

An Introduction to R

Thorsten Schmidt*

April 24, 2017

*Department of Mathematical Stochastics, University of Freiburg
web: www.stochastik.uni-freiburg.de/schmidt
mail: thorsten.schmidt@stochastik.uni-freiburg.de

I would like to thank the students of the QRM course in Madrid, 2012 for their interest and suggestions.

1 First steps

We try some first inputs in the command line of R. For the first time, type `17 + 2` and press enter and you will see the following:

```
> 17 + 2
[1] 19
```

The output `[1] 19` says that this is the first output, and the output is 19. If you do it again you will get `[2] 19` and so on. In the following the sign `>` will show you what to type and the following line(s) is the output.

R takes automatically control of brackets:

```
> sqrt (pi*x ^2
+          5)
```

The `+` sign shows that R expects you to finish the line. It appears by pressing enter after 2.

Storing data in variables can be done as follows:

```
> x <- 119 + 2
```

There is no output. But now 121 is stored in x and you can bring it up as follows:

```
> x
[1] 121
```

In the following we leave aside the outputs from R and just give the commands. Typically we use a table for assessing R commands as follows:

Objects in R	
<code>objects()</code>	Gives you a list of currently used storage variables, or objects. For example [1] "a" "b" "x"
<code>objects</code>	As this is a function, calling it without brackets give you the definition of the function. This is very helpful in accessing what happens on calling functions, in particular the ones which you write yourself.
<code>rm(x, y)</code>	Removes these objects from the workspace.

R has a powerful help. You can invoke it by

```
> help( mean)
```

or get a short cut by

```
> ? mean
```

One of the most often used function is `c`. It concatenates objects:

```
> c(1,2,5,78)
[1] 1 2 5 78
```

1.1.1 More complicated things

Next we experiment with sequences.

```
> 1:10
```

creates a sequence from 1 to 10. There are a number of powerful way to do this. Compute the mean of the first 100 integers.

```
> x <- seq(from=-1, to = 8, by = 0.1)      #This is a comment
> y <- x^2                                  #Store in y
```

1 First steps

And bring it to the screen:

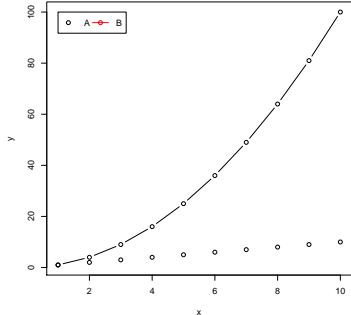
```
> plot(x,y,type = "l",main="The function x^2", xlab="x", ylab="y")
```

If you want to add a further line to an existing plot, you use `lines`:

```
> lines(x,x^3,type = "l",col="blue")
```

Exercise 1. Now get help on the command `plot`, try different functions, like square root and logarithm and so on. Some examples are listed below

The function plot	
<pre>plot(sin)</pre>	Functions are objects which have a direct plotting method. You can find out about all such objects with <code>methods(plot)</code>
<pre>plot(sin(seq(-pi,pi,by=0.01)), type="l",col=8,lwd=4,bty="l")</pre>	<code>type="l"</code> gives lines instead of just circles, <code>col="red"</code> gives the color, <code>lwd</code> : line width, <code>bty="l"</code> specifies the frame: "o" is standard, "n" is none. For legends see below.

Interactive Features	
<pre>x <- 1:10 y <- x^2 plot(x,y) identify(x,y) locator() plot(x,y,type="b") points(x,x,col="red") legend(1,100,legend= c("A","B"), lty=c(-1,1),pch=c(1,1),ncol= 2, col=c(1,2))</pre>	<p>Allows you by clicking on the point to identify the item. Right-Mouse click ends the session.</p> <p>Allows you to identify an arbitrary point in the plot, for example to place a legend at the right place</p> <p>Add a legend to the above graph. The place (1,100) is in the coordinates of the graph! It can simply be picked by using <code>locator</code>. <code>lty</code> gives the line types, -1 is no line, and <code>pch</code> gives the point type.</p>
	

1.1.2 Distributions

Typical distributions are included in the following form:

`ddist(x,) = $f_{\dots}(x)$` : density or probability functions

`pdist(x,) = $F_{\dots}(x)$` : cumulative distribution function: $\mathbb{P}(X \leq x)$ for $X \sim F$

`qdist(y,) = $F_{\dots}^{-1}(y)$` : quantilefunction

`rdist(n,)` : generates n random variables

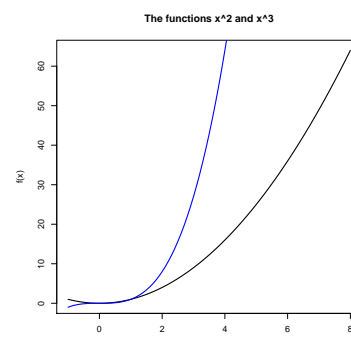
1 First steps

of the distribution *dist*. *dist* can be for example: norm, exp, binom, beta, gamma, t, norm and many others (see table in the appendix). So `dnorm` gives the density of the normal distribution, `pexp` gives $1 - e^{-x}$, the cdf of the Exp(1)-distribution, `qunif` is the inverse cdf of the uniform distribution, i.e. the identity on $[0, 1]$ and `rbeta(100,0.5,0.5)` simulates 100 i.i.d. Beta(0.5,0.5)-distributed random variables.

Exercise 2. Plot the densities function of the normal, the *t* and the Γ distributions. Vary with different parameters and colors in the plot.

For example, you could do the following:

```
x <- seq(-2,2,by = 0.01)
plot(x,dnorm(x),type="l")
lines (x,dnorm(x,mean=1,sd=1.5),
      type="l",col =2)
```



`dnorm(x,1,2)` gives a normal distribution with *standard deviation* of 2 and not variance of 2.

Simulation of random variables in R is very easy. Try for example the following:

Simulations

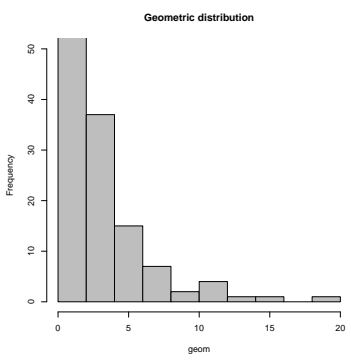
```
data <- rnorm(100)
mean(data)
var(data)
sd(data)
summary (data)
quantile( data, probs=c( 0, 0.2, 0.9))
```

This gives you some statistics on the data.

Exercise 3. You can experiment, plot and so on. Try different distributions like *t* and Γ to get a feeling on them. Try help on `sd`, `var`, `median`, `max`, `min`, `fivenum`, `IQR`.

1.1.3 Further statistics

Compute histograms:¹

<pre>geom <- rgeom(200,0.3) hist(geom) dotchart(geom) hist (geom, ylim=c(0,50), breaks= seq(min(geom),max(geom)+1,by=2), col=8) boxplot(geom)</pre>	<p>Simulate the geometric distribution and get a histogram.</p>  <table border="1"><caption>Approximate data for the Geometric distribution histogram</caption><thead><tr><th>geom value</th><th>Frequency</th></tr></thead><tbody><tr><td>0</td><td>52</td></tr><tr><td>2</td><td>37</td></tr><tr><td>4</td><td>15</td></tr><tr><td>6</td><td>7</td></tr><tr><td>8</td><td>3</td></tr><tr><td>10</td><td>2</td></tr><tr><td>12</td><td>1</td></tr><tr><td>14</td><td>1</td></tr><tr><td>16</td><td>1</td></tr><tr><td>18</td><td>1</td></tr></tbody></table>	geom value	Frequency	0	52	2	37	4	15	6	7	8	3	10	2	12	1	14	1	16	1	18	1
geom value	Frequency																						
0	52																						
2	37																						
4	15																						
6	7																						
8	3																						
10	2																						
12	1																						
14	1																						
16	1																						
18	1																						

Try also `smoothScatter` or the following example on nonparametric density estimation:

¹You can look at Bret Largets introduction: <http://www.stat.wisc.edu/~larget/R/eda-R.pdf> for further explanations and examples.

Kernel estimators

```

x <- rnorm(100)
y <- 2*x +rnorm(100)

plot(x,y)
lines(ksmooth(x,y,kernel="normal",
  bandwidth=1),col="green")
lines(ksmooth(x,y,kernel="normal",
  bandwidth=2.5),col="red")

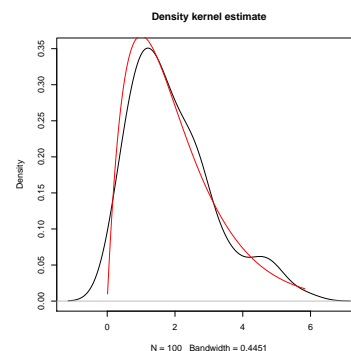
z <- rgamma(100,shape=2)
density(z)

plot(density(z))
x <- seq(0.01,max(z),by=0.01)
lines(x,dgamma(x,2),col="red")

```

This gives the Nadaraya-Watson kernel estimator with different bandwidths chosen.

A nonparametric method of estimating a density.



Now if you feel well with R, go ahead and try some of your favorite simulations.

Exercise 4. Monte-Carlo experiment. What is $\mathbb{E}(\xi^n)$ with $n = 1, 2, 3, 4$ when ξ is normally distributed. Or exponential, Gamma, t . Solve the question by Monte-Carlo:

```

> x <- rnorm(100)
> y <- x^4           #transformation with n=4
> mean(y)           #The Monte Carlo estimate of the 4th moment

```

The background is of course the strong law of large numbers.

If you feel an expert, then try the following (difficult) financial application:

Exercise 5. Simulate a Brownian motion. Simulate a geometric Brownian motion (the Black-Scholes Model). Compute the price of a call option via Monte-Carlo, i.e. simulate 1000 payoffs and compute the mean

1 First steps

Hints: if $(W_t)_{t \geq 0}$ is a Brownian motion, it has independent and stationary increments. Moreover the increments are normally distributed:

$$W_{t+\Delta} - W_t \sim \mathcal{N}(0, \Delta).$$

We obtain a Brownian motion by *accumulating* independent, normally distributed random variables. Use the command `cumsum` for this. Simulate paths and plot for a double check. Finally, simulated W_1 , the value of the Brownian motion at 1 1000 times and give a histogram. Is it a normal distribution (try `qqplot`). Has it mean zero? Has it variance 1? Yes - then you got it!

1.1.4 Defining our own functions

Of course a very powerful method is defining your own function. We start by giving a function for plotting a nice graph of a density. Note the `substitute` used for obtaining greek letters and inserting the values of variables in the title of the plot.

```
gnorm <- function(x) {  
  m=0    # mean  
  s=1    # sd  
  y <- dnorm(x,m,s)  
  plot(x,y,type="l",  
        main =substitute(paste("Normal Density with ",mu == m,  
                                ", ",sigma == s),list(m=mu,s=sigma)))  
}
```

We call this function for example by `gnorm(seq(-2,2,by=0.1))`. Simply typing `gnorm` recalls the definition of this function.

2 Copulas and multivariate random variables

This section will have a focus on simulation, with a small section on estimation at the end. We start with the simulation of multivariate distributions¹ (`.mnorm`, `.mt`, `.mghyp` in R) and give important implications for estimating correlations with heavy tails.

2.1 Multivariate models

Now we turn to the multivariate distributions. Define an easy function for the multivariate normal as follows². To address a covariance (or correlation) matrix simply by the correlations, we use `equicorr`

```
equicorr <- function (d, rho)
{
  if (rho < -(d - 1)^(-1))
    stop(paste("rho must be at least", -(d - 1)^(-1)))
  J <- matrix(rho, nrow = d, ncol = d)
  D <- diag(rep(1 - rho, d))
  J + D
}
```

Taking $X = \mu + AZ$ with Z being a vector of independent, standard normal variables gives a vector of normal variables with mean μ and covariance matrix $\Sigma = AA'$, which we use in the following. Recall that the Cholesky-Decomposition of Σ gives A .

```
rmnorm <- function (n, Sigma = equicorr(d, rho), mu = rep(0, d),
  d = 2, rho = 0.7)
{
  d <- dim(Sigma)[1]
```

¹The functions `.mnorm` `.mt` are taken from the QRM lib. See the appendix for details and code.

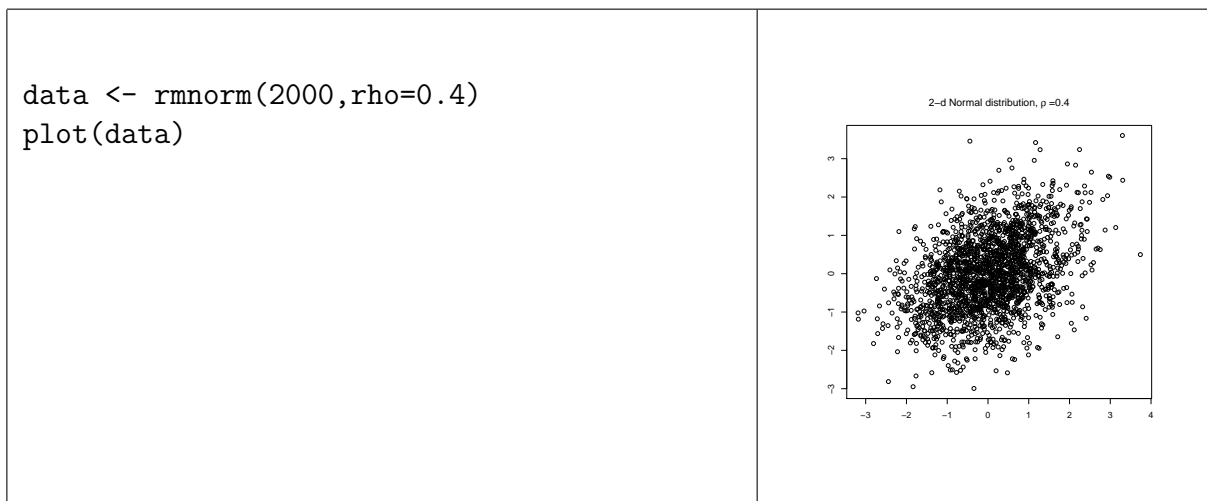
²This and the following functions are taken from the QRMLib. We list them again in the appendix, for your reference.

```

A <- t(chol(Sigma))
X <- matrix(rnorm(n * d), nrow = n, ncol = d)
mu.matrix <- matrix(mu, nrow = n, ncol = d, byrow = TRUE)
return(t(A %*% t(X)) + mu.matrix)
}

```

This function gives us an easy way to call a simulation of a multivariate normal distribution.



2.1.1 The probability and quantile transforms

An important idea needed here is the so-called *probability transform*: for a *continuous* random variable X with distribution function F the random variable

$$F(X)$$

is uniformly distributed.

Exercise 6. *Implement a test for this: Simulate a t -distribution and compute the (probability) transform $F(X)$. Test if this is uniformly distributed (Kolmogorov-Smirnov, or via a `qqplot`)*

From a uniform random variable U we obtain a random variable with cumulative distribution function via the *quantile* transform:

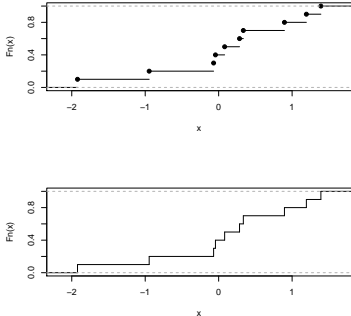
$$X \sim F^{-1}(U).$$

```

> x <- runif(100)
> y <- qbeta(x)
> hist(y)

```

We can also test this via a nonparametric estimate of the cdf, the *empirical distribution function*. This is implemented in the function `ecdf` from the package `stats`:

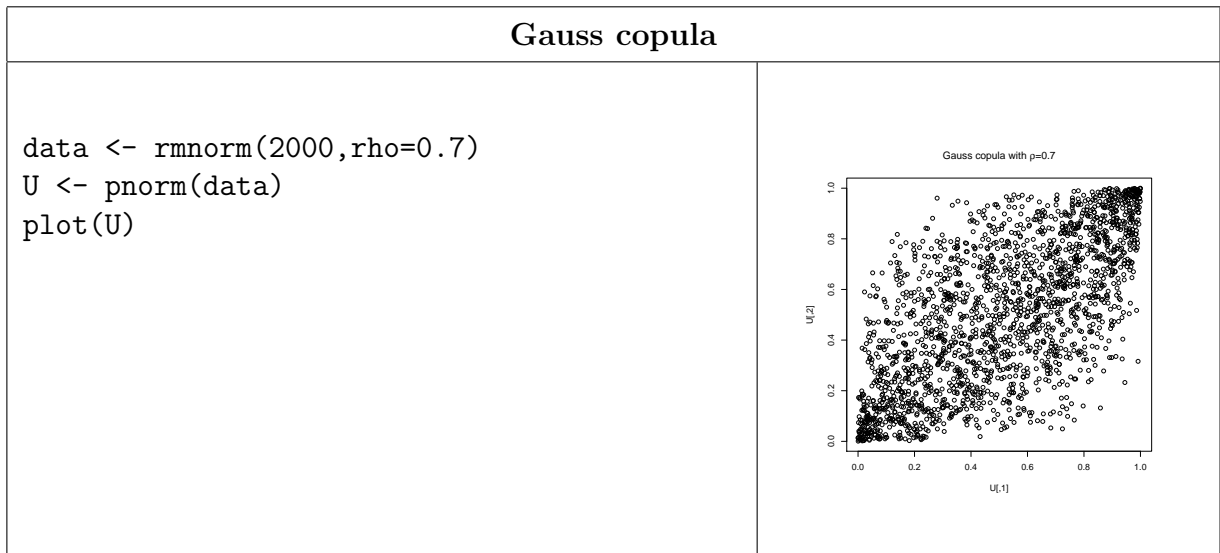
Empirical distribution	
<pre>require(stats) F10 <- ecdf(rnorm(10)) summary(F10) plot(F10) plot(F10, verticals= TRUE, do.points = FALSE)</pre>	<p>See <code>help(ecdf)</code> for more information.</p> 

2.2 Copulas

For computing or simulation a copula we take a random vector (X_1, X_2) and transform it to

$$(F_1(X_1), F_2(X_2))$$

The distribution of this vector is a *copula*. For any copula, there exists a random vector, such that its cdf is the copula. In this section we aim at simulating some copulas. The Gauss copula is obtained from the normal distribution:



The t -copula shall be compared to the Gauss copula and we define a function for simulating multivariate t -random variables.

```
rmt <- function (n, df = 4, Sigma = equicorr(d, rho),
                 mu = rep(0, d), d = 2, rho = 0.7)
{
  d <- dim(Sigma)[1]
  chi <- 2 * rgamma(n, shape = df/2)
  m1 <- rmnorm(n, Sigma = Sigma)
  m2 <- matrix(rep(sqrt(df)/sqrt(chi), d), ncol = d)
  mu.matrix <- matrix(mu, nrow = n, ncol = d, byrow = TRUE)
  return(m1 * m2 + mu.matrix)
}
```

Lets plot the t -random variables and the probability transforms.

```
data <- rmt(2000,rho=0.4,df=4)
plot(data,main="t")
U <- pt(data,df=4)
plot(U,main="t-copula")
```

Exercise 7. Simulate 100 Gaussian vectors (X_i, Y_i) with correlations $\rho = -0.8, 0, 0.8$ and transform X_i and Y_i to exponential, log-normal and normal variables with mean and variances of your choice. Plot the resulting data and the associated probability transforms (copula).

2-dimensional Normal and t -random variables	
<pre>data1 <- rmnorm(1000,d=2,rho=0.5) data2 <- rmt(1000,d=2,rho=0.5) var(data1) var(data2) cor(data1) cor(data2) cor.test(data2[,1],data2[,2])</pre>	<p>We use <code>rmnorm</code> and <code>rmt</code> to generate the 2-dimensional random variables. Then we check for the resulting variances. Correlation is estimated via <code>for</code>.</p> <p>If we want to have a confidence interval for the correlation, we use <code>cor.test</code></p>

Next, we visualize these data.

<pre>xmin=min(c(data1[,1],data2[,1])) xmax=max(c(data1[,1],data2[,1])) ymin=min(c(data1[,2],data2[,2])) ymax=max(c(data1[,2],data2[,2])) par(mfrow=c(2,1)) plot(data1,xlab="x_i",ylab="y_i", xlim=c(xmin,xmax),ylim=c(ymin,ymax), main="Multivariate Normal, rho=0.5") plot(data2,xlab="x_i",ylab="y_i", xlim=c(xmin,xmax),ylim=c(ymin,ymax), main="Multivariate t_3, rho=0.5")</pre>	<p>We first compute <code>xmin,...</code> to achieve the same scaling on both graphs. The output is split into two parts, using the <code>par</code>-command. <code>par(op)</code> switches back to the single output window³.</p> <div style="text-align: center;"> </div>
--	--

Clearly, the t -distribution shows much more outliers and heavy tails in comparison to the normal distribution. The variances, however, are different and after the following resealing still this difference appears.

```
data2[,1] <- sqrt(var(data1[,1])/var(data2[,1]))*data2[,1]
data2[,2] <- sqrt(var(data1[,2])/var(data2[,2]))*data2[,2]
```

Correlation as measure of dependence

In financial application it is very sensible to use correlation in the right way as we discuss in the courses. Here we illustrate this with the following example. We compute normal random variables with correlation of 0.8 and the transform them via the nonlinear function `exp`. The resulting correlation is much smaller! However, spearman's ρ_S does not show this effect. It is a dependence measure invariant to monotone transformations, a very suitable property in financial data. It is also less sensible to outliers.

<pre>set.seed(13) sigma <- 4 rho <- 0.8 Sigma <- matrix(c(1,rho*sigma , rho*sigma ,sigma^2),nrow=2) x <- rmnorm(100,Sigma) y <- exp(x) cor(y) cor(y,method="spearman")</pre>	<p>We use <code>set.seed</code> to guarantee we work all on the same random numbers. The estimated correlation (0.35) differs largely from the initial correlation (0.8). Why is this the case? Illustrate this by plotting the data.</p> <p>Try also alternatively Kendall's tau and Spearman's rho.</p>
---	---

If the data is heavy tailed, sensitive measures like mean or sample correlation may severely fail to estimate the correct values. We show this using kendalls'tau ρ_τ in comparison to linear correlation ρ on t_3 data. Recall the we have $\rho_S(X, Y) = \frac{6}{\pi} \arcsin \frac{\rho(X, Y)}{2} \approx \rho(X, Y)$ and $\rho_\tau(X, Y) = \frac{2}{\pi} \arcsin \rho(X, Y)$.

Robust measuring of correlation

```

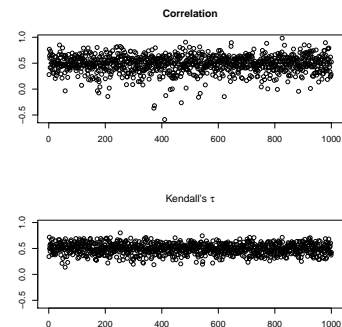
r1<-rep(0,1000)
r2<-rep(0,1000)

for (i in 1:1000){
  x <- rmt(90,df=3,rho=0.5)
  r1[i]<-cor(x)[1,2]
  r2[i]<-
    sin(cor(x,method="kendall")[1,2]*pi/2)
}

par(mfrow=c(2,1))
plot(r1,main="Correlation")
plot(r2,main="Kendalls tau")

```

We clearly see that robust measures like Spearman's ρ oder Kendall's τ are much better estimates if the data has heavy tails.



2.2.1 A first look at copulas

We simulate a number of copulas, starting with normal and t -copulas. The difference to our previous simulations with `rnorm` is that we have to transform the marginals to a uniform distribution. This can be done by using the empirical cumulative distribution function (cdf) (which is an approximation, then) or by using the underlying cdf, which we call *probability transform*.

Gauss and t -copula

```

par(mfrow=c(2,2))

data <- rmnorm(2000,rho=0.4)
plot(data,main="Normal",lwd=0.3)

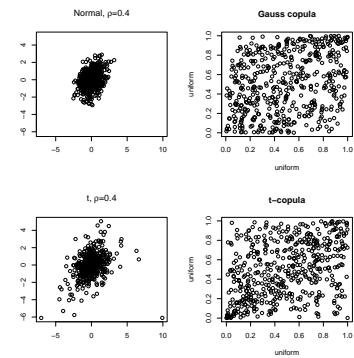
U <- pnorm(data)
plot(U,main="Gauss copula",lwd=0.3)

data <- rmt(2000,rho=0.4,df=4)
plot(data,main="t",lwd=0.3)

U <- pt(data,df=4)
plot(U,main="t-copula",lwd=0.3)

```

In this code we use the probability transform by `norm` and `pt` which are the cdf's of the normal and t distribution. The parameter `lwd=0.3` gives lighter circles.



Exercise Simulate different mean-variance mixture distributions and their copulas. Compare the copulas obtained by the probability transform to the ones obtained by the empirical cdf (use `ecdf`).

2.3 Estimating copulas

Typically, copulas can be estimated via maximum-likelihood methods. This requires a more sophisticated implementation, in particular because the density is not so simple. You find the code from the `QRMLib` in `QRMLibparts.R`. Alternatively, you simply install the `QRMLib`. If the package is installed you can load it by

```
library(QRMLib)
```

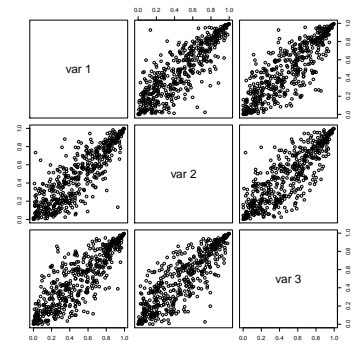
2.3.1 Estimation via rank correlations

If the number of degrees of freedom is known, it remains only to estimate the correlation parameter. As indicated above, this can be done via Spearman's ρ or Kendall's τ .

Estimation of the correlation coefficient	
<pre> N <- 100000 n <- 100 r1 <- rep(0,N);r2 <- r1;r3 <- r1 pb <- txtProgressBar(style=3) for (i in 1:N) { data <- rmt(15,rho=0.5) r1[i] <- cor(data)[1,2] r2[i] <- sin(cor(data,method="spearman")[1,2]*pi/6)*2 r3[i] <- sin(cor(data,method="kendall")[1,2]*pi/2) setTxtProgressBar(pb, i/N) } c(mean(r1), median(r1), var(r1)); ... </pre>	<p>We use the <code>txtProgressBar</code> to follow the progress of this longer simulation. Using a smaller <code>N</code> speeds it up, leading to similar results. The output is</p> <pre> [1] 0.489 0.499 0.0284 [1] 0.470 0.477 0.0081 [1] 0.491 0.498 0.0082 </pre>

Simulating from a Gumbel copula

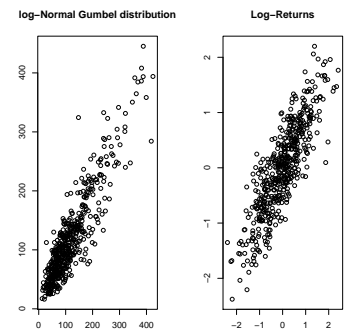
```
data <- rcopula.gumbel(1000,3,4)
pairs(data)
```



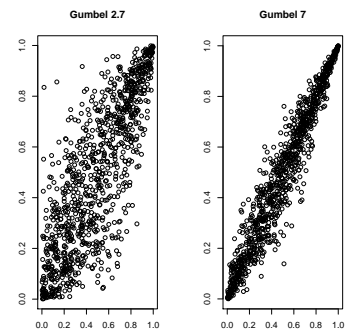
```
mu <- 0.02; sigma <- 0.6
stockdata <- 100*exp(mu+sigma*qnorm(data))
```

We transform to log-normal margins, and obtain a Meta lognormal-Gumbel distribution. We also plot the logreturns.

```
plot(stockdata)
plot(diff(log(stockdata)))
```

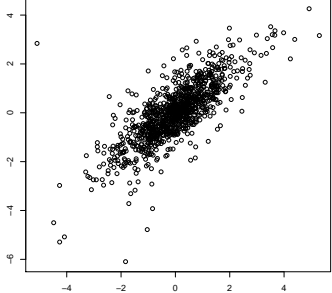


```
data2 <- rcopula.gumbel(1000,2.7,2)
data3 <- rcopula.gumbel(1000,7,2)
```



2 Copulas and multivariate random variables

As a next step we want to test the maximum-likelihood estimation of the copulas. We test some examples. In the QRMLib⁴ we find the necessary tools.

Estimating a t -copula	
<pre>data <- rmt(1000,df=5,rho=0.8)</pre>	
<pre>list(Means=c(mean(data[,1]),mean(data[,2])), Variances=var(data), Correlation=cor(data)[1,2])</pre>	<pre>\$Means [1] -0.017031 0.009217 \$Variances [,1] [,2] [1,] 1.472639 1.134481 [2,] 1.134481 1.452208 \$Correlation [1] 0.7757727</pre>
<pre>fit.norm(data)</pre>	<p>Gives the exact result as above with log-likelihood</p> <pre>ll.max = -2756.518</pre>
<pre>fit.mst(data)</pre>	<p>The result is nu=5.23 degrees of freedom and a correlation of 0.801 and</p> <pre>ll.max = -2648.338</pre>

⁴This library provides a lot of fitting algorithms: `fit.AC`, `fit.Archcopula2d`, `fit.gausscopula`, `fit.mNH`, `fit.mst`, `fit.norm`, `fit.tcopula`, `fit.tcopula.rank` besides other fitting routines for univariate distributions or extreme-value methods.

The higher likelihood under the t -copula suggest that this gives a better fit. This is *not* a test for normal copula against a t -copula, just a simple fitting argument (Can you name the difference?).

2.3.2 The generalised hyperbolic distribution

Next, we turn to the important class of generalized hyperbolic distributions. We use the notation $X \sim GH_d(\lambda, \chi, \psi, \mu, \Sigma, \gamma)$. The GH-distribution is a mixture:

$$X = \mu + \gamma W + \sqrt{W}AZ$$

where W has a generalized inverse Gaussian (GIG) distribution, $W \sim GIG(\lambda, \chi, \psi)$. So μ is the mean, γ is the skewness parameter and $\Sigma = AA'$ the covariance matrix (before mixing). As is easy to see by the Fourier-transform, the GH distribution is closed under linear transformations, i.e. $BX + b$ has distribution $GH_d(\lambda, \chi, \psi, B\mu + b, B\Sigma B', B\gamma)$.⁵ If $\lambda = \frac{1}{2}(d + 1)$, we call the distribution a d -dimensional hyperbolic distribution. We obtain the t_n distribution with $\lambda = -\frac{n}{2}, \chi = n, \psi = 0$; if $\gamma = 0$ this is the (classical) unskewed t_n -distribution.

⁵There are many special cases, hyperbolic, NIG, Variance Gamma and so on.

Densities and simulation of the generalized hyperbolic distribution

```

x <- seq(-2,5,by=0.01)
plot(x,dghyp(x,0.5,1,1,0,0))
lines(x,dghyp(x,0.5,7,1,0,0),col=19)
lines(x,dghyp(x,-0.5,5,1,0,0),col=22)
lines(x,dghyp(x,-0.5,5,1,0,-1),col=22)
lines(x,dghyp(x,-0.5,5,1,0,1),col=22)

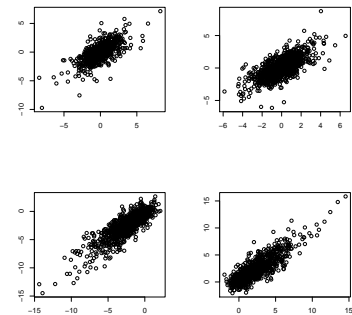
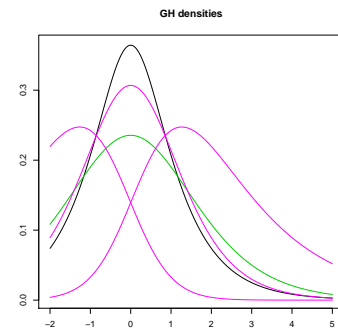
```

```

d1 <- rmghyp(1000,-0.5*4,4,0,rho=0.7)
d2 <- rmghyp(1000,-0.5,5,1,rho=0.7)
d3 <- rmghyp(1000,-0.5,5,1,gamma=-1,,rho=0.7)
d4 <- rmghyp(1000,-0.5,5,1,gamma=1,rho=0.7)

```

Some simulations. The first one is a t_4 -distribution. The second line illustrates the skewness effect of γ .



Next, we aim at estimating the parameters of the generalized hyperbolic distribution. First, in the univariate and then in the 2-dimensional case. Simulations are used to analyze the performance of the estimation.

<pre> N<- 100 n<- 500 psi <- rep(0,N) chi <- rep(0,N) rho <- rep(0,N) for (i in 1:N){ fit <- fit.mNH(rmghyp(1000,-0.5,1,1,rho=0.2)) psi[i] <- fit\$mix.pars[2] chi[i] <- fit\$mix.pars[3] rho[i] <- fit\$correlation[1,2] } </pre>	<p>we obtain</p> <table border="1"> <thead> <tr> <th>means</th> <th>variances</th> </tr> </thead> <tbody> <tr> <td>0.9926</td> <td>0.0172</td> </tr> <tr> <td>1.0503</td> <td>0.0396</td> </tr> <tr> <td>0.2010</td> <td>0.0013</td> </tr> </tbody> </table>	means	variances	0.9926	0.0172	1.0503	0.0396	0.2010	0.0013
means	variances								
0.9926	0.0172								
1.0503	0.0396								
0.2010	0.0013								

A data example

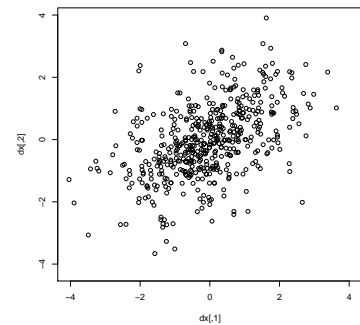
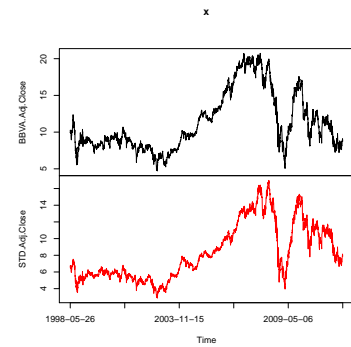
This section aims at analyzing financial data. We first download data conveniently from the yahoo finance sites with the package `fImport`. Alternatively, any data in `.csv` (or other) format can be easily imported, for example with the commands `read.table`, `read.csv`. We choose the stock prices from BBVA and Santander (NY), in Dollars.

Import data from finance.yahoo

```
require(fImport)

stocks <- yahooSeries(c("BBVA","STD"),
                      nDaysBack=5000)
x <- stocks[,c("BBVA.Adj.Close",
              "STD.Adj.Close")]
plot(stockdata)

dx <- array(c(x[,1],x[,2]),
            dim=c(length(x[,1]),2))
dx <- diff(log(dx))
plot(data)
```



<pre> mod.GAUSS <- fit.norm(dx) mod.NIG <- fit.mNH(dx , symmetric=FALSE,case="NIG") mod.HYP <- fit.mNH(dx , symmetric=FALSE,case="hyp") mod.t <- fit.mst(dx) mod.NIGs <- fit.mNH(dx , symmetric=TRUE,case="NIG") mod.HYPs <- fit.mNH(dx , symmetric=TRUE,case="hyp") mod.GAUSS\$ll.max # and similar </pre>	<p>The data analysis gives</p> <table border="1"> <thead> <tr> <th></th> <th>log likelihood max</th> </tr> </thead> <tbody> <tr> <td>GAUSS</td> <td>17332.2</td> </tr> <tr> <td>t</td> <td>18177.9</td> </tr> <tr> <td>NIG (symm)</td> <td>18198.5</td> </tr> <tr> <td>GH (symm)</td> <td>18165.9</td> </tr> <tr> <td>NIG</td> <td>18199.8</td> </tr> <tr> <td>GH</td> <td>18167.2</td> </tr> </tbody> </table> <p>The best fit is in both cases, symmetric and asymmetric provided by the NIG distribution.</p> <table border="1"> <thead> <tr> <th>lambda</th> <th>chi</th> <th>psi</th> </tr> </thead> <tbody> <tr> <td>-0.50</td> <td>0.00</td> <td>1310.6</td> </tr> <tr> <th>gamma</th> <td></td> <td></td> </tr> <tr> <td>-1.991</td> <td>-2.671</td> <td></td> </tr> </tbody> </table>		log likelihood max	GAUSS	17332.2	t	18177.9	NIG (symm)	18198.5	GH (symm)	18165.9	NIG	18199.8	GH	18167.2	lambda	chi	psi	-0.50	0.00	1310.6	gamma			-1.991	-2.671	
	log likelihood max																										
GAUSS	17332.2																										
t	18177.9																										
NIG (symm)	18198.5																										
GH (symm)	18165.9																										
NIG	18199.8																										
GH	18167.2																										
lambda	chi	psi																									
-0.50	0.00	1310.6																									
gamma																											
-1.991	-2.671																										

2.4 Definitions

This functions were used above, taken from QRMLib

```
# Generate a correlation matrix with equal correlations
```

```

equicorr <- function (d, rho)
{
  if (rho < -(d - 1)^(-1))
    stop(paste("rho must be at least", -(d - 1)^(-1)))
  J <- matrix(rho, nrow = d, ncol = d)
  D <- diag(rep(1 - rho, d))
  J + D
}

```

```
# Generate multivariate normal random variables
```

```

rmnorm <- function (n, Sigma = equicorr(d, rho),
  mu = rep(0, d), d = 2, rho = 0.7)
{
  d <- dim(Sigma)[1]
  A <- t(chol(Sigma))
  X <- matrix(rnorm(n * d), nrow = n, ncol = d)
  mu.matrix <- matrix(mu, nrow = n, ncol = d, byrow = TRUE)
}

```

2 Copulas and multivariate random variables

```
    return(t(A %*% t(X)) + mu.matrix)
  }
```

For the multivariate t -distribution we use the representation $t = \mu + \sqrt{W}AZ$ where Z is standardnormal and W has an inverse Gamma distribution.

```
rmt <- function (n, df = 4, Sigma = equicorr(d, rho),
                mu = rep(0, d), d = 2, rho = 0.7)
{
  d <- dim(Sigma)[1]
  chi <- 2 * rgamma(n, shape = df/2)
  m1 <- rmnorm(n, Sigma = Sigma)
  m2 <- matrix(rep(sqrt(df)/sqrt(chi), d), ncol = d)
  mu.matrix <- matrix(mu, nrow = n, ncol = d, byrow = TRUE)
  return(m1 * m2 + mu.matrix)
}
```

3 Risk measures

In this section we take a closer look at the estimation of risk measures, in particular value-at-risk and expected shortfall. We assume that we have a history of *losses* and want to predict or control outcomes of a future loss. Consider the i.i.d. case, i.e. where we observe X_1, \dots, X_n random variables which are independent and identically distributed and have the same distribution like $X = X_{n+1}$, the future loss. We want to estimate the risk regarding the unknown future outcome of X . Denote the cdf of X by F .

If F is continuous, or at least if $\mathbb{P}(X = \alpha)$, then the *value at risk* of X at confidence level α (typically 0.95 or 0.99) is

$$\text{VaR}_\alpha(X) := F^{-1}(\alpha).$$

$F^{-1}(\alpha)$ is a particular *quantile* of X . If X is continuous at α , the quantile to the level α is unique and we denote it by $q_\alpha(X)$. It coincides with $F^{-1}(\alpha)$ in this case. If $X \sim \mathcal{N}(\mu, \sigma^2)$, then

$$\text{VaR}_\alpha(X) = \mu + \sigma\Phi^{-1}(\alpha),$$

where Φ is the cdf of the standard normal distribution. In the normal context value-at-risk is coherent, otherwise it is not, because the subadditivity fails. The second risk measure what we want to consider is the expected shortfall (average value-at-risk or conditional value at risk). It is defined by

$$\text{ES}_\alpha(X) = \frac{1}{1 - \alpha} \int_{q_\alpha(X)}^1 q_u(X) du.$$

Note that for this definition we do not need continuity of X . If X is continuous, we also have that

$$\text{ES}_\alpha(X) = \mathbb{E}(X | X > \text{VaR}_\alpha(X)).$$

In the Gaussian case, i.e. if $X \sim \mathcal{N}(\mu, \sigma^2)$, we obtain

$$\text{ES}_\alpha(X) = \mu + \sigma \frac{\phi(\Phi^{-1}(\alpha))}{1 - \alpha}.$$

We start with an analysis of the IBEX index.

3 Risk measures

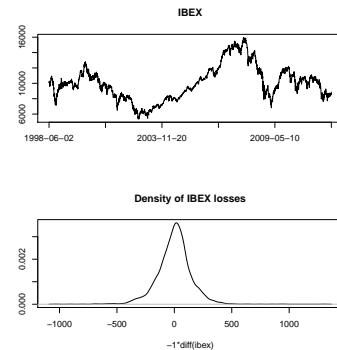
```
ibexd <- yahooSeries("^IBEX",nDaysBack=5000)
ibex <- ibexd[, "^IBEX.Adj.Close"]
plot(ibex,type="l")

ibex.d <- diff(ibex[1:length(ibex)])
ibex.loss.d <- -1*ibex.d
plot(density(ibex.loss.d))

mean(ibex.loss.d)
sd(ibex.loss.d)
```

We take the IBEX index as an example. Mean and sd compute to

-0.35 150



To compute the value-at-risk we first *assume* that the index loss has a normal distribution with mean -0.35 and sd of 150 .

```
VaR.Gauss <- function (alpha=0.95,mu,sigma)
{
mu + sigma* qnorm(alpha)
}

ES.Gauss <- function(alpha=0.95,mu,sigma)
{
mu + sigma/(1-alpha)*dnorm(qnorm(alpha))
}
```

The obtained risk measures are

VaR _{0.95}	246.37
ES _{0.95}	309.06
VaR _{0.99}	348.60
ES _{0.99}	399.43

3 Risk measures

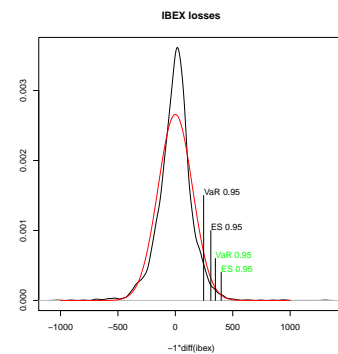
```
plot(density(ibex.loss.d)
x <- seq(-1000,1000,by=10)
lines(x,dnorm(x,mean=-0.35,sd=150),col="red")

#We mark the risk measures in the graph
lines(c(246.37,246.37),c(0,0.0015))
text(250,0.0015,"VaR 0.95",adj=c(0,0))

lines(c(309.07,309.07),c(0,0.001))
text(311,0.001,"ES 0.95",adj=c(0,0))

lines(c(348.60,348.60),c(0,0.0006))
text(350,0.0006,"VaR 0.95",adj=c(0,0))

lines(c(399.43,399.43),c(0,0.0004))
text(400,0.0004,"ES 0.95",adj=c(0,0))
```



on the graph we plot the 0.99 risk measures in green.

Next, we want to assess the quality of the risk measures. At the moment we employ two estimating possibilities:

- (i) Naive: we substitute the estimated parameters $\hat{\mu} = \text{mean}(X)$ and $\hat{\sigma} = \text{sd}(X)$. This gives the estimate

```
VaR.Gauss(alpha,mean(data),sd(data))
```

- (ii) Empirical: this uses the empirical quantile and is simply implemented by

```
VaR.emp <- function(x,alpha){
  quantile(x,alpha)
}
```

We analyze the quality of these measures by a Monte-Carlo simulation:

```

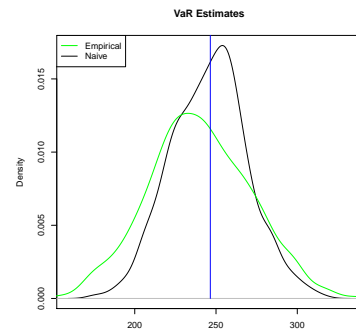
mu <- -0.3525174
sigma <- 150.0785

N<- 1000; n<- 100; alpha <- 0.95

var.est1 <- rep(0,N); var.est2 <- rep(0,N)
for (i in 1:N){
data <- rnorm(n,mu,sigma)
var.est1[i] <- VaR.emp(data,alpha)
var.est2[i] <-
  VaR.Gauss(alpha,mean(data),sd(data))
}

plot(density(var.est2))
lines(density(var.est1))
lines(c(246.5,246.5),c(0,0.020)) #True VaR

```



We obtain mean and sd's (true VaR=246.5)

```

$Empirical
[1] 239.76912  31.15811

$Naive
[1] 245.75854  23.45826

```

3.1 Performance of the risk estimates

This is, however, until now a statistical analysis. In a financial sense we are much more interested in the question *how often* is the calculated risk capital sufficient to cover the occurring losses? This shall be addressed in the following. We perform a method similar to a backtesting approach and the routines can simply be extended to this.

```

mu <- -0.3525174
sigma <- 150.0785

N<- 1000
n<- 100 # Learning period
M<- 100 # Backtesting period
alpha <- 0.95

var.true <- VaR.Gauss(alpha,mu,sigma)
var.est1 <- rep(0,N)
var.est2 <- rep(0,N)
var.est3 <- rep(0,N)
counts <- rep(0,4)

```

3 Risk measures

```

for (i in 1:N){
data <- rnorm(n,mu,sigma)
var.est1[i] <- VaR.emp(data,alpha)
var.est2[i] <- VaR.Gauss(alpha,mean(data),sd(data))
var.est3[i] <- mean(data)+sd(data)*qt(alpha,n-1)

newperiod <- rnorm(M,mu,sigma)
counts[1] <- counts[1]+ sum(newperiod>var.est1[i])
counts[2] <- counts[2]+ sum(newperiod>var.est2[i])
counts[3] <- counts[3]+ sum(newperiod>var.est3[i])
counts[4] <- counts[4]+ sum(newperiod>var.true)

}

```

The output is as follows (counts/M/N) :

Method	Empirical	Naive	t	True VaR
Perc. failure	0.05860	0.05320	0.05166	0.05005

Surprisingly, all the risk measures perform quite bad. This is due to the well-known effect that the distribution has to be estimated and estimating the parameters changes the original normal distribution to a t distribution. Still, it is surprising that the according t -estimate propagated in the literature performs so bad. This is due to the fact that the estimator is *biased in probability*. There is a theoretical solution to this, which we discussed in class. Here we aim at finding an empirical solution to this. We aim at computing the quantile factor q such that

$$\text{VaR}_{0.95}^* := \text{mean}(\text{data}) + \text{sd}(\text{data}) * q$$

does produce an average number of $1 - \alpha$ failures.

```

# Estimate the quantile level q empirically
N<- 1000
n<- 100 # Learning period
M<- 100 # Backtesting period

newperiod <- rep(0,N*M)
for (i in 1:N){
  data <- rnorm(n,mu,sigma)
  newperiod[(1+(i-1)*M):(i*M)] <- (rnorm(M,mu,sigma)-mean(data))/sd(data)
}

```

3 Risk measures

The outcome is surprisingly clear:

```
> quantile(newperiod,0.95)
 95%
1.668395
```

By simply adding

```
var.est4[i] <- mean(data)+sd(data)*1.668395
```

We can incorporate our new VaR-estimator in the above test and obtain

Method	Empirical	Naive	t	New VaR	True VaR
Perc. failure	0.05860	0.05320	0.05166	0.5051	0.05005

In accordance with our theoretical considerations the new factor is very close to

$$\sqrt{\frac{n+1}{n}}t_{n-1}.$$

Accordingly we obtain the *modified* VaR estimator by

$$\text{VaR}_\alpha^* := \text{mean}(\text{data}) + \text{sd}(\text{data}) * \text{qt}(\alpha) * \text{sqrt}((n+1)/(n)). \quad (3.1)$$

Exercise 8. Translate this considerations to a t -distribution and proceed with the following steps:

- Estimate a t (or a generalised inverse Gaussian) distribution from the IBEX data.
- Propose VaR-estimates: naive estimation, empirical estimation and an estimation with is according to (3.1).
- Study mean, variance and distribution of these estimates with a simulation study.
- Implement a backtesting procedure and analyze as above how this estimations perform.

Exercise 9. What are appropriate criteria for the expected shortfall? How can this mechanisms be transported?

4 Estimation of diffusions and affine models

We discuss the implementation of some estimation methods for continuous, affine, one-dimensional diffusions. A great survey on the estimation methods can be found in [?]. Essentially, we consider models of the class

$$dX_t = \mu(X_t, \theta)dt + \sigma(X_t, \theta)dW_t,$$

i.e. continuous Markov processes. Famous examples are the affine models and, in particular, the Vasicek and the Cox-Ingersoll-Ross models.

4.1 Simulation

The *Vasicek* model is a special Ornstein-Uhlenbeck process. In classical notation we state it by

$$dX_t = \kappa(\theta - X_t)dt + \sigma dW_t.$$

It has the explicit solution¹

$$X_t = \theta + (X_0 - \theta)e^{-\kappa t} + \sigma e^{-\kappa t} \int_0^t e^{\kappa s} dW_s.$$

For a simulation we can use this explicit solution, which is however not feasible for more general diffusions. In these cases we have the possibility to approximate the solution by discretization. The simplest one is the *Euler-Maruyama* discretization,

$$x_{t_i} = \mu(x_{t_{i-1}})(t_i - t_{i-1}) + \sigma(x_{t_{i-1}})\epsilon_i$$

where $\epsilon_1, \epsilon_2, \dots$ are independent and $\epsilon_i \sim \mathcal{N}(0, t_i - t_{i-1})$. Of a faster convergence is the Milstein scheme, where

$$x_{t_i} = x_{t_{i-1}} + \mu(x_{t_{i-1}})(t_i - t_{i-1}) + \sigma(x_{t_{i-1}})\epsilon_i + \frac{\sigma(x_{t_{i-1}})}{2} \frac{\partial \sigma(x_{t_{i-1}})}{\partial x} (\epsilon_i^2 - (t_i - t_{i-1}))$$

In the Vasicek model, both methods coincide. For $t_i - t_{i-1}$ small enough, the methods are very close:

¹See Cucchiero, for example

```

vas.sim <- function (n,theta,sigma,kappa,Delta,vas0,method="exact") {
  simul <- rep(0,n); simul[1] <- vas0
  for (i in 2:n){
    simul[i] <- theta + (simul[i-1]-theta)*exp(-1*kappa *Delta)
      + sigma*rnorm(1,0,sqrt((1-exp(-2*kappa*Delta))/2/kappa))
  }
  return (simul)
}

```

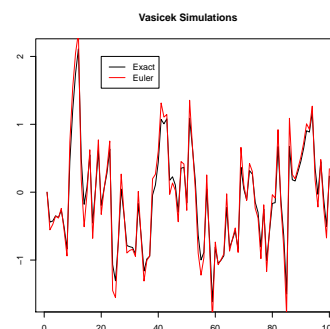
For the Euler method we use

```

simul[i] <- simul[i-1]
  +kappa*(theta-simul[i-1])*Delta
  +sigma*rnorm(1,0,sqrt(Delta))

```

instead. For delta smaller than 0.1 the difference is typically very small. The Milstein method is only relevant for the CIR process.



Exercise 10. Implement the simulation of the CIR process with all three methods and compare their performance.

The simulation of the CIR process is much more difficult, as you will experience. What to do if the process gets below zero?

4.2 Maximum-Likelihood estimation

For the maximum-likelihood estimation we rely on R's routine `optim`. In the Vasicek model we have the exact density and can simply maximise the log-likelihood. We assume that we observe $x_{i\Delta}$, $i = 0, \dots, n-1$, i.e. n data points. The first point x_0 is deterministic. Then

$$\begin{aligned}
 l(x; \theta) &= -\frac{n-1}{2} \ln(2\pi) - \frac{n-1}{2} \ln(V) - \frac{1}{2V} \sum_{i=1}^{n-1} (x_{i\Delta} - \beta - (x_{(i-1)\Delta} - \beta)e^{-\alpha\Delta})^2, \\
 V &= \frac{\sigma^2}{2\alpha} (1 - e^{-2\alpha\Delta}).
 \end{aligned}$$

We maximize² this function with respect to $\theta = (\alpha, \beta, \sigma)'$. We will see that estimation of β and σ is by far simpler than estimating α .

```

vas.ll <- function (params,Delta,Data) {
  alpha <- params[1]
  beta <- params[2]
  sigma <- params[3]
  n <- length(Data)
  vassigma2 <- (sigma^2)*(1-exp(-2*alpha*Delta))/2/alpha
  summe <- -1/2/vassigma2* sum ((Data[2:n]-beta-
    (Data[1:(n-1)]-beta)*exp(-1*alpha*Delta))^2)
  return((1-n)/2*log(2*pi) - (n-1)/2*log(vassigma2)+summe)
}

vas.mle<- function (Delta,Data,output=FALSE){
  startvalue <- c(0.1,0.1,0.1) # arbitrary
  MLE.out <- optim (startvalue,vas.ll,gr = NULL, method = "Nelder-Mead",
    lower = -Inf, upper = Inf, control = list(fnscale=-1),
    hessian = FALSE, Delta,Data)
  return (c(MLE.out[[1]][1],MLE.out[[1]][2],MLE.out[[1]][3]))
}

```

It should be noted that `optim` is a numerical procedure which depends on the start value. The Nelder-Mead method seems to be quite robust. The BFGS method has a better performance, but choosing $\alpha = \beta = \sigma = 0.5$ and starting with 0.1 leads to convergence but no sensible result.

We compare our results to the ones from Hurn et. al (2007) in the following exercise.

Exercise 11. *Simulate 2000 runs of a MLE estimation of a Vasicek model with parameters $\alpha = 0.2, \beta = 0.08$ and $\sigma = 0.1$. Assume that you observe in each run 500 data points with $\Delta = 1/12$ and $x_0 = \beta$.*

We state the outcome in terms of mean error and root mean square error ($N^{-1} \sum_{i=1}^N (\hat{\theta}_i - \theta)$ and $\sqrt{N^{-1} \sum_{i=1}^N (\hat{\theta}_i - \theta)^2}$). We use the parameters $\theta_1 = (0.2, 0.08, 0.1)$ and $\theta_2 = (0.3, 0.4, 0.5)$ for a simulation study. The start value of the optimization is $\theta_0 = (0.1, 0.1, 0.1)'$.

²Typically it may happen that the maximization does not converge. This should be covered by using checking if `MLE.out` returns non-zero `convergence`.

	Mean error in α (RMS error in α)	Mean error in β (RMS error in β)	Mean error in σ (RMS error in σ)
MLE θ_1	0.110200 (0.183036)	0.000246 (0.077009)	-0.002569 (0.023177)
MLE θ_2	0.100273 (0.197201)	-0.031805 (0.288133)	0.001530 (0.016250)
MLE. corr θ_1	0.108214 (0.180004)	-0.000351 (0.076279)	0.000142 (0.003229)

It is important that these estimators are *biased*! Even if the mean error is very small, a `t.test` rejects the hypothesis that the bias is zero immediately. So for `MLE.corr` we crudely estimated the bias by simulation and subtracted it. In contrast to typical bootstrap approaches this did not give an increasing variance, so this seems to be currently the best way to estimate the parameters. The estimation of α should be possible to improve, however. The `t.test` for the bias now does not reject on $\hat{\beta}$ and $\hat{\sigma}$. Note !

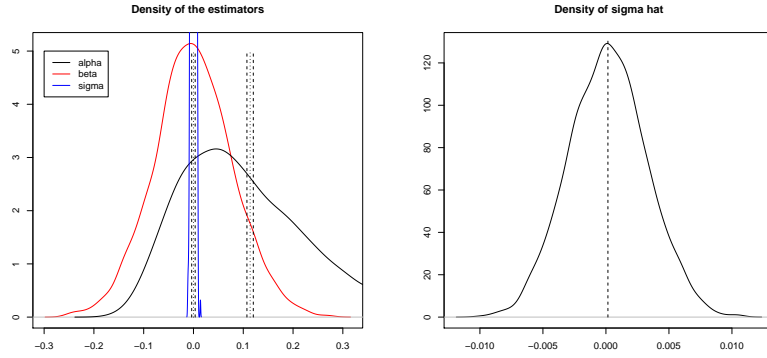
We used the following code:

```
alpha=0.2; beta=0.08; sigma=0.1; delta=1/12
n=500; N=2000
thetahat = array(rep(0,3*N),dim=c(N,3))
for (i in 1:N){
  thetahat[i,]= (vas.mle(delta,vas.sim(n,alpha,beta,sigma,delta,0))
  - c(alpha,beta,sigma))
}

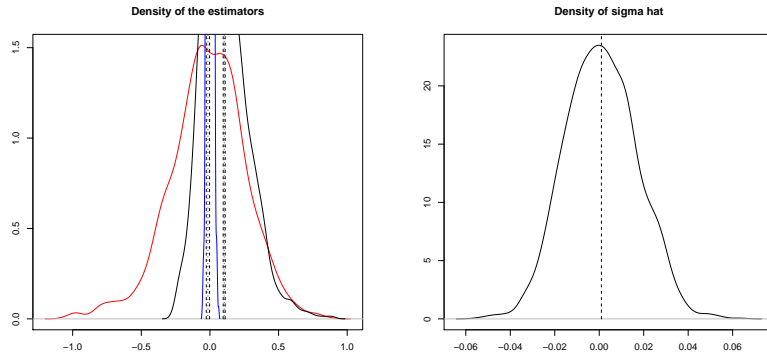
cat("\n Mean errors: \n")
print(c(mean(thetahat[,1]),mean(thetahat[,2]),mean(thetahat[,3])))
cat("\n RMS erros: \n")
print(c(sqrt(mean(thetahat[,1]^2)),sqrt(mean(thetahat[,2]^2))
, sqrt(mean(thetahat[,3]^2))))
cat("\n Test for bias: \n")
print(c(t.test(thetahat[,1],alternative="two.sided")$p.value,
t.test(thetahat[,2],alternative="two.sided")$p.value,
t.test(thetahat[,3],alternative="two.sided")$p.value),digits=3)
```

The density of the (centered) estimators are easily plotted:

4 Estimation of diffusions and affine models



and for the second sets of parameters:



In particular, the plots of the estimators for α and β show significant skewness. Also the bias is illustrated (not always, though). It is important to recall that we have $N = 2000$ observations.

4.2.1 GMM

The generalized method of moments (GMM) is a flexible and powerful method for estimation. We follow Chaussé (2010) and give a short outline and the implementation for the estimation of affine processes with R. In this regard we will be able to profit from the `gmm`-package available.

Assume that we aim at estimating $\theta_0 \in \mathbb{R}^p$ and we have $q > p$ moment conditions of the form

$$\mathbb{E}(g(\theta_0, X_i)) = 0, \quad i = 1, \dots, n$$

X_1, \dots, X_n is the observation or a transformation of it, in our case we would see a discrete observation of a stochastic process. θ_0 is the unique solution of the above

equation and element of a compact space. We also need some boundedness of higher moments (see ...).

As q is typically greater than q we have no solution of

$$\bar{g}(\theta) := \frac{1}{n} \sum_{i=1}^n g(\theta, x_i),$$

(which otherwise would be our estimator). We therefore rely on *minimizing a quadratic distance*

$$\bar{g}(\theta)^\top W \bar{g}(\theta).$$

It can be shown that the optimal weight matrix W is

$$W^* = \left(\lim_{n \rightarrow \infty} \text{Var}(\sqrt{n} \bar{g}(\theta_0)) \right)^{-1}.$$

This matrix, however, must be estimated and there are different schemes available for this (two-step as in Hansen (1982) or iterative procedures as in Hansen et. al (1996)). Under weak conditions we have asymptotic normality and consistency.

Example 4.2.1. Suppose we have i.i.d. observations which are $\mathcal{N}(\mu, \sigma^2)$ distributed. We could use the following moment conditions:

$$\mathbb{E} \begin{pmatrix} X_i - \mu \\ (X_i - \mu)^2 - \sigma^2 \\ X_i^3 - \mu(\mu^2 + 2\sigma^2) \end{pmatrix} = 0$$

to estimate with GMM. Note that we do not use the density, i.e. the method will be clearly less efficient as maximum-likelihood. On the other side we gain in robustness against miss specification of the density. Implementation is done as follows (borrowed from Chaussé, 2010)

```

g1 <- function(tet, x) {
  m1 <- (tet[1] - x)
  m2 <- (tet[2]^2 - (x - tet[1])^2)
  m3 <- x^3 - tet[1] * (tet[1]^2
    + 3 * tet[2]^2)
  f <- cbind(m1, m2, m3)
  return(f) }

Dg <- function(tet, x) {
  G <- matrix(c(1, 2 * (-tet[1] + mean(x)),
    -3 * tet[1]^2 - 3 * tet[2]^2,
    0, 2 * tet[2], -6 * tet[1] * tet[2]),
    nrow = 3, ncol = 2)
  return(G) }

x <- rnorm(200, mean = 4, sd = 2)
gmm(g1, x, c(mu = 0, sig = 0), grad = Dg)

```

This runs the `hmm` function with moment conditions given in the function `g1` and data in `x`. The starting vector is simply $(0, 0)$ with names given. The gradient is provided in `Dg`, which improves the performance.

Many existing methods can be seen as a special case of GMM, as for example the maximum-likelihood estimation.

4.2.2 GMM for affine models

Considering the Vasicek model, we again use the Euler discretization

$$x_{t_i} - x_{t_{i-1}} \approx \alpha(\beta - x_{t_{i-1}})(t_i - t_{i-1}) + \epsilon_i;$$

ϵ_1, \dots are independent and $\sim \mathcal{N}(0, \sigma^2(t_i - t_{i-1}))$. This leads to the following moment conditions;

$$\mathbb{E} \begin{pmatrix} \epsilon_i \\ \epsilon_i X_{t_{i-1}} \\ \epsilon_i^2 - \sigma^2 * (t_i - t_{i-1}) \\ (\epsilon_i^2 - \sigma^2 * (t_i - t_{i-1})) X_{t_{i-1}} \end{pmatrix} = 0.$$

```

vas.g <- function(theta, x) {
  t <- length(x)
  et <- diff(x) - theta[1]*(theta[2] - x[-t])*Delta
  ht <- et^2 - (theta[3])^2*vas.gmm.Delta
  g <- cbind(et, et * x[-t], ht, ht * x[-t])

```

```

    return(g)
}

```

We call of the function `hmm` as follows: we set `vas.gmm.Delta` as *global variable* using `<<-`. The function does not always converge in which case we obtain a warning. Then the starting values must be improved, eg. by using a linear regression or a moment estimator. It is remarkable that using the explicit solution of the moments does *not* provide an improvement of the method.

```

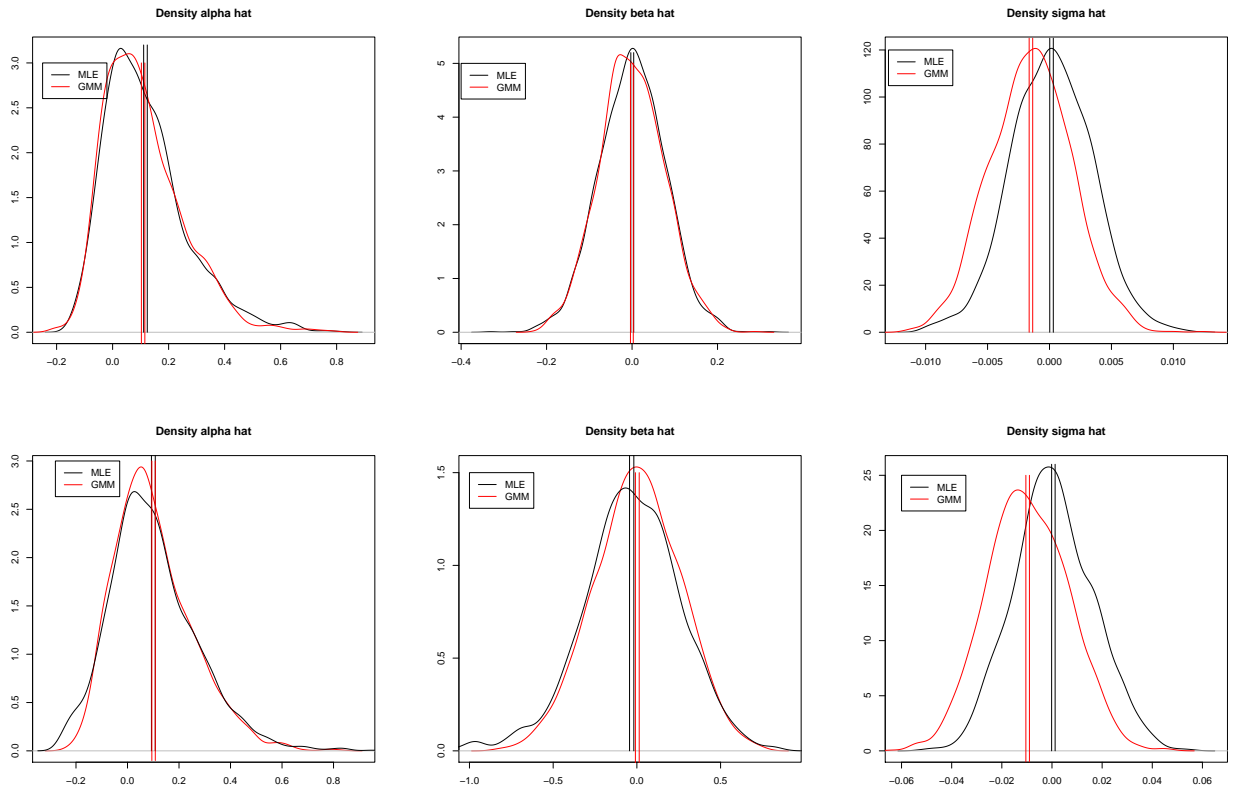
vas.gmm<- function (Delta,data,output=FALSE,tet0 = c(0.1,0.1,0.1) )
{
  vas.gmm.Delta <<- Delta # global
  gmm_res <- gmm(vas.g, data, t0 = tet0,
    control = list(maxit = 1000, reltol = 1e-10))
  coef(gmm_res)
}

```

	Mean error in α (RMS error in α)	Mean error in β (RMS error in β)	Mean error in σ (RMS error in σ)
MLE θ_1	0.110200 (0.183036)	0.000246 (0.077009)	-0.002569 (0.023177)
MLE θ_2	0.100273 (0.197201)	-0.031805 (0.288133)	0.001530 (0.016250)
GMM ³ θ_1	0.111979 (0.181764)	-0.002128 (0.076595)	0.000158 (0.003086)
GMM θ_2	0.101213 (0.183155)	0.002594 (0.259188)	-0.009704 (0.018861)

The performance of the estimators is best viewed with the density plots of the bias of our 2000 simulations: for $\theta_1 = (0.2, 0.08, 0.1)'$ we obtain a small advantage in the first coordinate while the estimator of σ has a bias. Note that the precision in σ is by far higher as in the two other coordinates. For $\theta_2 = (0.3, 0.4, 0.5)'$ this pattern becomes clearer: for β the GMM performs better while for σ there is a clear bias.

4 Estimation of diffusions and affine models



Exercise 12. *Implement a GMM estimator of a CIR model.*

A Commands

The distributions (norm, gamma, beta...) are listed in the appendix.

Command	Examples	Comment
<code>c</code>	<code>c(1,2)</code>	concatenates things
<code>d...</code>	<code>dnorm(0)</code>	density of a distribution.
<code>dim</code>	<code>dim(X)</code>	dimension of a matrix
<code>function</code>	<code>rtnew <- function(N,n=2) { rt(N,n) }</code>	Define a new function
<code>matrix</code>	<code>matrix(1:4,ncol=2)</code>	converts to a matrix
<code>par</code>	<code>par(mfrow=c(2,1)), par(op)</code>	splits graphical output
<code>q...</code>	<code>qnorm(0.95)</code>	the quantile of a distribution.
<code>r...</code>	<code>rnorm(100)</code>	random numbers from distributions.
<code>rep</code>	<code>rep(2,100)</code>	repeat numbers
<code>seq</code>	<code>seq(1,10,by=0.1)</code>	sequences of numbers
<code>t</code>	<code>t(X)</code>	transpose of a matrix
<code>%*%</code>	<code>A %*% t(X)</code>	matrix multiplication

Mathematical functions

`sin, cos, sqrt, exp, log`

A.1 Distributions

...(-)Distribution	S-Name	Parameters
Beta	beta	shape1, shape2, ncp= 0
Binomial	binom	size, prob
Cauchy	cauchy	location= 0, scale= 1
χ^2	chisq	df, ncp= 0
Exponential	exp	rate= 1
F	f	df1, df2 (ncp= 0)
Gamma	gamma	shape, rate= 1
Geometrische	geom	prob
Hypergeometrische	hyper	m, n, k
Log-Normal	lnorm	meanlog= 0, sdlog= 1
Logistische	logis	location= 0, scale= 1
Multinomial	multinom	size, prob
Multivariate Normal (Im package mvtnorm)	mvnorm	mean= rep(0, d), sigma= diag(d) (with d = Dimension)
Multivariate t (Im package mvtnorm)	mtnorm	(complicated; nur p...., q.... und r....)
Negative Binomial	nbinom	size, prob
Normal	norm	mean= 0, sd= 1
Poisson	pois	lambda
Wilcoxon's signed- sum of ranks	signrank	n
Students t	t	df, ncp= 0
Uniforme	unif	min= 0, max= 1
Weibull	weibull	shape, scale= 1
Wilcoxon's sum of ranks	wilcox	m, n

We will meet further multivariate distributions in the `QRMLib`