

Introduction to Computing for Economics and Management

Lecture 2: Introduction to R, variables, vectors



Previous Lecture: Assistants

Teaching Assistants



Nadin Kökciyan

nadin.kokciyan@boun.edu.tr



Ufuk Serkan Yıldırım

ufukserkan@gmail.com

Previous Lecture: Assistants

Student Assistants

- ▯ Cem Ozdemir, cem.ozdemir55@gmail.com
- ▯ Emre Boran, emreboran@live.com
- ▯ Abdullah Coskun, csknabdllh@gmail.com
- ▯ Yusuf Turan, yusufturan645@hotmail.com

Previous Lecture: What are the Course Objectives ?

- Learn basic concepts of programming
- Course topics include
 - Introduction into basic data structures (vector, matrix, lists, etc.)
 - Program control statements (conditional execution, loops, etc.)
 - Data visualization (scatter plots, histograms, etc.)
 - Input and Output (import internet data, export graphics, etc.)
- We use the free R programming language and environment for statistical computing and graphics which is available for Windows, MacOS and Linux

How can we achieve the course objectives ?

- No rush!
- Lecture: Stop me, ask questions, try to answer my questions.
- Problem Session: Repeat the lecture with examples.
- Lab Session: Hands-on experience!

How can we achieve the course objectives ?

Attend the lecture every Wednesday 13:00 – 15:00,

- Room NH405
- Attend the 1 hour Problem Session directly after the lecture in NH405
 - Problem session starts this week, Wednesday 28th of September
- **Use R and practice at home every week**
- Attend the Lab Session
 - CMPE140.01: Friday 13:00 - 15:00, CMPE Building, Room B4
 - CMPE140.02: Friday 15:00 - 17:00, CMPE Building, Room B4

Attendance check

- We provide for each student a one-time code
 - at each lecture
 - at each lab session
 - at each problem session
- Each student has to submit the Student ID together with the one-time code
- Only submissions on the day of lecture/lab are valid

Submission page: <http://goo.gl/kGFkPj>

Attendance check

- We provide for each student a one-time code
 - at each lecture
 - at each lab session
 - at each problem set
- Each student has to submit the ID together with the one-time code
- Only submissions on time are valid

Submission page: <https://www.bogazici.edu.tr/kGFkFj>

Scores

Lecture and problem session attendance	10 %
Lab session (quiz)	30 %
Midterm	25 %
Final	45 %

- Attend to your own section in Lab sessions!
 - Automatic grading
- Midterm (21st Oct) and Final:
 - Multiple-choice
- Quizzes:
 - Multiple choice (and programming)

Course Material – Web page

- The material is adapted from Bert Arnrich's slides.
- You find all course material on my lecture page:
- www.cmpe.boun.edu.tr/~emre/courses/CMPE140/

Emre Ugur
Ph.D.

[Home](#)
[Publications](#)
[Projects](#)
[Current Research](#)
[Courses](#)
[More About Me](#)
[Contact](#)

CMPE140 Introduction to Computing for Economics and Management
2016/2017 Fall

Aim:

CMPE 140 introduces basic concepts of computing with the R programming language. Course topics include an introduction into basic data structures (vector, matrix, lists, data frames, etc.), program control statements (conditional execution, for and while loops, etc.), data visualization and input/output.

Instructor: Emre Ugur ([contact](#))

Office hours: Tuesday, 16:00-17:00

[Mailing-list: Please send email if you are not registered!](#)

Timetable:

Lecture	Wednesday 13:00-15:00	New Hall Building, Room NH405
Problem Session	Wednesday 15:00-16:00	New Hall Building, Room NH405
Lab for CMPE104.1	Friday 13:00-15:00	CMPE Building, Room B4
Lab for CMPE104.2	Friday 15:00-17:00	CMPE Building, Room B4

Schedule:

Chapter	Topics	Lecture Slides
Introduction	- Course objectives and organization, - R Intro, - R Download and Installation, - Simple calculations, - R help and documentation, - R Programming Environment,	Week 1
Vectors	- Variable name conventions - Creating data vectors - Data vector indexing - Data vector filtering - Data vector sorting	



[All](#)
[Maps](#)
[News](#)
[Images](#)
[Videos](#)
[More](#)
[Search tools](#)

About 887 results (0.30 seconds)

CmpE 140 Introduction to Computing for Economics and Ma
www.cmpe.boun.edu.tr/courses/cmpe140 ▾
CMPE 140 introduces basic concepts of exploratory data analysis and programmin
Department of Computer Engineering, Boğaziçi University,

[PDF] CMPE 140 Introduction to Computing for Economics and
https://www.cmpe.boun.edu.tr/.../CmpE%20140/cmpe_140_spring_2016_s
Jan 23, 2016 - CMPE 140 introduces basic concepts of computing with the R progr
Course topics include an introduction into basic data ...

CmpE 140 Introduction to Computing for Economics and Ma
www.cmpe.boun.edu.tr/courses/cmpe140/2016/spring ▾ [Translate this pa](#)
CMPE 140 introduces basic concepts of computing with the R programming langua
Bilgisayar Mühendisliği Bölümü, Boğaziçi Üniversitesi,

CmpE 140 Introduction to Computing for Economics and Ma
www.cmpe.boun.edu.tr/courses/cmpe140/2015/fall ▾ [Translate this page](#)
CmpE 140 Introduction to Computing for Economics and Management 2015 Fall ...
Mühendisliği Bölümü, Boğaziçi Üniversitesi, 34342 Bebek ...

Boğaziçi University - Department of Computer Engineering
www.boun.edu.tr/Default.aspx?SectionID=561 ▾ Boğaziçi University ▾
CMPE 352, Fundamentals of Software Engineering, 2, 5. CMPE 362 ... CMPE 140
Computing for Economics and Management (2+1+2) 3 ECTS 5

Course Material – Web page

- The material is adapted from Bert Arnrich's slides.
- You find all course material on my lecture page:
- www.cmpe.boun.edu.tr/~emre/courses/CMPE140/

Emre Ugur

P.H.D.

[Home](#)
[Publications](#)
[Projects](#)
[Current Research](#)
[Courses](#)
[More About Me](#)
[Contact](#)

CMPE140 Introduction to Computing for Economics and Management

2016/2017 Fall

Aim:

CMPE 140 introduces basic concepts of computing with the R programming language. Course topics include an introduction into basic data structures (vector, matrix, lists, data frames, etc.), program control statements (conditional execution, for and while loops, etc.), data visualization and input/output.

Instructor: Emre Ugur ([contact](#))

Office hours: Tuesday, 16:00-17:00

Mailing-list: Please send email if you are not registered!

Timetable:

Lecture	Wednesday 13:00-15:00	New Hall Building, Room NH405
Problem Session	Wednesday 15:00-16:00	New Hall Building, Room NH405
Lab for CMPE104.1	Friday 13:00-15:00	CMPE Building, Room B4
Lab for CMPE104.2	Friday 15:00-17:00	CMPE Building, Room B4

Schedule:

Chapter	Topics	Lecture Slides
Introduction	- Course objectives and organization, - R Intro, - R Download and Installation, - Simple calculations, - R help and documentation, - R Programming Environment,	Week 1
Vectors	- Variable name conventions - Creating data vectors - Data vector indexing - Data vector filtering - Data vector sorting - Data vector operations - Creating regular sequences - Creating repeated values	
Matrices	- Matrix creation - Matrix modification - Matrix operations - Matrix indexing - Matrix filtering - Matrix function apply() - Programming own functions - Higher-dimensional arrays	
Lists	- Shortcomings of vectors and matrices - Creating lists - List indexing - Adding/deleting list elements - Concatenate lists - Vectors as list components - Example: word list - Accessing list components/values - Example: sort word list alphabetically - Applying functions to lists - Example: sort word list by word frequency	
Data frames	- Shortcomings of vectors and matrices - Creating data frames - Accessing data frames - Data frame indexing - Data frame modifications - Data import from file - Data frame summary - Scatter plot	Midterm!

Google

cmpe 140 bogazici

[All](#)
[Maps](#)
[News](#)
[Images](#)
[Videos](#)
[More](#)
[Search tools](#)

About 887 results (0.30 seconds)

CmpE 140 Introduction to Computing for Economics and M
www.cmpe.boun.edu.tr/courses/cmpe140 ▾
 CMPE 140 introduces basic concepts of exploratory data analysis and programmi
 Department of Computer Engineering, Boğaziçi University,

[PDF] CMPE 140 Introduction to Computing for Economics an
https://www.cmpe.boun.edu.tr/.../CmpE%20140//cmpe_140_spring_2016_
 Jan 23, 2016 - CMPE 140 introduces basic concepts of computing with the R prog
 Course topics include an introduction into basic data ...

CmpE 140 Introduction to Computing for Economics and M
www.cmpe.boun.edu.tr/courses/cmpe140/2016/spring ▾ Translate this p
 CMPE 140 introduces basic concepts of computing with the R programming langu
 Bilgisayar Mühendisliği Bölümü, Boğaziçi Üniversitesi,

CmpE 140 Introduction to Computing for Economics and M
www.cmpe.boun.edu.tr/courses/cmpe140/2015/fall ▾ Translate this page
 CmpE 140 Introduction to Computing for Economics and Management 2015 Fall .
 Mühendisliği Bölümü, Boğaziçi Üniversitesi, 34342 Bebek ...

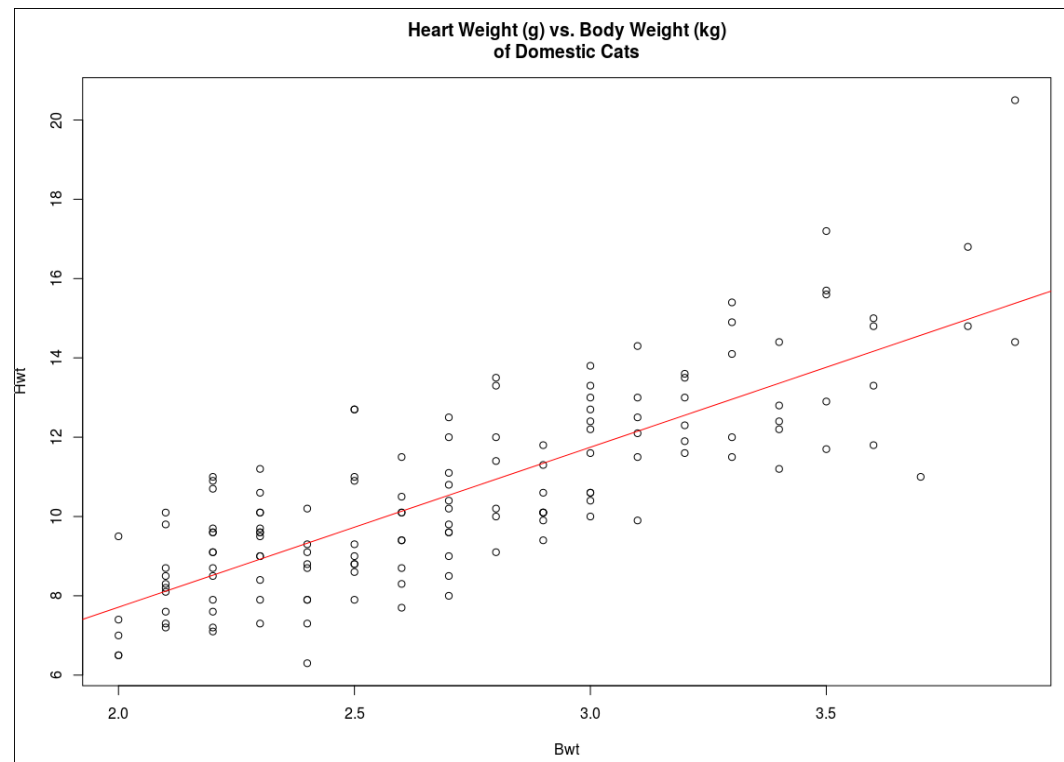
Boğaziçi University - Department of Computer Engineering
www.boun.edu.tr/Default.aspx?SectionID=561 ▾ Boğaziçi University ▾
 CMPE 352, Fundamentals of Software Engineering, 2, 5. CMPE 362 ... CMPE 14
 Computing for Economics and Management (2+1+2) 3 ECTS 5

Course Material – Mailing list

- All off-line communication through mailing list
- `cmpe140@listeci.cmpe.boun.edu.tr`
 - All registered students are registered to the mailing-list with their emails in the boun registration system.
- If you have not received this email, let me know!

Some cool stuff for modelling and prediction

```
library("MASS")  
data(cats)  
with(cats, plot(Bwt, Hwt))  
title(main="Heart Weight (g) vs. Body Weight (kg)\nof Domestic Cats")  
lm.out = lm(Hwt ~ Bwt, data=cats)  
abline(lm.out, col="red")
```



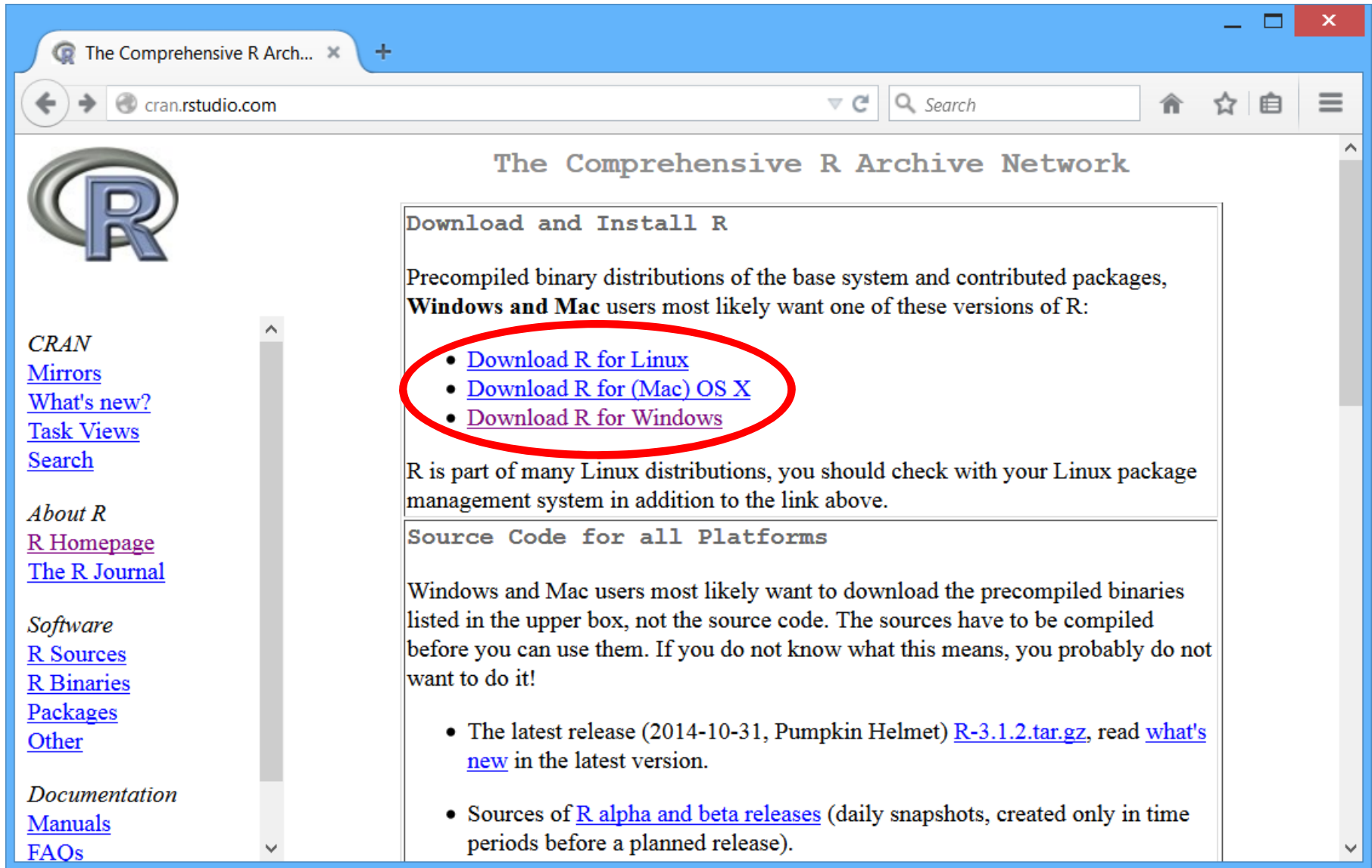
Some cool stuff for visualization

- Stock Analysis using R
- <http://www.r-chart.com/2010/06/stock-analysis-using-r.html>
- Want to do some quick, in depth technical analysis of Apple stock price using R? There is a package for that!

```
library('quantmod')  
getSymbols("AAPL")  
chartSeries(AAPL, subset='last 3  
months')  
addBBands()
```



Previous lecture: R download and installation



The screenshot shows the CRAN (Comprehensive R Archive Network) website. The browser address bar shows `cran.rstudio.com`. The page title is "The Comprehensive R Archive Network". The main content area is titled "Download and Install R" and contains the following text:

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

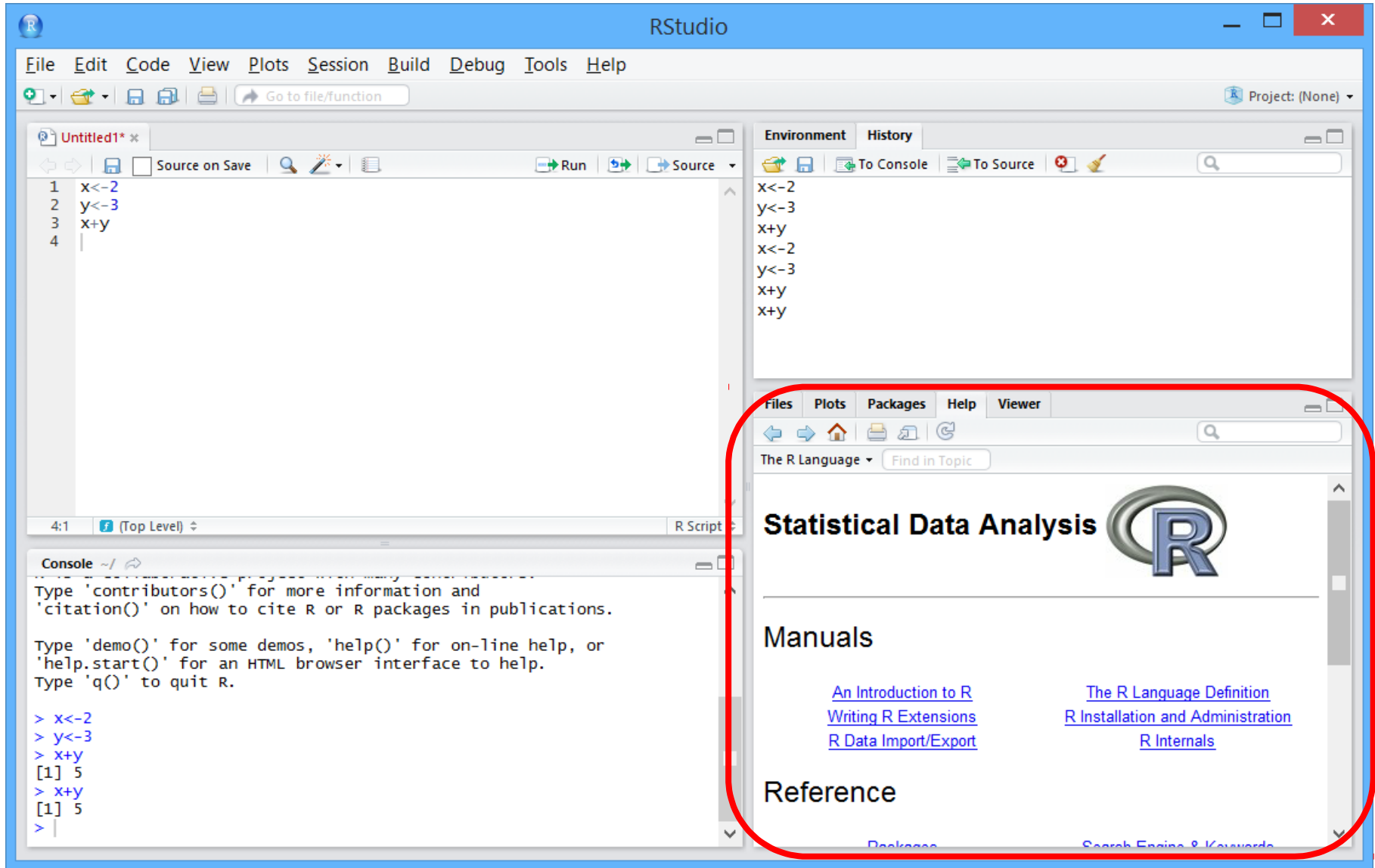
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2014-10-31, Pumpkin Helmet) [R-3.1.2.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).

The left sidebar contains the following links:

- CRAN
- [Mirrors](#)
- [What's new?](#)
- [Task Views](#)
- [Search](#)
- About R
- [R Homepage](#)
- [The R Journal](#)
- Software
- [R Sources](#)
- [R Binaries](#)
- [Packages](#)
- [Other](#)
- Documentation
- [Manuals](#)
- [FAQs](#)

Previous lecture: RStudio



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main workspace is divided into four panes:

- Source Editor:** Contains a script named 'Untitled1*' with the following code:

```
1 x<-2
2 y<-3
3 x+y
4 |
```
- Environment:** Shows the current environment with variables x, y, x+y, and their values.
- History:** Shows the execution history of the code.
- Console:** Displays the output of the code execution:

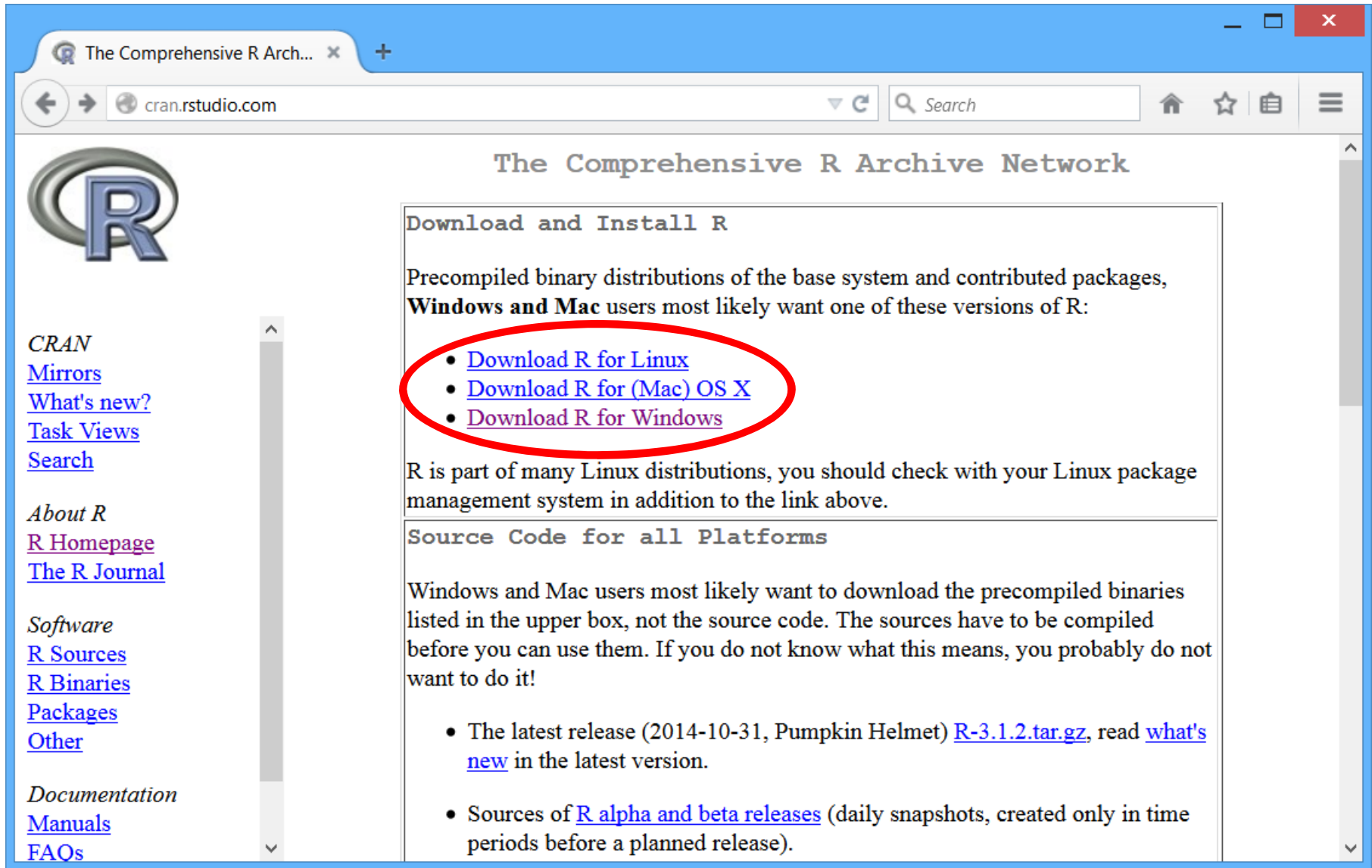
```
> x<-2
> y<-3
> x+y
[1] 5
> x+y
[1] 5
> |
```

The bottom pane is the **Help Viewer**, which is highlighted with a red rounded rectangle. It shows the 'The R Language' help page, titled 'Statistical Data Analysis' with the R logo. The page includes links to various manuals and reference materials:

- [An Introduction to R](#)
- [Writing R Extensions](#)
- [R Data Import/Export](#)
- [The R Language Definition](#)
- [R Installation and Administration](#)
- [R Internals](#)

The bottom of the help viewer also shows a 'Reference' section and a search bar.

Previous lecture: R download and installation



The screenshot shows the CRAN (Comprehensive R Archive Network) website. The browser address bar shows `cran.rstudio.com`. The page title is "The Comprehensive R Archive Network". The main content area is titled "Download and Install R" and contains the following text:

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2014-10-31, Pumpkin Helmet) [R-3.1.2.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).

The left sidebar contains the following links:

- CRAN
- [Mirrors](#)
- [What's new?](#)
- [Task Views](#)
- [Search](#)
- About R
- [R Homepage](#)
- [The R Journal](#)
- Software
- [R Sources](#)
- [R Binaries](#)
- [Packages](#)
- [Other](#)
- Documentation
- [Manuals](#)
- [FAQs](#)

Previous lecture: Interactive R session

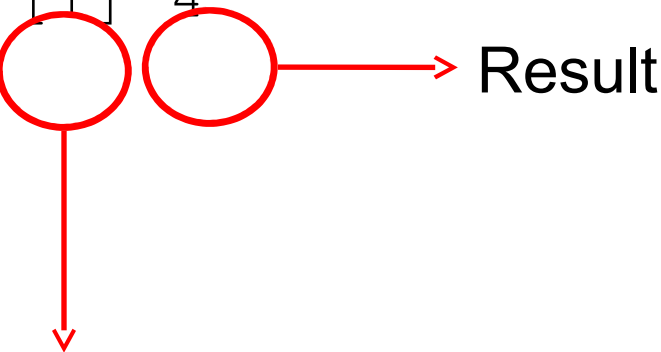
Enter an arithmetic expression and receive a result

```
> 2 + 2
```

```
[1] 4
```

```
> 2 + 2
```

```
[1] 4
```



Index of the first number on a result line: we will see later that the index is helpful when we receive more than a single result

Previous lecture: Interactive R session

Do some standard calculations

```
> sqrt(4)
```

```
[1] 2
```

```
> exp(-2)
```

```
[1] 0.1353353
```

```
> cos(pi)
```

```
[1] -1
```

```
> 2 < 4
```

```
[1] TRUE
```

Data types:

- numerical
- character
- logical

Previous lecture: Interactive R session

Assign a value to the objects x and y by using the assignment operator `<-`

```
> x <- 2
```

```
> y <- 3
```

Not assignment if space introduced in between `<-`

```
> y < -3
```

Check the values of both objects

```
> x
```

```
[1] 2
```

```
> y
```

```
[1] 3
```

Previous lecture: Interactive R session

```
> x+y  
[1] 5
```

```
> x*y  
[1] 6
```

```
> x*y+3  
[1] 9
```

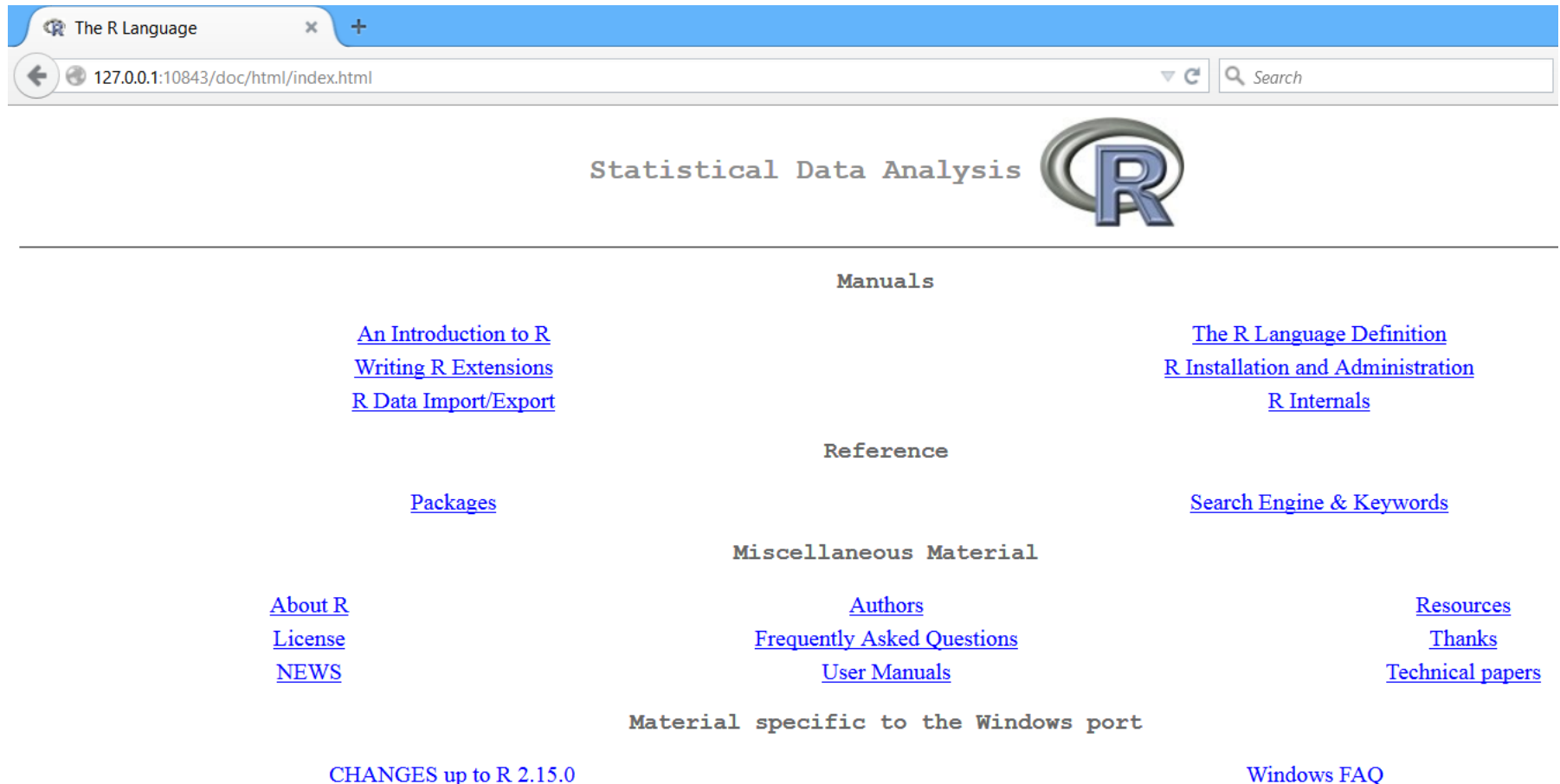
```
> sqrt(x*y)+3  
[1] 5.44949
```

Previous lecture: getting help and documentation



Open a Web browser interface to R help

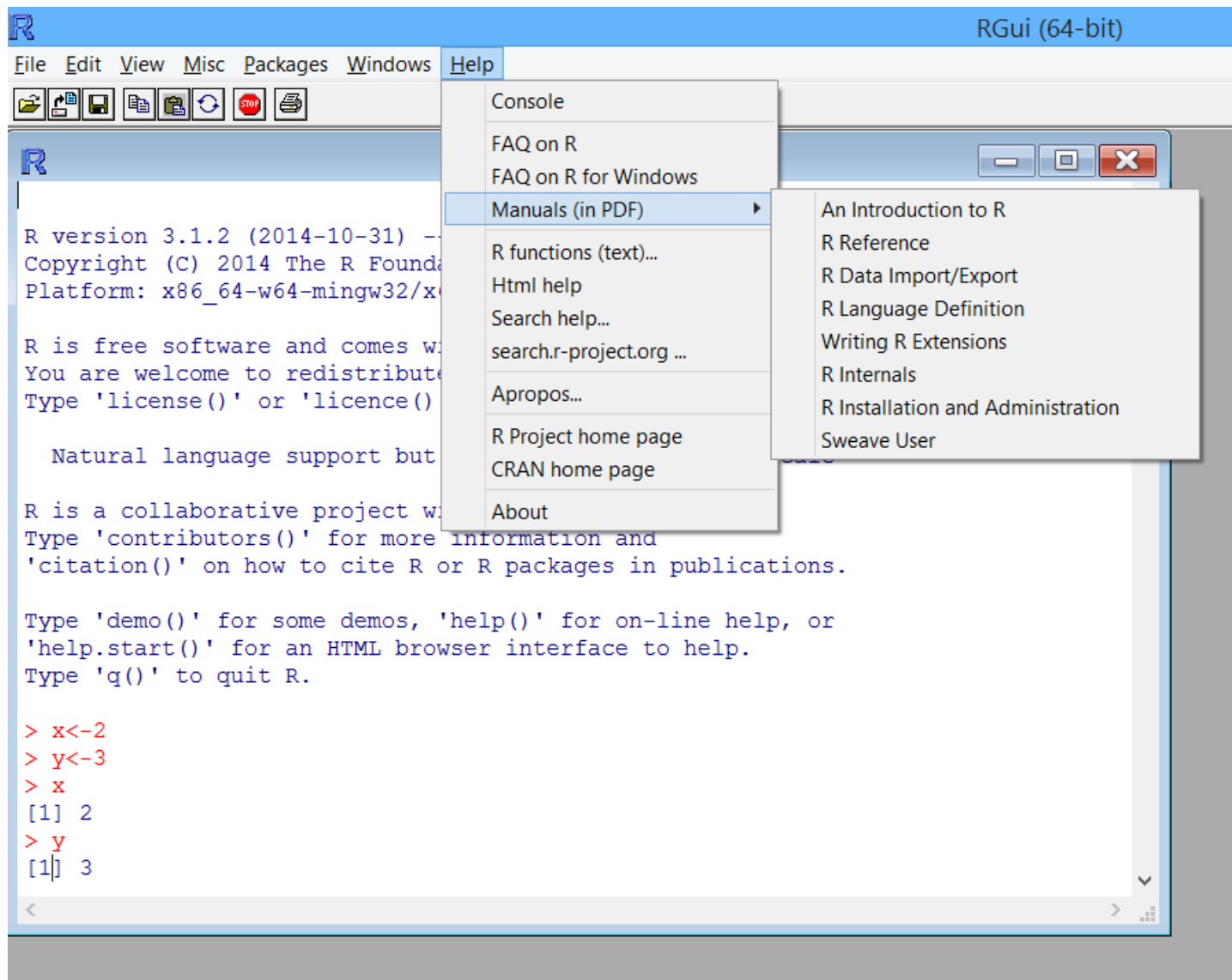
```
> help.start()
```



The screenshot shows a web browser window with the title "The R Language". The address bar displays "127.0.0.1:10843/doc/html/index.html". The main content area features the text "Statistical Data Analysis" next to the R logo. Below this, there are several sections of links:

- Manuals**
 - [An Introduction to R](#)
 - [Writing R Extensions](#)
 - [R Data Import/Export](#)
 - [The R Language Definition](#)
 - [R Installation and Administration](#)
 - [R Internals](#)
- Reference**
 - [Search Engine & Keywords](#)
- Miscellaneous Material**
 - [About R](#)
 - [License](#)
 - [NEWS](#)
 - [Authors](#)
 - [Frequently Asked Questions](#)
 - [User Manuals](#)
 - [Resources](#)
 - [Thanks](#)
 - [Technical papers](#)
- Material specific to the Windows port**
 - [CHANGES up to R 2.15.0](#)
 - [Windows FAQ](#)

Previous lecture: R manuals



Program today

- Name conventions
- Creating data vectors
- Data vector indexing
- Data vector filtering
- Data vector sorting
- Data vector operations
- Creating regular sequences
- Creating repeated values

Name conventions

Object names can contain letters, digits, underscore (_), and the dot symbol (.)

```
> person1.height <- 1.70
> person1.weight <- 65
> bmi.person1 <- person1.weight /
person1.height^2
> bmi.person1
[1] 22.49135
```

```
> person2.height <- 1.75
> person2.weight <- 66
> bmi.person2 <- person2.weight /
person2.height^2
> bmi.person2
■ [1] 21.55102
```

Name conventions

Name must **not start with a number**

```
> 2a <- 3
```

■ `Error: unexpected symbol in "2a"`

Name **must not start with a dot** followed by a digit

```
> .2x <- 2
```

`Error: unexpected symbol in ".2x"`

■ `> .xy <- 3`

■ Since names that start with a dot are special, we should not introduce them in our scripts to avoid confusion

Descriptive names

- Better use descriptive names like `person.height` instead of just `h`
- Descriptive names make scripts easier to read and understand
- With descriptive names we can usually avoid conflicts with names that are already used by the system as we will see soon

Case-sensitivity

Names are case-sensitive, e.g. `x` and `X` do not refer to the same object

```
> x <- 2
```

```
> X <- 3
```

```
> x + x
```

```
> x + X
```

■

Case-sensitivity

Names are case-sensitive, e.g. `x` and `X` do not refer to the same object

```
> x <- 2
```

```
> X <- 3
```

```
> x + x
```

```
[1] 4
```

```
> x + X
```

```
■ [1] 5
```

Names used by R

- Some names are already used by the R system, e.g. `c`, `q`, `t`, `C`, `D`, `F`, `I`, `T`, `df`, `pt`
- You should not use those names for your objects since it will create confusion when other people read your script
- In particular you should avoid the names `F` and `T` since they are the standard abbreviations for `FALSE` and `TRUE` and no longer work as such if you redefine them

Name check

You can check **whether** an object name **is already used** by the system when you enter the name in the console

```
> c
```

```
function (... , recursive = FALSE)  
.Primitive("c")
```

```
> my.c
```

```
■ Error: object 'my.c' not found
```

- Whenever you get the message that object is not found, you can understand that the name is not in use

Data vectors

- The fundamental data type in R is the vector
- It is hard to imagine a R session without using vectors
- The objects x and y that we have seen so far can be considered as one-element vectors

Why we need data vectors

Operating with one-element objects is cumbersome

```
> person1.height <- 1.70  
> person1.weight <- 65  
> bmi.person1 <- person1.weight /  
person1.height^2  
> bmi.person1  
[1] 22.49135
```

```
> person2.height <- 1.75  
> person2.weight <- 66  
> bmi.person2 <- person2.weight /  
person2.height^2  
> bmi.person2  
■ [1] 21.55102
```

Data vector creation

- Instead of using a single object for a single data entry, we rather create an array of numbers which we call **data vector**

Data vectors are created with the construct **c**

```
> persons.height <- c(1.70, 1.75, 1.62)
```

```
> persons.weight <- c(65, 66, 61)
```

```
> persons.height  
[1] 1.70 1.75 1.62
```

```
> persons.weight  
▪ [1] 65 66 61
```

Data vector creation

- With the construct `c` we can also concatenate vectors which is useful when we need to extend an existing vector

For example, we have data from the first two persons and we create a vector from it

```
> person.height <- c(1.70, 1.75)
```

■

- Now, we **extend** the vector with data from the third person

```
> person.height <- c(person.height, 1.62)
```

■

```
> person.height
```

```
[1] 1.70 1.75 1.62
```

Data vector modes

- Vector elements must all have the same **mode/type**

Integer mode

```
> person.weight <- c(65, 66, 61)
```

▪

- **Numeric** (floating-point number)

```
> person.height <- c(1.70, 1.75, 1.62)
```

▪

- **Character** (string)

```
> person.name <- c("Can", "Cem", "Hande")
```

Data vector modes

Logical (Boolean)

- `> person.female <- c(FALSE, FALSE, TRUE)`

Complex

- `> complex.numbers <- c(1+2i, -1+0i)`

Other

Missing values

- In many datasets, we often have **missing data**, i.e. observations for which values are missing
- In R we denote a missing value with **NA**

Any vector might contain missing values

```
> person.weight <- c(65, NA, 61)
```

```
▪ > person.name <- c("Can", "Cem", NA)
```

Data vector element names

We can assign names to the elements of a data vector to make the vector more readable

```
> person.height <- c(Can=1.70, Cem=1.75,  
Hande=1.62)
```

```
> person.height  
Can    Cem Hande  
1.70   1.75   1.62
```

```
> person.weight <- c(Can=65, Cem=66, Hande=61)
```

```
> person.weight  
Can    Cem Hande  
65     66    61
```

Data vector indexing

We can **access a single element** of a vector by providing the index of the element in square brackets **[..]**

```
> person.height  
Can    Cem  Hande  
1.70   1.75   1.62
```

```
> person.height[1]  
Can  
1.7
```

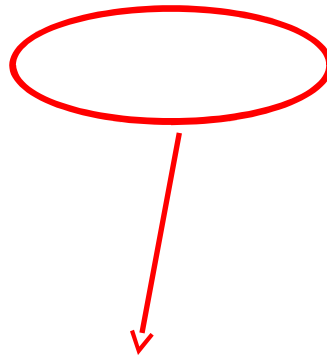
```
> person.height[2]  
Cem  
■ 1.75
```

Data vector indexing

- Beside accessing a single element, we can select sub-vectors
- One way to select a subvector is to specify a range and use brackets

Example: select 1st and 2nd element from `person.height` >
`person.height[1:2]`

Can	Cem
1.70	1.75



We provide a vector with range in the square brackets for selecting the subvector

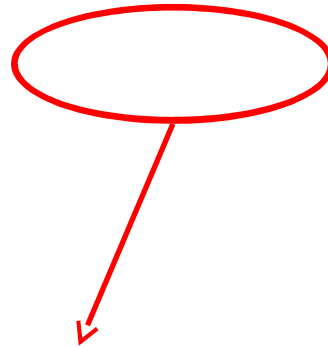
Data vector indexing

- Beside accessing a single element, we can select sub-vectors
- One way to select a subvector is to specify **TRUE** for the elements we would like to select and **FALSE** for the others

Example: select 1st and 3rd element from `person.height` >
`person.height[c(T,F,T)]`

Can Hande

1.70 1.62



We provide a Boolean index vector in the square brackets for selecting the subvector

Data vector indexing

- As alternative to a Boolean index vector, we can specify the element indices directly
- Specifying element **indices** is more convenient in particular for large vectors

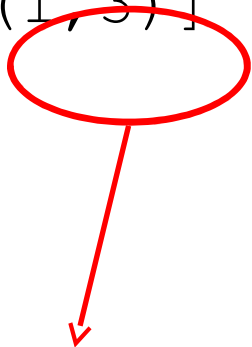
Example: select 1st and 3rd element from `person.height`

```
> person.height[c(1, 3)]
```

Can Hande

1.70 1.62

■



We provide an index vector in the square brackets for selecting the subvector

Data vector indexing

We can select duplicates as well

```
> person.height[c(1,1,3,2,3)]
```

Can	Can	Hande	Cem	Hande
1.70	1.70	1.62	1.75	1.62

We exclude elements with neg. indices, e.g. exclude 1st entry

```
> person.height[-1]
```

Cem	Hande
1.75	1.62

Example: exclude first and third element

```
> person.height[c(-1, -3)]
```

■

Data vector indexing

We can select duplicates as well

```
> person.height[c(1,1,3,2,3)]
```

Can	Can	Hande	Cem	Hande
1.70	1.70	1.62	1.75	1.62

We exclude elements with neg. indices, e.g. exclude 1st entry

```
> person.height[-1]
```

Cem	Hande
1.75	1.62

Example: exclude first and third element

```
> person.height[c(-1, -3)]
```

Cem
1.75

Data vector indexing

- We can change the values of the selected elements

Example: change the value of the first element

```
> person.height  
  Can    Cem Hande  
1.70  1.75  1.62
```

- >

Data vector indexing

- We can change the values of the selected elements

Example: change the value of the first element

```
> person.height  
  Can    Cem Hande  
1.70  1.75  1.62
```

```
> person.height[1] <- 1.72
```

```
> person.height  
  Can    Cem Hande  
▪ 1.72  1.75  1.62
```

Data vector indexing

We can use the index notation to insert values in an existing vector

```
> person.height  
Can    Cem Hande  
1.70   1.75   1.62
```

```
> person.height <- c(person.height[1:2],  
Lale=1.76, person.height[3])
```

```
> person.height
```

■

Data vector indexing

We can use the index notation to insert values in an existing vector

```
> person.height  
Can    Cem Hande  
1.70   1.75   1.62
```

```
> person.height <- c(person.height[1:2],  
Lale=1.76, person.height[3])
```

```
> person.height  
Can    Cem Lale Hande  
1.70   1.75 1.76 1.62
```

Data vector indexing

We can use the index notation to delete values from an existing vector

```
> person.height
  Can   Cem  Lale Hande
1.70  1.75  1.76  1.62

> person.height <- person.height[-3]

> person.height
■
```

Data vector indexing

We can use the index notation to delete values from an existing vector

```
> person.height
```

Can	Cem	Lale	Hande
1.70	1.75	1.76	1.62

```
> person.height <- person.height[-3]
```

```
> person.height
```

Can	Cem	Hande
1.70	1.75	1.62

Data vector filtering

- The idea behind filtering is to first apply a Boolean evaluation function to each element of the vector
- For each single element, the Boolean evaluation function **returns** `TRUE` in case of a positive evaluation and `FALSE` in case of a negative evaluation

Example: persons taller than 1.65

```
> person.height > 1.65
```

```
Can    Cem Hande
```

- `TRUE TRUE FALSE`

Data vector filtering

In a second step, we use the results of the evaluation function for the filtering

```
> person.taller <- person.height > 1.65
```

```
> person.taller  
Can    Cem Hande  
TRUE   TRUE FALSE
```

```
> person.height[person.taller]
```

■

Data vector filtering

In a second step, we use the results of the evaluation function for the filtering

```
> person.taller <- person.height > 1.65
```

```
> person.taller  
Can    Cem Hande  
TRUE   TRUE FALSE
```

```
> person.height[person.taller]  
Can    Cem  
■ 1.72  1.75
```

Data vector filtering

We can perform evaluation and filtering in one line

```
> person.height[person.height > 1.65]
```

Can Cem

■ 1.72 1.75

We provide the evaluation function directly in the square brackets for selecting the subvector which elements fulfill the evaluation function

```
> X <- person.height > 1.65
```

```
> X
```

Can Cem Hande

TRUE TRUE FALSE

Data vector filtering

- We can use filtering to selectively change those elements of a vector which fulfill the evaluation function

Example: Decrease weight by 1kg of those persons whose weight is larger than 65kg

```
> person.weight[person.weight > 65] <-  
person.weight[person.weight > 65] - 1
```

```
> person.weight  
Can    Cem Hande  
▪    65    65    61
```

Data vector filtering

Function `which()` returns the indices of those elements that fulfill the evaluation function

```
> person.height  
  Can   Cem Hande  
1.72  1.75  1.62
```

```
> which(person.height > 1.65)
```

```
Can  Cem
```

```
■    1    2
```

Data vector sorting

We use the function `sort` for sorting a vector

```
> sort(person.height)
```

Hande	Can	Cem
1.62	1.70	1.75

```
> sort(person.weight)
```

Hande	Can	Cem
61	65	66

- `help(sort)` shows us that the sort function has a second **parameter** `decreasing`
- Since `decreasing = FALSE` is the default value, the default sorting is in ascending order

Data vector sorting

We can obtain a sorting in descending order by specifying
`decreasing = TRUE`

```
> sort(person.height, decreasing = TRUE)
  Cem    Can Hande
1.75  1.70  1.62
```

```
> sort(person.weight, decreasing = TRUE)
  Cem    Can Hande
■    66    65    61
```

Data vector sorting

- Often we need to sort a vector according to the values of some other vector

We first compute an ordering

```
> person.height  
Can    Cem Hande  
1.70   1.75   1.62
```

- `> order(person.height)`
- As a result of the ordering we achieve the index positions in the right order: person 3 is first, person 1 is second, person 2 is third

Data vector sorting

- Often we need to sort a vector according to the values of some other vector

We first compute an ordering

```
> person.height  
Can    Cem Hande  
1.70   1.75   1.62
```

```
> order(person.height)
```

```
▪ [1] 3 1 2
```

- As a result of the ordering we achieve the index positions in the right order: person 3 is first, person 1 is second, person 2 is third

Data vector sorting

We can use the resulting sorted index for sorting some other vector

```
> person.weight[order(person.height)]
```

Hande	Can	Cem
61	65	66

- `person.weight` was sorted according to the order of `person.height`

Vector recycling

- When applying an operation to two vectors which requires them to be the same length, **the shorter one will repeated** until it is long enough to match the longer one
- This automatic lengthening of vectors is called recycling

Example: vector addition

```
> c(1, 2, 3) + c(1, 2, 3, 4)
```

```
▪ [1] 2 4 6 5
```

The shorter vector was automatically “recycled” to be as

```
> c(1, 2, 3, 1) + c(1, 2, 3, 4)
```

```
▪ [1] 2 4 6 5
```

Vector recycling

We usually get a warning message in case an automatic recycling appears

```
> c(1, 2, 3) + c(1, 2, 3, 4)
[1] 2 4 6 5
```

Warning message:

In `c(1, 2, 3) + c(1, 2, 3, 4)` :

- longer object length is not a multiple of shorter object length

Conditional element selection with the `ifelse()` function

- We provide the `ifelse(test, yes, no)` function with a Boolean vector `test` and two vectors `yes` and `no`
- `ifelse` returns a vector which is created from selected elements from the vectors `yes` and `no`: `yes[i]` is selected if `test[i]` is true and `no[i]` is selected if `test[i]` is false

Example (which uses recycling):

```
> ifelse(person.height > 1.7, "tall", "small")
      Can      Cem      Hande
▪ "small"    "tall"    "small"
```

Conditional element selection with the `ifelse()` function

- If we need more than the two alternatives `yes` and `no` we can replace `yes` or `no` with **another** `ifelse()` function

Example: create three height groups `small`, `tall` and `huge`

- `> ifelse(?)`

Can	Cem	Hande
"tall"	"huge"	"small"

Conditional element selection with the `ifelse()` function

- If we need more than the two alternatives `yes` and `no` we can replace `yes` or `no` with **another** `ifelse()` function

Example: create three height groups small, tall and huge

```
> ifelse(person.height > 1.7, "huge",  
ifelse(person.height > 1.65, "tall", "small"))
```

Can	Cem	Hande
-----	-----	-------

- | | | |
|--------|--------|---------|
| "tall" | "huge" | "small" |
|--------|--------|---------|

Data vector operations

Single element operation possible but rarely used in practice

```
> person.weight[1] / person.height[1]^2
```

Can

21.97134

```
> person.weight[2] / person.height[2]^2
```

Cem

21.55102

```
> person.weight[3] / person.height[3]^2
```

Hande

■ 23.24341

Data vector operations

- We can perform calculations with vectors just like ordinary numbers

Element wise operations

```
> person.height^2
```

```
Can    Cem  Hande
```

```
■ 2.9584 3.0625 2.6244
```

Vector addition, e.g. persons have gained/lost weight

```
> person.weight + c(1.5, 1.75, -0.5)
```

```
Can    Cem  Hande
```

```
■ 66.50 67.75 60.50
```

Data vector operations

Operations on multiple vectors

```
> person.weight / person.height^2
```

Can	Cem	Hande
-----	-----	-------

```
■ 21.97134 21.55102 23.24341
```

The result of a vector calculation can be assigned to a new data vector for further processing

```
> bmi <- person.weight / person.height^2
```

```
> bmi
```

Can	Cem	Hande
-----	-----	-------

21.97134	21.55102	23.24341
----------	----------	----------

```
■
```

Data vector operations

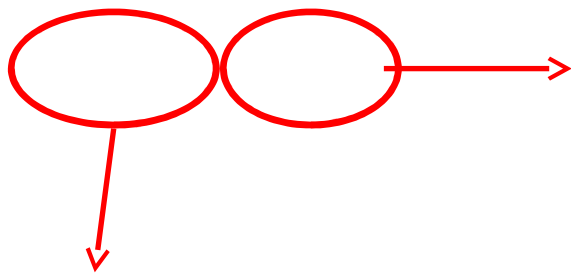
- R has many **built-in functions** that operate on vectors

For example, we would like to **round** the resulting BMI values

```
> round(bmi)
```

Can	Cem	Hande
22	22	23

- ```
> round(bmi)
```



Arguments of the function are provided in parentheses

R function round

# Rounding of numbers

- The help page `help(round)` shows us more options and ways how to round numeric values

`round` has an additional option `digits` to indicate the number of decimal places to be used when rounding

```
> round(bmi, digits=1)
```

| Can  | Cem  | Hande |
|------|------|-------|
| 22.0 | 21.6 | 23.2  |

```
> round(bmi, digits=2)
```

| Can   | Cem   | Hande |
|-------|-------|-------|
| 21.97 | 21.55 | 23.24 |



# Rounding of numbers

`ceiling` returns a numeric vector containing the smallest integers not less than the corresponding elements of the input vector

```
> bmi
```

| Can      | Cem      | Hande    |
|----------|----------|----------|
| 21.97134 | 21.55102 | 23.24341 |

```
> ceiling(bmi)
```

| Can | Cem | Hande |
|-----|-----|-------|
| 22  | 22  | 24    |

# Rounding of numbers

`floor` returns a numeric vector containing the largest integers not greater than the corresponding elements of the input vector

```
> bmi
```

| Can      | Cem      | Hande    |
|----------|----------|----------|
| 21.97134 | 21.55102 | 23.24341 |

```
> floor(bmi)
```

| Can | Cem | Hande |
|-----|-----|-------|
| 21  | 21  | 23    |

# Rounding of numbers

`trunc` returns a numeric vector containing the integers formed by truncating the values in the input vector toward 0

```
> bmi
```

| Can      | Cem      | Hande    |
|----------|----------|----------|
| 21.97134 | 21.55102 | 23.24341 |

```
> trunc(bmi)
```

| Can | Cem | Hande |
|-----|-----|-------|
| 21  | 21  | 23    |

Lecture 2 finished here

# Rounding of numbers

`signif` rounds the values of the input vector to the specified number of significant digits

```
> signif(bmi, digits=1)
```

| Can | Cem | Hande |
|-----|-----|-------|
| 20  | 20  | 20    |

```
> signif(bmi, digits=2)
```

| Can | Cem | Hande |
|-----|-----|-------|
| 22  | 22  | 23    |

```
> signif(bmi, digits=3)
```

| Can  | Cem  | Hande |
|------|------|-------|
| 22.0 | 21.6 | 23.2  |