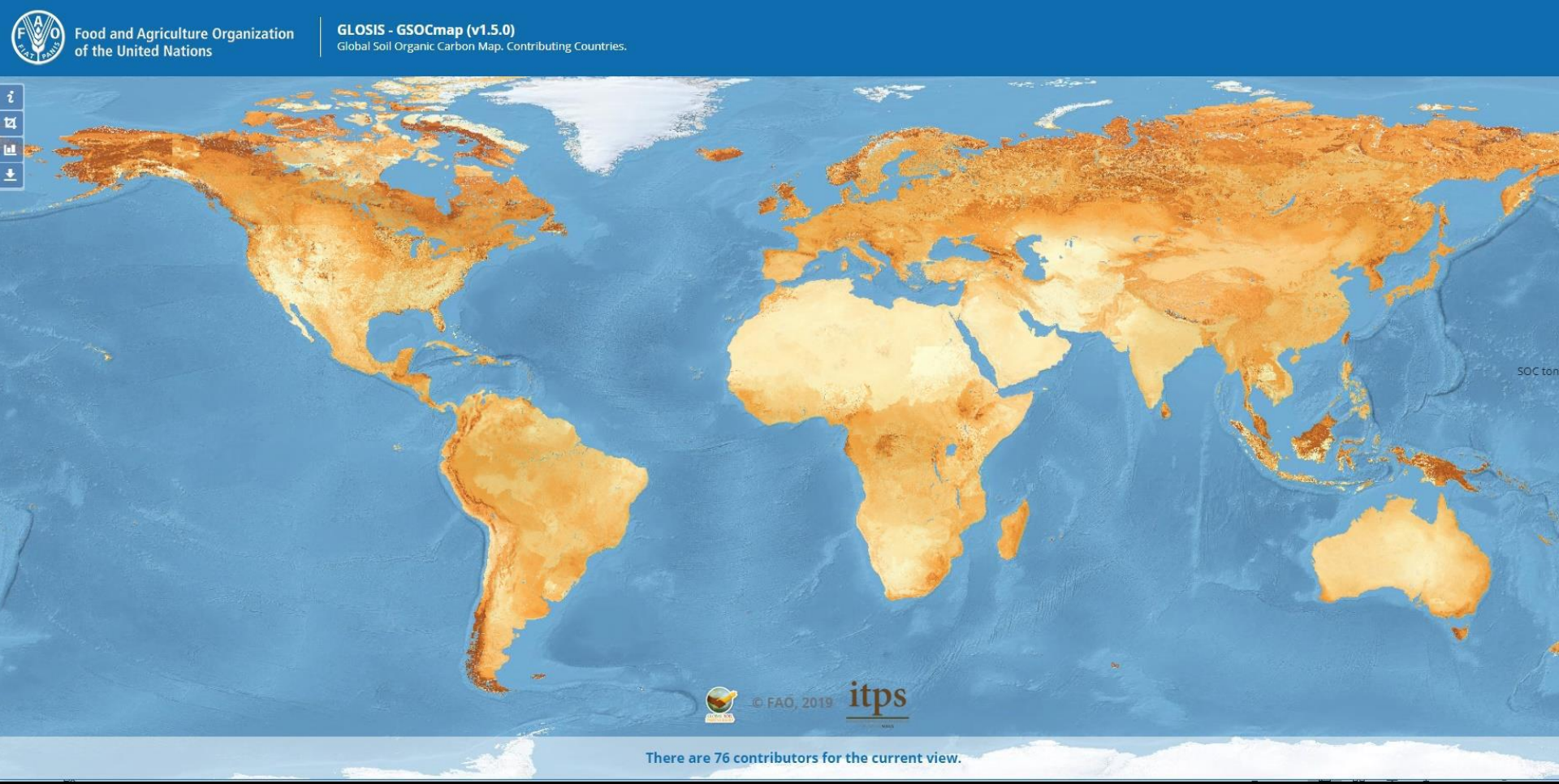


Introduction to R for spatial modeling

Ruhollah Taghizadeh
Soil Science and Geomorphology

E-mail: *ruhollah.taghizadeh-mehrjardi@mnf.uni-tuebingen.de*

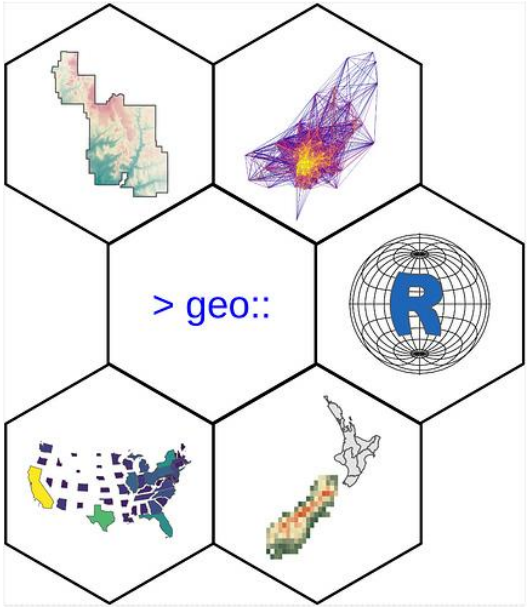
Content

❑Part 1

- ▪R and RStudio
- ▪Setting up an R session
- ▪Data types
- ▪Data structures
- ▪Selecting subsets
- ▪Functions
- ▪Import/export of data
- ▪Plotting
- ▪Working with spatial data in R

❑ Part 2

- ▪Lets practice!!



What is R?

- **R is a free programming language for statistics**
- **R is command line based**
- **R can be expanded through packages**
- **R is used to analyze and visualize (geographic) data**
- **R is comparatively easy to access**

Why R?

- It is free
- Available for the most popular operating systems
- Low system requirements
- Continuous further development
- Active user community with its own conference (useR!)
- Application in many natural sciences
- PYPL Popularity of Programming Language: 7th place

First steps

- https://www.r-project.org/
  - **R website, where you will find the program itself, FAQs, manuals and other useful information about and around R:**
  - **CRAN (Comprehensive R Archive Network; cran.r-project.org)**
  - **Manuals (cran.r-project.org/manuals.html)**
  - **Mailing lists**
  - **Books list**

R software

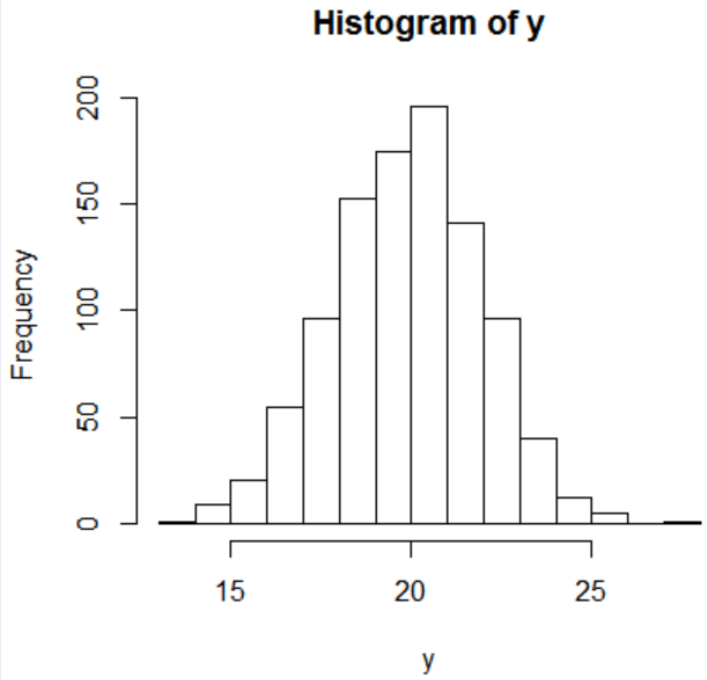
RGui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'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.

[Previously saved workspace restored]

> x = 20
> x + 9
[1] 29
> y = rnorm(1000, mean = x, sd =2)
> hist(y)
> c(mean(y),sd(y))
[1] 19.891741  2.055908
>
```

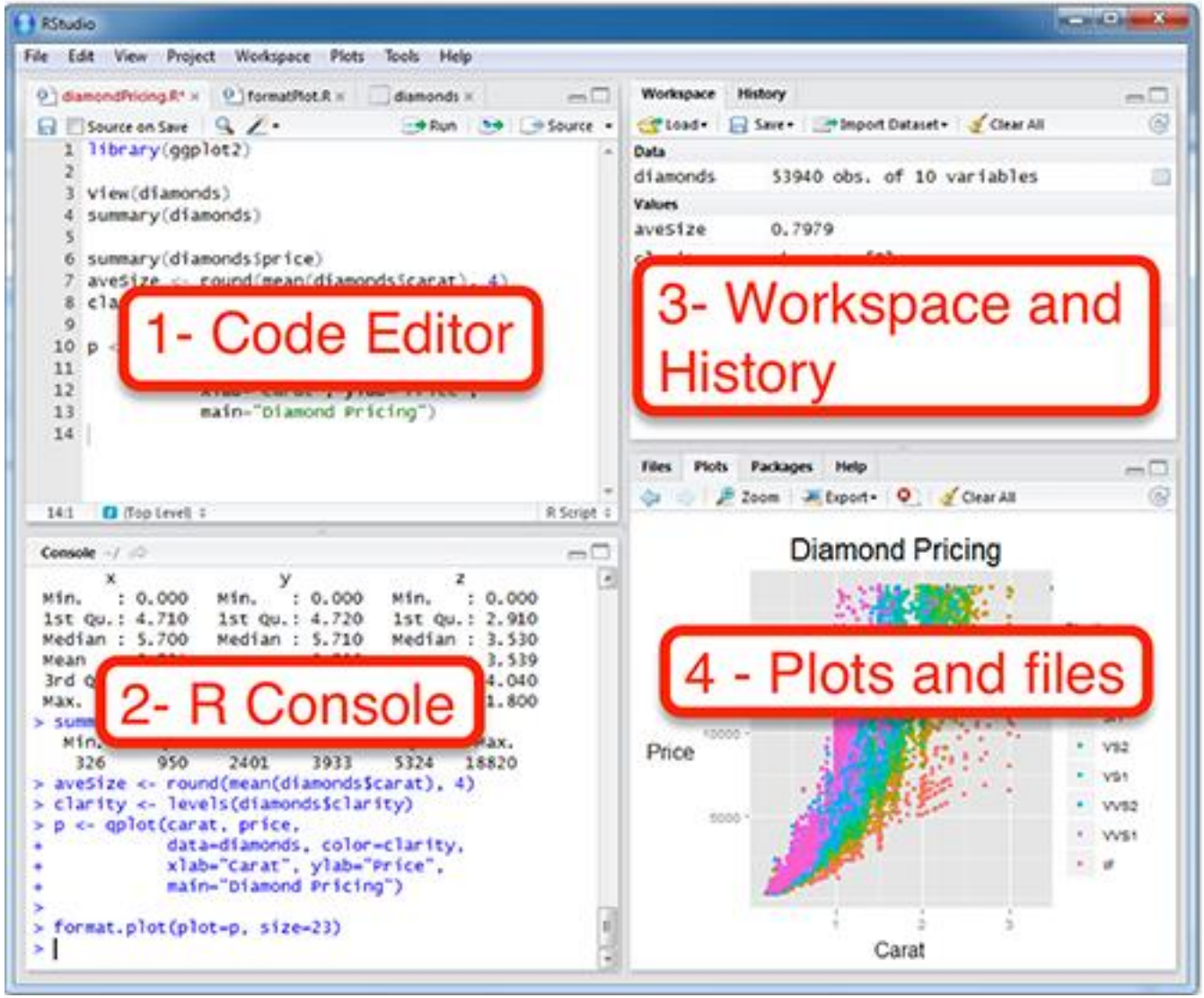
R Graphics: Device 2 (ACTIVE)



First steps

- **Other user interfaces**
  - **R Commander (https://www.rcommander.com/)**
  - **RStudio(https://rstudio.com/) (https://rstudio.com/products/rstudio/download/)**
  - **And some others (e.g., Tinn-R)**
  - **Open source and commercial user interfaces**
  - **Desktop and server variants**

RStudio


RStudio
File Edit View Project Workspace Plots Tools Help
diamondPricing.R* formatPlot.R diamonds
Source on Save Run Source
1 library(ggplot2)
3 View(diamonds)
4 summary(diamonds)
6 summary(diamonds$price)
7 aveSize <- round(mean(diamonds$carat), 4)
10 p <-
13 main="Diamond Pricing")
14:1 (Top Level) R Script
1- Code Editor
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
3- Workspace and History
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
Carat
4 - Plots and files
Console
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean 3.539
3rd Qu. 4.040
Max. 1.800
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
> format.plot(plot=p, size=23)
2- R Console

First steps-Help

- **R-help(Mailing-Listen)**

  Most important list for user questions
  (Pay attention to posting)

- **R Community, most problems are not new and solutions can be found online**

  Stackoverflow

  R-bloggers,

  Quick-R,

- **Books and eBooks**

  R Cookbook

  R for Data Science, Spatial Data Science,

  Geocomputationwith R, The R Inferno,
  Rstudio Cheatsheets

Essentials of R

- **Basic arithmetic functions:**

+ (addition), - (subtraction), * (multiplication), / (division),

^ (exponentiation)

```
7 + 4 # => 11
7 - 4 # => 3
7 / 2 # => 3.5
7 * 2 # => 14
7 ^ 2 # => 49
```

Essentials of R

- **Basic arithmetic operations:**

  Logarithms and exponentials: log2(x), log10(x), exp(x)

  Trigonometric functions: cos(x), sin(x), tan(x), acos(x), asin(x), atan(x)

  Other mathematical functions: abs(x): absolute value; sqrt(x): square root.

```
log2(4)  # => 2
abs(-4)  # => 4
sqrt(4)  # => 2
```

Essentials of R

- **Assigning values to variables:**

```
s <- 5
5
```

```
s <- seq(from=20, to=0, by=-2)
20 18 16 14 12 10 8 6 4 2 0
```

```
s <- rep(1:4, 2)
1 2 3 4 1 2 3 4
```

```
s <- rnorm(10, mean = 10, sd = 1)
 9.3 10.5 10.4 13.2 10.1 10.9 11.5 10.4  9.1  9.6
```

Essentials of R

- **Assigning values to variables:**

```
Clay   <-  40
Sand   <-  40
Silt    <-   20

Sum <- Clay + Sand + Silt

print (Sum) # => 100
```

Note: R is case sensitive!

Data types

- **Basic data types:**
  - **Numeric**

```
> x <- 10.1
> x
[1] 10.1
> class(x)
[1] "numeric"
```

  - **Integer**

```
> y <- as.integer(x)
> y
[1] 10
> class(y)
[1] "integer"
```

  - **Character**

```
> z <- "Bas"
> z
[1] "Bas"
> class(z)
[1] "character"
```

  - **Logical**

```
> x <- 1; y<-2
> z <- x > y
> z
[1] FALSE
> class(z)
[1] "logical"
>
```

class(x)

Data structures

- **Vectors:**
  - **Sequence of elements of the same basic type (character, logical, integer or numeric).**

```
x <- c(1, 2, 3)
x <- 1:3
y <- c(1, 1, 1)
y <- rep(2, 10)
z <- as.character(1:3)
z <- c("a", "b", "c")
```

```
length(z)
names(x) <- z
x[2:3]
x["b"]
as.vector(x)
is.vector(x)
```

Data structures

- **Factors:**
  - **A vector with categorical values (classes)**

```
x <- c(1, 2, 3)
y <- as.factor(x)
```

```
> y
[1] 1 2 3
Levels: 1 2 3
```

Data structures

- **Matrix:**
  - **Matrices are similar to vectors, but have two dimensions**

```
x <- 1:20
x <- matrix(x, 5, 4)              # matrix(x, nrow = 5, ncol = 4)
x[2, 3]
x[c(1, 5), 2:4]
x[, 2:4]

dim(x); nrow(x);  ncol(x)
length(x)
as.matrix(x)
is.matrix(x)
x[, "b"]
x[, c("a", "b")]
```

Data structures

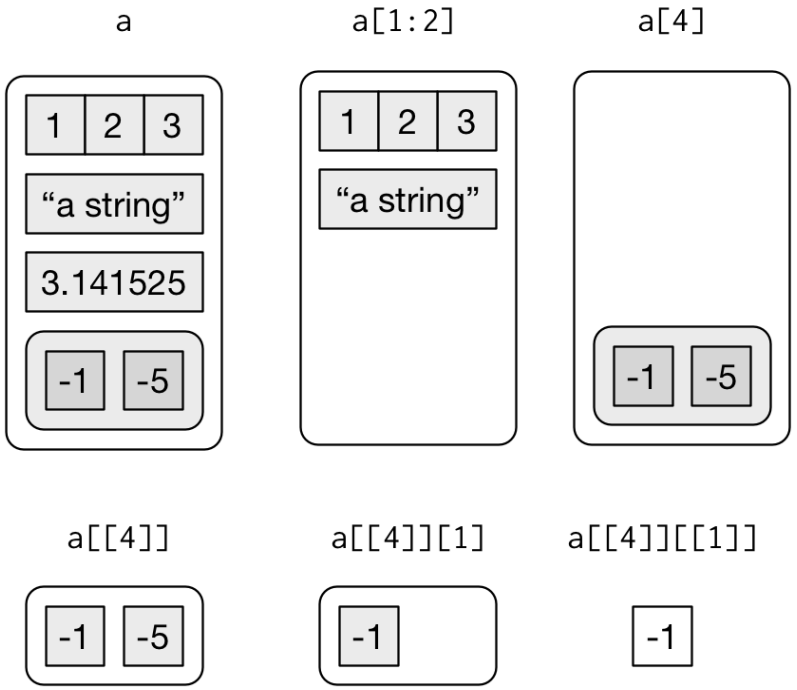
- **Arrays:**
  - **Arrays are similar to matrices, but can have more than two dimensions**

```
x <- 1:60
x <- array(x, c(5, 4, 3))
x[2, 3, 1]
x[1, 2:4, 3]
x[, , 1]

dim(x); dimnames(x)
nrow(x); ncol(x)
length(x)
as.array(x)
is.array(x)
```

Data structures

- **List:**
  - **Ordered sequence of objects of the same or dissimilar type**

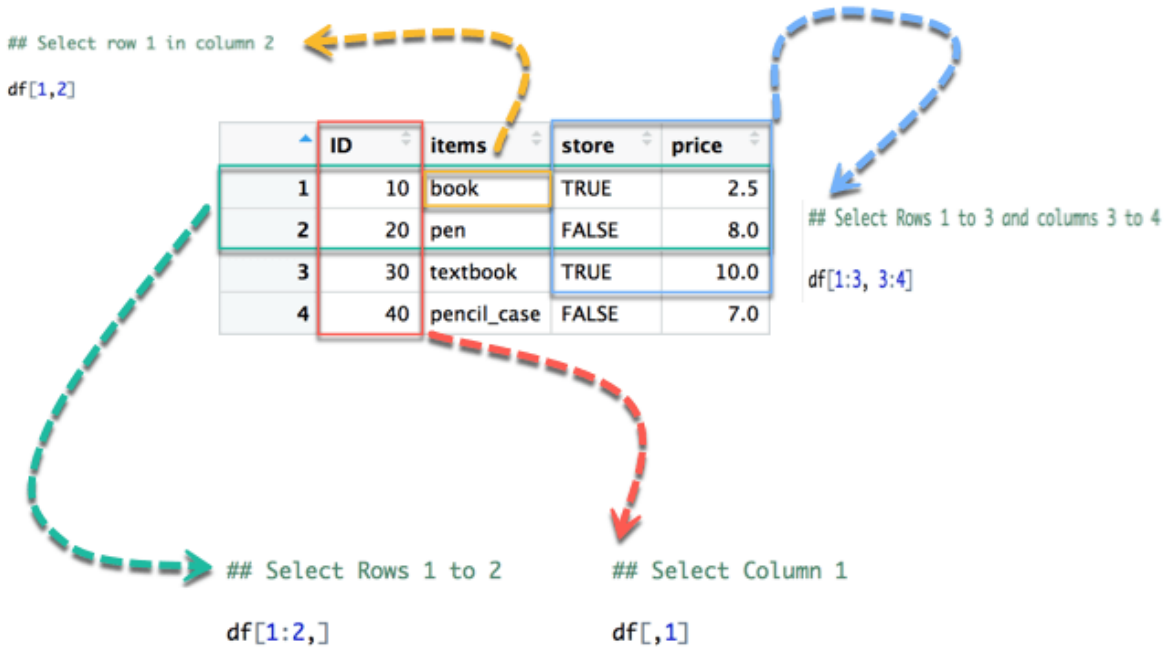
```
x <- list(Eins = 11:15, Zwei = c("a", "b", "c"), Drei = (1:4) > 0)
y <- list(x = x, Vier = 1:3)
x$Eins
y$x$Eins
y$Vier
y[[2]]
length(x)
length(y)
y$Fuenf <- names(x)
```



Data structures

- **Data frame:**
  - **Data frames are similar to matrices but with the same or different types**

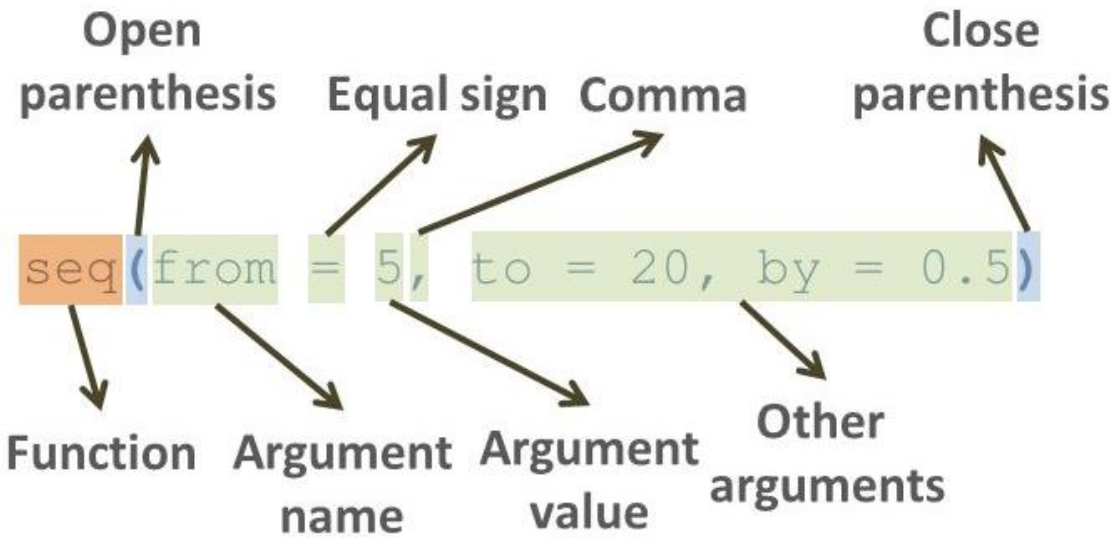
```
x <- data.frame(N = 11:14, C = c("a", "b", "c", "d"), L = (1:4) > 0)
dim(x)
nrow(x)
ncol(x)
length(x)
names(x)
as.data.frame(x)
is.data.frame(x)
```



Functions, Arguments, and Packages

- **Functions:**

Data analyses and modelling is done through functions.



hist(x)
plot(x)

args(plot)

Functions, Arguments, and Packages

- **Packages:**

install.packages*("package name")*

install.packages("ggplot2")

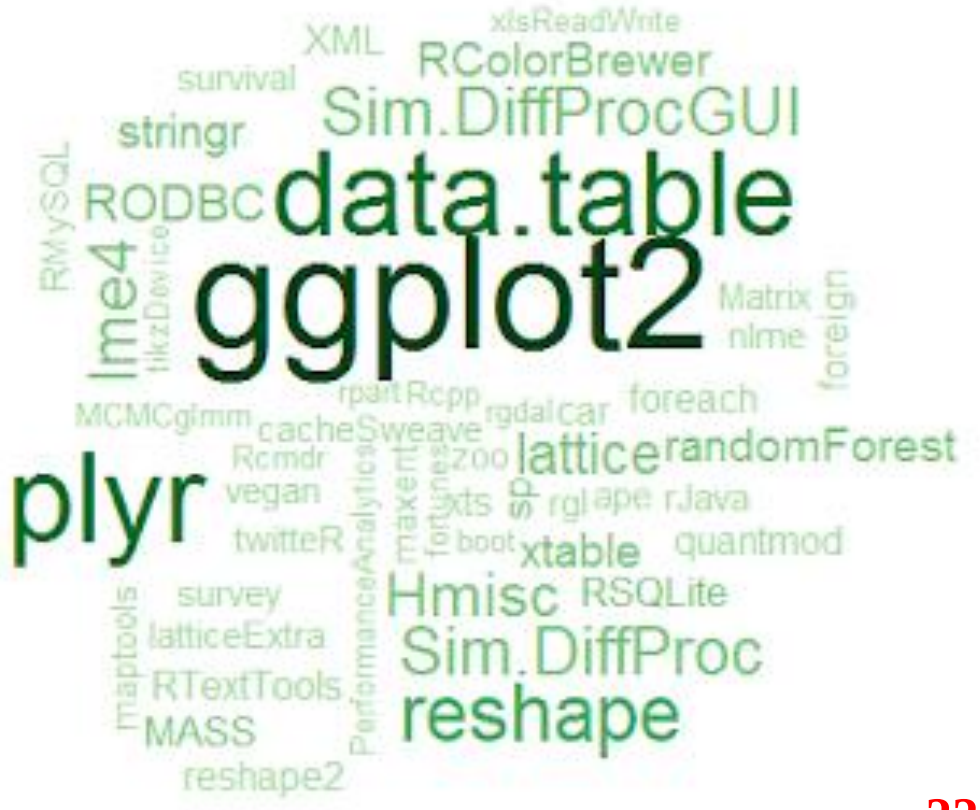
library(ggplot2)

detach("ggplot2 ")

installed.packages()

remove.packages()

update.packages()



Useful commands

Command	Purpose
help()	Obtain documentation for a given R command
example()	View some examples on the use of a command
c(), scan()	Enter data manually to a vector in R
seq()	Make arithmetic progression vector
rep()	Make vector of repeated values
data()	Load (often into a data.frame) built-in dataset
View()	View dataset in a spreadsheet-type format
str()	Display internal structure of an R object
read.csv(), read.table()	Load into a data.frame an existing data file
library(), require()	Make available an R add-on package
dim()	See dimensions (# of rows/cols) of data.frame
length()	Give length of a vector
ls()	Lists memory contents
rm()	Removes an item from memory
names()	Lists names of variables in a data.frame
hist()	Command for producing a histogram
histogram()	Lattice command for producing a histogram
stem()	Make a stem plot
table()	List all values of a variable with frequencies
xtabs()	Cross-tabulation tables using formulas
mosaicplot()	Make a mosaic plot
cut()	Groups values of a variable into larger bins
mean(), median()	Identify "center" of distribution
by()	apply function to a column split by factors
summary()	Display 5-number summary and mean
var(), sd()	Find variance, sd of values in vector
sum()	Add up all values in a vector
quantile()	Find the position of a quantile in a dataset
barplot()	Produces a bar graph
barchart()	Lattice command for producing bar graphs
boxplot()	Produces a boxplot
bwplot()	Lattice command for producing boxplots

Command	Purpose
plot()	Produces a scatterplot
xyplot()	Lattice command for producing a scatterplot
lm()	Determine the least-squares regression line
anova()	Analysis of variance (can use on results of lm())
predict()	Obtain predicted values from linear model
nls()	estimate parameters of a nonlinear model
residuals()	gives (observed - predicted) for a model fit to data
sample()	take a sample from a vector of data
replicate()	repeat some process a set number of times
cumsum()	produce running total of values for input vector
ecdf()	builds empirical cumulative distribution function
dbinom(), etc.	tools for binomial distributions
dpois(), etc.	tools for Poisson distributions
pnorm(), etc.	tools for normal distributions
qt(), etc.	tools for student t distributions
pchisq(), etc.	tools for chi-square distributions
binom.test()	hypothesis test and confidence interval for 1 proportion
prop.test()	inference for 1 proportion using normal approx.
chisq.test()	carries out a chi-square test
fisher.test()	Fisher test for contingency table
t.test()	student t test for inference on population mean
qqnorm(), qqline()	tools for checking normality
addmargins()	adds marginal sums to an existing table
prop.table()	compute proportions from a contingency table
par()	query and edit graphical settings
power.t.test()	power calculations for 1- and 2-sample t
anova()	compute analysis of variance table for fitted model

Base R
Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.

help.search('weighted mean')
Search the help files for a word or phrase.

help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.

class(iris)
Find the class an object belongs to.

Using Libraries

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x)
Return x sorted.

rev(x)
Return x reversed.

table(x)
See counts of values.

unique(x)
See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

x[x == 10]	Elements which are equal to 10.
x[x < 0]	All elements less than zero.
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.

Named Vectors

x['apple']	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence){
   Do something
}
```

Example

```
for (i in 1:4){
   j <- i + 10
   print(j)
}
```

While Loop

```
while (condition){
   Do something
}
```

Example

```
while (i < 5){
   print(i)
   i <- i + 1
}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

Functions

```
function_name <- function(var){
   Do something
   return(new_variable)
}
```

Example

```
square <- function(x){
   squared <- x*x
   return(squared)
}
```

Reading and Writing Data

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/ write.table.
load('file.RData')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions

a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

Learn more at web page or vignette • package version • Updated: 3/1

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.
`as.factor`	`'1', '0', '1', levels: '1', '0'`	Character strings with preset levels. Needed for some statistical models.

Maths Functions

`log(x)`	Natural log.	`sum(x)`	Sum.
`exp(x)`	Exponential.	`mean(x)`	Mean.
`max(x)`	Largest element.	`median(x)`	Median.
`min(x)`	Smallest element.	`quantile(x)`	Percentage quantiles.
`round(x, n)`	Round to n decimal places.	`rank(x)`	Rank of elements.
`signif(x, n)`	Round to n significant figures.	`var(x)`	The variance.
`cor(x, y)`	Correlation.	`sd(x)`	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

The Environment

`ls()`	List all variables in the environment.
`rm(x)`	Remove x from the environment.
`rm(list = ls())`	Remove all variables from the environment.

You can use the environment panel in RStudio to browse variables in your environment.

Matrixes

`m <- matrix(x, nrow = 3, ncol = 3)`
Create a matrix from x.


`m[2, ]` - Select a row


`m[ , 1]` - Select a column


`m[2, 3]` - Select an element

`t(m)`
Transpose
`m %*% n`
Matrix Multiplication
`solve(m, n)`
Find x in: m * x = n

Lists

`l <- list(x = 1:5, y = c('a', 'b'))`
A list is collection of elements which can be of different types.

`l[[2]]`	`l[1]`	`l$x`	`l['y']`
Second element of l.	New list with only the first element.	Element named x.	New list with only element named y.

Also see the **dplyr** library.

Data Frames

`df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))`
A special case of a list where all elements are the same length.

x	y
1	a
2	b
3	c

List subsetting

`df$x` 
`df[[2]]` 

Understanding a data frame

`View(df)`	See the full data frame.
`head(df)`	See the first 6 rows.

Matrix subsetting

`df[ , 2]` 
`df[2, ]` 
`df[2, 2]` 

`nrow(df)`
Number of rows.
`ncol(df)`
Number of columns.
`dim(df)`
Number of columns and rows.

`cbind` - Bind columns.


`rbind` - Bind rows.


Strings

Also see the **stringr** library.

`paste(x, y, sep = ' ')`	Join multiple vectors together.
`paste(x, collapse = ' ')`	Join elements of a vector together.
`grep(pattern, x)`	Find regular expression matches in x.
`gsub(pattern, replace, x)`	Replace matches in x with a string.
`toupper(x)`	Convert to uppercase.
`tolower(x)`	Convert to lowercase.
`nchar(x)`	Number of characters in a string.

Factors

`factor(x)`
Turn a vector into a factor. Can set the levels of the factor and the order.

`cut(x, breaks = 4)`
Turn a numeric vector into a factor but 'cutting' into sections.

Statistics

`lm(x ~ y, data=df)`
Linear model.

`glm(x ~ y, data=df)`
Generalised linear model.

`summary`
Get more detailed information out a model.

`t.test(x, y)`
Preform a t-test for difference between means.

`pairwise.t.test`
Preform a t-test for paired data.

`prop.test`
Test for a difference between proportions.

`aov`
Analysis of variance.

Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	`rnorm`	`dnorm`	`pnorm`	`qnorm`
Poison	`rpois`	`dpois`	`ppois`	`qpois`
Binomial	`rbinom`	`dbinom`	`pbinom`	`qbinom`
Uniform	`runif`	`dunif`	`punif`	`qunif`

Plotting

Also see the **ggplot2** library.


`plot(x)`
Values of x in order.


`plot(x, y)`
Values of x against y.


`hist(x)`
Histogram of x.

Dates

See the **lubridate** library.

Working with spatial data in R

- **Spatial Data in R:**
  - **R offers a wide variety of packages and tools that can handle**
  - **Relevant packages:**

– sp : handling spatial data
– raster : reading/manipulating/writing spatial raster data
– rgdal : reading/writing spatial data

Working with spatial data in R

- **Spatial data class summary:**

gridded

fullgridded

–**sp**:

- SpatialPointsDataFrame
- SpatialPixelDataFrame
- SpatialGridDataFrame
- SpatialPolygonDataFrame
- SpatialLinesDataFrame

- Format: **shapefile**

- Format: **GeoTiff**

–**raster**:

- RasterLayer(single layer)
- RasterStack/ RasterBrick(multiple layers)

Working with spatial data in R

- **Importing and exporting spatial data:**

  - rgdal: **readOGR**(vector), **readGDAL**(raster)
  - raster: **raster**

  - rgdal: **writeOGR**(vector), **writeGDAL**(raster)
  - raster: **writeRaster**

Working with spatial data in R

- **Projections:**

•Once you have loaded your spatial data in R, you might need to tell R its geographic projection.

•Check the current projection: **proj4string** function (sppackage).

•Setting a projection: **CRS**function (sppackage).

•Reprojectingto another coordinate system: **spTransform**function (sppackage) or **projectRaster**(raster package).

Please install R and RStudio

Please install SAGA and Google Earth