Applied Statistics and Data Analysis
Lab 3a: A review of inference concepts - Statistical models

Luca Grassetti and Paolo Vidoni
Department of Economics and Statistics, University of Udine

September, 2019

1 Some introductory examples

1.1 Example: temperatures

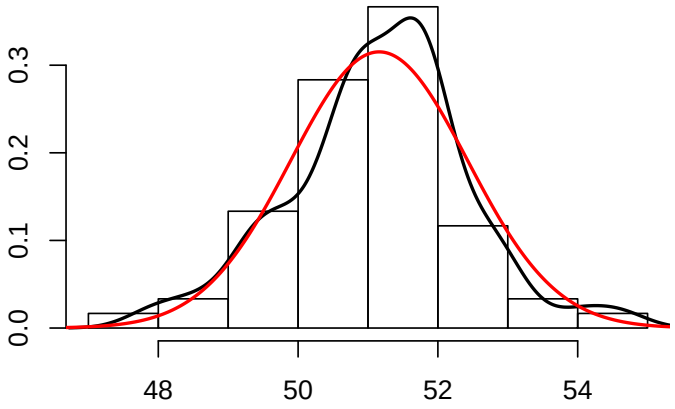
The `nhtemp` data set contains the mean annual temperature in degrees Fahrenheit in New Haven, Connecticut, from 1912 to 1971. It is a time series of 60 observations. The unknown density function of the interest phenomenon is estimated by considering the histogram calculated using the function `hist`. The plot includes two additional curves obtained by means of the function `lines`, namely:

- the black one, which is the continuous approximation of the true density function given by the kernel density estimation procedure; the result is obtained using the function `density`;
- the red one, which is the continuous approximation of the true density function identified within the family of the normal distributions; the parameters of the normal density, specified using the function `dnorm`, are the sample mean `mean(nhtemp)` and the sample standard deviation `sqrt(var(nhtemp))`.

With regard to function `density`, the following main options have to be considered:

- `bw` and `adjust`, which product defines the smoothing bandwidth of the estimation procedure: `adjust` is only a multiplier (the default value is 1) and `bw` can be numeric or it can be a character string giving a rule to choose the bandwidth (the default option is `"nrd0"`, giving the Silverman's 'rule-of-thumb');
- `kernel`, which allows to choose the smoothing kernel to be used (the default is `"gaussian"` but it is possible to define also `"rectangular"`, `"triangular"`, `"epanechnikov"`, `"biweight"`, `"cosine"` or `"optcosine"`).

```
hist(nhtemp,freq=F,main=' ',xlab=' ',ylab=' ')
lines(density(nhtemp),lwd=2)
lines(seq(45,60,0.01),dnorm(seq(45,60,0.01),
        mean(nhtemp),sqrt(var(nhtemp))),col='red',lwd=2)
```



The empirical distribution of temperatures can be conveniently approximated by a Gaussian theoretical distribution, although the tails are thicker than the normal ones. Furthermore, the following commands are used to compute some numerical summaries and the main location and variability indices.

```
summary(nhtemp)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  47.90   50.58   51.20   51.16   51.90   54.60

mean(nhtemp)

[1] 51.16

median(nhtemp)

[1] 51.2

var(nhtemp)

[1] 1.601763
```

Skewness and kurtosis indices are obtained by using the corresponding sample moments. The library `moments` allows the direct computation of these indices with specific commands.

```
mean((nhtemp-mean(nhtemp))^3)/sqrt(var(nhtemp))^3
```

```
[1] -0.07178758
```

```
mean((nhtemp-mean(nhtemp))^4)/sqrt(var(nhtemp))^4
```

```
[1] 3.383275
```

1.2 Example: roller data

The `roller` data set of the package `DAAG` collects ten replications of a simple experiment where the effect of the `weight` of a roller, on the depth `depression` made in the grass, is measured. The scatterplot gives a graphical representation of the interest phenomenon. The `plot` function is used here with the following specific arguments:

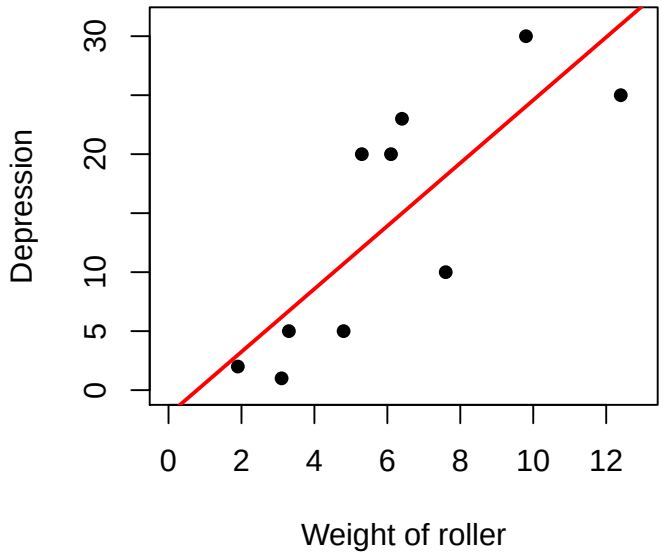
- `depression ~ weight`, which means that the `depression` behavior is represented conditionally to the `weight`;
- `data`, which is used to define the source of data;
- `xlim` and `ylim`, which define the x and y limits of the plot area;
- `xlab` and `ylab`, which define the labels of the axes;
- `pch`, which allows to define different symbols (`pch=16` corresponds to a solid circle).

In order to describe the linear component of the relationship between the two variables, the regression line is also reported in the same scatterplot using the function `abline`.

The `abline` function can be specified by considering alternative options. For example, the intercept and the slope of the line can be defined directly using the `a` and `b` options (the `coef` argument can be used instead); indeed, a vertical or an horizontal line can be added to the existing graph using the `v` or `h` options, respectively.

The regression line can be drawn also by using the `reg` option, that is by considering, as argument of the function `abline`, the result of function `lm`, which fits a linear model.

```
library(DAAG)
plot(depression ~ weight, data = roller,
     xlim=c(0,1.04*max(weight)),ylim=c(0,1.04*max(depression)),
xlab = 'Weight of roller', ylab = 'Depression', pch = 16)
roller.lm <- lm(depression ~ weight,data=roller)
abline(roller.lm,col='red',lwd=2)
```



Function `lm` returns an object of class `"lm"`, namely a linear regression object. Its attributes are

```
attributes(roller.lm)

$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"
```

Thus, it is a list which contains, for example, the coefficients of the regression line, that can be obtained by asking for the element `coefficients` (the abbreviation `coef` can be considered, since it defines a unique correspondence with the attribute `coefficients`).

```
roller.lm$coef

(Intercept)      weight
  -2.087148    2.666746
```

If the object is processed by using the function `summary`, the result presents some other interesting attributes.

```
attributes(summary(roller.lm))

$names
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"

$class
[1] "summary.lm"
```

The sub-object `coefficients` is now more complex than before.

```
summary(roller.lm)$coef

             Estimate Std. Error    t value    Pr(>|t|)
(Intercept) -2.087148  4.7542813 -0.4390038 0.672274166
weight       2.666746  0.7002426  3.8083171 0.005175013
```

The Ordinary Least Squares estimates for model parameters are obtained by minimizing the sum of the squared residuals. The procedure can be developed also using one of the optimizer included in R, as for instance `optim`.

First, the objective function is defined as follows

```
OLSfunct <- function(coef, y, x)
{
  return(sum((y-coef[1]-coef[2]*x)^2))
}
```

Then, the function `optim` is considered in order to minimize the function `OLSfunct` with respect to the parameters values `param`.

```
param <- c(0,0)
res <- optim(param, OLSfunct, y=roller$depression, x=roller$weight)
```

The point estimates of the regression parameters are

```
res$par

[1] -2.089796  2.666938
```

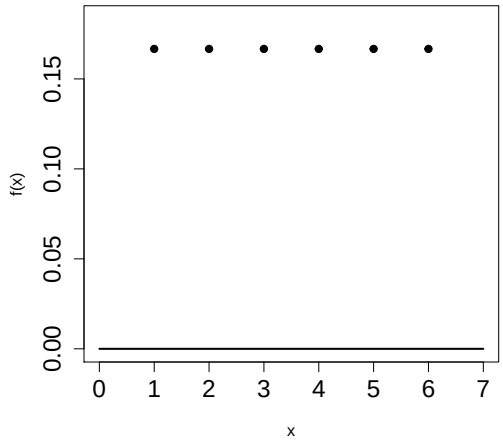
We obtain exactly the same values given by the `lm` function.

2 Basic statistical models

2.1 Discrete uniform distribution

The discrete uniform distribution can be easily defined without using specific R functions. In fact, the probability or mass function simply specifies a probability mass $1/n$ to each of the n potential observations of the random phenomenon.

```
xx<-c(1:6)
plot(xx,rep(1/6,6),pch=19,xlim=c(0,7),ylim=c(0,1.1/6),
     cex.axis=1.5,xlab="x",ylab="f(x)")
segments(0,0,7,0,lwd=2)
```



The `sample` command can be used to simulate observations from a discrete uniform random variable. The procedure is as follows:

- define the vector of possible observations (for example, letters from `"a"` to `"j"`);

```
xx <- letters[1:10]
```

- extract, using the `sample` function, a sample of size 100 with replacement from vector `xx` (the command `set.seed(11)` sets the seed of the pseudo-random number generation so that the results are reproducible);

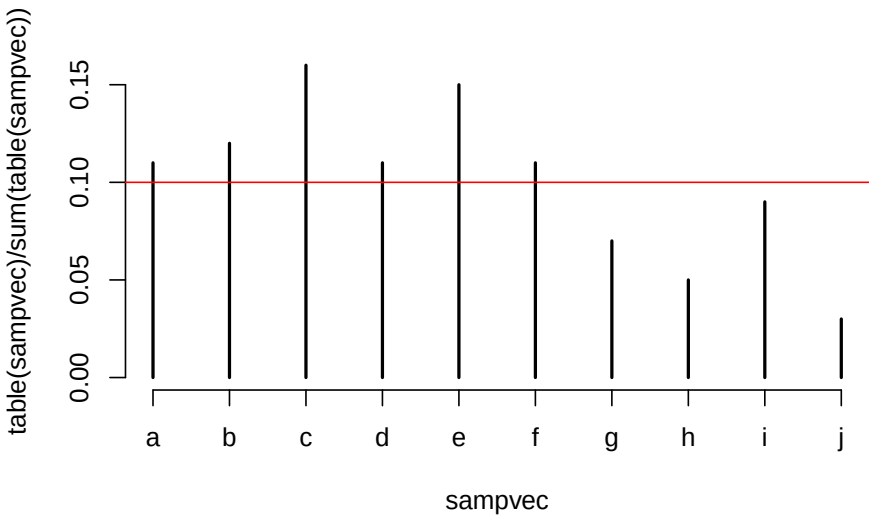
```
set.seed(11)
sampvec <- sample(xx, 100, replace=T)
```

- calculate, using the `table` function, the relative frequency of the observed values in order to define the empirical probability distribution;

```
table(sampvec)/sum(table(sampvec))


sampvec
   a    b    c    d    e    f    g    h    i    j
0.11 0.12 0.16 0.11 0.15 0.11 0.07 0.05 0.09 0.03
```

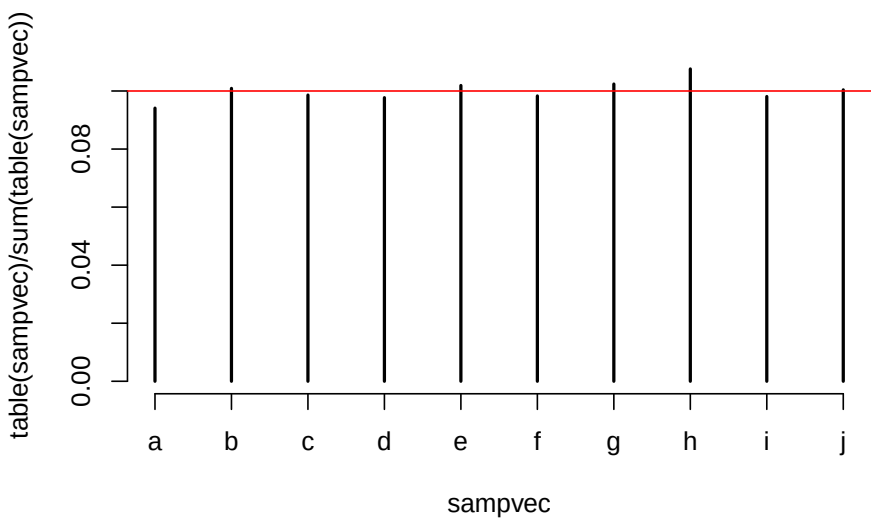
- plot the empirical distribution with an additional red horizontal line, which corresponds to the value assigned by the true uniform probability distribution.

```
plot(table(sampvec)/sum(table(sampvec)))
abline(h=0.1, col=2)
```



Since the sample size could be too low to obtain a good approximation of the theoretical distribution, we consider a larger sample (10000 observations). The empirical distribution is expected to be closer to true one.

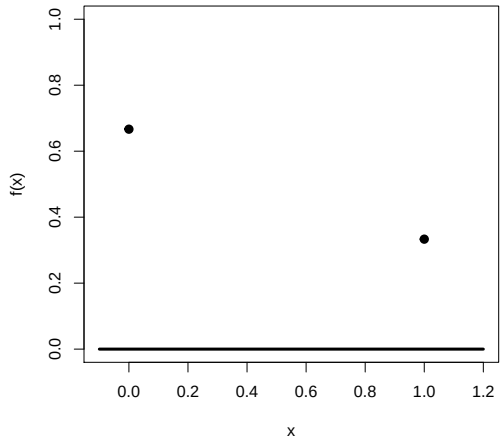
```
set.seed(11)
sampvec <- sample(xx, 10000, replace=T)
table(sampvec)/sum(table(sampvec))

sampvec
     a      b      c      d      e      f      g      h      i      j
0.0941 0.1009 0.0986 0.0977 0.1019 0.0983 0.1024 0.1076 0.0981 0.1004

plot(table(sampvec)/sum(table(sampvec)))
abline(h=0.1, col=2)
```



2.2 Bernoulli distribution

The Bernoulli distribution corresponds to an elementary random experiment where the success (coded as 1) or the failure (coded as 0) are the only two possible outcomes. If the success probability is $P(X = 1) = 1/3$, the probability function can be described using the following commands.

```
plot(c(0,1),c(2/3,1/3),pch=19,lwd=2,xlab='x',
     ylab='f(x)',xlim=c(-0.1,1.2),ylim=c(0,1))
segments(-0.1,0,1.2,0,lwd=3)
```



The `sample` function can be used to generate samples from this distribution. More precisely,

- define the possible outcomes;

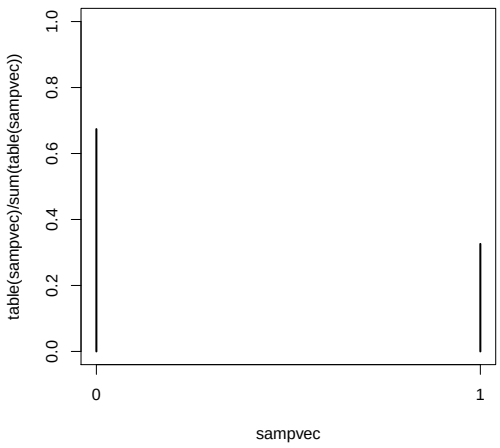
```
xx <- c(0,1)
```

- extract a sample of size 1000 with outcomes probabilities 2/3 and 1/3, respectively;

```
set.seed(101)
sampvec <- sample(xx, 1000, prob=c(2/3,1/3), replace=T)
```

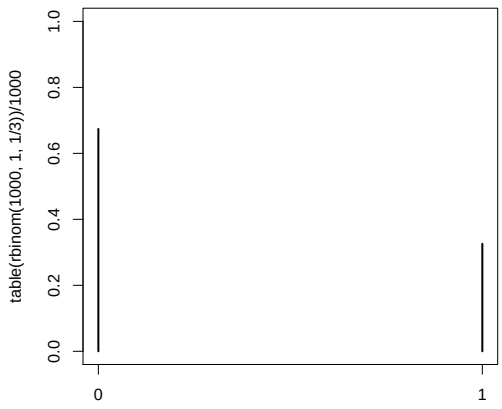
- tabulate the sample distribution and use the barplot to represent it.

```
table(sampvec)/sum(table(sampvec))

sampvec
    0     1
0.674 0.326

plot(table(sampvec)/sum(table(sampvec)), ylim=c(0,1))
```



Since a Bernoulli distribution corresponds to a binomial distribution with $n = 1$ trials, the same result can be obtained using function `rbinom` with the option `size=1`.

```
set.seed(101)
plot(table(rbinom(1000,1,1/3))/1000, ylim=c(0,1))
```



2.3 Binomial distribution

The binomial distribution describes the number of successes in $n \geq 1$ independent Bernoulli experiments with the same success probability p. The following functions are included in R in order to work with this distribution (where `size` is the number of replications and `prob` is the probability of success in each Bernoulli experiment) :

- `rbinom(n, size, prob)`, which is used to generate data, where `n` is the sample size;
- `dbinom(x, size, prob)`, which is used to obtain the probability function at a given data point `x`;
- `qbinom(p, size, prob)`, which is used to calculate the theoretical quantiles for a given probability level `p`;
- `pbinom(q, size, prob)`, which is used to obtain the distribution function at a given data point `q`.

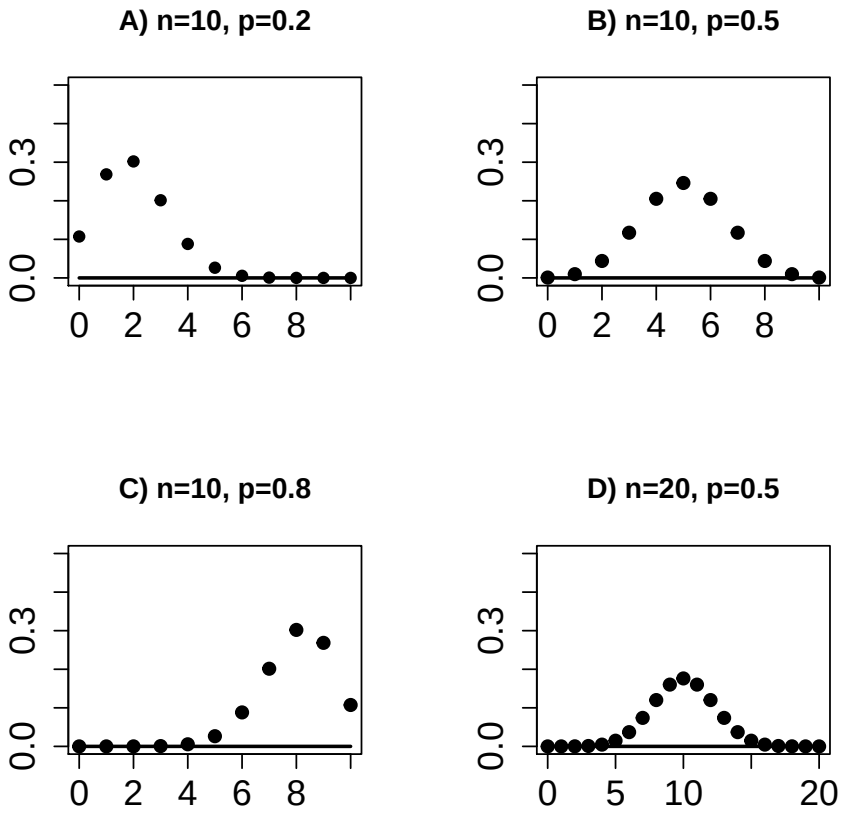
R has similar functions available for a number of different probability distributions.

The following code is used to

1. prepare the graphical device for a figure with four different plots;
2. define the variable support `xx` (namely, the potential values of the binomial experiment);
3. draw the scatterplot with coordinates `xx` and `dbinom(xx, 10, 0.2)` (the probabilities corresponding to the support points); the option `cex.axis` gives the magnification to be used for the axis and `main` defines the main title for the plot;
4. add the x-axis on on the plot using the `segments` function, where `lwd` gives the line width.

The figure represents the probability functions of binomial random variables with parameters $n = 10$, $p = 0.2$ (plot A), $n = 10$, $p = 0.5$ (plot B), $n = 10$, $p = 0.8$ (plot C) and $n = 20$, $p = 0.5$ (plot D).

```
par(mfrow=c(2,2)) # Step 1
xx<-seq(0,10,1) # Step 2
plot(xx,dbinom(xx,10,0.2),pch=19,ylim=c(0,0.5),
     cex.axis=1.5,xlab=" ",ylab=" ",main="A) n=10, p=0.2") # Step 3
segments(0,0,10,0,lwd=2) # Step 4

plot(xx,dbinom(xx,10,0.5),pch=19,ylim=c(0,0.5),lwd=2,
     cex.axis=1.5,xlab=" ",ylab=" ",main="B) n=10, p=0.5") # Step 3
segments(0,0,10,0,lwd=2) # Step 4

plot(xx,dbinom(xx,10,0.8),pch=19,ylim=c(0,0.5),lwd=2,
     cex.axis=1.5,xlab=" ",ylab=" ",main="C) n=10, p=0.8") # Step 3
segments(0,0,10,0,lwd=2) # Step 4

xx<-seq(0,20,1) # Step 2
plot(xx,dbinom(xx,20,0.5),pch=19,ylim=c(0,0.5),lwd=2,
     cex.axis=1.5,xlab=" ",ylab=" ",main="D) n=20, p=0.5") # Step 3
segments(0,0,20,0,lwd=2) # Step 4
```




```
par(mfrow=c(1,1))
```

The final line of code restores the graphical device configuration to the default (that is, a single plot device).

The deciles of the probability distribution and the distribution function at the support values can be obtained by using the following commands.

```
qbinom(seq(0,1,0.1), 10, 0.2)
```

```
 [1]  0  0  1  1  2  2  2  3  3  4 10
```

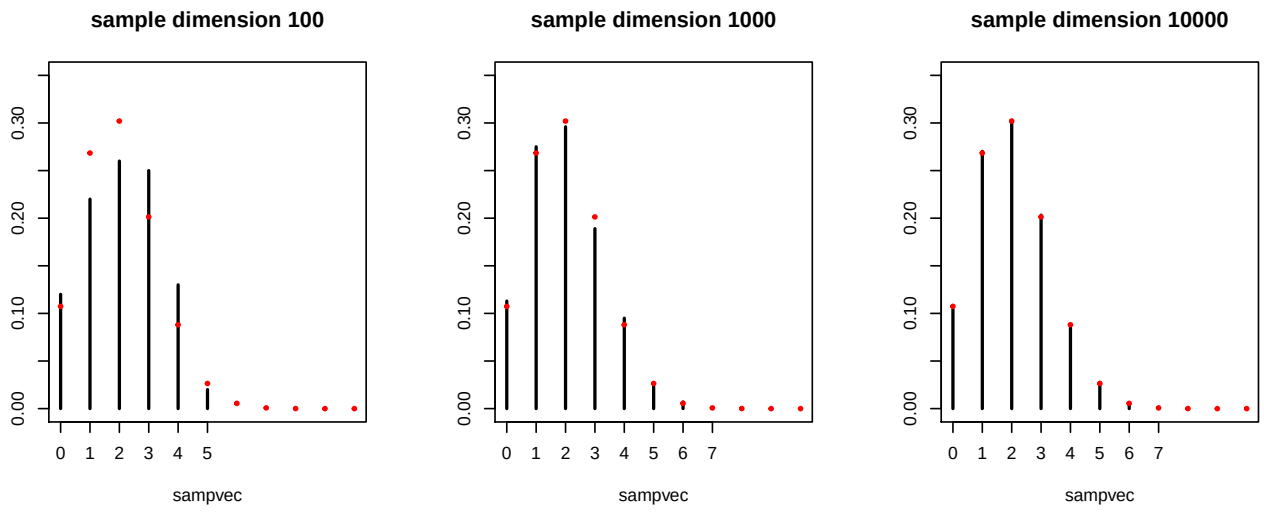
```
pbinom(0:10, 10, 0.2)
```

```
 [1] 0.1073742 0.3758096 0.6777995 0.8791261 0.9672065 0.9936306
 [7] 0.9991356 0.9999221 0.9999958 0.9999999 1.0000000
```

As in the previous section, it is also possible to generate a sample from a binomial distribution and to draw the empirical probability function (namely, the function which gives the relative

frequencies associated to the outcomes). Incrementing the sample size it is possible to observe that the empirical and the theoretical probability functions get closer. The red circles represent the theoretical probabilities.

```
par(mfrow=c(1,3))
## 100
set.seed(101)
nn <- 100
sampvec <- rbinom(nn,10,0.2)
plot(table(sampvec)/sum(table(sampvec)), xlim=c(0,10), ylim=c(0,0.35), ylab="",
     main="sample dimension 100")
points(0:10, dbinom(0:10,10,0.2), col=2, lwd=2, cex=0.3)
## 1000
set.seed(101)
nn <- 1000
sampvec <- rbinom(nn,10,0.2)
plot(table(sampvec)/sum(table(sampvec)), xlim=c(0,10), ylim=c(0,0.35), ylab="",
     main="sample dimension 1000")
points(0:10, dbinom(0:10,10,0.2), col=2, lwd=2, cex=0.3)
## 10000
set.seed(101)
nn <- 10000
sampvec <- rbinom(nn,10,0.2)
plot(table(sampvec)/sum(table(sampvec)), xlim=c(0,10), ylim=c(0,0.35), ylab="",
     main="sample dimension 10000")
points(0:10, dbinom(0:10,10,0.2), col=2, lwd=2, cex=0.3)
```



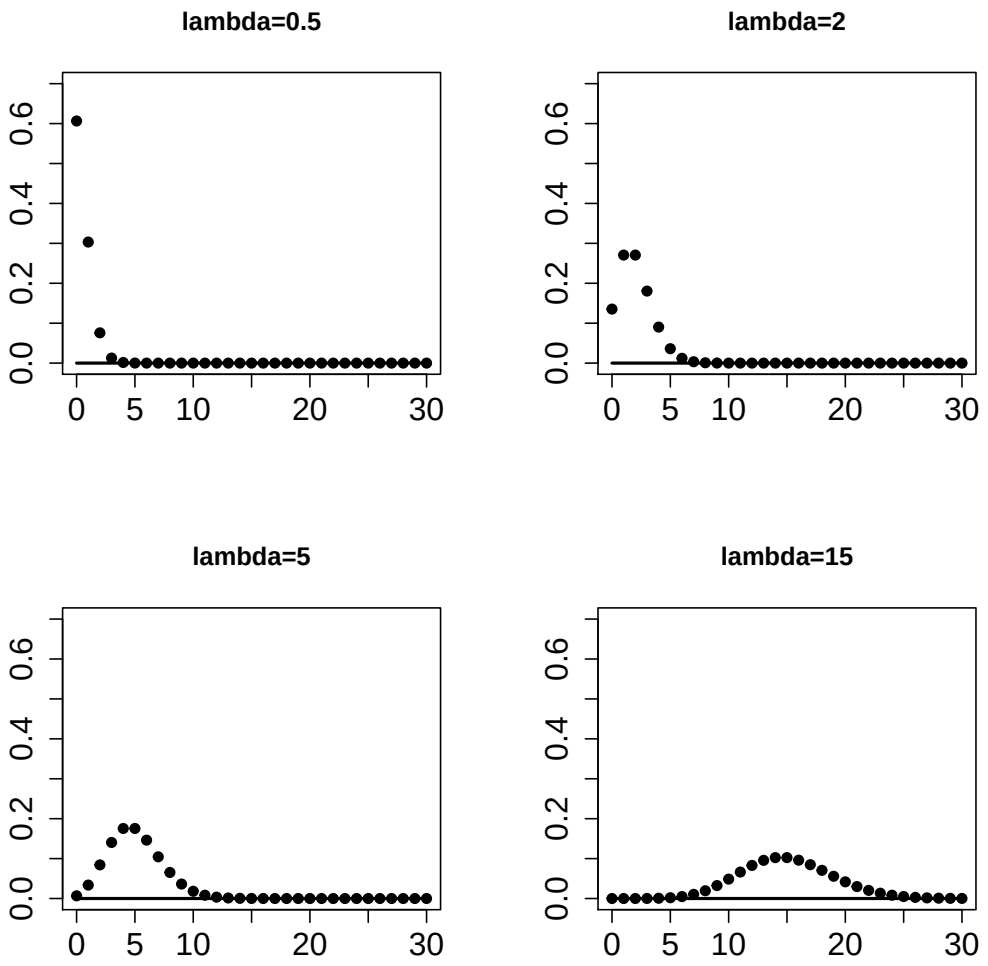

```
par(mfrow=c(1,1))
```

2.4 Poisson distribution

The Poisson distribution is used to describe count data. As for the distributions considered above, there are specific commands which give the probability function (`dpois`), the distribution function (`ppois`) and the quantiles (`qpois`). Indeed, the `rpois` function can be used to generate data from a Poisson distribution. The distribution is characterized by the parameter λ, which is both the mean and the variance.

Some probability functions, with different values for λ, are represented here by considering a subset of the support values, that is the integer sequence ranging from 0 to 30 given by the command `seq(0,30,1)`.

```
par(mfrow=c(2,2))
xx<-seq(0,30,1)
plot(xx,dpois(xx,0.5),pch=19,ylim=c(0,0.7),
     cex.axis=1.5,xlab=" ",ylab=" ",main="lambda=0.5")
segments(0,0,30,0,lwd=2)
plot(xx,dpois(xx,2),pch=19,ylim=c(0,0.7),
     cex.axis=1.5,xlab=" ",ylab=" ",main="lambda=2")
segments(0,0,30,0,lwd=2)
plot(xx,dpois(xx,5),pch=19,ylim=c(0,0.7),
     cex.axis=1.5,xlab=" ",ylab=" ",main="lambda=5")
segments(0,0,30,0,lwd=2)
plot(xx,dpois(xx,15),pch=19,ylim=c(0,0.7),
     cex.axis=1.5,xlab=" ",ylab=" ",main="lambda=15")
segments(0,0,30,0,lwd=2)
```



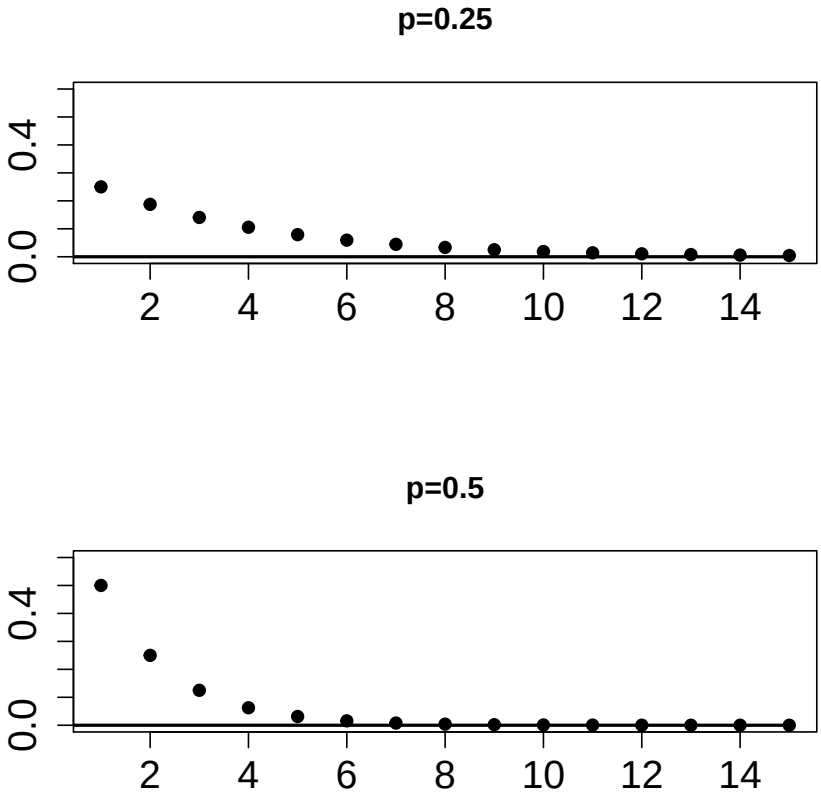

```
par(mfrow=c(1,1))
```

2.5 Geometric distribution

The geometric distribution is used to describe the number of trials for the first success in a sequence of independent Bernoulli experiments with the same probability of success p. Instead of considering the number of trials, R specifies the distribution focusing on the number of failures before the first success. Then, in order to compute the probability function, the trick is to use the `dgeom` function on a support sequence with values reduced by 1. The probability functions for a geometric distribution with $p = 0.25$ and $p = 0.5$ are drawn by means of the following lines of code.

```
par(mfrow=c(2,1))
xx<-seq(1,15,1)
plot(xx,dgeom(xx-1,0.25),pch=19,ylim=c(0,0.6),
```

```
     cex.axis=1.5,xlab=" ",ylab=" ",main="p=0.25")
segments(0,0,15,0,lwd=2)
plot(xx,dgeom(xx-1,0.5),pch=19,ylim=c(0,0.6),
     cex.axis=1.5,xlab=" ",ylab=" ",main="p=0.5")
segments(0,0,15,0,lwd=2)
```



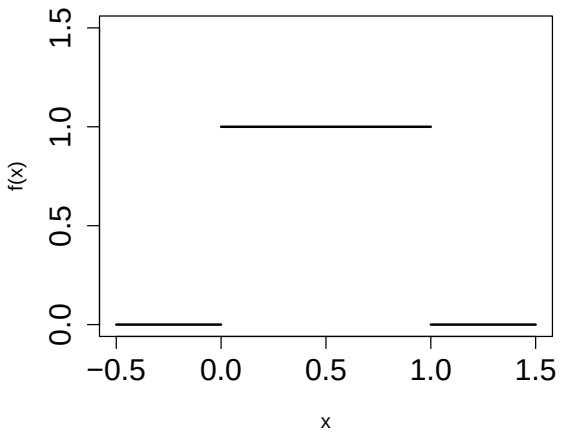

```
par(mfrow=c(1,1))
```

2.6 Continuous uniform distribution

The uniform distribution is characterized by a constant density function, defined on an interval support. It describes equiprobability for continuous experiments. The reciprocal of the support width defines the level of the density function. The density function of a uniform random variable with support $[0, 1]$ is given below.

```
xx<-seq(0,1,0.01)
plot(xx,dunif(xx,0,1),xlim=c(-0.5,1.5),ylim=c(0,1.5),
     type='l',lwd=2,cex.axis=1.5,xlab="x",ylab="f(x)")
```

```
segments(-0.5,0,0,0,lwd=2)
segments(1,0,1.5,0,lwd=2)
```

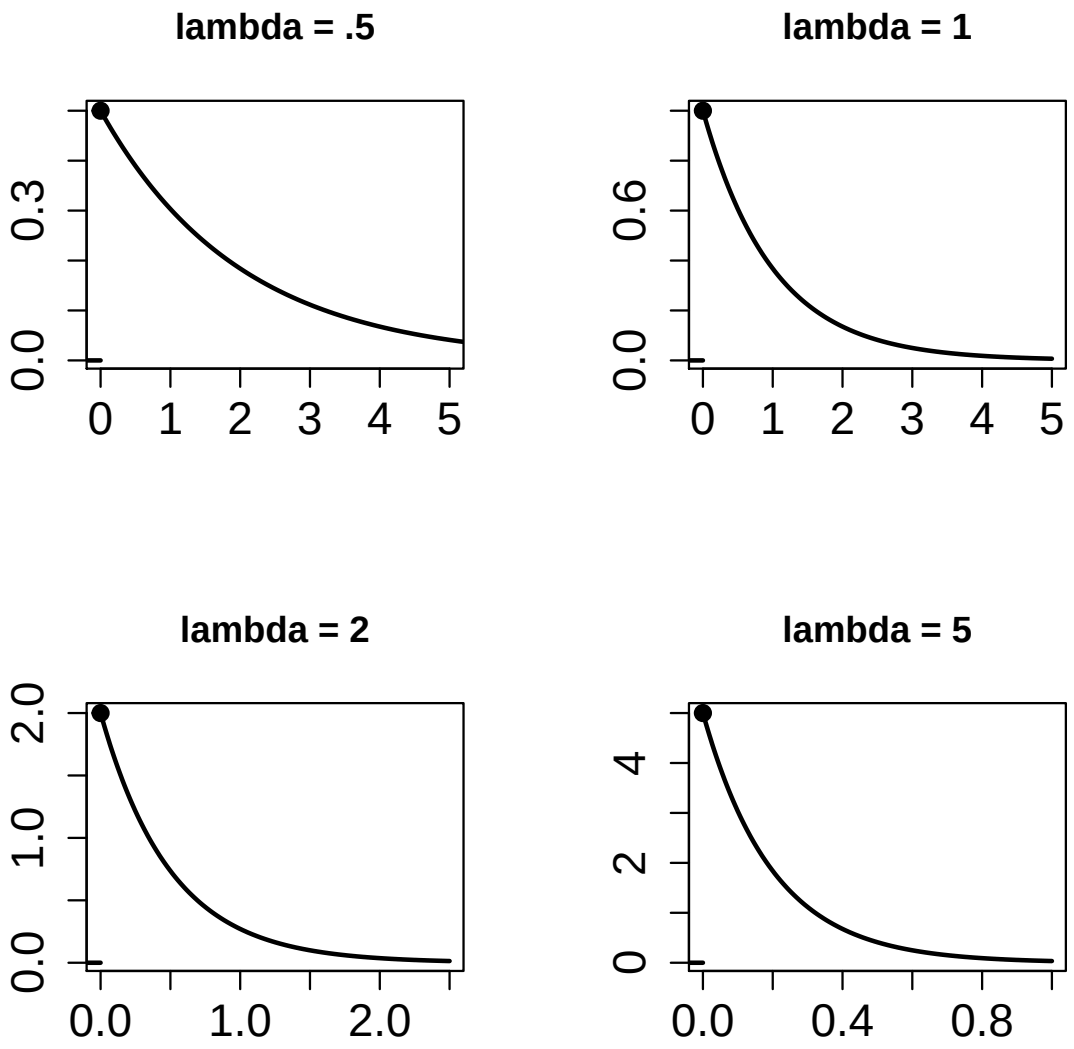


2.7 Exponential distribution

The exponential distribution is often used to describe durations, failure times or waiting times, assuming a constant hazard rate given by the parameter λ. The density function for exponential random variables with $\lambda = 0.5, 1, 2, 5$ is generated by the following code.

```
par(mfrow=c(2,2))
lam <- 0.5
xx<-seq(0,5/lam,0.01/lam)
plot(xx,dexp(xx,lam),xlim=c(0,5),
     type='l',lwd=2,cex.axis=1.5,xlab=" ",ylab=" ", main="lambda = .5")
points(0,dexp(0,lam),pch=19,lwd=2)
segments(-1,0,0,0,lwd=2)
lam <- 1
xx<-seq(0,5/lam,0.01/lam)
plot(xx,dexp(xx,lam),xlim=c(0,5/lam),
     type='l',lwd=2,cex.axis=1.5,xlab=" ",ylab=" ", main="lambda = 1")
points(0,dexp(0,lam),pch=19,lwd=2)
segments(-1,0,0,0,lwd=2)
lam <- 2
xx<-seq(0,5/lam,0.01/lam)
plot(xx,dexp(xx,lam),xlim=c(0,5/lam),
     type='l',lwd=2,cex.axis=1.5,xlab=" ",ylab=" ", main="lambda = 2")
points(0,dexp(0,lam),pch=19,lwd=2)
segments(-1,0,0,0,lwd=2)
lam <- 5
```

```
xx<-seq(0,5/lam,0.01/lam)
plot(xx,dexp(xx,lam),xlim=c(0,5/lam),
     type='l',lwd=2,cex.axis=1.5,xlab=" ",ylab=" ", main="lambda = 5")
points(0,dexp(0,lam),pch=19,lwd=2)
segments(-1,0,0,0,lwd=2)
```




```
par(mfrow=c(1,1))
```

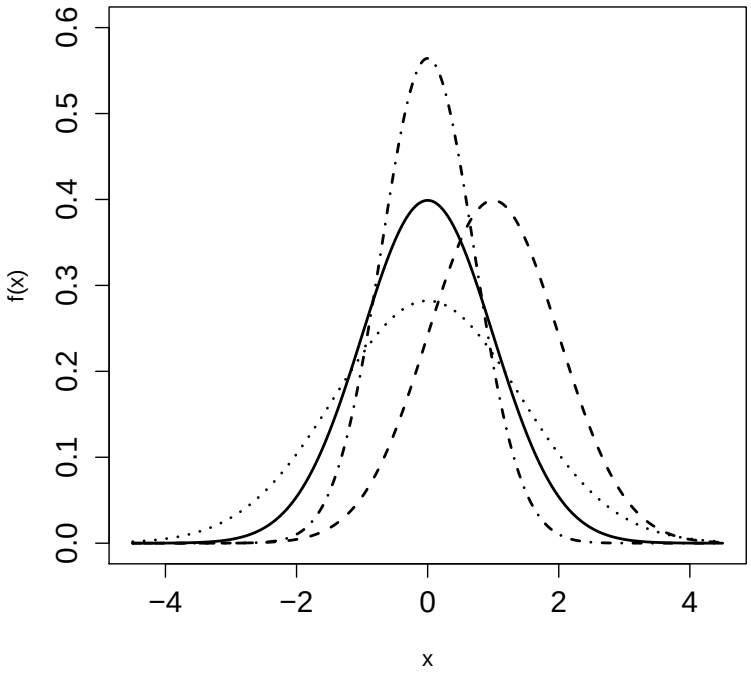
2.8 Normal distribution

The normal or Gaussian distribution is characterized by a bell-shaped density function. The distribution is specified by giving the mean value μ and the variance σ^2. Note that R requires the

standard deviation σ instead of σ^2.

The following lines of code are used to compare the standard normal distribution, namely with $\mu = 0$ and $\sigma^2 = 1$ (solid line), the distribution with $\mu = 1$ and the same variance (dashed line), the distribution with mean $\mu = 0$ and double the variability, that is $\sigma^2 = 2$ (dotted line) and the distribution with mean $\mu = 0$ and half the variability, that is $\sigma^2 = 1/2$ (dash-dotted line).

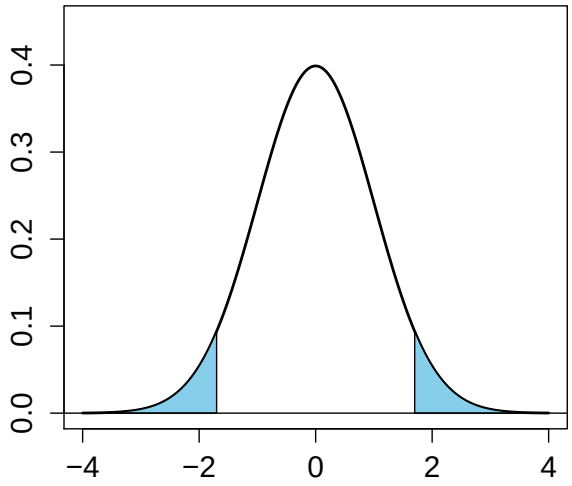
```
xx<-seq(-4.5,4.5,0.01)
plot(xx,dnorm(xx,0,1),ylim=c(0,0.6),
     type='l',lwd=2,cex.axis=1.3,xlab="x",ylab="f(x)") # Step 1
lines(xx,dnorm(xx,1,1),lwd=2,lty=2) # Step 2
lines(xx,dnorm(xx,0,sqrt(2)),lwd=2,lty=3) # Step 3
lines(xx,dnorm(xx,0,sqrt(1/2)),lwd=2,lty=4) # Step 4
```



The following plot gives a clear representation of the symmetry of the Gaussian density function: the two symmetric tails, highlighted using function `polygon`, have the same area.

```
xx<-seq(-4,4,0.01)
plot(xx,dnorm(xx,0,1),ylim=c(0,0.45),type='l',lwd=2,cex.axis=1.3,xlab=" ",ylab=" ")
cord.x <- c(-4,seq(-4,-1.7,0.01),-1.7)
cord.y <- c(0,dnorm(seq(-4,-1.7,0.01)),0)
polygon(cord.x,cord.y,col='skyblue')
cord.x <- c(1.7,seq(1.7,4,0.01),4)
```

```
cord.y <- c(0,dnorm(seq(1.7,4,0.01)),0)
polygon(cord.x,cord.y,col='skyblue')
abline(0,0)
```



The symmetry of the density function can be further verified using the following commands, related to the normal distribution. With regard to the standard normal distribution, we find that the α-level and the $(1-\alpha)$-level quantiles, with $\alpha \in [0, 0.5]$, correspond to opposite support values.

```
p <- 0.2
qnorm(p, 0, 1)

[1] -0.8416212

qnorm(1-p, 0, 1)

[1] 0.8416212
```

Moreover, opposite support values give distribution function levels which sum up to one, while the corresponding values for the density function are equal.

```
z <- 1.96
pnorm(z, 0, 1)

[1] 0.9750021

1 - pnorm(-z, 0, 1)
```

```
[1] 0.9750021

z <- 1.96
dnorm(z, 0, 1)

[1] 0.05844094

dnorm(-z, 0, 1)

[1] 0.05844094
```

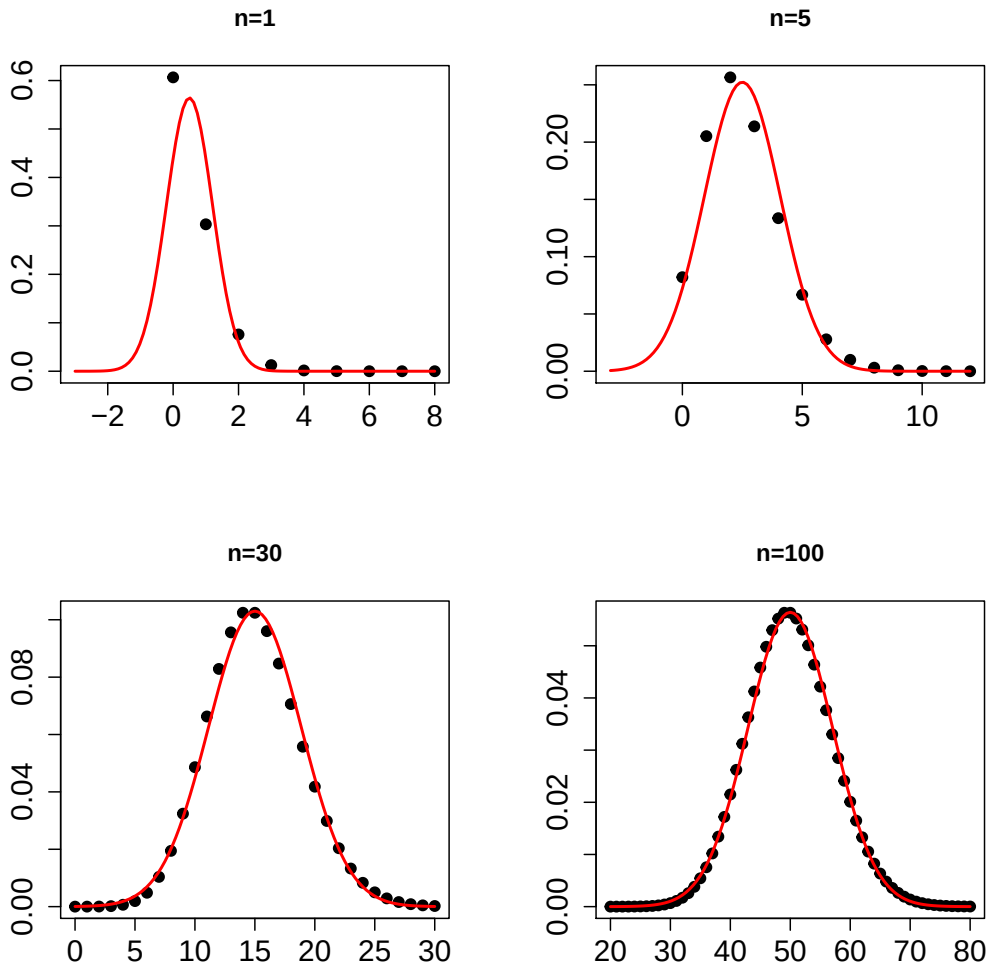
2.9 The central limit theorem

The central limit theorem states that, under suitable assumptions, the limit distribution, as the sample size increases, of the mean and of the sum of independent, identically distributed random variables is Gaussian. We shall present three simple applications concerning samples from the Poisson and the Bernoulli distribution and samples from an skewed bimodal distribution.

It is well-known that a Poisson distribution with parameter $n\lambda$ can be interpreted as the distribution of the sum of n independent Poisson distributed random variables with parameter λ. Then, according to the central limit theorem, as n increases the Poisson distribution gets closer to a Gaussian distribution with mean $\mu = n\lambda$ and variance $\sigma^2 = n\lambda$. In order to visualize this result, the probability function of four Poisson distributions with $\lambda = 0.5$ and $n = 1, 5, 30, 100$ are plotted along with the corresponding normal density approximation. Here we consider function `curve`, using options which are very similar to those ones defined for the `plot` function.

```
par(mfrow=c(2,2))
xx<-seq(0,8,1)
plot(xx,dpois(xx,0.5),pch=19,lwd=2,xlim=c(-3,8),cex.axis=1.5,
     xlab=" ",ylab=" ",main="n=1")
curve(dnorm(x,0.5,sqrt(0.5)),-3,8,lwd=2,col='red',add=T)

xx<-seq(0,12,1)
plot(xx,dpois(xx,0.5*5),pch=19,lwd=2,xlim=c(-3,12),cex.axis=1.5,
     xlab=" ",ylab=" ",main="n=5")
curve(dnorm(x,0.5*5,sqrt(0.5*5)),-3,12,lwd=2,col='red',add=T)

xx<-seq(0,30,1)
plot(xx,dpois(xx,0.5*30),pch=19,lwd=2,xlim=c(0,30),cex.axis=1.5,
     xlab=" ",ylab=" ",main="n=30")
curve(dnorm(x,0.5*30,sqrt(0.5*30)),0,30,lwd=2,col='red',add=T)

xx<-seq(20,80,1)
```

```
plot(xx,dpois(xx,0.5*100),pch=19,lwd=2,xlim=c(20,80),cex.axis=1.5,
     xlab=" ",ylab=" ",main="n=100")
curve(dnorm(x,0.5*100,sqrt(0.5*100)),20,80,lwd=2,col='red',add=T)
```

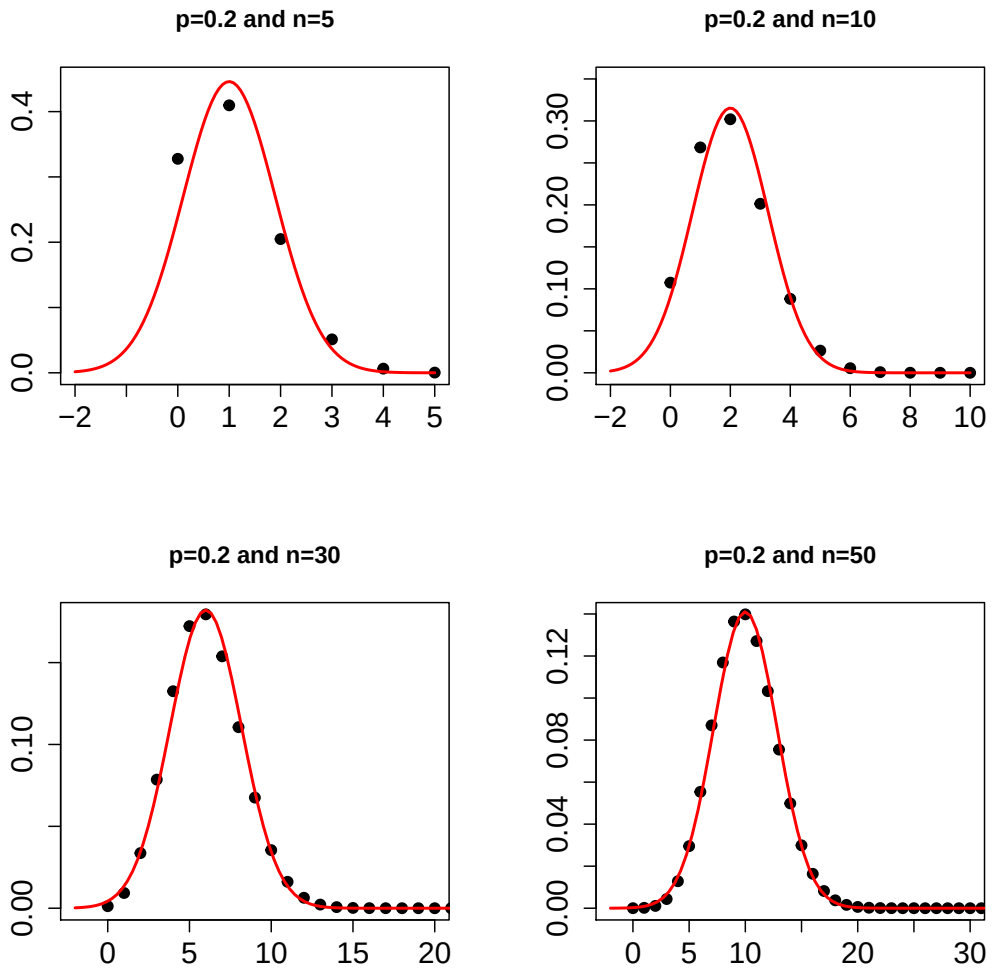



```
par(mfrow=c(1,1))
```

The sum of $n \geq 1$ independent Bernoulli random variables, with common success probability p, follows a binomial distribution with parameters n and p. As a simple application of the central limit theorem, we aim at comparing the binomial probability function with the approximating normal density with $\mu = np$ and $\sigma^2 = np(1-p)$. In particular, we consider $p = 0.2$ and $n = 5, 10, 30, 50$ and we show that the approximation improves as n increases.

```
par(mfrow=c(2,2))
n <- 5
p <- 0.2
```

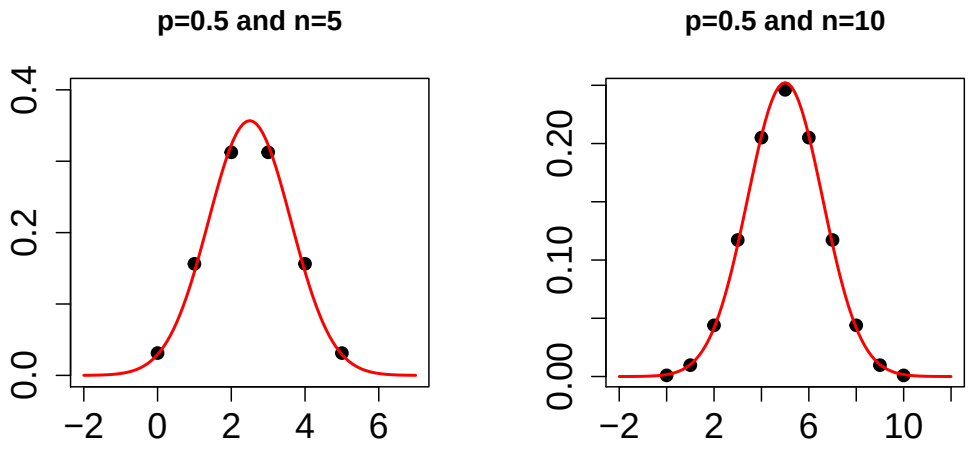
```r
plot(0:n,dbinom(0:n,n,p),pch=19,lwd=2,xlim=c(-2,n),ylim=c(0,0.45),cex.axis=1.5,
     xlab=" ",ylab=" ",main="p=0.2 and n=5")
curve(dnorm(x,n*p,sqrt(n*p*(1-p))),-2,n,lwd=2,col='red',add=T)
n <- 10
plot(0:n,dbinom(0:n,n,p),pch=19,lwd=2,xlim=c(-2,n),ylim=c(0,0.35),cex.axis=1.5,
     xlab=" ",ylab=" ",main="p=0.2 and n=10")
curve(dnorm(x,n*p,sqrt(n*p*(1-p))),-2,n,lwd=2,col='red',add=T)
n <- 30
plot(0:n,dbinom(0:n,n,p),pch=19,lwd=2,xlim=c(-2,n-10),cex.axis=1.5,
     xlab=" ",ylab=" ",main="p=0.2 and n=30")
curve(dnorm(x,n*p,sqrt(n*p*(1-p))),-2,n,lwd=2,col='red',add=T)
n <- 50
plot(0:n,dbinom(0:n,n,p),pch=19,lwd=2,xlim=c(-2,n-20),cex.axis=1.5,
     xlab=" ",ylab=" ",main="p=0.2 and n=50")
curve(dnorm(x,n*p,sqrt(n*p*(1-p))),-2,n,lwd=2,col='red',add=T)
```



```
par(mfrow=c(1,1))
```

If we consider $p = 0.5$, the Bernoulli distribution turns out to be symmetric and the convergence to the limiting Gaussian distribution is very fast.

```
par(mfrow=c(1,2))
n <- 5
p <- 0.5
plot(0:n,dbinom(0:n,n,p),pch=19,lwd=2,ylim=c(0,0.4),xlim=c(-2,n+2),
     cex.axis=1.5,xlab=" ",ylab=" ",main="p=0.5 and n=5")
curve(dnorm(x,n*p,sqrt(n*p*(1-p))),-2,n+2,lwd=2,col='red',add=T)
n <- 10
plot(0:n,dbinom(0:n,n,p),pch=19,lwd=2,xlim=c(-2,n+2),
     cex.axis=1.5,xlab=" ",ylab=" ",main="p=0.5 and n=10")
curve(dnorm(x,n*p,sqrt(n*p*(1-p))),-2,n+2,lwd=2,col='red',add=T)
```

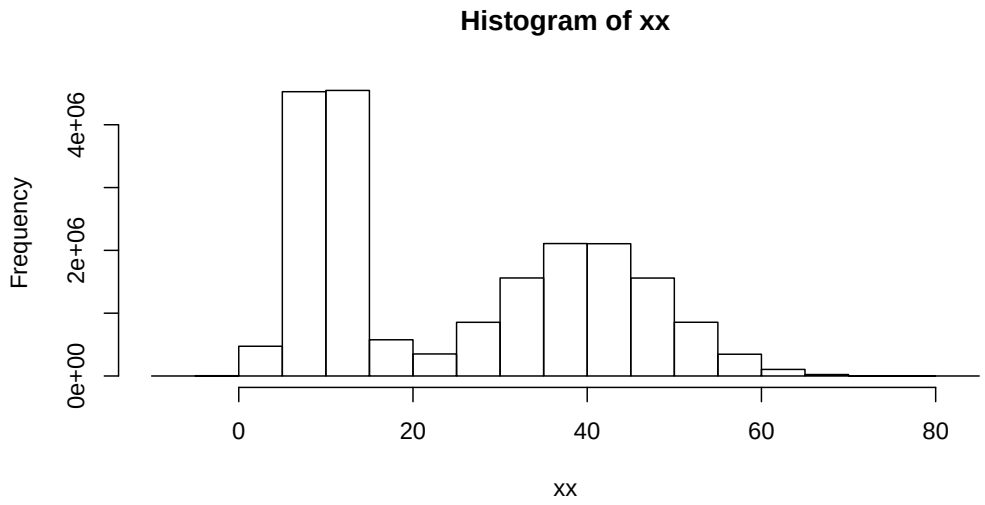


In the last application, we shall consider a random sample based on independent observations from a somewhat peculiar distribution. The analysis is based on the following steps:

1. define (in this case, empirically, by means of a simulation procedure) a peculiar distribution with a bimodal and asymmetric density function;
2. prepare the graphical device;
3. set the number of random samples (1000 replications, in this case) to be considered in order to obtain values for the sample mean;
4. set the sample size ($n = 2, 5, 15, 40$ for the four cases);
5. select the samples and compute the values for the sample mean;

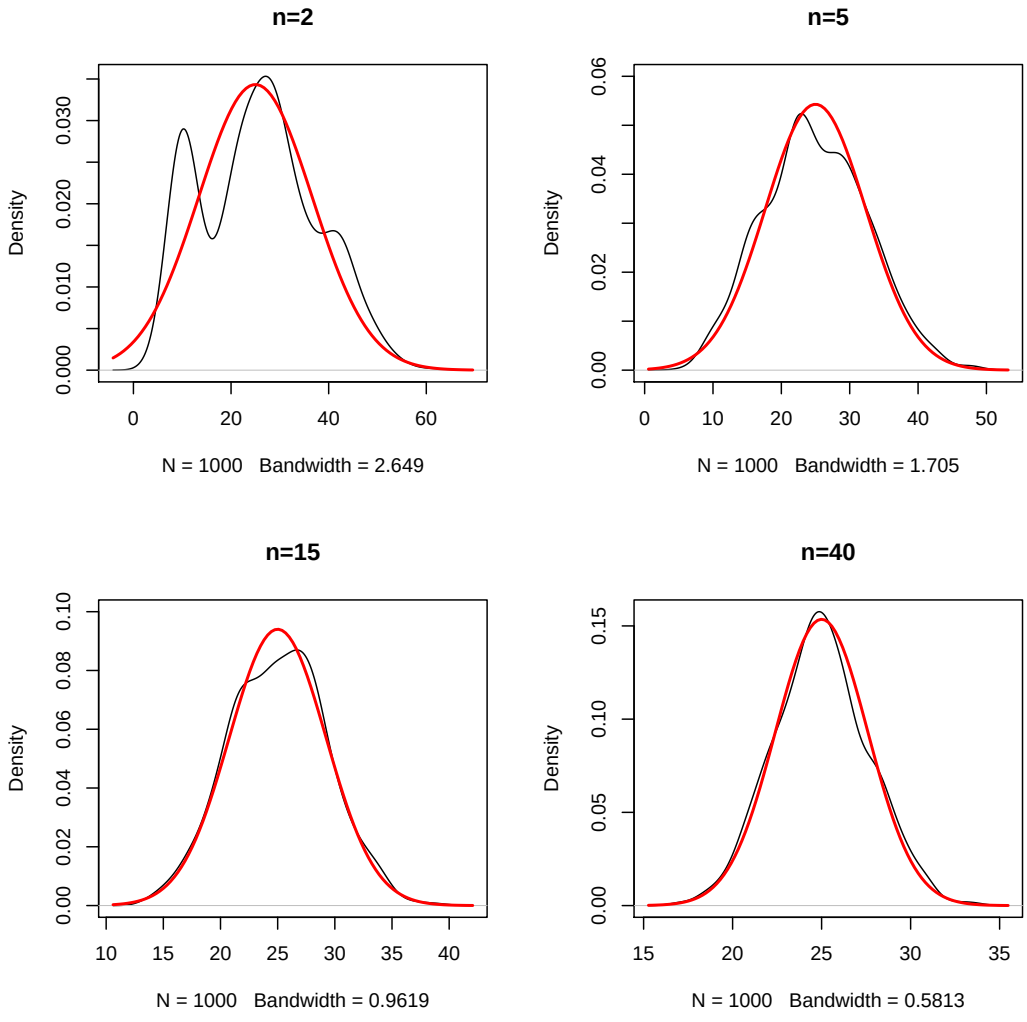
6. draw the kernel density estimation for the sample mean (interpreted as a nearly exact approximation of the true density), to be compared with the approximating Gaussian density (in red).

```
set.seed(25)
xx <- c(rnorm(10000000,10,3),rnorm(10000000,40,9)) # Step 1
hist(xx)
```




```
par(mfrow=c(2,2)) # Step 2

nsamples <- 1000 # Step 3

sampmean <- NULL # Step 4
size <- 2
set.seed(21)
for(i in 1:nsamples){sampmean <- c(sampmean,
                                   mean(sample(xx,size,replace=T)))} # Step 5
plot(density(sampmean), main="n=2") # Step 6
curve(dnorm(x,mean(xx),sqrt(var(xx)/size)),lwd=2,col='red',add=T)

sampmean <- NULL # Step 4
size <- 5
set.seed(25)
for(i in 1:nsamples){sampmean <- c(sampmean,
                                   mean(sample(xx,size,replace=T)))} # Step 5
plot(density(sampmean),ylim=c(0,0.06), main="n=5") # Step 6
curve(dnorm(x,mean(xx),sqrt(var(xx)/size)),lwd=2,col='red',add=T)
```

```r
sampmean <- NULL # Step 4
size <- 15
set.seed(22)
for(i in 1:nsamples){sampmean <- c(sampmean,
                                   mean(sample(xx,size,replace=T)))} # Step 5
plot(density(sampmean),ylim=c(0,0.1), main="n=15") # Step 6
curve(dnorm(x,mean(xx),sqrt(var(xx)/size)),lwd=2,col='red',add=T)

sampmean <- NULL # Step 4
size <- 40
set.seed(25)
for(i in 1:nsamples){sampmean <- c(sampmean,
                                   mean(sample(xx,size,replace=T)))} # Step 5
plot(density(sampmean), main="n=40") # Step 6
curve(dnorm(x,mean(xx),sqrt(var(xx)/size)),lwd=2,col='red',add=T)
```

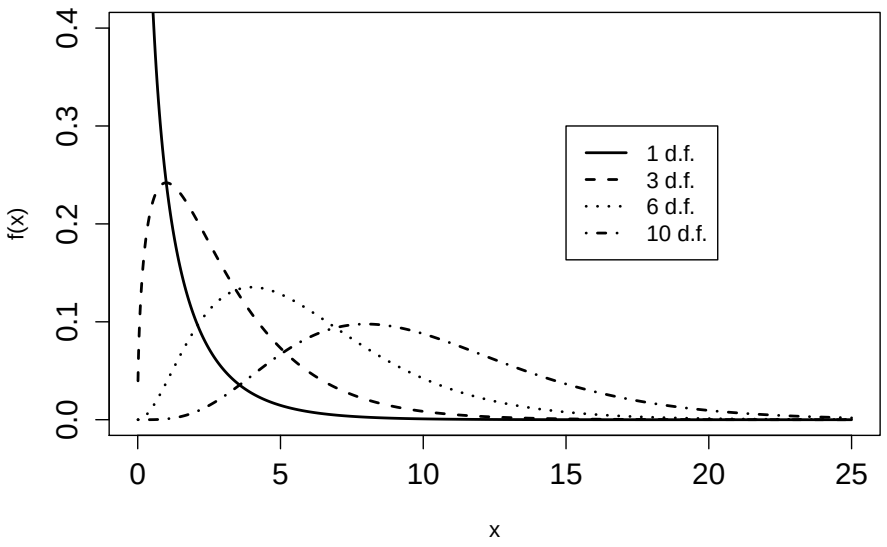


```
par(mfrow=c(1,1))
```

2.10 Chi-squared distribution

The chi-squared distribution, with parameter (degrees of freedom) $k \geq 1$, is specified by a continuous density function on non-negative support values. The density functions, with $1, 3, 6, 10$ degrees of freedom (d.f.), are generated by the following code.

```
xx<-seq(0.01,25,0.01)
plot(xx,dchisq(xx,1),ylim=c(0,0.4),xlim=c(0,25),type='l',lwd=2,
     cex.axis=1.3,xlab="x",ylab="f(x)")
lines(xx,dchisq(xx,3),lwd=2,lty=2)
lines(xx,dchisq(xx,6),lwd=2,lty=3)
lines(xx,dchisq(xx,10),lwd=2,lty=4)
legend(15,0.3,legend=c("1 d.f.","3 d.f.","6 d.f.","10 d.f."), lty=1:4, lwd=2)
```

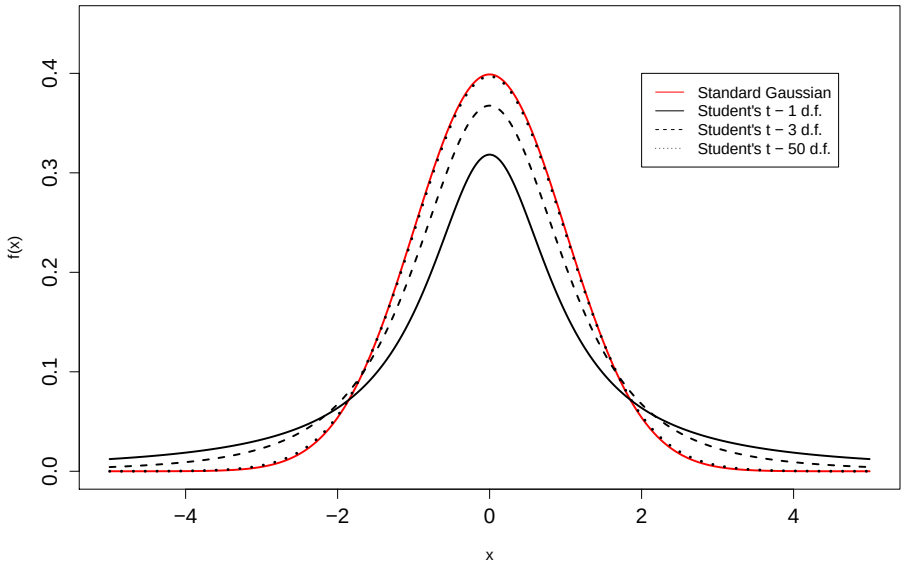



```
par(mfrow=c(1,1))
```

2.11 Student's t distribution

The Student's t distribution , with parameter (degrees of freedom) $k \geq 1$, is specified by a continuous density function which is similar to the Gaussian one but with heavier tails (the extreme values are less rare). When the degrees of freedom tend to infinity the Student's t distribution tends to the standard normal distribution. The density functions, with $1, 3, 50$ degrees of freedom, are generated by the following code, and drawn with the standard Gaussian density (in red).

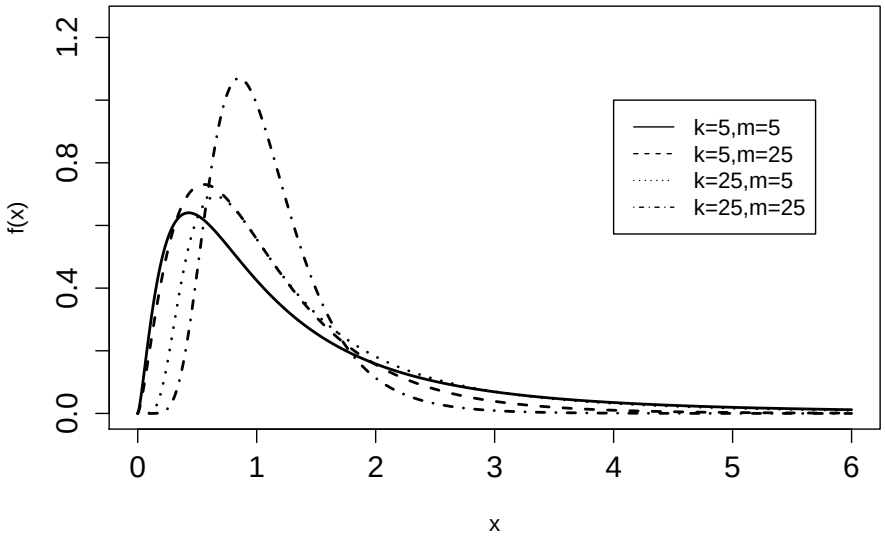
```
xx<-seq(-5,5,0.01)
plot(xx,dnorm(xx),type='l',ylim=c(0,0.45),lwd=2,cex.axis=1.3,xlab="x",
     ylab="f(x)", col='red')
lines(xx,dt(xx,1),lwd=2)
lines(xx,dt(xx,3),lwd=2,lty=2)
lines(xx,dt(xx,50),lwd=3,lty=3)
legend(2,.4,legend=c("Standard Gaussian","Student's t - 1 d.f.",
                     "Student's t - 3 d.f.","Student's t - 50 d.f."),
                     lty=c(1,1,2,3), col=c(2,1,1,1))
```



2.12 *F* distribution

The F distribution, also known as the Fisher or the Snedecor distribution, is a model for a continuous random variable defined as the ratio between two independent chi-squared distributed random variables divided by their degrees of freedom. The parameters of an F distribution are the degrees of freedom of the two component chi-squared random variables. Density functions for an F distribution with different degrees of freedom are generated by the following code.

```
xx<-seq(0,6,0.01)
plot(xx,df(xx,5,5),type='l',ylim=c(0,1.25),lwd=2,
     cex.axis=1.3,xlab="x",ylab="f(x)",
     main=" ")
lines(xx,df(xx,5,25),lwd=2,lty=2)
lines(xx,df(xx,25,5),lwd=2,lty=3)
lines(xx,df(xx,25,25),lwd=2,lty=4)
legend(4,1,legend=c("k=5,m=5","k=5,m=25","k=25,m=5","k=25,m=25"), lty=1:4)
```



3 Simulation of random samples

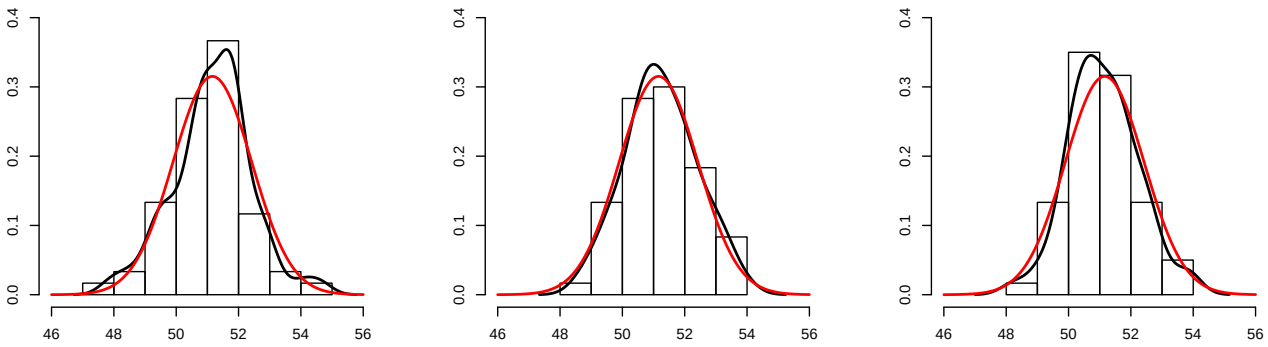
3.1 Simulations from a normal distribution

Let us consider the data set `nhtemp`, containing the mean annual temperatures in New Haven. The following code is used to compare the original data set and two simulated data sets obtained from a Gaussian distribution (considered as a plausible model for the observed phenomenon) with mean and variance equal to the sample mean and the sample variance based on the available data.

The left panel gives the histogram and the kernel density estimator based on the original data. The central and the right panels give the histogram and the kernel density estimation based on two distinct data sets simulated from the assumed Gaussian model (which density is given by the red line). The original and the simulated data sets have the same dimension $n = 60$. The histograms and the kernel density estimators presents a similar behavior in both the situations.

```
par(mfrow=c(1,3),pty="s")
x <- seq(46,56,0.05)
mean(nhtemp)

[1] 51.16

var(nhtemp)

[1] 1.601763
```

```
hist(nhtemp,freq=F,xlim=c(46,56),ylim=c(0,0.45),main=' ',xlab=' ',ylab=' ')
lines(density(nhtemp),lwd=2)
lines(x,dnorm(x,mean(nhtemp),sqrt(var(nhtemp))),col='red',lwd=2)
set.seed(123)
for(i in 1:2){
  y <- rnorm(60,mean(nhtemp),sqrt(var(nhtemp)))
  hist(y,freq=F,main=' ',xlim=c(46,56),ylim=c(0,0.45),xlab=' ',ylab=' ')
  lines(density(y),lwd=2)
  lines(x,dnorm(x,mean(nhtemp),sqrt(var(nhtemp))),col='red',lwd=2)
}
```




```
par(mfrow=c(1,1))
```

3.2 Simulation of the Bernoulli sample mean

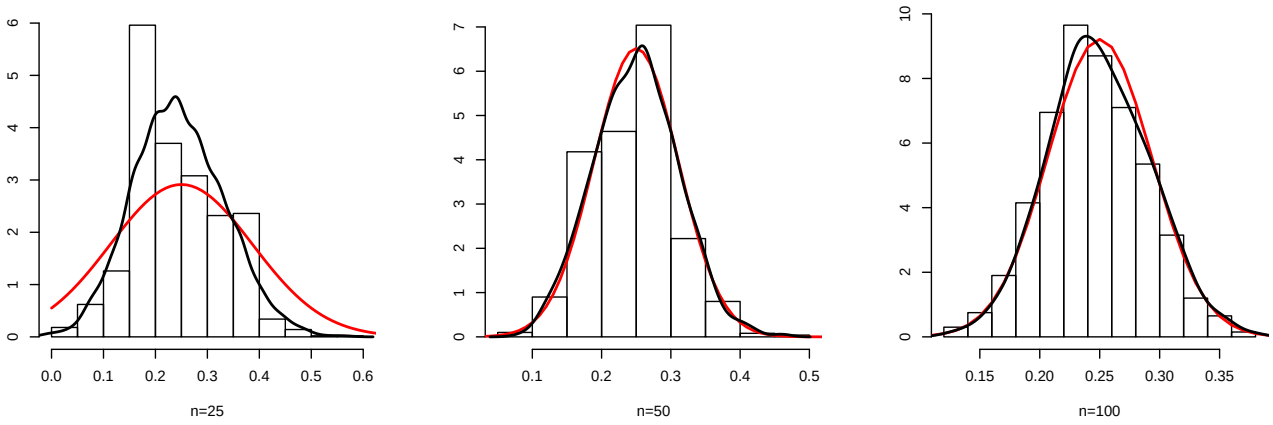
In the following simulation study we analyze the behavior of the (empirical) distribution of the sample mean, as the sample size increases. According to the central limit theorem, the empirical distribution is expected to converge to a suitable Gaussian distribution.

Firstly, 1000 random samples of size $n = 25, 50, 100$ are simulated from a Bernoulli distribution with $p = 0.25$, using the function `rbinom`.

```
set.seed(4)
x <- seq(0,1.5,0.01)
sim1<-rbinom(1000,25,0.25)/25 # 1000 simulated sample means with n=25
sim2<-rbinom(1000,50,0.25)/50 # 1000 simulated sample means with n=50
sim3<-rbinom(1000,100,0.25)/100 # 1000 simulated sample means with n=100
```

Then, for each case, the empirical distribution of the sample mean is estimated using the histogram and the kernel density estimation procedure. The results are reported in the following plots with the approximating Gaussian distribution (in red).

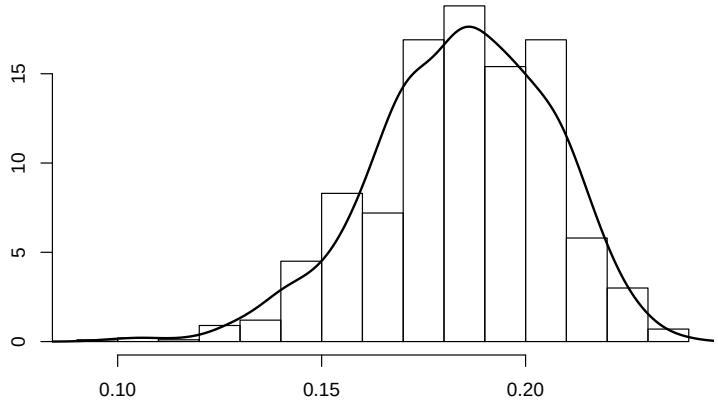
```
par(mfrow=c(1,3),pty="s")
hist(sim1,freq=F,xlab="n=25",ylab=' ',main=' ')
lines(x,dnorm(x,0.25,sqrt(0.25*0.75/10)),lwd=2,col='red')
lines(density(sim1),lwd=2)
hist(sim2,freq=F,xlab="n=50",ylab=' ',main=' ')
lines(x,dnorm(x,0.25,sqrt(0.25*0.75/50)),lwd=2,col='red')
lines(density(sim2),lwd=2)
hist(sim3,freq=F,xlab="n=100",ylab=' ', main=' ')
lines(x,dnorm(x,0.25,sqrt(0.25*0.75/100)),lwd=2,col='red')
lines(density(sim3),lwd=2)
```




```
par(mfrow=c(1,1))
```

In this second simulation study, the aim is to estimate the sampling distribution of the Bernoulli sample variance using both the histogram and the kernel density approach. Since the variance of a Bernoulli distribution is $p(1-p)$, the sample variance is simply obtained by substituting p with the sample mean. From a Bernoulli distribution with $p = 0.25$, 1000 random samples of size $n = 100$ are generated and for each sample the sample mean and the Bernoulli sample variance are obtained. The unknown density of the Bernoulli sample variance is estimated by histogram and the kernel density estimation drawn in the following plot.

```
set.seed(4)
sim<-rbinom(1000,100,0.25)/100 # 1000 simulated sample means with n=100
varest <- sim*(1-sim)
hist(varest,freq=F,xlab=" ",ylab=' ',main=' ')
lines(density(varest),lwd=2)
```

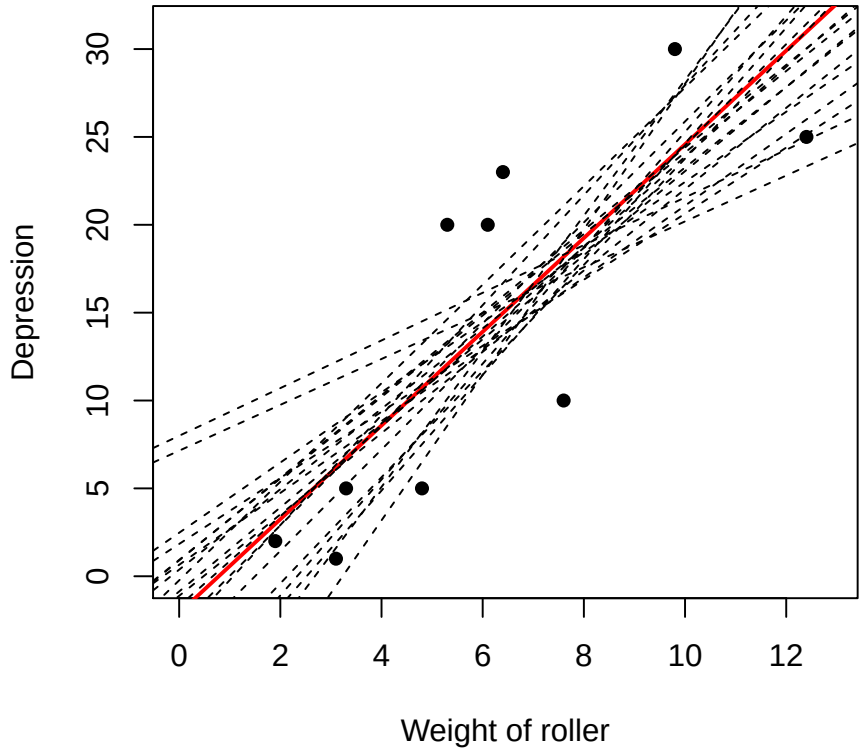


3.3 Simulation of regression data

In this application we simulate observations from a simple linear regression model, estimated using the `roller` data set. This simulation of regression data is developed in order to observe the variability of the model parameters estimates, and then of the regression line. The analysis is based on the following steps:

1. the scatterplot for the `roller` data is drawn;
2. the regression line is estimated and plotted in the graph;
3. the estimated regression model is used to generate 20 regression data sets, saved in the matrix `roller.sim`; to this end, we consider the function `simulate`;
4. the regression lines, estimated using the simulated data sets, are included in the original scatterplot by means of a suitable `for` cycle.

```
plot(depression ~ weight, data = roller, xlim=c(0,1.04*max(weight)), # Step 1
     ylim=c(0,1.04*max(depression)), xlab = 'Weight of roller',
     ylab = 'Depression', pch = 16)

roller.lm <- lm(depression ~ weight,data=roller) # Step 2
abline(roller.lm,col='red',lwd=2)

roller.sim <- simulate(roller.lm,nsim=20) # Step 3
for(i in 1:20) { # Step 4
abline(lm(roller.sim[,i]~ roller$weight),lty=2)
}
```



4 Model assumptions

Many statistical methods are based on the assumption that the data are normally distributed. The assumption that an interest phenomenon can be conveniently described by means of a suitable class of distributions, in particular the class of the Gaussian models, may be checked considering both graphical tools and more formal numerical procedures.

4.1 Example: onions data

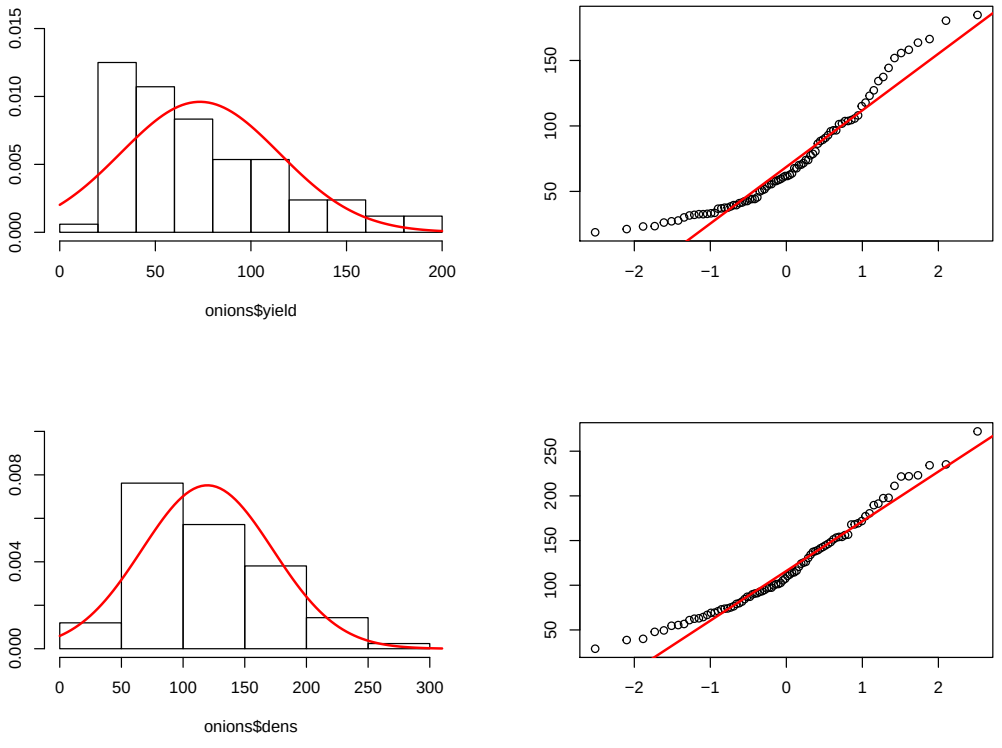
The `onions` data set comes from an experiment which aims at measuring the productivity of white Spanish onions in two South Australian locations. The sample size is $n = 84$ and the data set contains measurements for the areal density of plants (plant per m^2) and for the onion yield (gr per plant). In what follows the onions data set is analysed by considering the original data and their logarithmic transformation.

With regard to the original data set, the analysis is based on the following steps:

1. the data set, available in the working directory, is imported using the `read.table` function, where the column names are specified using the option `col.names`;
2. the Gaussian model assumed for the areal density (`yield`) is checked by considering the histogram and the quantile-quantile plot; this second plot is obtained by combining the `qqnorm` and `qqline` functions;
3. the same results are obtained for the plants density (`dens`).

We draw also the normal density estimated from the available data (in red) and it is immediate to conclude that, in both cases, the skewness in the empirical distribution is evident.

```
onions<-read.table("onions.dat", col.names=c("yield","dens","location")) # Step 1
par(mfrow=c(2,2))

hist(onions$yield,freq=F,main=' ',ylim=c(0,0.016),xlim=c(0,200),ylab=' ') # Step 2a
curve(dnorm(x,mean(onions$yield),sqrt(var(onions$yield))),0,200,add=T,lwd=2,col='red')
qqnorm(onions$yield,main=' ',xlab=' ',ylab=' ') # Step 2b
qqline(onions$yield,col='red',lwd=2)

hist(onions$dens,freq=F,main=' ',ylim=c(0,0.010),xlim=c(0,310),ylab=' ') # Step 3a
curve(dnorm(x,mean(onions$dens),sqrt(var(onions$dens))),0,310,add=T,lwd=2,col='red')
qqnorm(onions$dens,main=' ',xlab=' ',ylab=' ') # Step 3b
qqline(onions$dens,col='red',lwd=2)
```



```
par(mfrow=c(1,1))
```

The hypothesis that a Gaussian distribution gives a convenient representation of the interest phenomenon can be studied also by considering suitable statistical tests for normality. Some of them are included in the base R distribution, while further tests are specified in the `fBasics` library. We shall consider the Kolmogorov-Smirnov normality test, the Shapiro-Wilk's test for normality, the Jarque-Bera test for normality and the D'Agostino normality test, which allows to distinguish between non-normality due to skewness and due to kurtosis.
An in-depth presentation of statistical testing procedures will be considered in what follows. In order to interpret the following results, whenever the p-value is low (for example, lower than 0.05 or 0.01) there is a strong empirical evidence against the distributional assumption under investigation.

```
ks.test(onions$yield,pnorm)
```

```
Warning in ks.test(onions$yield, pnorm):  ties should not be present for the Kolmogorov-Smirn
test
```

```

One-sample Kolmogorov-Smirnov test

data:  onions$yield
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
ks.test(onions$dens,pnorm)
```

```

One-sample Kolmogorov-Smirnov test

data:  onions$dens
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
# install.packages("fBasics", repos="https://cran.rstudio.com/")
library(fBasics)
```

```
Loading required package:  timeDate
Loading required package:  timeSeries
```

```
# ksnormTest(onions$yield)
# ksnormTest(onions$dens)
shapiroTest(onions$yield)
```

```

Title:
 Shapiro - Wilk Normality Test
```

```
Test Results:
  STATISTIC:
    W: 0.9107
  P VALUE:
    2.328e-05

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
shapiroTest(onions$dens)
```

```

Title:
 Shapiro - Wilk Normality Test

Test Results:
  STATISTIC:
    W: 0.9589
  P VALUE:
    0.008958

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
jarqueberaTest(onions$yield)
```

```

Title:
 Jarque - Bera Normalality Test

Test Results:
  STATISTIC:
    X-squared: 11.4967
  P VALUE:
    Asymptotic p Value: 0.003188

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
jarqueberaTest(onions$dens)
```

```

Title:
 Jarque - Bera Normalality Test
```

```
Test Results:
  STATISTIC:
    X-squared: 6.0026
  P VALUE:
    Asymptotic p Value: 0.04972

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
dagoTest(onions$yield)
```

```

Title:
 D'Agostino Normality Test

Test Results:
  STATISTIC:
    Chi2 | Omnibus: 10.5133
    Z3   | Skewness: 3.2242
    Z4   | Kurtosis: 0.3431
  P VALUE:
    Omnibus  Test: 0.005213
    Skewness Test: 0.001263
    Kurtosis Test: 0.7315

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
dagoTest(onions$dens)
```

```

Title:
 D'Agostino Normality Test

Test Results:
  STATISTIC:
    Chi2 | Omnibus: 5.9171
    Z3   | Skewness: 2.4321
    Z4   | Kurtosis: -0.0425
  P VALUE:
    Omnibus  Test: 0.0519
    Skewness Test: 0.01501
    Kurtosis Test: 0.9661

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

Additional functions for testing normality can be found in the `nortest` package, with particular regard to the Anderson-Darling normality test, the Cramer-von Mises normality test, the Lilliefors (Kolmogorov-Smirnov) normality test, the Pearson chi-squared normality test and the Shapiro-Francia normality test.

```
# install.packages("nortest", repos="https://cran.rstudio.com/")
library(nortest)

# adTest(onions£yield) # it's not working on my computer
# adTest(onions£dens)  # it's not working on my computer
cvmTest(onions$yield)


Title:
 Cramer - von Mises Normality Test

Test Results:
  STATISTIC:
    W: 0.3612
  P VALUE:
    6.53e-05

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente

cvmTest(onions$dens)


Title:
 Cramer - von Mises Normality Test

Test Results:
  STATISTIC:
    W: 0.1647
  P VALUE:
    0.01483

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente

lillieTest(onions$yield)


Title:
 Lilliefors (KS) Normality Test
```

```
Test Results:
  STATISTIC:
    D: 0.127
  P VALUE:
    0.00191

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
lillieTest(onions$dens)
```

```


Title:
 Lilliefors (KS) Normality Test

Test Results:
  STATISTIC:
    D: 0.1025
  P VALUE:
    0.02929

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
pchiTest(onions$yield)
```

```


Title:
 Pearson Chi-Square Normality Test

Test Results:
  PARAMETER:
    Number of Classes: 12
  STATISTIC:
    P: 32.8571
  P VALUE:
    Adhusted: 0.0001414
    Not adjusted: 0.0005549

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
pchiTest(onions$dens)
```

```
Title:
 Pearson Chi-Square Normality Test

Test Results:
  PARAMETER:
    Number of Classes: 12
  STATISTIC:
    P: 10.8571
  P VALUE:
    Adhusted: 0.2856
    Not adjusted: 0.4553

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
sfTest(onions$yield)
```

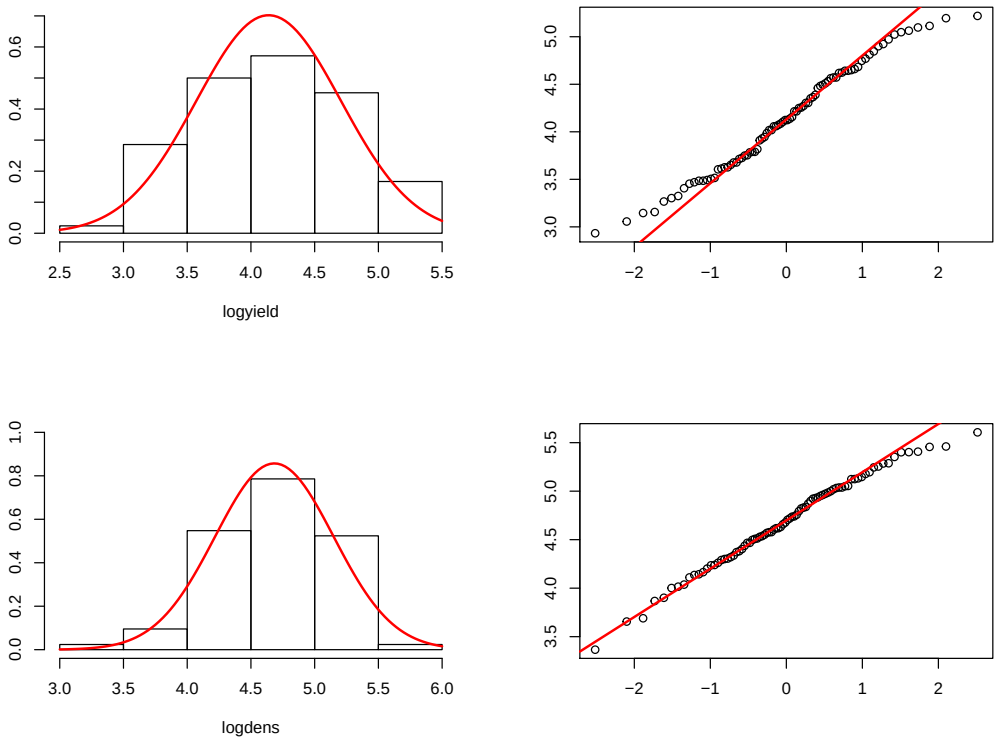
```
Title:
 Shapiro - Francia Normality Test

Test Results:
  STATISTIC:
    W: 0.9154
  P VALUE:
    0.0001038

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

```
sfTest(onions$dens)
```

```
Title:
 Shapiro - Francia Normality Test

Test Results:
  STATISTIC:
    W: 0.9614
  P VALUE:
    0.01486

Description:
 Thu Sep 26 12:38:11 2019 by user: Utente
```

The same graphical and numerical analyses can be replicated for the logarithmic transformation of the two data sets. The only difference in the code is the specification of the vectors `logyield` and `logdens`, containing the data in the log-scale. With regard to the statistical tests for normality, only the D'Agostino normality test is taken into account.

```
onions<-read.table("onions.dat", col.names=c("yield","dens","location"))
logyield<-log(onions$yield)
logdens<-log(onions$dens)
par(mfrow=c(2,2))
hist(logyield,freq=F,main=' ',ylim=c(0,0.7),xlim=c(2.5,5.5),ylab=' ')
curve(dnorm(x,mean(logyield),sqrt(var(logyield))),2.5,5.5,add=T,lwd=2,col='red')
qqnorm(logyield,main=' ',xlab=' ',ylab=' ')
qqline(logyield,col='red',lwd=2)
hist(logdens,freq=F,main=' ',ylim=c(0,1),xlim=c(3,6),ylab=' ')
curve(dnorm(x,mean(logdens),sqrt(var(logdens))),3,6,add=T,lwd=2,col='red')
qqnorm(logdens,main=' ',xlab=' ',ylab=' ')
qqline(logdens,col='red',lwd=2)
```




```
par(mfrow=c(1,1))
```

The qqplot for the `logyield` vector identifies the presence of thinner tails than those of the normal density. This conclusion is confirmed by the result of the corresponding D'Agostino's test of

normality, since the p-value suggests to reject the null hypothesis of normality, emphasizing that the rejection is due to the presence of a negative excess of kurtosis, called platykurtosis. The same analysis, performed on the `logdens` vector, assures that, in this case, the normality assumption can be considered as valid.

```
dagoTest(logyield)


Title:
 D'Agostino Normality Test

Test Results:
  STATISTIC:
    Chi2 | Omnibus: 6.7649
    Z3  | Skewness: 0.0306
    Z4  | Kurtosis: -2.6008
  P VALUE:
    Omnibus  Test: 0.03396
    Skewness Test: 0.9756
    Kurtosis Test: 0.009302

Description:
 Thu Sep 26 12:38:12 2019 by user: Utente

dagoTest(logdens)


Title:
 D'Agostino Normality Test

Test Results:
  STATISTIC:
    Chi2 | Omnibus: 1.674
    Z3  | Skewness: -1.2671
    Z4  | Kurtosis: -0.2617
  P VALUE:
    Omnibus  Test: 0.433
    Skewness Test: 0.2051
    Kurtosis Test: 0.7936

Description:
 Thu Sep 26 12:38:12 2019 by user: Utente
```