# Introduction to financial modeling in R platform

Perm State National Research University

CHAIR
OF INFORMATION SYSTEMS
AND MATHEMATICAL METHODS
IN ECONOMICS

**Arbuzov V.**
arbuzov@prognoz.ru

Perm R group

r-group.mifit.ru

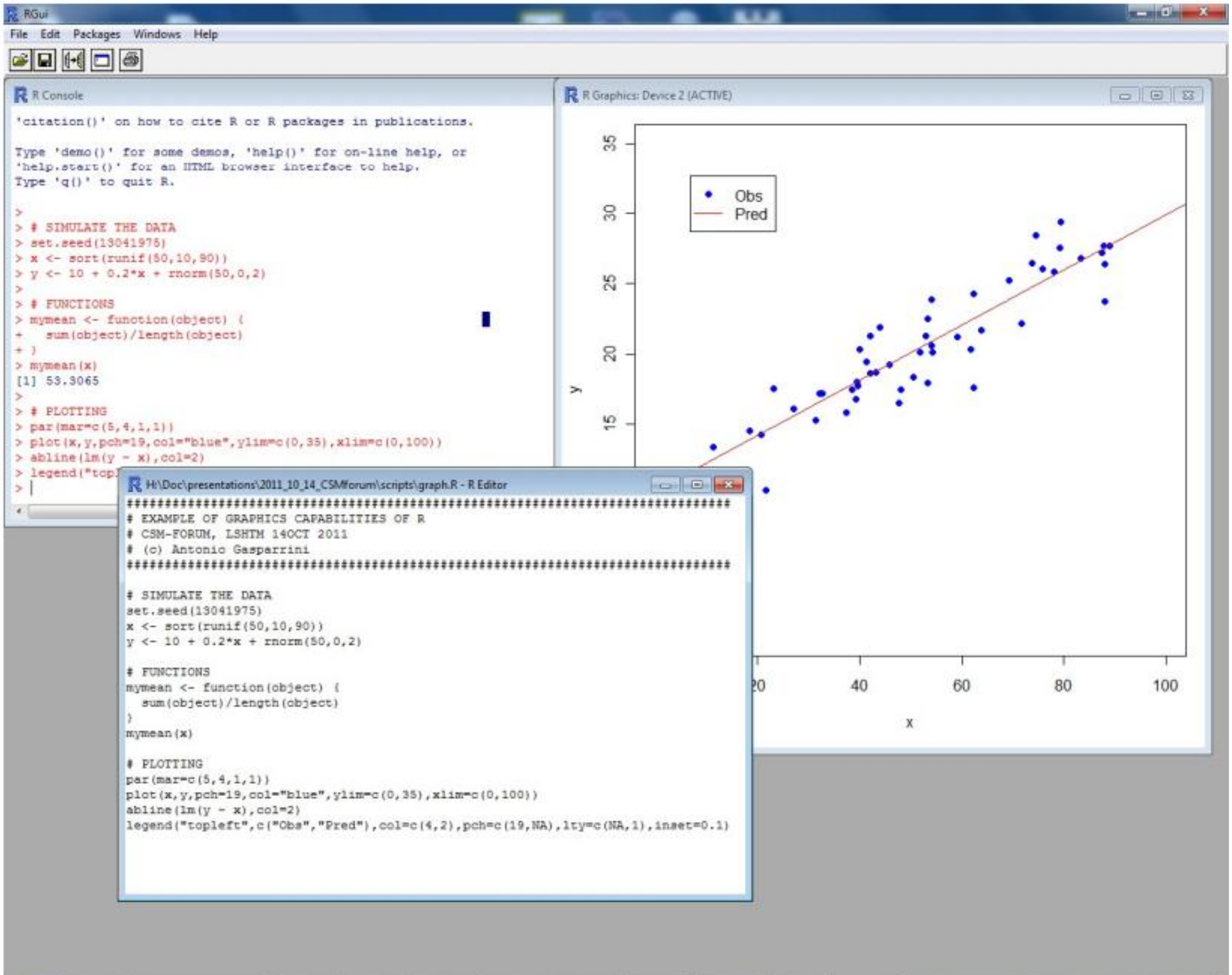
**Perm Winter School 2014**

**Russia, Perm, 31 January 2014**

**workshop**

# Basic knowledgement about R

# R console


RGui
File Edit Packages Windows Help
R Console
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> # SIMULATE THE DATA
> set.seed(13041975)
> x <- sort(runif(50,10,90))
> y <- 10 + 0.2*x + rnorm(50,0,2)
> # FUNCTIONS
> mymean <- function(object) {
+ sum(object)/length(object)
+ }
> mymean(x)
[1] 53.3065
> # PLOTTING
> par(mar=c(5,4,1,1))
> plot(x,y,pch=19,col="blue",ylim=c(0,35),xlim=c(0,100))
> abline(lm(y ~ x),col=2)
R Graphics: Device 2 (ACTIVE)
Obs
Pred
y
x
H:\Doc\presentations\2011_10_14_CSMforum\scripts\graph.R - R Editor
# EXAMPLE OF GRAPHICS CAPABILITIES OF R
# CSM-FORUM, LSHTM 14OCT 2011
# (c) Antonio Gasparrini
# SIMULATE THE DATA
set.seed(13041975)
x <- sort(runif(50,10,90))
y <- 10 + 0.2*x + rnorm(50,0,2)
# FUNCTIONS
mymean <- function(object) {
sum(object)/length(object)
}
mymean(x)
# PLOTTING
par(mar=c(5,4,1,1))
plot(x,y,pch=19,col="blue",ylim=c(0,35),xlim=c(0,100))
abline(lm(y ~ x),col=2)
legend("topleft",c("Obs","Pred"),col=c(4,2),pch=c(19,NA),lty=c(NA,1),inset=0.1)

## Simple calculation

```
4+3
5-1
4*6
10/2
2^10
9%%2
9%/%2
abs(-1)
exp(1)
sqrt(25)
sin(1)
pi
cos(pi)
sign(-106)
log(1)
```

## Mathematical functions in R

| | |
|---|---|
| log(x) | log to base e of $x$ |
| exp(x) | antilog of $x$ ($e^x$) |
| log(x,n) | log to base $n$ of $x$ |
| log10(x) | log to base 10 of $x$ |
| sqrt(x) | square root of $x$ |
| factorial(x) | $x!$ |
| choose(n,x) | binomial coefficients $n!/(x!\ (n-x)!)$ |
| gamma(x) | $\Gamma(x)$, for real $x$ $(x-1)!$, for integer $x$ |
| lgamma(x) | natural log of $\Gamma(x)$ |
| floor(x) | greatest integer $< x$ |
| ceiling(x) | smallest integer $> x$ |
| trunc(x) | closest integer to $x$ between $x$ and 0 trunc(1.5) $= 1$, trunc(-1.5) $= -1$ trunc is like floor for positive values and like ceiling for negative values |
| round(x, digits=0) | round the value of $x$ to an integer |
| signif(x, digits=6) | give $x$ to 6 digits in scientific notation |
| runif(n) | generates $n$ random numbers between 0 and 1 from a uniform distribution |
| cos(x) | cosine of $x$ in radians |
| sin(x) | sine of $x$ in radians |
| tan(x) | tangent of $x$ in radians |
| acos(x), asin(x), atan(x) | inverse trigonometric transformations of real or complex numbers |
| acosh(x), asinh(x), atanh(x) | inverse hyperbolic trigonometric transformations of real or complex numbers |
| abs(x) | the absolute value of $x$, ignoring the minus sign if there is one |

# Coding

```
x<-1:10
```

| | |
|---|---|
| max(x) | maximum value in $x$ |
| min(x) | minimum value in $x$ |
| sum(x) | total of all the values in $x$ |
| mean(x) | arithmetic average of the values in $x$ |
| median(x) | median value in $x$ |
| range(x) | vector of min($x$) and max($x$) |
| var(x) | sample variance of $x$ |
| cor(x,y) | correlation between vectors $x$ and $y$ |
| sort(x) | a sorted version of $x$ |
| rank(x) | vector of the ranks of the values in $x$ |
| order(x) | an integer vector containing the permutation to sort $x$ into ascending order |
| quantile(x) | vector containing the minimum, lower quartile, median, upper quartile, and maximum of $x$ |
| cumsum(x) | vector containing the sum of all of the elements up to that point |
| cumprod(x) | vector containing the product of all of the elements up to that point |
| cummax(x) | vector of non-decreasing numbers which are the cumulative maxima of the values in $x$ up to that point |
| cummin(x) | vector of non-increasing numbers which are the cumulative minima of the values in $x$ up to that point |
| pmax(x,y,z) | vector, of length equal to the longest of $x$, $y$ or $z$, containing the maximum of $x$, $y$ or $z$ for the $i$th position in each |

# Coding

## Useful commands

**to create a vector**

1:10
seq(1,10)
rep(1,10)

**assignment operator**

**<-** or **->**
x<-10
10->X

**working with vectors**

A<-1:10
B<-c(2,4,8)
A*B
A>B

**case sensitive**

X
x

**R is a case sensitive language. FOO, Foo, and foo are three different objects!**

## Matrix

```
y <- matrix(nrow=2,ncol=2)
y[1,1] <- 1
y[2,1] <- 2
y[1,2] <- 3
y[2,2] <- 4
x <- matrix(1:4, 2, 2)
```

A *matrix* is a vector with two additional attributes: the number of rows and the number of columns

## Matrix Operations

```
x %*% y
x* y
3*y
x+y
x+3
x[,2]
x[1,]
rbind(x,y)->z
cbind(x,y)
z[1:2,]
z[z[,1]>1,]
z[which(z[,1]>1),1]
```

## List

```
j <- list(name="Joe", salary=55000, union=T)
j$salary
j[["salary"]]
j[[2]]
j$name [2]<-"Poll"
j$salary[2]<-10000
j$union [2]<-"F"
```

In contrast to a vector, in which all elements must be of the same mode, R's **list** structure can combine objects of different types.

## Data Frame

```
kids <- c("Jack","Jill")
ages <- c(12,10)
d <- data.frame(kids,ages)
d[1,]
d[,1]
d[[1]]
```

## Arrays

```
my.array <- array(1:24, dim=c(3,4,2))
```

a **data frame** is like a matrix, with a two-dimensional rows-andcolumns structure. However, it differs from a matrix in that each column may have a different mode.

***Loops***

***for***

```
x <- c(5,12,13)
for (n in x) print(n^2)

for (i in 1:10)
{
x<-i*3
cos(x)->x
print(x)
}
```

**while**

```
i <- 1
while (i <= 10) i <- i+4

i <- 1
while (i <= 10)
{
x<-i*3
cos(x)->x
print(c(x,i))
i <- i+1
}
```

***if-else***

```
i<-3
if (i == 4) x <- 1 else x <- 3
```

# Import/Export data

***Import from text files or csv***

```
mydata <- read.table("d:/my.data.txt",
header=TRUE,
sep=",")
```

***Import from Excel***

```
library(xlsx)
mydata<-read.xlsx("d:/my.data.xlsx",1)
```

**Exporting Data**

```
write.table(mydata, "c:/mydata.txt", sep="\t")
write.xlsx(mydata, "c:/mydata.xls")
```

**Import/Export Data to clipboard**

```
read.table("clipboard",sep="\t")
write.table(arg1,"clipboard",sep="\t")
```

P.S.

```
rb <- function(arg1){read.table("clipboard",sep="\t")}
cb <- function(arg1){write.table(arg1,"clipboard",sep="\t")}
```

**Creating a Graph**

```
attach(mtcars)
plot(wt, mpg)
abline(lm(mpg~wt))
title("Regression of MPG on Weight")
```

**Bar Plot**

```
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution",
  xlab="Number of Gears")
```

**Pie Charts**

```
slices <- c(10, 12,4, 16, 8)
lbls <- c("US", "UK", "Australia",
"Germany", "France")
pie(slices, labels = lbls, main="Pie
Chart of Countries")
```

**Boxplot**

```
boxplot(mpg~cyl,data=mtcars, mai
n="Car Milage Data",
  xlab="Number of Cylinders",
ylab="Miles Per Gallon")
```

# Advanced knowledgement about R

# Select data from matrix

```
x <- matrix(0,50,2)

x[,1]

x[1,]

x[,1]<-rnorm(50)

x

x[1,]<-rnorm(2)

x

x <- matrix(rnorm(100),50,2)

x[which(x[,1]>0),]
```

Operators
and &
or |

## Basic distributions in R

| | |
|---|---|
| **Beta** | ?beta |
| Binomial | ?binom |
| **Cauchy** | ?cauchy |
| **Chi-squared** | ?chisq |
| **Exponential** | ?exp |
| **F** | ?f |
| **Gamma** | ?gamma |
| Geometric | ?geom |
| Hypergeometric | ?hyper |
| **Log-normal** | ?lnorm |
| Multinomial | ?multinom |
| Negative binomial | ?nbinom |
| **Normal** | ?norm |
| Poisson | ?pois |
| **Student's t** | ?t |
| **Uniform** | ?unif |
| **Weibull** | ?weibull |

**d – density**
**q – quantile**
**r – random**

**Plot density of chosen distribution…**

```
x <- seq(-4, 4, length=100)
dx <- d?????(x)
plot(x, hx, type="l")
```

**Generate random variables of chosen distribution …**

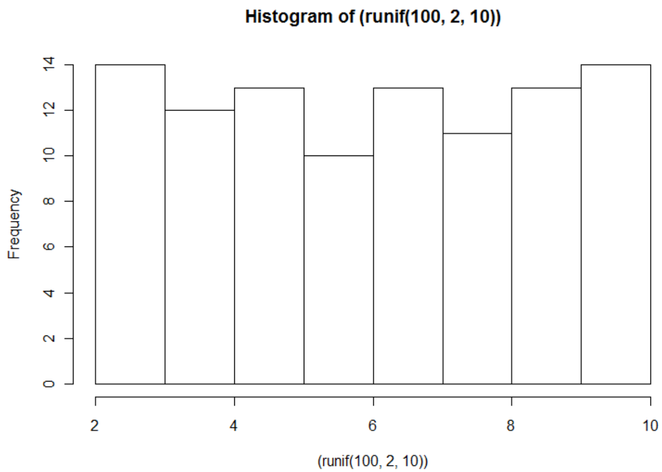
```
x<-rnorm(1000)
```

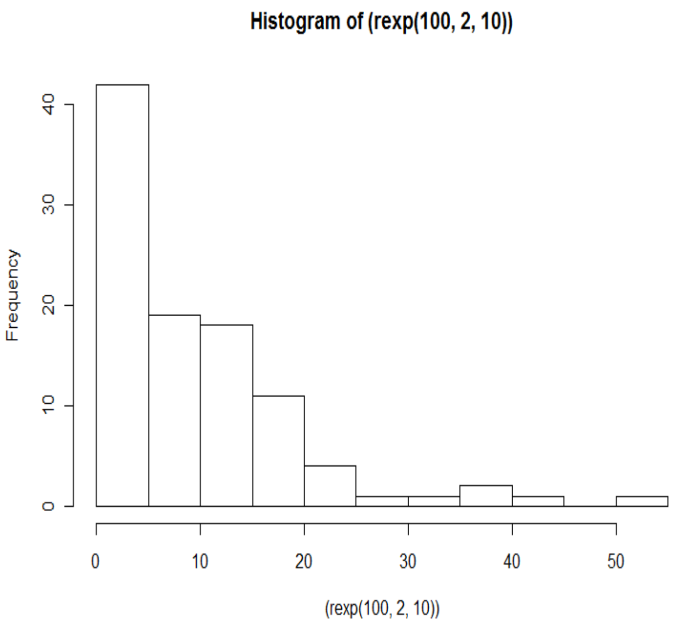
# Distributions

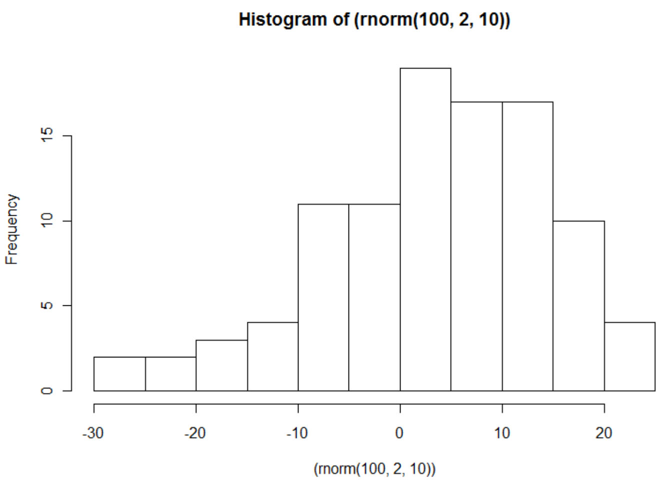
## What is this distribution ?

**hist (?)**
**density (?)**

?



# Estimation of parameters

**Let's estimate parameters of chosen distribution….**

```
library(MASS)

fitdistr(x,"normal")

params<-fitdistr(x,"normal")$estimate
```

**Compare theoretical and empirical distributions…**

```
hist(x, freq = FALSE,ylim=c(0,0.4))

curve(dnorm(x, params[1], params[2]), col = 2, add = TRUE)
```

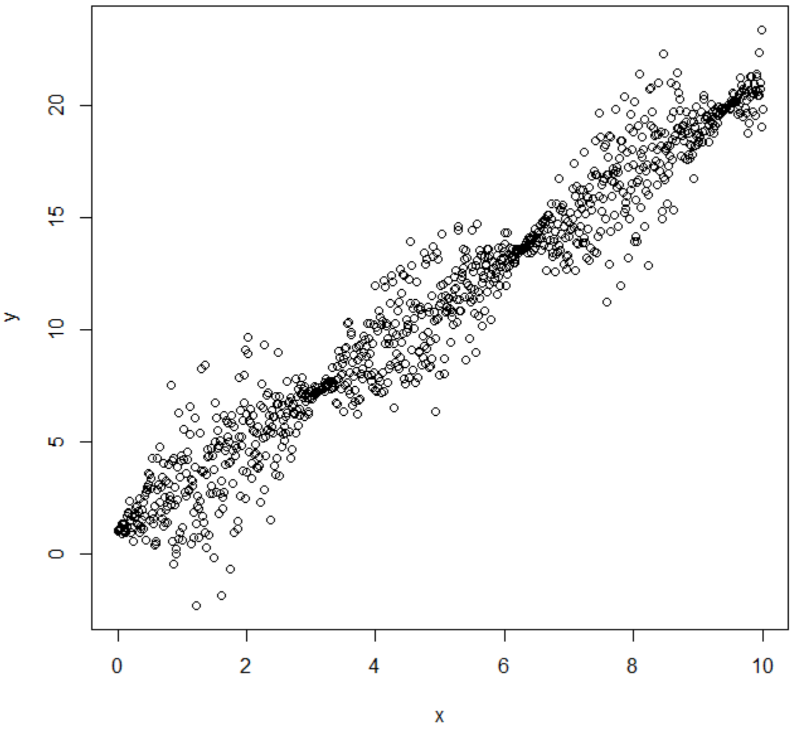
# Correlations

```
y<-rnorm(1000)

cor(x,y)

cor(x,y, ,method = ?????)

acf(y)
```

# Linear least-squares method

```
x<-seq(0,10,length=1000)
```

```
y<-1+2*x+sin(x)*rnorm(1000,0,2)
plot(x,y)
```




How to estimate this parameters?

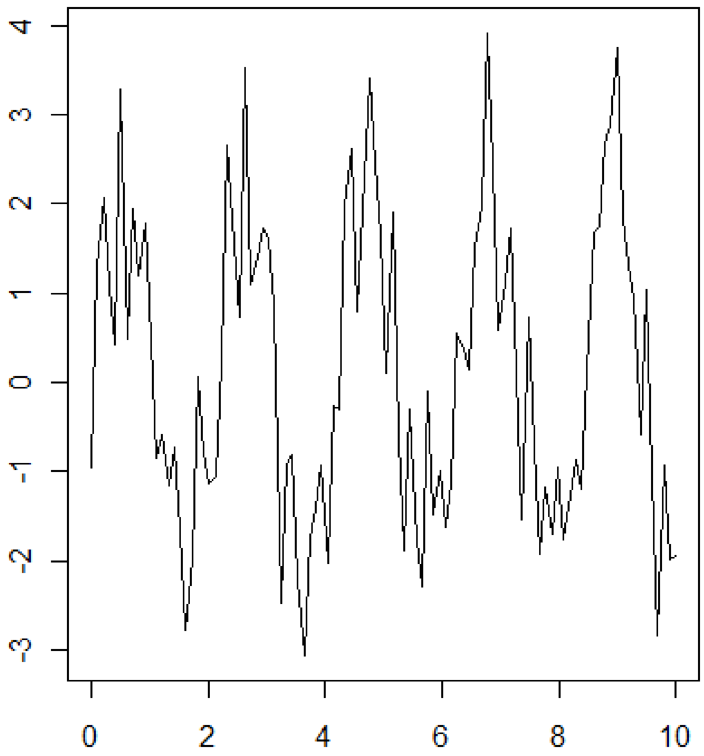
$$y = \alpha + \beta x$$

```
lm(y ~ x)
summary(lm(y ~ x))
```

```
abline(lm(y ~ x),col="red",lwd=3)
```

# Nonlinear least-squares method

```
x<-seq(0,10,length=1000)

y<-2*sin(3*x)+rnorm(1000,0,0.8)
```




How to estimate this parameters?

$$y = \alpha \sin(\beta x)$$

```
help(nls)

nls(y ~ A*sin(B*x))

nls(y ~ A*sin(B*x),start=list(A=1.8,B=3.1))
```

Add few diagrams to graph

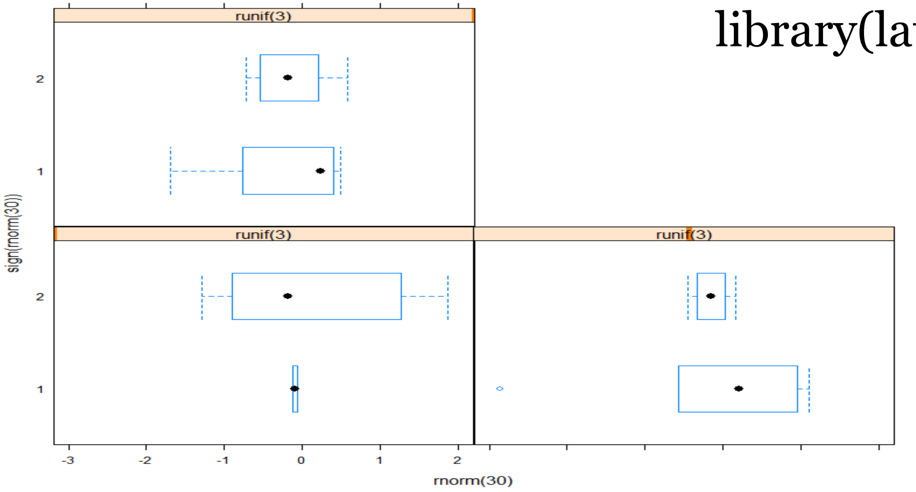
```
par(mfrow=c(2,2))

plot(rnorm(100),rbeta(100,12,1))
```

Add legend to graph

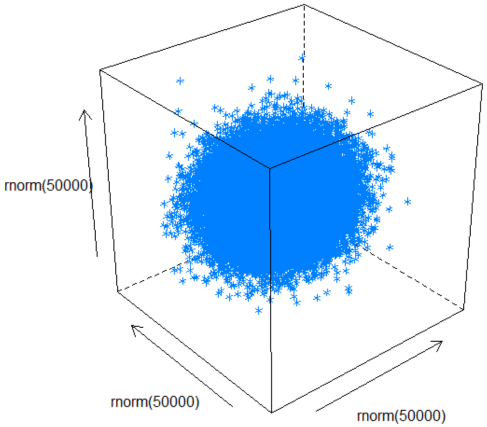
**legend**("topright", inset=.05, title="legend", c("4","6","8"), horiz=TRUE)

# Package lattice

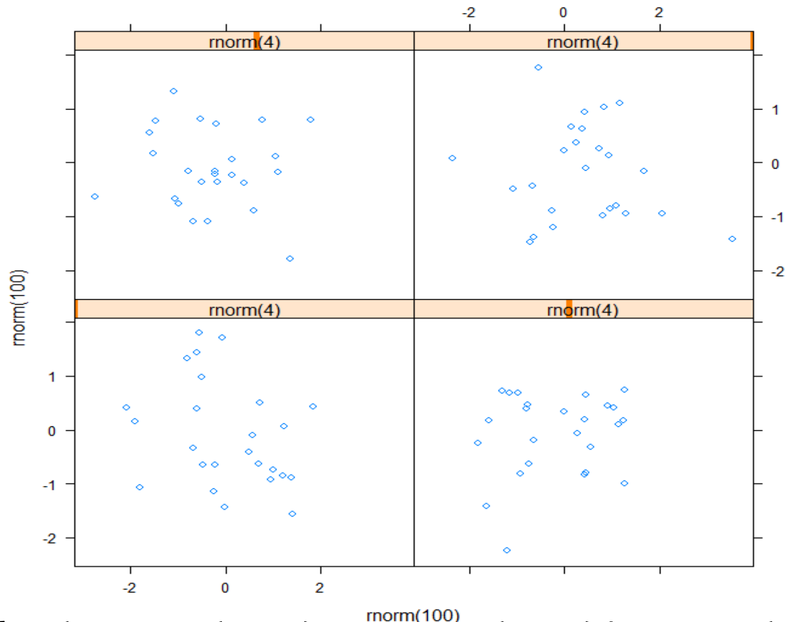
library(lattice)



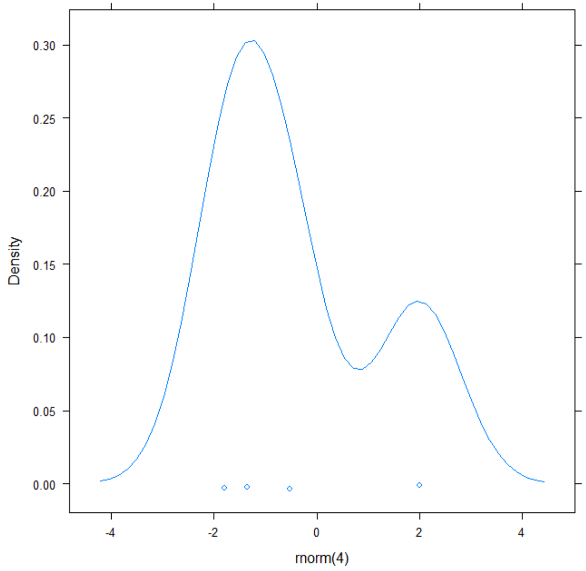

bwplot(sign(rnorm(30))~rnorm(30)|runif(3))




cloud(y~x*y)



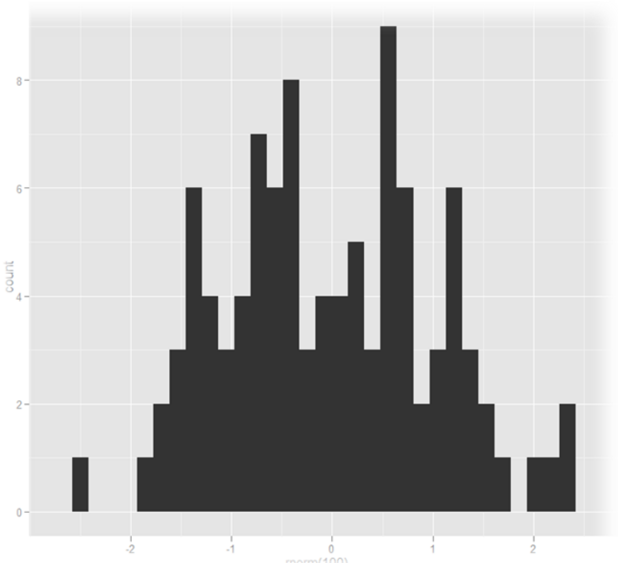

xyplot(rnorm(100)~rnorm(100)|rnorm(4))



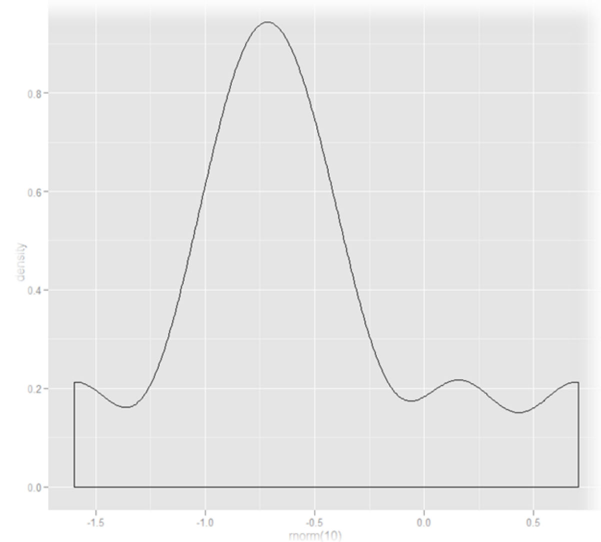

densityplot(rnorm(4))

library(ggplot2)

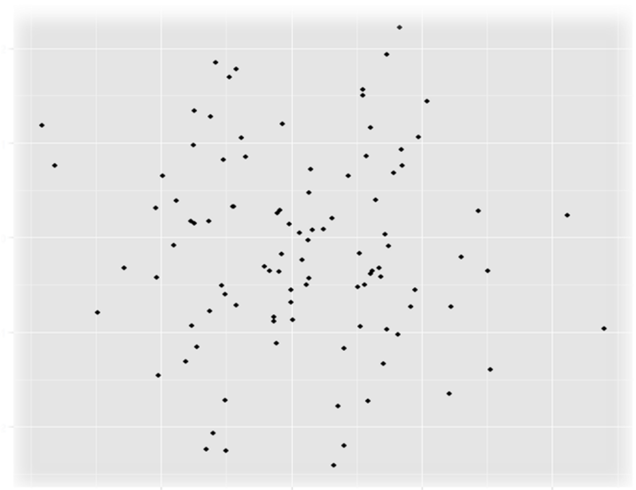
qplot(rnorm(100))



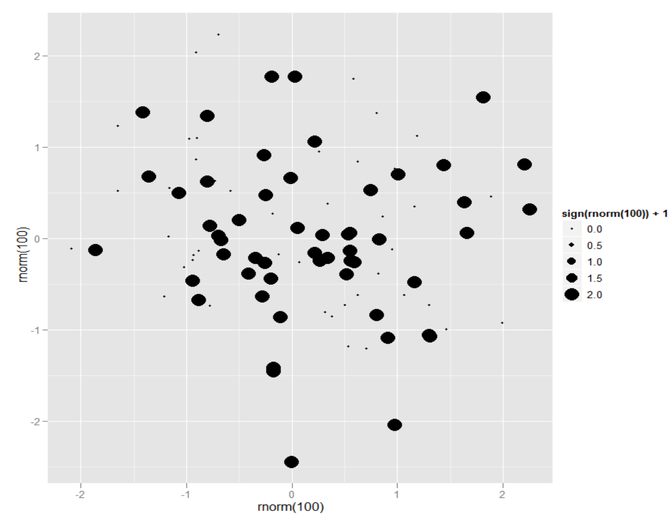
qplot(rnorm(100),geom='density')



qplot(rnorm(100),rnorm(100))

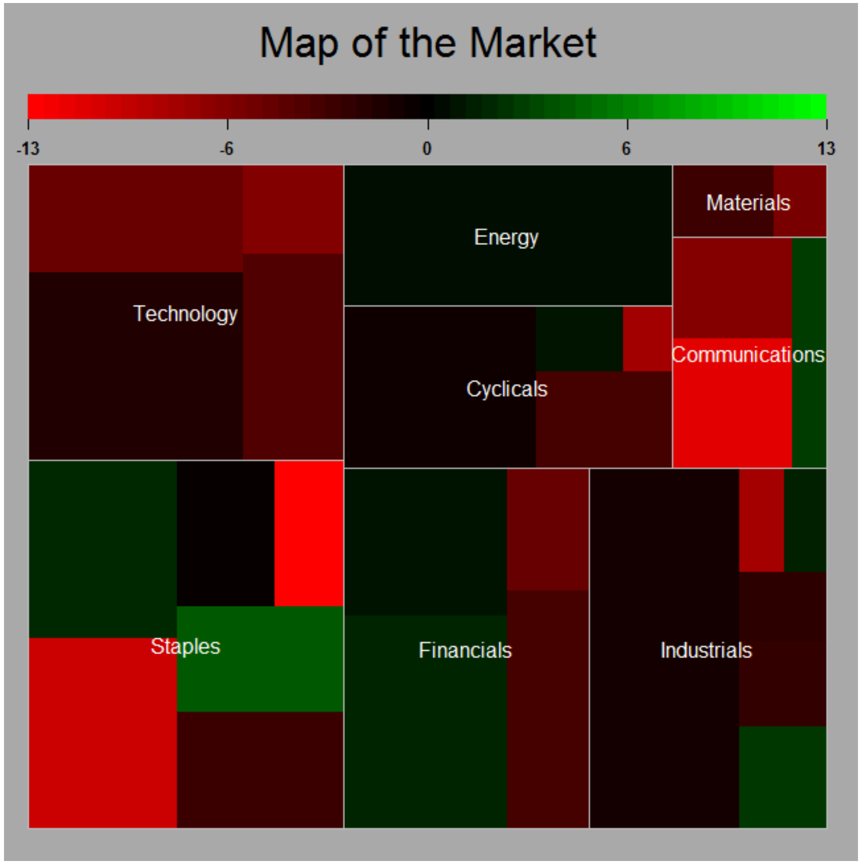


qplot(rnorm(100),rnorm(100),
size=sign(rnorm(100))+1)



# Tree maps

```
library(portfolio)

data(dow.jan.2005)

map.market(id    = dow.jan.2005$symbol,
area  = dow.jan.2005$cap.bil,
group = dow.jan.2005$sector,
color = 100 * dow.jan.2005$month.ret)
```



# Download Data

Package **quantmod**

getSymbols("GOOG",src="yahoo", from = "2007-01-01“, to = Sys.Date())

getSymbols("USD/EUR",src="oanda")

Package **rusquant**

getSymbols("SPFB.RTS", from="2011-01-01", src="Finam“, period="hour" , auto.assign=FALSE)

1min, 5min, 10min, 15min, 30min, hour, day, week, month

# Working with package quantmod

## Data visualization

```
barChart(AAPL)
candleChart(AAPL,multi.col=TRUE,theme="white")
chartSeries(AAPL,up.col='white',dn.col='blue')
```

## Add technical indicators

```
addMACD()
addBBands()
```

## Select data

```
AAPL['2007']
AAPL['2007-03/2007']
AAPL['/2007']
AAPL['2007-01-03']
```

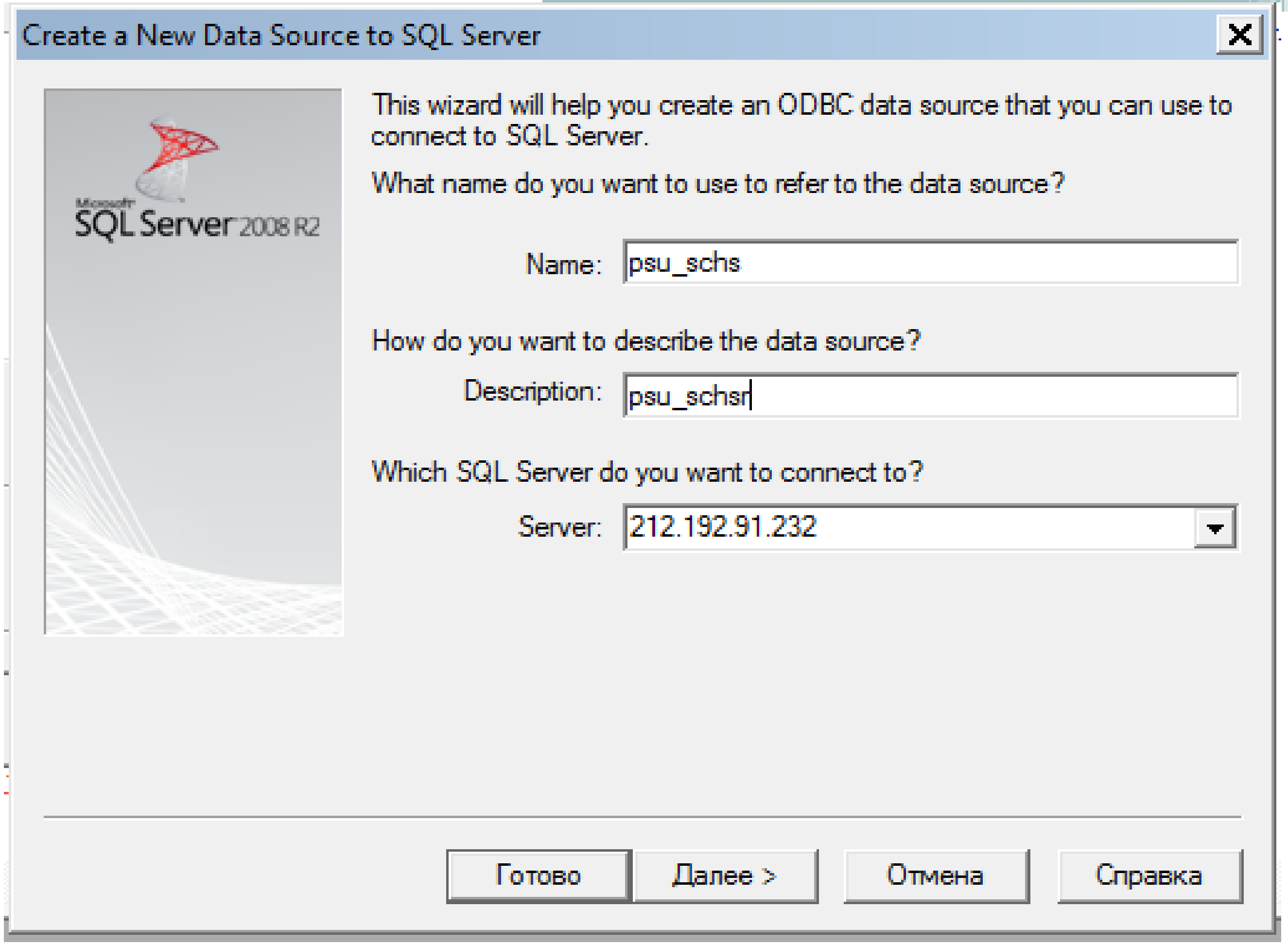
## Data management

```
to.weekly(AAPL)
to.monthly(AAPL)
dailyReturn(AAPL)
weeklyReturn(AAPL)
monthlyReturn(AAPL)
```

# Load Data from Database

Package RODBC

library(RODBC)




```
odbcDriverConnect(“”)
channel <- odbcConnect("psu_schs",“student","Qwerty1")
sqlQuery(channel, “select * from  LPPL_MODELS”)
```

# Practice

# Practice № 1. Working with data

**TASK:**

a. Download Data of your instrument
b. Plot price
c. Add technical indicators
d. Calculate price returns

Commands to help :

```
barChart(AAPL)
chartSeries(AAPL,up.col='white',dn.col='blue')
AAPL['2007-03/2007']
addMACD()
dailyReturn(AAPL)
```

# Practice № 2. Distribution of price returns

**TASK :**

a. Download Data of your instrument
b. Calculate returns of close prices
c. Plot density of distribution
d. Estimate parameters of distribution
e. Plot in one graph empirical and theoretical distributions

Commands to help :

```
getSymbols("AAPL", auto.assign=FALSE)

library(MASS)
fitdistr(x,"normal")

hist(x)
density(x)

curve(dnorm(x, params[1], params[2]), col = 2, add = TRUE)
```

# Practice № 3. Estimation of correlation

**TASK :**

a. Download Index Data (ticker: “MICEX”)
b. Download Data of your instrument
c. Calculate returns of close prices
d. Calculate correlation of returns
e. Calculate correlation of returns in 2012 year
f. Calculate correlation of returns in 2008 year
g. Calculate autocorrelation function of returns

Commands to help :

```
getSymbols(" MICEX ",
src="Finam", period="day" , auto.assign=FALSE)
```

```
AAPL['2007']
AAPL['2007-03/2007']
AAPL['/2007']
AAPL['2007-01-03']
```

# Practice № 4. Volatility estimation

**TASK :**

a. Download Data of your instrument
b. Calculate returns of close prices
c. Plot clusterization of volatility
d. Estimate model **garch**
e. garchFit(data=x) @sigma.t

Commands to help :

```
AAPL['2007']
AAPL['2007-03/2007']
AAPL['/2007']
AAPL['2007-01-03']
```

# The practical task №5. Calculation of VaR

**TASK :**

a. Download Data of your instrument
b. Calculate returns of close prices
c. Calculate historical VaR
d. Calculate parametric VaR
e. library(PerformanceAnalytics)
f. help(VaR)

Commands to help :

quantile(x,0.95, na.rm=TRUE)

AAPL['2007']
AAPL['2007-03/2007']
AAPL['/2007']
AAPL['2007-01-03']

# The practical task № 6. Estimate LPPL model

**TASK :**

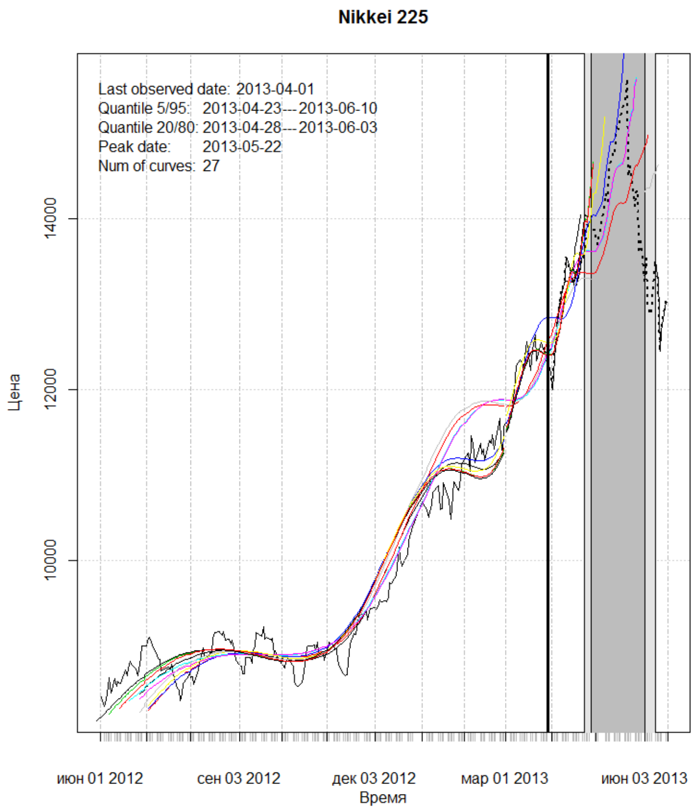
a. Download Nikkei Index Data(ticker: “^N225”) from June 2012 to June 2013
b. Estimate parameters of model LPPL

**MODEL LPPL:**

$$ln[p(t)] = A + B(t_c - t)^m + C(t_c - t)^m \cos[\omega \log(t_c - t) - \varphi]$$

Commands to help :

help(nsl)



# Q&A

**arbuzov@prognoz.ru**