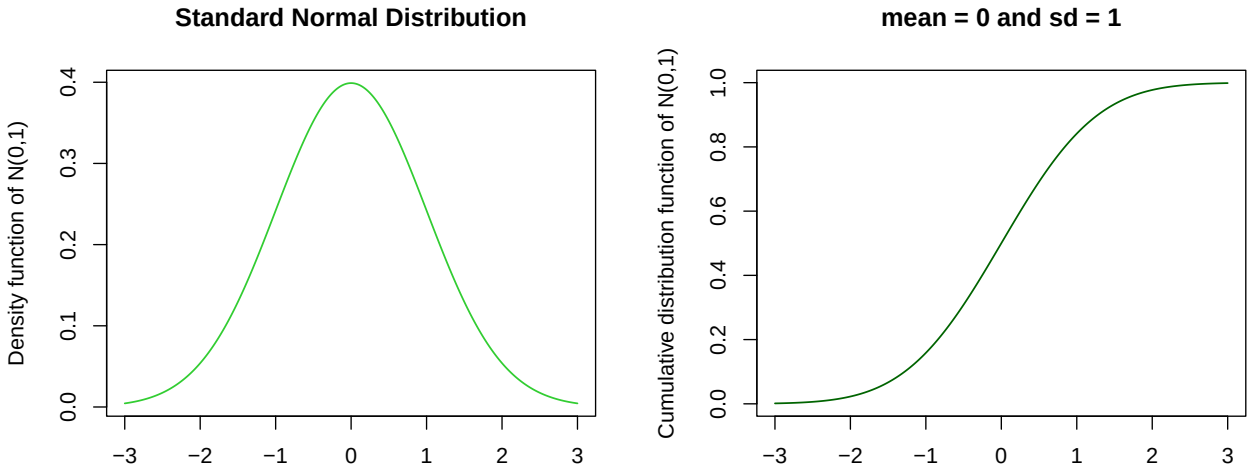
Normal Distribution

The most prevalent continuous distribution is the Normal distribution (also called the Gaussian distribution after German mathematician Carl F. Gauss, 1777-1855). A reason for its wide applicability is that, in numerous settings, the distribution of sum of independent random variables tends towards a Normal distribution. (This deep mathematical fact called the Central Limit Theorem will be demonstrated a bit at the end of the lecture.) A consequence is that complex properties, such as variation between individuals in IQ or height, or susceptibility to coronary artery disease, that all result from an interplay between a huge number of individual factors, both from the genome and from the environment, follow approximately a Normal distribution in the population. Read more: http://www.bmj.com/content/310/6975/298.full.

Normal distribution is defined by 2 parameters: mean (mu, μ) and standard deviation (sigma, σ). Often (outside R) variance (sigma^2, σ^2) is used in place of standard deviation to define the second parameter. Always pay attention to which one is in question since mixing up numerical values of these parameters badly mixes up all the statistics! For now, remember that the basic R functions take in standard deviation `sigma`.

The standard normal distribution, N(0,1), has mean = 0 and sd = 1. Let's plot the density function `dnorm(,0, 1)` and the cumulative distribution function `pnorm(, 0, 1)` of N(0,1) next to each other. We use `par(mfrow = c(1,2))` to set the plotting parameter `mfrow` to split the plotting region into 1 row and 2 columns, and the two plots will be shown next to each other.

```
x = seq(-3, 3, 0.001) #range of x-values where we evaluate dnorm() and pnorm()
d = dnorm(x, 0, 1) #values of density function of N(0,1)
p = pnorm(x, 0, 1) #values of cumulative distribution function of N(0,1)
par(mfrow = c(1,2)) #plotting region split to 1 x 2 area to show two plots next to each other
plot(x, d, xlab = "", ylab = "Density function of N(0,1)",
     main = "Standard Normal Distribution",
     t = "l", lwd = 1.3, col = "limegreen")
plot(x, p, xlab = "", ylab = "Cumulative distribution function of N(0,1)",
     main = "mean = 0 and sd = 1",
     t = "l", lwd = 1.3, col = "darkgreen")
```



The density plot shows that the peak is at the mean of the distribution, and the density drops from there symmetrically making a bell-shaped curve characteristic to a Normal distribution.

The cumulative distribution plot shows how the probability mass accumulates as we move from small values (here starting from -3) to larger values (here up to 3). We know that 95% of the probability mass of the N(0,1) distribution is between values -1.96 and 1.96:

```
qnorm(c(0.025, 0.975), 0, 1) #95% of mass of N(0,1) is between these two points
```

```
## [1] -1.959964  1.959964
```

(This fact was actually the motivation to use 1.96 as the multiplier of the SE in Lecture 3 when we derived approximate 95% confidence intervals for a proportion parameter.)

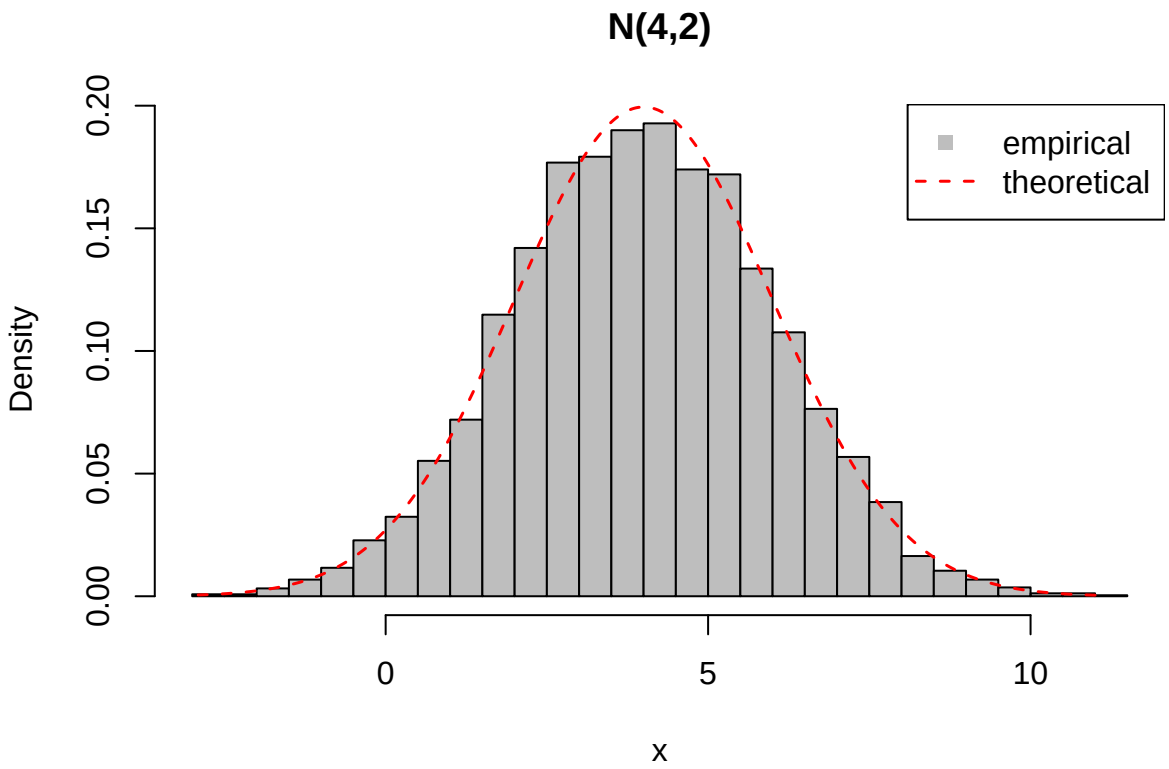
Let's generate samples from a Normal distribution, say with mean = 4 and sd = 2, using `rnorm()` and plot the data using a histogram. Standard `hist()` shows on y-axis the counts of observations in each bin, but by setting `prob = TRUE` we can make the y-axis to scale to the values of a density function (making the total area of the histogram = 1). Then we can also show the theoretical density function `dnorm(,4,2)` in the same plot and compare the two.

```
n.samples = 5000 #samples from distribution
mu = 4 #mean
sigma = 2 #standard deviation

x = rnorm(n.samples, mu, sigma) #random sample from normal distribution
c(mean = mean(x), sd = sd(x)) #show empirical mean and sd of data
```

```
##     mean       sd
## 3.985483 1.994015
```

```
hist(x, breaks = 40, col = "gray", prob = TRUE, main = "N(4,2)")
x.grid = seq(min(x), max(x), length.out = 1000) #grid of x-values to evaluate dnorm(, mu, sigma)
lines(x.grid, dnorm(x.grid, mu, sigma), col = "red", lwd = 1.5, lty = 2) #add dashed line to the curren
#Let's make a legend text to appear in the top right corner
#pch is plotting symbol (15 square; -1 no symbol); lty is line type (0 no line; 2 dashed line)
legend("topright", col = c("gray","red"), legend = c("empirical","theoretical"),
       lwd = 1.5, pch = c(15,-1), lty = c(0,2))
```

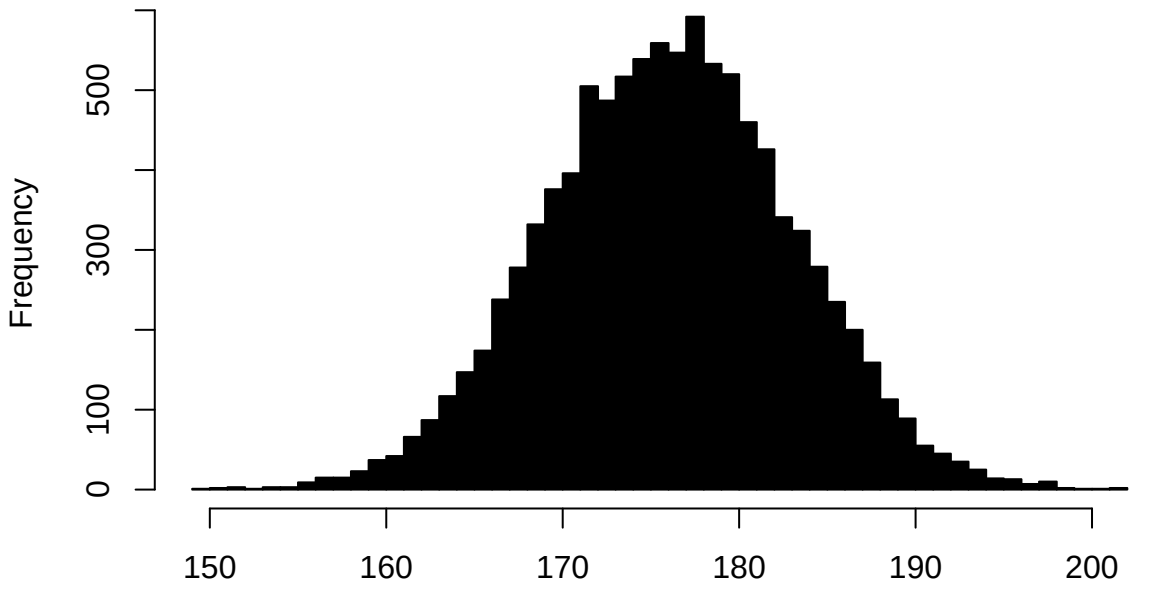


We see that the histogram of 5000 random samples from N(4,2) matches quite well with the theoretical density function of N(4,2), as it should. With more samples, the match would get even tighter.

Examples 4.1

(1) Distribution of height in Finnish men is Normal with mean = 176 and sd = 7 (in cm). Generate example data set of heights of $n = 10000$ Finnish men and plot a histogram of the data.

```
x = rnorm(10000, 176, 7)
hist(x, main = "", xlab = "", col = "black", breaks = 50)
```



(2) What should be the theoretical variance of these data? Compute empirical variance of simulated data in R.

Theoretical variance is sd squared, i.e., $7^2 = 49$. Let's validate:

```
c(theoretical = 7^2, empirical = var(x))
```

```
## theoretical   empirical
##    49.00000    48.88782
```

(3) Use `pnorm()` to estimate the expected proportion of Finnish men that are ≤ 165 cm. Compare to the empirical estimate of the same proportion in the simulated data set.

```
pnorm(165, 176, 7) #theoretical proportion of values of N(176, var = 7^2) that are <= 165
```

```
## [1] 0.05804157
```

```
mean(x <= 165) #empirical proportion of values in x that are <= 165
```

```
## [1] 0.0571
```

(4) How tall is the tallest 1% of Finnish men? Use `qnorm()` for theoretical estimate and `quantile()` for an empirical estimate from the simulated sample.

```
qnorm(0.99, 176, 7) #theoretical cut-point with 99% in the left tail so 1% in the right tail
```

```
## [1] 192.2844
```

```
quantile(x, 0.99) #empirical cut-point that leaves 99% of the values on the left side
```

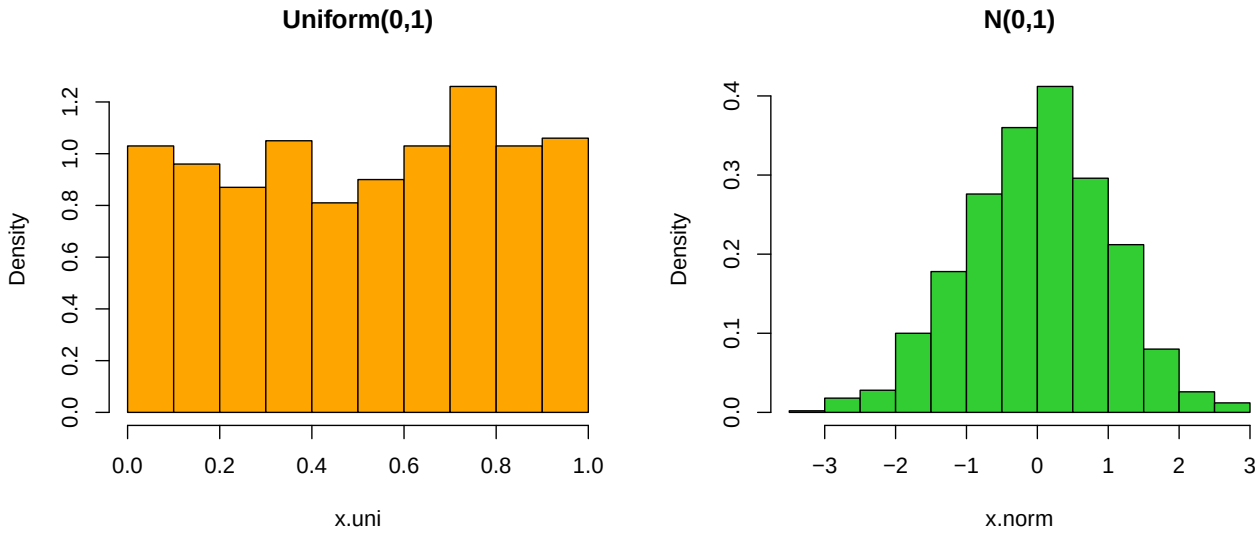
```
##      99%
## 192.2204
```

Assessing normality of data

Does my data set follow a Normal distribution?

Let's generate two data sets of size $n = 1000$. First from the uniform distribution in (0,1) and the second from N(0,1). Uniform distribution has "equal probability to pick any value in the interval", and therefore its density function is a flat line. Let's plot the two distributions using histograms.

```
n.samples = 1000
x.uni = runif(n.samples, 0, 1) #runif takes in left (0) and right (1) endpoints of interval
x.norm = rnorm(n.samples, 0, 1) #standard normal N(0,1)
par(mfrow = c(1,2))
#we can scale histogram to probability density by prob = TRUE
hist(x.uni, prob = TRUE, main = "Uniform(0,1)", col = "orange")
hist(x.norm, prob = TRUE, main = "N(0,1)", col = "limegreen")
```



We can see that the left-hand plot does not look like a "bell-shaped" Normal distribution, while the right-hand plot does look roughly as Normal.

Let's run **Shapiro-Wilk test** for normality for both samples. It returns a P-value under the null hypothesis that the data are Normally distributed.

```
shapiro.test(x.uni)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x.uni
## W = 0.94784, p-value < 2.2e-16
```

```
shapiro.test(x.norm)
```

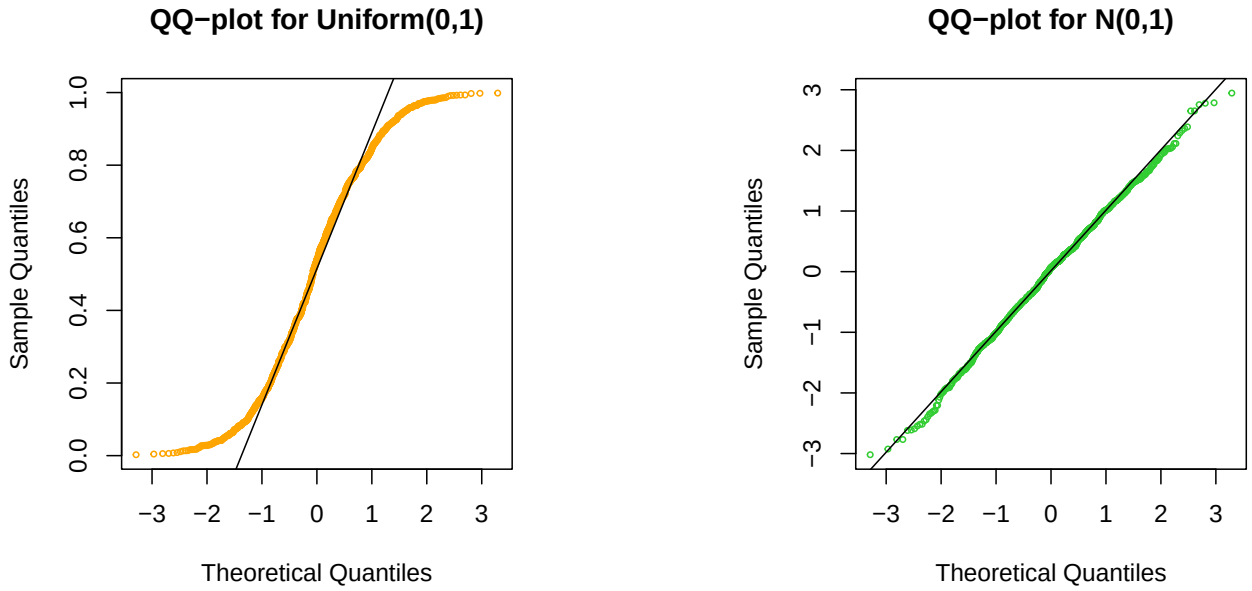
```
##
##  Shapiro-Wilk normality test
##
## data:  x.norm
## W = 0.99806, p-value = 0.3073
```

A low P-value for dataset `x.uni` means that it is unlikely to come from a Normal distribution, while a high P-value for `x.norm` does not indicate deviation from the Normal distribution.

The quantitative tests (like Shapiro-Wilk above) may have little discriminatory power for small samples (that is, they do not detect deviation from Normality with statistical significance) and too much for large samples (that is, they detect tiny deviations with statistical significance). It is always good to also visually inspect Normality of data since that can inform what kind of deviations from Normality we may have and whether these deviations might be important in practice. QQ-plot is a tool for that.

QQ-plot A visual checking for Normality of a data set is often done with a quantile-quantile plot (QQ-plot) using first `qqnorm( )` and then `qqline( )` to add a reference line to the plot, as below.

```r
par(pty = "s") #make plots exact squares
par(mfrow = c(1,2)) #split plotting area into 1 x 2 panels
qqnorm(x.uni, main = "QQ-plot for Uniform(0,1)", cex = 0.5, col = "orange")
qqline(x.uni) #adds line to the existing QQ-plot
qqnorm(x.norm, main = "QQ-plot for N(0,1)", cex = 0.5, col = "limegreen")
qqline(x.norm) #adds line to the existing QQ-plot
```



In QQ-plots, each data point from the observed data ("Sample Quantiles") is plotted against the corresponding Theoretical Quantile from the Normal distribution. That is, there is one point whose x-coordinate is the minimum from the Normal distribution and y-coordinate is the minimum of the observed values. Similarly, for every i, there is one point whose x-coordinate is the ith largest value from the theoretical Normal distribution (of a sample of this size), and y-coordinate is the ith largest value of the observed data. If the shapes of the two distributions (observed data and theoretical distribution) match each other, then these points appear approximately on a line. The reference line is put there by `qqline()` command. We see that on the right side the observed data seem to follow a Normal distribution because points are approximately on the line whereas on the left side the points are not on the line and hence the data are unlikely to come from a Normal distribution.

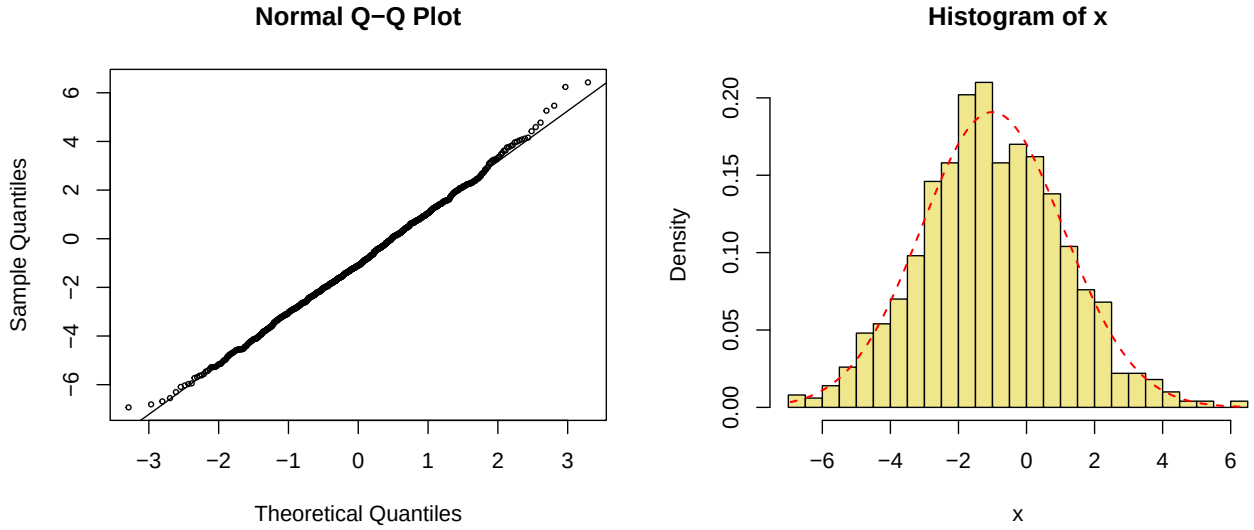
Example 4.2

(1) Generate $n = 1000$ values from N(mean = -1, var = 4). Compute Shapiro-Wilk test P-value to assess Normality of data. Make a QQ-plot to assess normality of data. Plot an empirical histogram of the data and add on top of it a density function of the Normal distribution with the same mean and standard deviation as in the observed data.

```r
x = rnorm(1000, -1, 2) #NOTE: var = 4 --> sd = sqrt(4) = 2
shapiro.test(x) #Look at the P-value
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.99833, p-value = 0.4502
```

```r
par(mfrow = c(1,2))
qqnorm(x, cex = 0.5)
qqline(x)
hist(x, col = "khaki", xlab = "x", breaks = 25, prob = TRUE)
x.seq = seq(min(x), max(x), length = 1000) #evaluate density function at these points
lines(x.seq, dnorm(x.seq, mean(x), sd(x)), col = "red", lwd = 1.5, lty = 2) #use empirical mean and sd
```



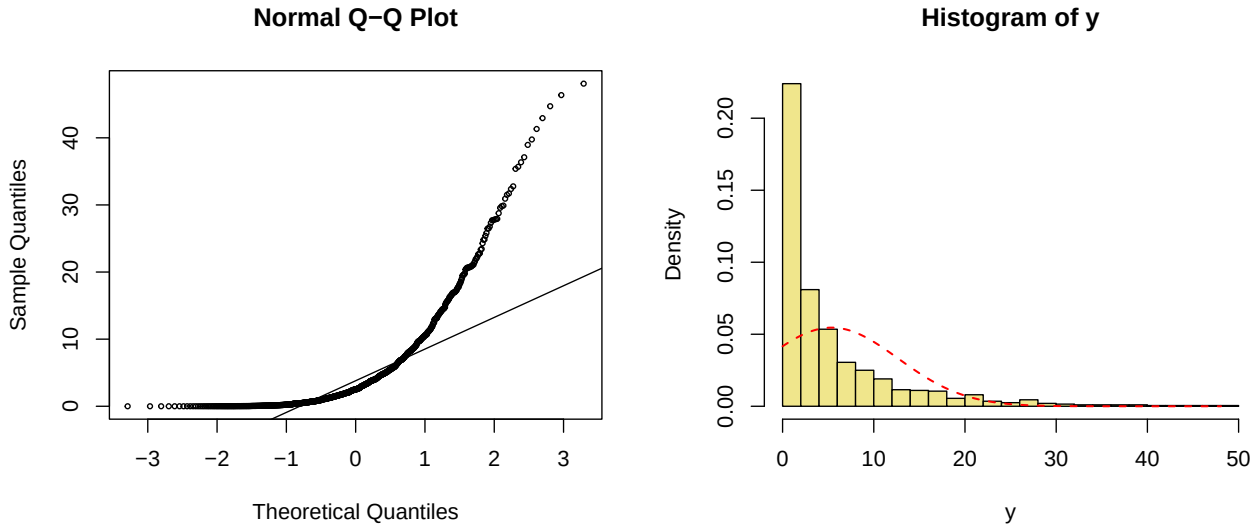
We see that `x` looks like Normally distributed (P-value is not small, QQ-plot is on the line, histogram looks Normal).

(2) Set $y = x^2$ where x is the data from part 1. Repeat the same analyses for these transformed data y.

```r
y = x^2
shapiro.test(y)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.71674, p-value < 2.2e-16
```

```r
par(mfrow = c(1,2))
qqnorm(y, cex = 0.5)
qqline(y)
hist(y, col = "khaki", xlab = "y", breaks = 25, prob = TRUE)
x.seq = seq(min(y), max(y), length = 1000) #evaluate density function at these points
lines(x.seq, dnorm(x.seq, mean(y), sd(y)), col = "red", lwd = 1.5, lty = 2) #use empirical mean and sd
```



We see that `y=x^2` is not at all Normally distributed (P-value is small, QQ-plot is not on the line, histogram does not look Normal).

t-tests

When data are Normally distributed, a set of tests called t-tests are available http://www.bmj.com/about-bmj/resources-readers/publications/statistics-square-one/7-t-tests. These are used to compare

- the mean of a sample to a specific value (**one-sample t-test**), or
- the means of two independent samples against each other (**two-sample t-test**), or
- the difference between paired samples under two conditions (**paired t-test**).

In the last two situations, it is important to make a difference between independent data sets and correlated data sets when choosing a t-test. Two samples are independent when each sampled individual from each population can be assumed chosen at random without any dependence on which other values have been sampled from the population. Contrast this to, say, situation where same set of individuals have each been measured two times, say, first before a treatment and second after the treatment. When we are interested in the treatment effect from such a design, we should treat the data as **paired**, each individual's two measurements forming a pair. In particular, the sets of measurements under the two conditions are not independent because there is likely to be factors that an individual possess under both conditions (such as body-mass, pain tolerance, genetics etc.) that can affect the measurement (e.g. how much drug is needed for an individual to feel painless). Thus we would need to use the paired t-test.

The t-tests analyze means of continuous measurements and can therefore be seen as continuous version of `prop.test( )` that we used previously to analyze proportions that are means of binary outcomes.

By using the function `t.test()` we not only do the test, but also, and often more importantly, get estimates for the means / differences between the means and their confidence intervals. Comparing the means using t-tests is quite robust to the distributional assumption and in large samples works also for non-normal data.

One-sample t-test: (Try `?t.test`.)

```
n.samples = 100
x = rnorm(n.samples, 3, 1)

#test whether mean is 2.5
t.test(x, mu = 2.5)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 6.1481, df = 99, p-value = 1.658e-08
## alternative hypothesis: true mean is not equal to 2.5
## 95 percent confidence interval:
##  2.913969 3.308503
## sample estimates:
## mean of x
##  3.111236
```

We can read from the output the (two-sided) P-value under the null hypothesis that the mean is 2.5, which here is very small (2e-8) indicating that the population mean is unlikely to be 2.5. We also have the empirical mean from the data and a 95%CI for the population mean. From this 95%CI, we see that 2.5 is quite far from it, which also says that the population mean is unlikely to be 2.5.

The t-test is based on the fact that if x_i for each $i = 1, ..., n$ comes from $N(\mu, \sigma^2)$, then the empirical mean $\overline{x} = \frac{1}{n}\sum_i x_i$ is distributed as $N(\mu, \frac{\sigma^2}{n})$, and thus the test statistic

$$z = \frac{\overline{x} - \mu}{(\sigma/\sqrt{n})} \text{ follows } N(0, 1),$$

and this can be used to derive P-values. Idea is that if the observed mean $\overline{x}$ is far from the hypothesized mean μ, then z-score has so large magnitude (either positive or negative) that it is a very unlikely observation under the standard Normal distribution, thus giving a small P-value under the hypothesis that the mean is μ.

However, usually we do not know the exact value of σ but first have to estimate it from the data using sample standard deviation

$$\widehat{\sigma} = \sqrt{\frac{\sum_i (x_i - \overline{x})^2}{n-1}},$$

then the test statistic

$$t = \frac{\overline{x} - \mu}{(\widehat{\sigma}/\sqrt{n})}$$

follows *t-distribution with* $n-1$ *degrees of freedom*. When $n > 30$ or so, this distribution is very close to the Standard Normal distribution but for smaller sample sizes there are some differences between the two. This test statistic and its distribution is where the t-test has got its name.

Let's test empirically that this is indeed the t-statistic reported by R above.

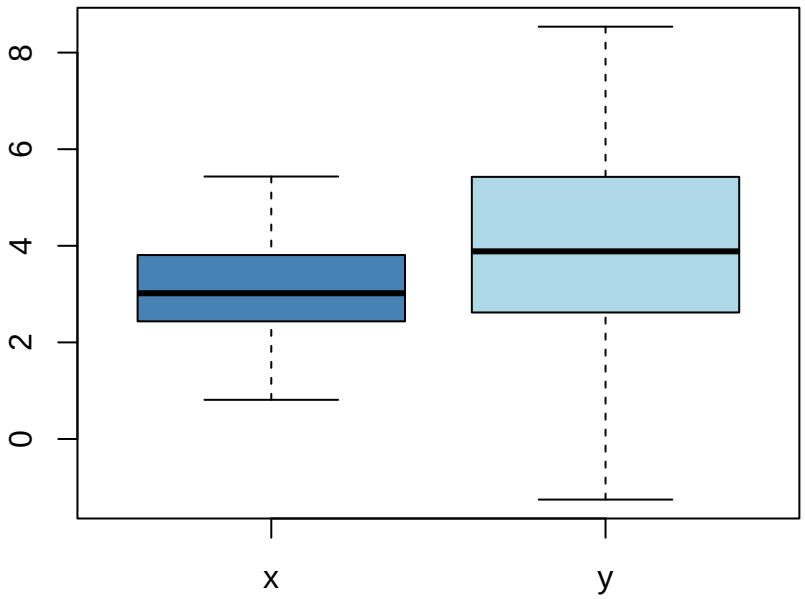
```
mu = 2.5
(mean(x) - mu)/(sd(x) / sqrt(n.samples)) #should be t-statistic given by t.test above
```

```
## [1] 6.148132
```

Two-sample t-test for independent data sets:

Let's then compare two independent Normal samples for their similarity in means. Let's generate two data sets `x` and `y` and display them with a **boxplot**.

```
n.samples = 100
x = rnorm(n.samples, 3, 1)
y = rnorm(n.samples, 4, 2)
boxplot(x, y, names=c("x","y"), col = c("steelblue","lightblue") )
```



What do you see in the boxplot? For each data set, the box shows the interquartile range, i.e., the middle 50% of the values, starting from the first quartile (25%) and extending to the 3rd quartile (75%). In the middle of the box, the median is marked with a black stripe. The lines outside box extend up to 1.5 times the interquartile range and the remaining outlier observations would be shown as individual points. Boxplot starts to be useful when there are more than about 20 observations. If there are less, `stripchart()`, that shows every observation, may be more informative than the boxplot.

```
#Do Welch t-test to test whether mean of x and y are equal:
t.test(x, y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -4.2017, df = 154.55, p-value = 4.465e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.3905990 -0.5011722
## sample estimates:
## mean of x mean of y
##  3.087692  4.033578
```

A smallish P-value says that the means are unlikely to be the same. Welch t-test can be used in cases where the variances of the two distributions need not be the same (like in our case here). The test statistic for Welch t-test is

$$w = \frac{\overline{x} - \overline{y}}{\sqrt{\frac{\widehat{\sigma}_x^2}{n_x} + \frac{\widehat{\sigma}_y^2}{n_y}}}$$

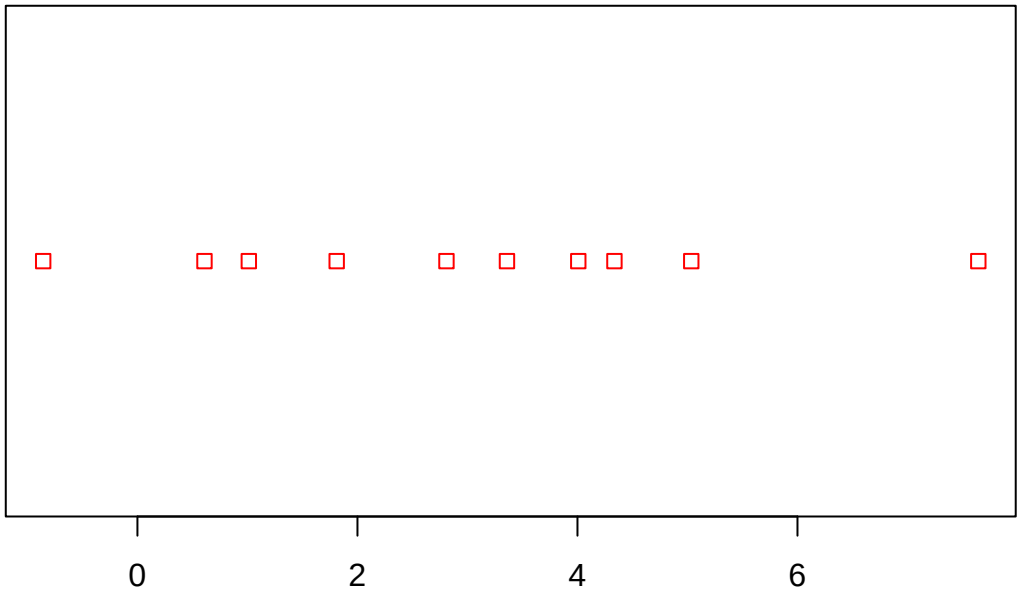
and it is compared to a t-distribution with certain degrees of freedom (not explained here).

In addition to Welch t-test, there is also a version of 2-sample t-test called **Student's t-test** which is specifically for cases where the variance is known to be the same between the groups. Since usually we do not know this, the Welch t-test should be our default option. Indeed, `t.test()` has its default parameter `var.equal = FALSE`, meaning that it uses Welch test by default. However, in cases where variances are the same, Student's t-test is more *powerful* than Welch test and that is why it is kept as an option in `t.test()`. (We will talk about what power means in the next lecture.)

Example 4.3

(1) Assume that a population mean for a particular biomarker is $\mu = 6$ and that the distribution of the biomarker in patients infected by a specific virus is N(3,2^2) (2^2 means that variance is 2^2=4 and hence the standard deviation is 2). Suppose that you had a sample of $n = 10$ infected individuals and you would like to compare their biomarker levels to the population mean. Generate a sample of $n = 10$ values from N(3,2^2). Display the data with a `stripchart()`. Use a t-test to test whether the mean of the patient population seems to be μ based on these data. What is the 95% confidence interval for the mean?

```
x = rnorm(10, 3, 2)
stripchart(x, col = "red")
```




```
t.test(x, mu = 6) #'mu' is t.test parameter for null hypothesis mean
```

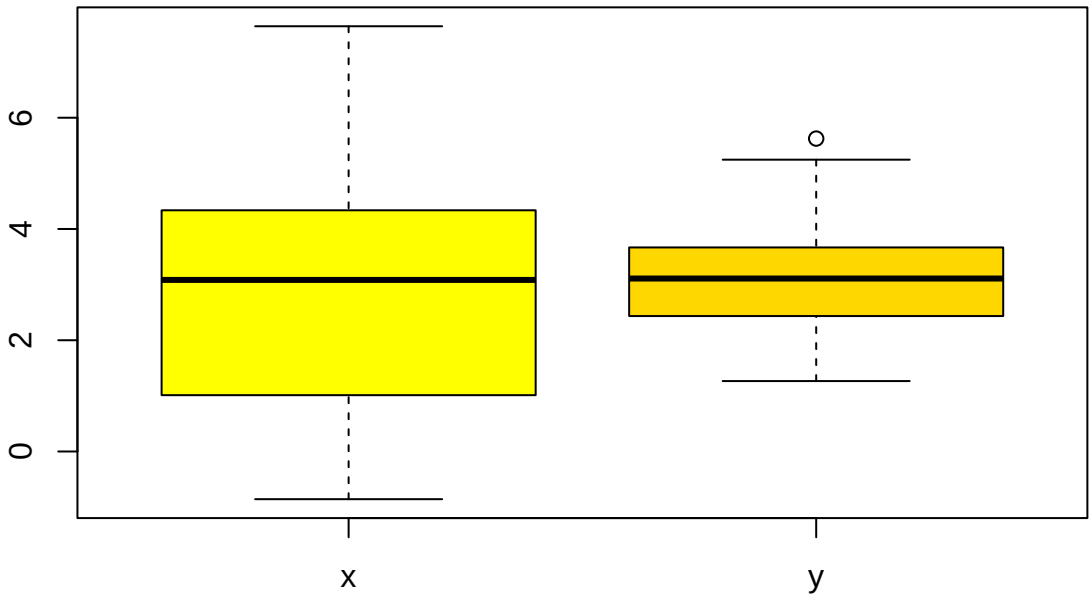
```
##
##  One Sample t-test
##
## data:  x
## t = -3.881, df = 9, p-value = 0.003726
## alternative hypothesis: true mean is not equal to 6
## 95 percent confidence interval:
##  1.214385 4.738885
## sample estimates:
## mean of x
##  2.976635
```

From the chart we see that nearly all values are below the reference value of $\mu = 6$ and hence it is not surprising that the P-value is small indicating a deviation of the sample mean from the reference value.

In the one-sample t-test, the 95% CI is given for the mean of the distribution. Here, the interval is wide because the sample size is so small, but it is still quite far from the reference value of $\mu = 6$.

(2) Suppose that you have also another set of infected individuals collected from a different hospital area. Suppose that their biomarker distribution is N(3,1^2). Generate a sample of $n = 40$ values from this distribution. Make a boxplot of both these data and the data from part (1). Use a t.test to test whether the means of the two populations are equal. What is the 95% confidence interval for the difference in means?

```
y = rnorm(40, 3, 1)
boxplot(x,y, names = c("x","y"), col = c("yellow","gold"))
```




```
t.test(x, y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -0.22407, df = 9.9011, p-value = 0.8273
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.959101  1.601522
## sample estimates:
## mean of x mean of y
##  2.976635  3.155424
```

In the two-sample t-test, the 95% CI is for the difference between the means of `x` and `y`. Here it is (-1.96,..,1.60), and there is no evidence that the population means would be different from each other because value 0 is well within this interval. Note also that the P-value is large (0.83), which says that we do not detect any clear deviation from the null hypothesis that the means are equal. This result is not surprising since the means were indeed the same in the data generation, only the standard deviations differed, which is not tested by the t-test.

Paired sample t-test.

Consider the default example data set `sleep` in R's `datasets` library. It shows how much two soporific drugs (here 1 and 2) increase in hours of sleep compared to control on 10 patients. (Write `?sleep` for info and `sleep` to see the data.)

```
sleep
```

```
##    extra group ID
## 1    0.7     1  1
## 2   -1.6     1  2
## 3   -0.2     1  3
## 4   -1.2     1  4
## 5   -0.1     1  5
## 6    3.4     1  6
## 7    3.7     1  7
## 8    0.8     1  8
## 9    0.0     1  9
## 10   2.0     1 10
## 11   1.9     2  1
## 12   0.8     2  2
## 13   1.1     2  3
## 14   0.1     2  4
## 15  -0.1     2  5
## 16   4.4     2  6
## 17   5.5     2  7
## 18   1.6     2  8
## 19   4.6     2  9
## 20   3.4     2 10
```

So we have 20 observations (`extra`) on 10 individuals (`ID`), each measured once with each of the two drugs (`group`). The question is whether there is a difference in extra sleep individuals get depending on which drug they are given. We should use the paired t-test because the **same** individuals are measured for both drugs.

We can do this paired t-test by making two vectors of the extra sleep times corresponding to the two drugs and calling `t.test(, paired = TRUE)`.

```
drug.1 = sleep$extra[sleep$group == 1]
drug.2 = sleep$extra[sleep$group == 2]
#Check that we got the data correct
rbind(drug.1, drug.2)
```

```
##        [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## drug.1  0.7 -1.6 -0.2 -1.2 -0.1  3.4  3.7  0.8  0.0   2.0
## drug.2  1.9  0.8  1.1  0.1 -0.1  4.4  5.5  1.6  4.6   3.4
```

```
t.test(drug.1, drug.2, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  drug.1 and drug.2
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -2.4598858 -0.7001142
## sample estimates:
## mean difference
##           -1.58
```

It seems that with drug 2 people sleep longer. The mean of the differences is -1.58 (-2.46, -0.70) and the P-value is 0.003 (under the null hypothesis that the mean difference was 0).

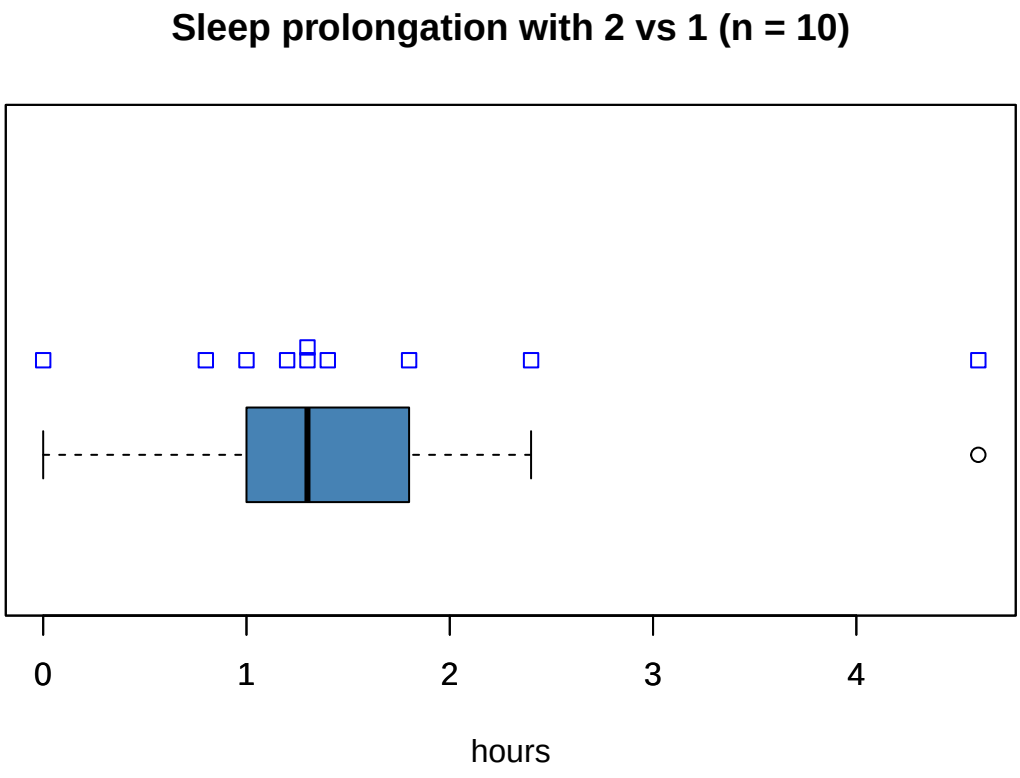
NOTE: To avoid writing complicated expressions like `sleep$extra[sleep$group == 1]`, we could use `with(sleep,)` structure which means that all the individual variables (here `extra` and `group`) are taken from the `sleep` data set. So the paired t-test can be done like this:

```
with(sleep, t.test(extra[group == 1], extra[group == 2], paired = TRUE))
```

```
##
##  Paired t-test
##
## data:  extra[group == 1] and extra[group == 2]
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -2.4598858 -0.7001142
## sample estimates:
## mean difference
##           -1.58
```

Let's visualize the differences between 2 and 1 using both a stripchart and a boxplot.

```
sleep.d = drug.2 - drug.1
stripchart(sleep.d, method = "stack", xlab = "hours", col = "blue",
           main = "Sleep prolongation with 2 vs 1 (n = 10)")
#Let's add a horizontal boxplot to the current plot (add = TRUE), at y = 0.6 (at = .6)
boxplot(sleep.d, horizontal = TRUE, add = TRUE, at = .6, col = "steelblue")
```



Actually, the paired t-test is nothing but a one-sample t-test ran on the set of within sample differences. Let's confirm.

```
t.test(sleep.d) #One-sample t-test testing whether mean of within pair differences is 0
```

```
##
##  One Sample t-test
##
## data:  sleep.d
## t = 4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.7001142 2.4598858
## sample estimates:
## mean of x
##      1.58
```

Example 4.4

(1) You have collected 40 men and 40 women and want to compare whether there is a difference between their haemoglobin levels? Would you use a paired or independent two-sample test?

The group of men are independent of the group of women (or at least no links between sampling strategies of the men and women have been described). Therefore, use the independent sample t-test.

(2) You study families who have children and you have collected both fathers and mothers from 40 families. Again you want to compare whether there is a difference in the haemoglobin levels between men and women. Would you use a paired or independent two-sample test?

Now the data come in as 40 father-mother pairs. Within families, the parents share many environmental factors, and may share some factors also because they have paired up in the first place. Therefore, use the paired t-test to account for the shared background within families.

Non-parametric tests

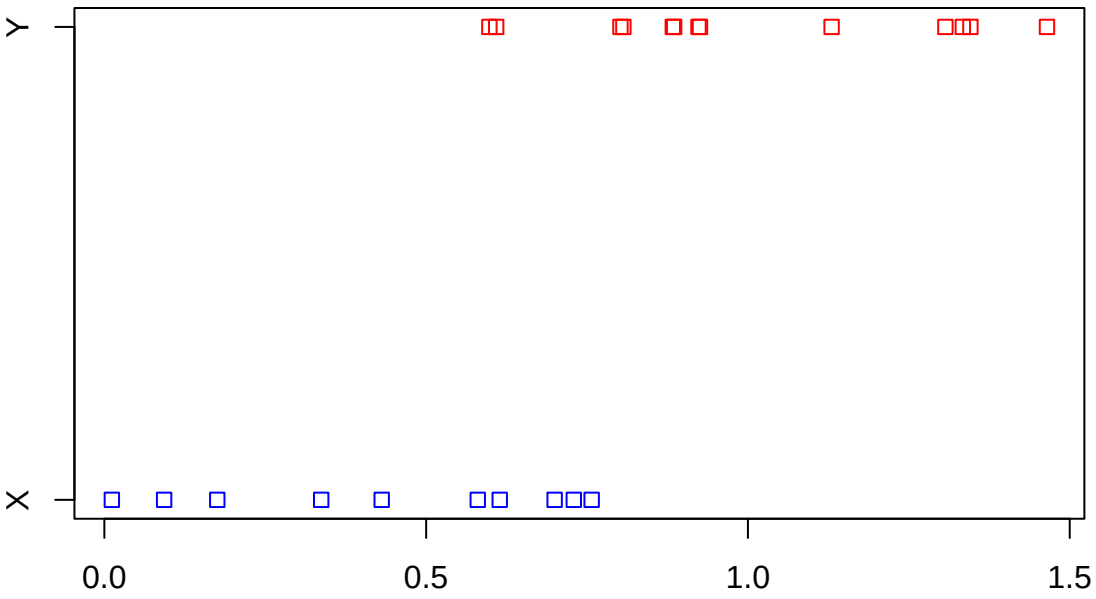
What if continuous data are not normally distributed and you are unsure whether the methods you would like to use are robust enough for the non-normality? There are *non-parametric* methods available for these cases. Non-parametric means that we do not assume any specific form or shape about the underlying population distribution. It other words, the method is statistically valid whether the data follow Normal or Uniform or Binomial distribution, for example. Under Normality, non-parametric methods typically lose some statistical power (topic of next lecture) compared to methods that assume Normality, but the obvious advantage of non-parametric tests is that they are robust to deviations from the exact distributional assumptions.

Consider the question whether the values (of some biomarker) in population X (patients) tend to be smaller or larger than in another population Y (healthy). Collect n samples from X and m samples from Y and order the $n+m$ values in ascending order. If distributions of X and Y are similar then the sum of ranks of values from X should come from the distribution of sum of ranks of n randomly chosen values from among all possible ranks $1, ..., n+m$. If the sum of ranks of X is clearly lower than under the null distribution, then values from X tend to be smaller than those from Y. If sum of ranks from X is clearly larger than under the null, then X tends to be larger than Y. This can be tested by **Wilcoxon rank sum test** (also known as **Mann-Whitney U-test**). http://www.bmj.com/about-bmj/resources-readers/publications/statistics-square-one/10-rank-score-tests

As an example, let's make samples of $n = 10$ and $m = 13$ samples from two uniform distributions that have ranges (0,1) and (0.5, 1.5), respectively, and apply Wilcoxon test.

```r
n = 10 #samples from X
m = 13 #samples from Y

# Sample X from U(0,1) and Y from U(0.5,1.5)
# Distributions are not the same but on average Y>X
X = runif(n, 0, 1)
Y = runif(m, 0.5, 1.5)
# Visualize X and Y together with different colors
stripchart(list(X,Y), col = c("blue","red"), group.names = c("X","Y"))
```



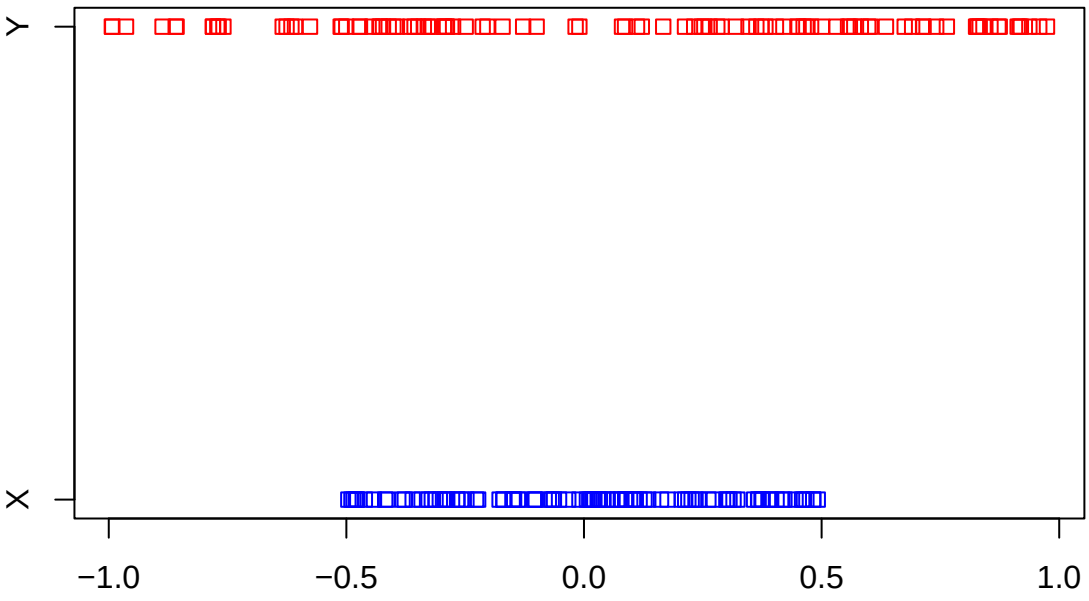

```r
# We want to test whether X and Y come from the same distribution
# We use Wilcoxon rank sum test.
wilcox.test(X,Y)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  X and Y
## W = 8, p-value = 0.0001171
## alternative hypothesis: true location shift is not equal to 0
```

We see from the plot that X tends to be smaller than Y and Wilcoxon test P-value (0.0001) indicates a possible difference as well.

Example 4.5 Generate $n = 100$ values from Uniform(-0.5, 0.5) distribution and also from Uniform(-1, 1) distribution. Use `stripchart( )` to visualize the observed data using different colors for the two samples. Which one seems to have higher values? Apply Wilcoxon rank sum test to data? What is P-value?

```r
n = 100
X = runif(n, -0.5, 0.5)
Y = runif(n, -1, 1)
# Visualize X and Y together with different colors
stripchart(list(X,Y), col = c("blue","red"), group.names = c("X","Y"))
```




```
wilcox.test(X,Y)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  X and Y
## W = 4655, p-value = 0.3999
## alternative hypothesis: true location shift is not equal to 0
```

Here the ranges of the distributions are clearly different (X is concentrated on a subrange of Y), but their median values are similar (actually 0), and there is no tendency of either of them being smaller than the other. Hence Wilcoxon P-value is not significant.

Example analysis: Blood pressure data

Let's finish with a summary of what we have done in this lecture. Let's read in the blood pressure / cholesterol data from a file `systbp_ldlc.txt`.

```
x = read.table("systbp_ldlc.txt", as.is = TRUE, header = TRUE)
str(x) #Show structure of the data
```

```
## 'data.frame':    965 obs. of  3 variables:
##  $ sex   : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ systbp: int  123 150 160 113 139 145 124 126 127 135 ...
##  $ ldlc  : num  3.4 4.2 2.7 2.3 3.5 3.4 3.1 4 3.6 3.3 ...
```

```
head(x) #Show the first few lines
```

```
##   sex systbp ldlc
## 1   2    123  3.4
## 2   2    150  4.2
## 3   2    160  2.7
```

```
## 4   2    113  2.3
## 5   2    139  3.5
## 6   2    145  3.4
```

```
#Sex is 1=Male, 2=Female, systolic blood pressure in mmHg, ldl-cholesterol in mmol/l
summary(x) #summary of the columns
```

```
##       sex            systbp           ldlc
##  Min.   :1.000   Min.   : 93.0   Min.   :1.400
##  1st Qu.:1.000   1st Qu.:122.0   1st Qu.:2.900
##  Median :2.000   Median :132.0   Median :3.500
##  Mean   :1.544   Mean   :134.4   Mean   :3.571
##  3rd Qu.:2.000   3rd Qu.:145.0   3rd Qu.:4.100
##  Max.   :2.000   Max.   :208.0   Max.   :7.500
```

```
#How many males and females?
table(x$sex) # '$' denotes variables of data frame.
```

```
##
##   1   2
## 440 525
```

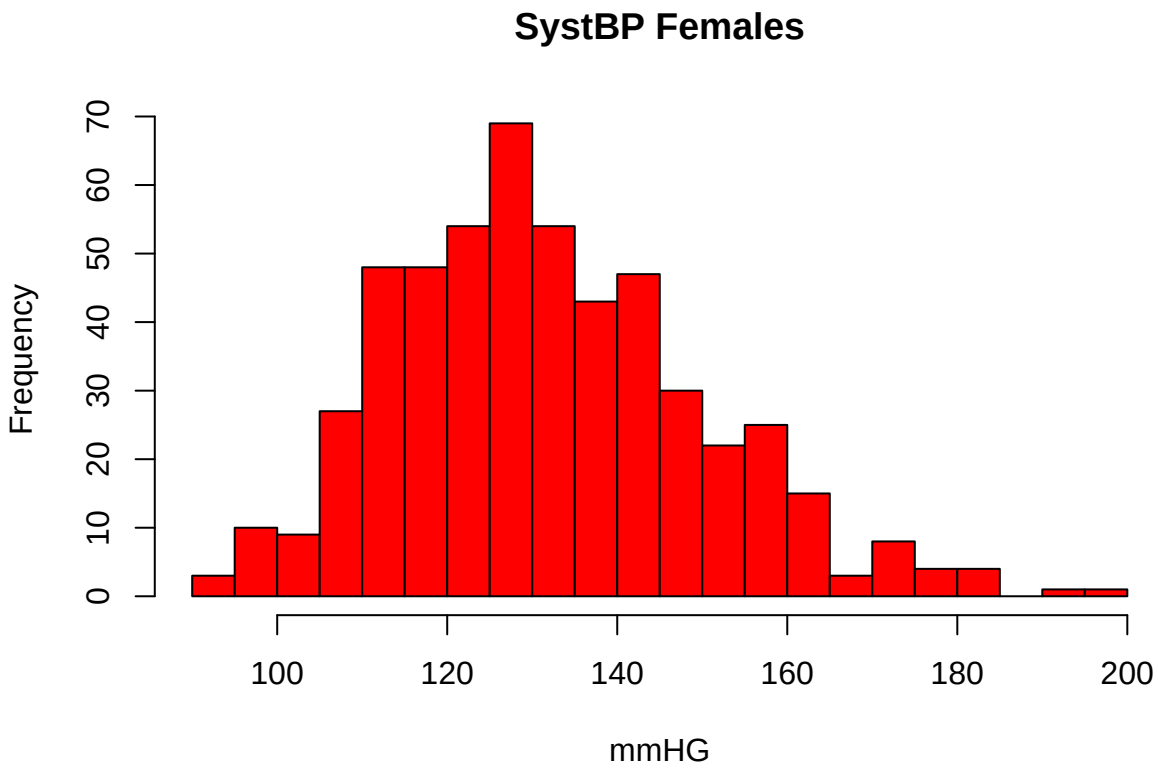
```
#We can also do things to the rows that match the condition 'sex == 1' (i.e. are males)
summary(x[x$sex == 1,]) #summary for males
```

```
##       sex        systbp           ldlc
##  Min.   :1   Min.   : 97.0   Min.   :1.500
##  1st Qu.:1   1st Qu.:125.0   1st Qu.:3.100
##  Median :1   Median :134.0   Median :3.700
##  Mean   :1   Mean   :137.1   Mean   :3.765
##  3rd Qu.:1   3rd Qu.:148.0   3rd Qu.:4.325
##  Max.   :1   Max.   :208.0   Max.   :7.300
```

```
#Or we can enclose the command in "with(x, )" and then R knows to pick variable 'sex' automatically fro
with(x, summary(x[sex == 2, ])) #summary for females
```

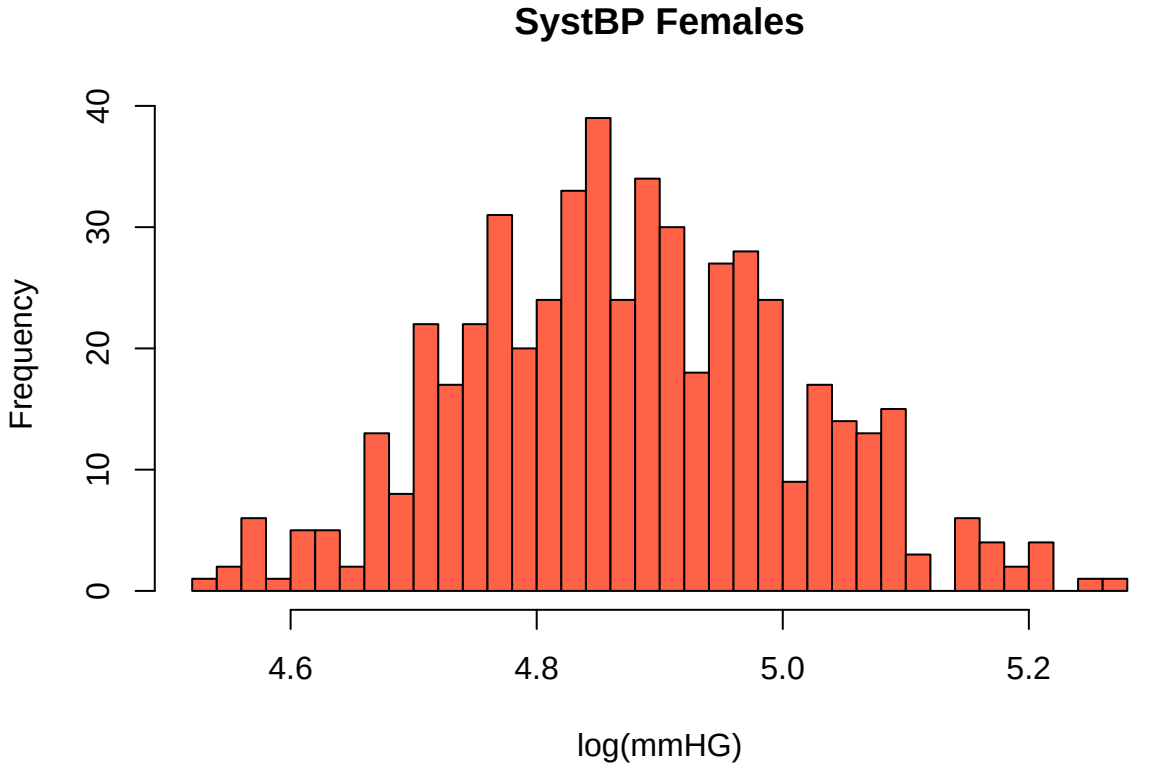
```
##       sex        systbp           ldlc
##  Min.   :2   Min.   : 93.0   Min.   :1.400
##  1st Qu.:2   1st Qu.:119.0   1st Qu.:2.800
##  Median :2   Median :130.0   Median :3.300
##  Mean   :2   Mean   :132.2   Mean   :3.408
##  3rd Qu.:2   3rd Qu.:143.0   3rd Qu.:3.900
##  Max.   :2   Max.   :196.0   Max.   :7.500
```

```
#Above we saw mean and quantiles but let's plot the whole distribution.
#Use systbp in females.
y = x[x$sex == 2,"systbp"] #now 'y' is shorthand for our data
hist(y, breaks = 30, xlab = "mmHG", main = "SystBP Females", col = "red")
```



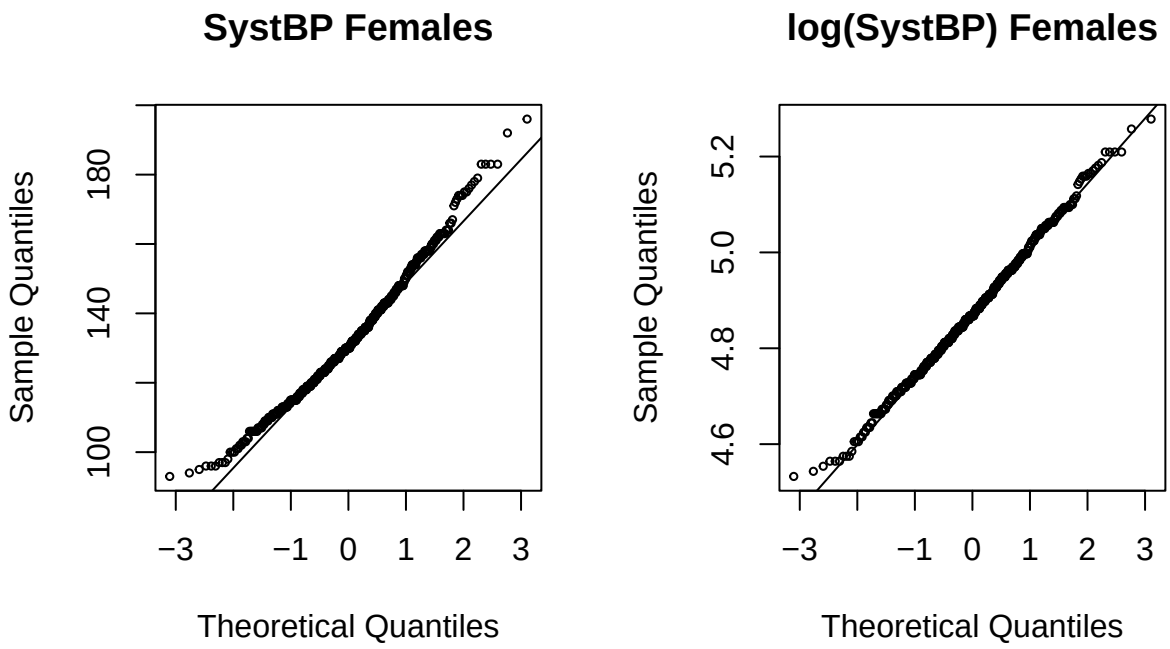
It is not symmetric around the mode but has skew to the right, towards the high values. This is common in positively valued measurements. Often logarithm of the values is much more symmetric and closer to a Normal distribution.

```
hist(log(y), breaks = 30, xlab = "log(mmHG)", main = "SystBP Females", col = "tomato")
```



Let's make QQ-plots to assess Normality of both raw values and the log-transformed values.

```
par(mfrow = c(1,2)) #2 plots next to each other (plotting area divided into 1 row, 2 columns)
par(pty = "s") #make plots to exact squares
qqnorm(y, main = "SystBP Females", cex = 0.5)
qqline(y)
qqnorm(log(y), main = "log(SystBP) Females", cex = 0.5)
qqline(log(y))
```



We see that the original data have more higher values than a Normal distribution due to the skew to right whereas the log-transformed data look like Normally distributed.

Are the means in males and females different? Even though the distributions were not perfectly normal, the sample sizes (~500 for males and females) are large enough that the confidence interval from `t.test()` is a good approximation for the difference in means.

```
y.m = x[x$sex == 1, "systbp"]
y.f = x[x$sex == 2, "systbp"]
t.test(y.m, y.f)
```

```
##
##  Welch Two Sample t-test
##
## data:  y.m and y.f
## t = 4.2583, df = 944.71, p-value = 2.266e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.621554 7.103381
## sample estimates:
## mean of x mean of y
##  137.0682  132.2057
```

```r
#With log-transformed data we have more perfectly normally distributed data:
logy.m = log(y.m)
logy.f = log(y.f)
t.test(logy.m, logy.f)
```

```
##
##  Welch Two Sample t-test
##
## data:  logy.m and logy.f
## t = 4.4962, df = 954.69, p-value = 7.769e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.02111718 0.05382917
## sample estimates:
## mean of x mean of y
##  4.912726  4.875253
```

```r
#What about non-parametric test?
wilcox.test(y.m, y.f)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  y.m and y.f
## W = 133548, p-value = 2.84e-05
## alternative hypothesis: true location shift is not equal to 0
```

```r
#Note that non-parametric test is immune to any monotonic transformation.
#Thus it gives exactly the same P-value for log-transformed data as it gave for the original data:
wilcox.test(logy.m, logy.f)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  logy.m and logy.f
## W = 133548, p-value = 2.84e-05
## alternative hypothesis: true location shift is not equal to 0
```

Conclusions: Means of `systbp` are statistically higher in males than in females. (Same conclusion by all 3 tests with P-values $<$ 1e-4.) The difference in means is 4.9 mmHG (95%CI 2.6,. . . ,7.1).

Extra material - Read if you are interested - not required in exercises

Read more about logarithms http://www.bmj.com/content/312/7032/700.full and transformations http://www.bmj.com/content/312/7033/770.full. Note that in conclusion, we gave a 95%CI for the difference in means in the original scale (mmHG) even though the distributions were not perfectly normal. This is based on a large sample size and Central Limit Theorem but would not be a good practice for small data sets. For that, read about CI of the difference in means of transformed data http://www.bmj.com/content/312/7039/1153.full.

Central Limit Theorem (CLT)

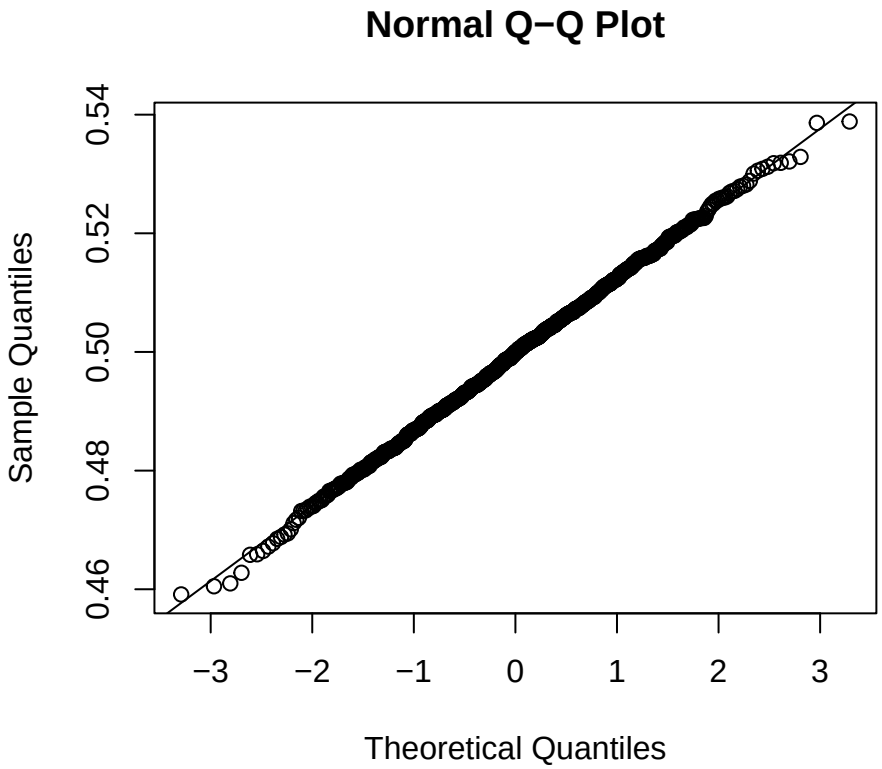
Assume that dataset X contains n (independent) samples from some distribution with mean=μ and standard deviation=σ but X does not necessarily need to follow Normal, binomial or any other distribution we have ever heard about. CLT says that the distribution of the point estimate of the mean of X is approximately Normal(mean=μ,sd= $\sigma/\sqrt{n}$) in LARGE samples. This result can be used to derive confidence intervals for the estimated mean. For example, a 95%CI is the observed mean $\pm 1.96\, s/\sqrt{n}$, where s is the observed standard deviation of X. The importance of this result is its complete generality with respect to the shape of the underlying distribution. However, it requires a large sample to work in practice: Rather hundreds than a few dozens.

Let's try it out with Uniformly distributed values. Let's compute a mean of $n = 500$ U(0,1) distributed values, and repeat this computation 1000 times. We'll do that by making a big data matrix where rows are 1000 repetitions and columns are 500 observations. According to the theory, the mean of U(0,1) is 0.5 and its sd is $1/\sqrt{12} \approx 0.289$. Thus, we expect that the mean of 500 samples from U(0,1) has a mean of 0.5 and sd of $1/\sqrt{12 \cdot 500} \approx 0.0129$. Our main interest is whether the distribution of the mean over the data sets is Normal as CLT claims.

```
n = 500 #sample size in each repetition
m = 1000 #number of repetitions
X = matrix(runif(n*m), nrow = m, ncol = n) #data matrix
means = rowSums(X) / n #collect 1000 means here
c(mean(means), sd(means)) #what is the mean and sd of the estimates of the mean
```

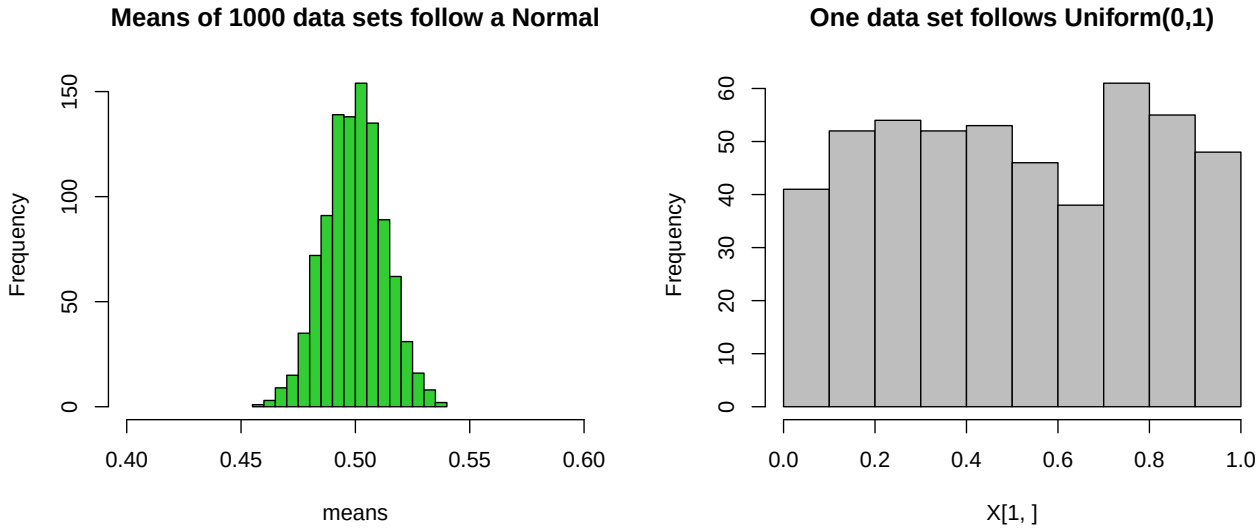
```
## [1] 0.49958043 0.01287575
```

```
qqnorm(means) #see whether means seem Normally distributed
qqline(means)
```



Indeed, the distributions of means look like Normal according to the QQ-plot. Let's see the histogram of the means, and let's also show a histogram of one of the individual data sets to see that it indeed looked like uniform (and is far from Normal!).

```
par(mfrow = c(1,2))
hist(means, xlim = c(0.4,0.6), breaks = 15, col = "limegreen", main="Means of 1000 data sets follow a N
#plot the first data set just to make sure that the original data look like U(0,1)
hist(X[1,], breaks = 10, col = "gray", main="One data set follows Uniform(0,1)")
```



From the histograms we see how a single data set looks uniformly distributed on [0,1] (right) but how the means of 1000 such data sets is tightly concentrated around 0.5 and looks like Normally distributed. This is CLT in action.

From this result, we can derive the standard Normal approximation to the confidence interval for the mean of values from any distribution. Suppose that we have n values from a distribution D. The endpoints of the 95% confidence interval for the mean of D are

$$\overline{x} \pm 1.96 \times \widehat{s}/\sqrt{n},$$

where $\overline{x}$ is the empirical mean of the n values, $\widehat{s}$ is their empirical standard deviation and 1.96 `=qnorm(1-0.05/2)` is the quantile point from the standard Normal distribution below which probability mass $0.975 = 1 - 0.05/2$ lies.