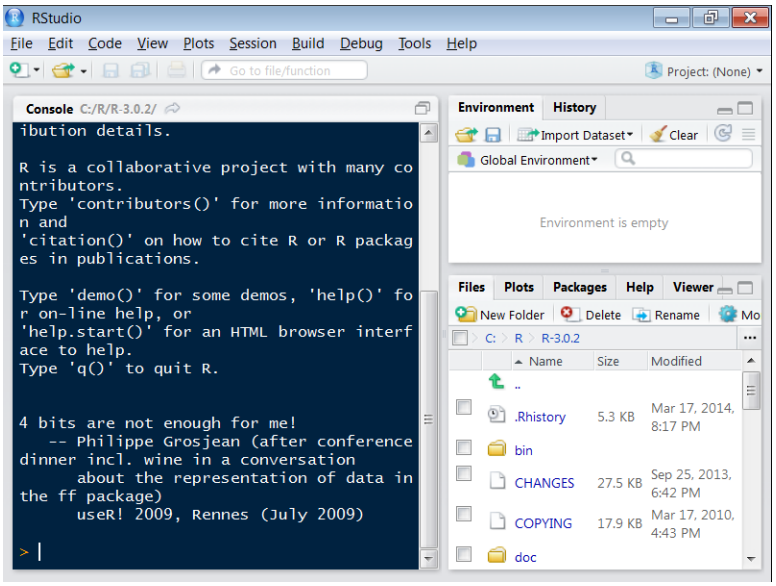
Introductie R en RStudio

Ivy Jansen, Pieter Verschelde, Thierry Onkelinx

R vs Rstudio

- R
  - Taal waarmee je gegevens (statistisch) kan verwerken
  - Beschikt over een rudimentaire GUI (Graphical User Interface)
- RStudio
  - Toolbox rond R
  - Vereenvoudigt het gebruik van R zeer sterk
  - Sterk aanbevolen door BMK
  - Andere toolboxen als TINN-R, Eclipse, Emacs, ... zijn toegelaten
    - Niet door BMK ondersteund
- Wat te doen na een eerste installatie of een upgrade van R en RStudio?
  - INBO tutorials website
  - RStudio
  - R

Basisscherm



Coding basics

R als rekenmachine

```
1 / 200 * 30
(59 + 73 + 2) / 3
sin(pi / 2)
sqrt(169)
```

Nieuwe objecten creëren

```
x <- 3 * 4
y <- sqrt(169)
z <- (x > y)
naam <- "Ivy Jansen"
```

Alle R commando's waarmee een object aangemaakt wordt, **assignments**, hebben dezelfde vorm

```
object_name <- value
```

Sneltoets voor " <- " : ALT + "-"

What's in a name?

- Object namen starten met een letter
- Bevat alleen letters, cijfers, _ en .
- Beschrijvende naam
- Verschillende conventies

```
i_use_snake_case
otherPeopleUseCamelCase
some.people.use.periods
```

```
this_is_a_really_long_name <- 2.5
```

Sneltoets om lange naam te vervolledigen : TAB

Sneltoets om vorige commando's terug op te roepen : ↑

```
r_rocks <- 2 ^ 3
```

Laten we dit object eens inspecteren

```
r_rock
#> Error: object 'r_rock' not found
R_rocks
#> Error: object 'R_rocks' not found
```

There's an implied contract between you and R: it will do the tedious computation for you, but in return, you must be completely precise in your instructions.

- Typos matter
- Case matters

Functies oproepen

R heeft een grote collectie van ingebouwde functies, die je als volgt oproept

```
function_name(arg1 = val1, arg2 = val2, ...)
```

```
sin(pi / 2)
sqrt(169)
```

```
seq(1, 10)
round(5.78)
```

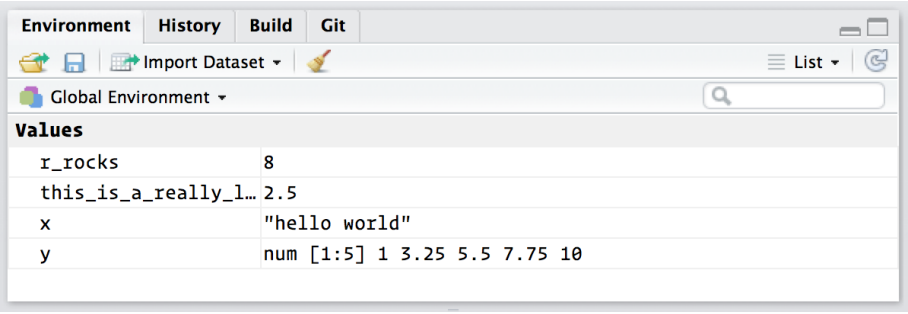
R helpt waar mogelijk met haakjes en aanhalingstekens

```
x <- "hello world"
```

```
> x <- "hello
+
```

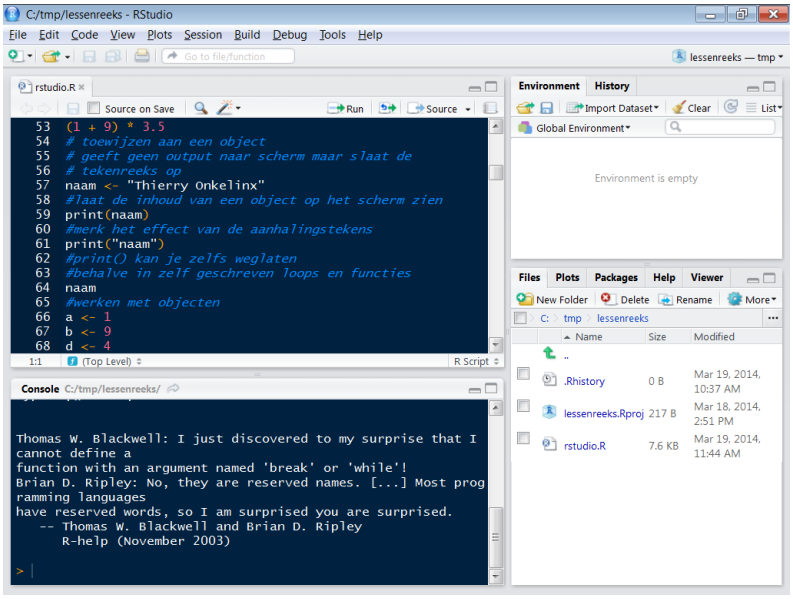
Een + aan het begin van de regel betekent dat R wacht op meer input. Meestal betekent dit dat je een " of een) vergeten bent. Voeg het ontbrekende teken toe en duw op ENTER, of duw op ESCAPE om het commando af te breken.

Environment



Scripts

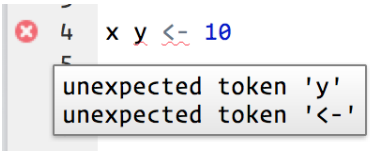
- Eenvoudig tekstbestand met .R extentie
- Bevat een set van R commando's in een logische volgorde
- Gemakkelijk commentaar toe te voegen
- Principe
  - Alle code in een script
  - Code van script naar console sturen om uit te voeren
    - CTRL + ENTER stuurt huidige regel naar console (en gaat naar de volgende regel)
    - Indien tekst geselecteerd, dan wordt de volledige selectie doorgestuurd
  - Je past de code in het script aan tot dat er gebeurt wat jij wenst
  - Je bewaart het script om het later opnieuw te gebruiken (of aan verder te werken)
- Nieuw script starten
  - Via menu `File` –> `New file` –> `R Script`
  - Geef je scripts een zinvolle naam
- Bestaand script openen
  - Via menu `File` –> `Recent files`
  - Via menu `File` –> `Open file ...`
  - Via tabblad `Files` dubbelklikken op het bestand
- Je kan meerdere scripts naast elkaar openen
  - Elk script wordt een apart tabblad



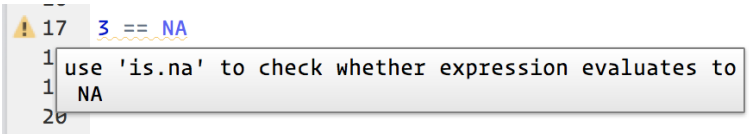
Handige weetjes i.v.m. scripts

- RStudio beschikt over *syntax highlighting*

- Beweeg over het kruisje om te zien wat het probleem is

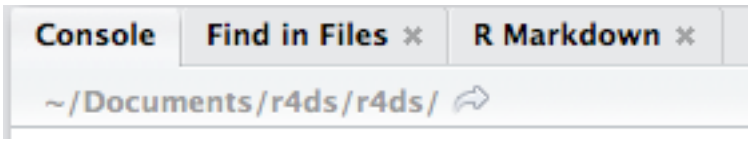


- Ook mogelijke problemen worden aangegeven



- Als je de cursor op een haakje plaatst, dan zal het overeenkomstige haakje oplichten
- CTRL + SHIFT + C zet de selectie (of huidige regel) om naar commentaar regel(s)
  - Commentaarregels beginnen met `#`
  - Indien het commentaarregels betrof, worden de commentaar tekens verwijderd
- Bestanden met een asterix (*) achter hun (rode) naam bevatten wijzigingen die nog niet bewaard werden
  - Bestand bewaren met CTRL + S of `File` –> `Save`
  - RStudio zorgt continu voor een autosave van alle scripts

Waar bevind ik me?

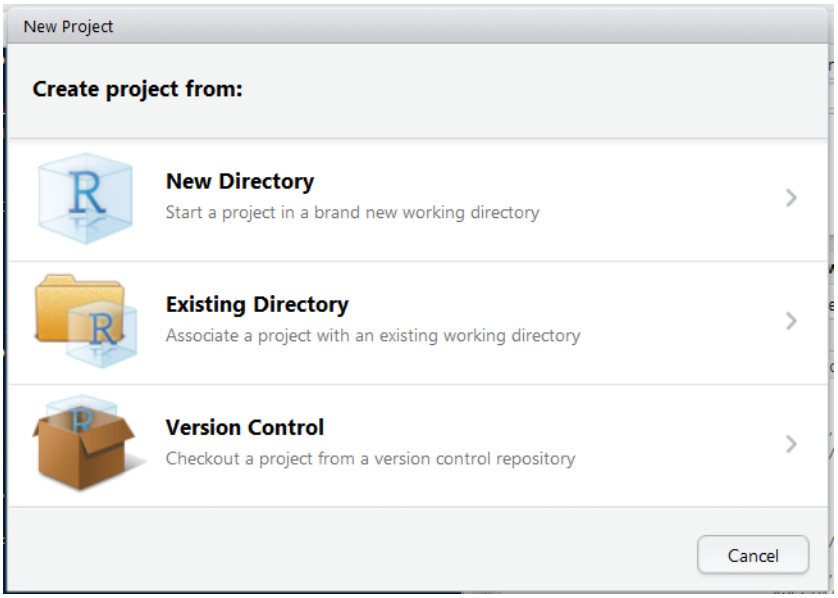


- Working directory
  - `getwd()`
  - `setwd("/path/to/my/CoolProject")`
  - `Files` venster –> `More` –> `Set As Working Directory`
- Gebruik van paden
  - Om data in te lezen
  - Om resultaten weg te schrijven
  - Keuze tussen absolute en relatieve paden
- Verwarrend, en vooral **vervelend** als je de structuur van je pc verandert, of werk wil doorgeven aan een collega

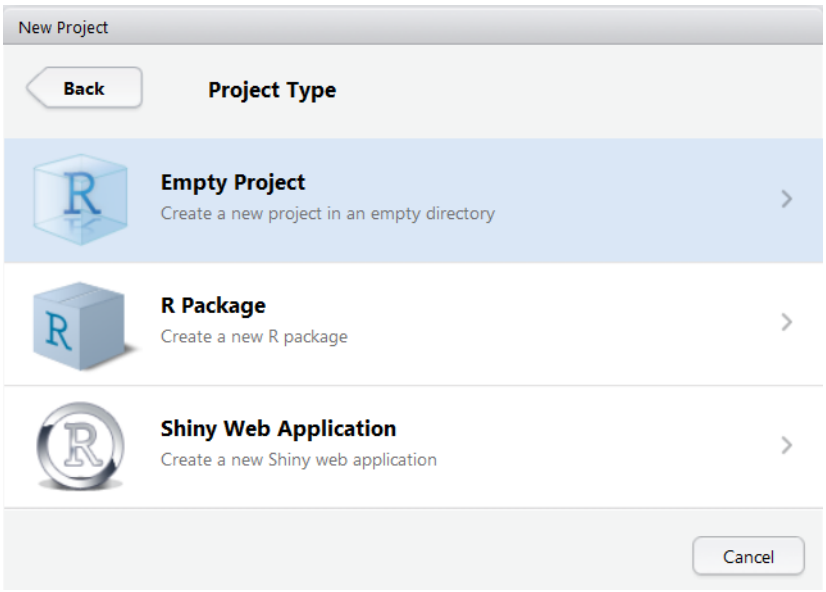
Projecten

- Een handige manier om je R werk te structuren
- Voordelen
  - Projectspecifieke basisdirectory
    - Alle scripts en data kan je per project samenzetten
  - Geopende scripts bij het afsluiten zullen terug geopend worden
  - Meerdere projecten kunnen naast elkaar geopend worden
    - Start hiervoor een RStudio sessie per project
  - Hele project gemakkelijk doorgeven aan een collega
- **Suggestie**: gebruik minstens één RStudio project per (onderdeel van een) JIRA project
- Starten met een nieuw project
  - In RStudio via het menu `File` –> `New project...`
- Bestaand project openen
  - In Verkenner dubbelklikken op `.Rproj` bestand
  - In RStudio via menu `File` –> `Recent projects`
  - Of `File` –> `Open project...`
  - Of knop rechtsboven

Nieuw project

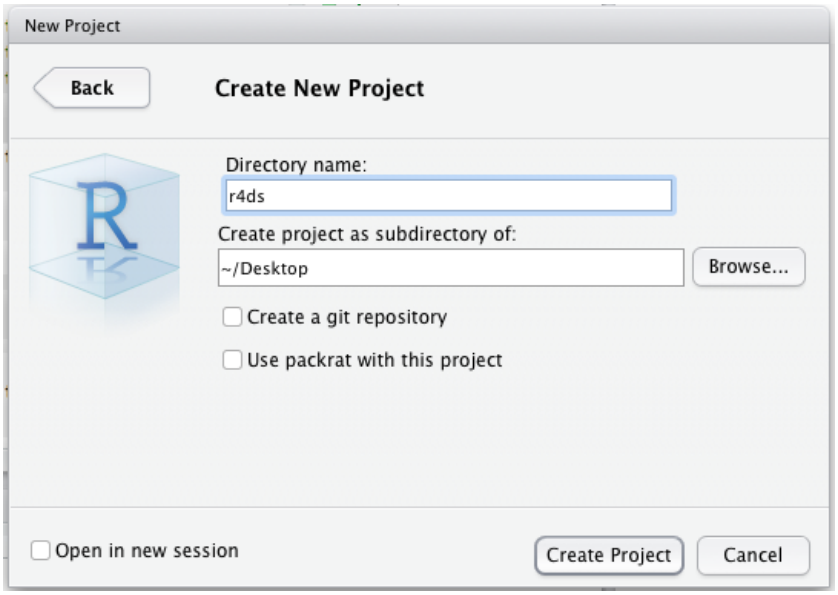


- **New directory**:
  - Maak het project in een nieuwe directory
- **Existing directory**:
  - Maak een project op basis van een bestaande directory
  - In een volgende stap selecteer je de gewenste directory
- **Version control**:
  - Start een project met versiebeheer
  - Geavanceerde versie van *track changes* in Word
  - Behoorlijk geavanceerd en buiten scope van deze cursus



- **Empty project**
  - Een leeg project
  - Je kiest

    * de naam van het project (`Directory name`)
    * de naam van een directory waarin het project als subdirectory gemaakt wordt
  - Klik daarna op *Create project*
- **R Package**
  - Voor wie zelf code wil documenteren onder de vorm van een R package
  - Buiten de scope van deze cursus
  - Slides van workshop beschikbaar bij BMK
- **Shiny Web Application**
  - Jouw R analyse beschikbaar stellen als een webapplicatie
  - Buiten de scope van deze cursus



Aanbevolen structuur projectmap

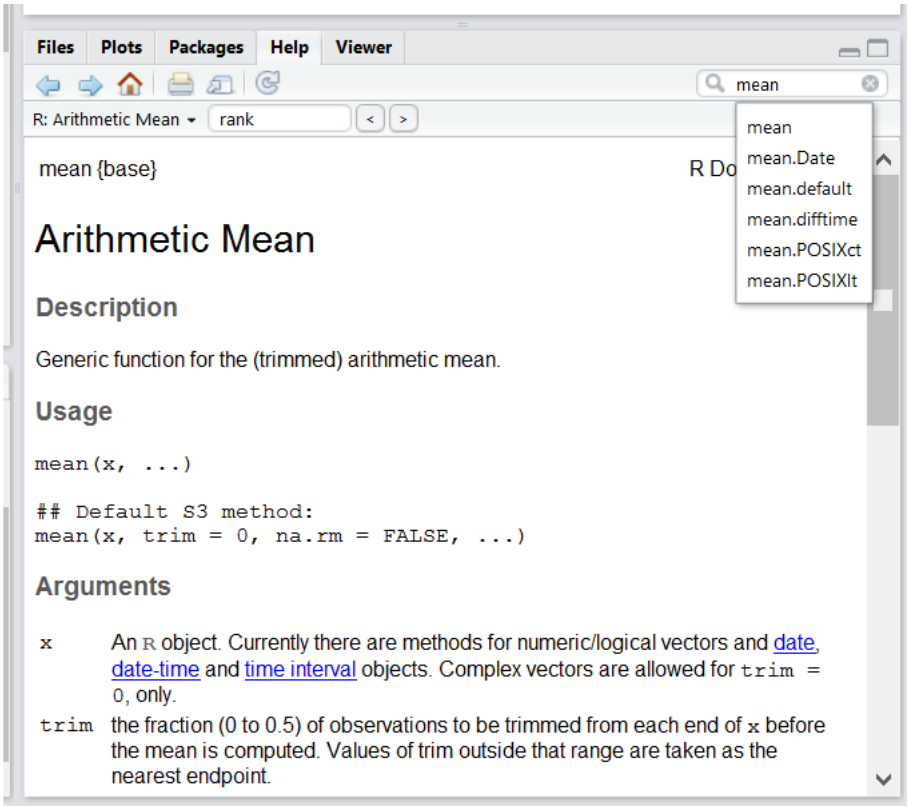
- MijnProject.Rproj
- ./data
  - Alle datasets (xls, txt, csv, . . .)
- ./media
  - Alle figuren en dergelijke
- ./src
  - Alle scripts
- Nieuwe mappen gemakkelijk aan te maken via `Files` venster

Packages

- Een R **package** is een collectie van functies, data en documentatie ter uitbreiding van base R.
- Deze moeten eerst geïnstalleerd worden met het commando `install.packages("Name_Package")`. Dit dient slechts éénmalig uitgevoerd te worden en mag in de console
  - `install.packages("readr")`
- Kan ook via het `Packages` venster –> `Install`
- Zorg dat je verbonden bent met het internet !!!
- Je kan de functies, objecten en help files van een package pas gebruiken nadat het geladen is met het commando `library()`. Zet dit commando steeds bovenaan je R-script, zodat alle packages die nodig zijn voor de analyse, vanaf de start geladen zijn
  - `library(readr)`
- Kan ook via het `Packages` venster, en dan de box voor het package aanvinken, maar dit is niet aan te raden, wegens niet reproduceerbaar

Help

- Gebruik de built-in RStudio help interface voor meer info over R functies



- Ik ken de naam van de functie die ik wil gebruiken, maar weet niet goed hoe
  - Gebruik het vraagteken
  - `?mean`
- Ik wil een functie gebruiken die `X` doet. Er moet zo een functie bestaan, maar ik weet niet welke
  - Gebruik `help.search()` of het dubbele vraagteken `??`
  - `??kruskal`
  - Dit zoekt enkel in de reeds geïnstalleerde packages

- rdocumentation.org website
  - Doorzoekt de help files van alle bestaande packages
- Generieke zoektocht op Google "R <task>"
  - Package documentatie
  - Forum waar al iemand anders jouw vraag gesteld heeft
- Ik zit vast... Ik krijg een error message en ik begrijp het niet
  - Google de error message
  - Werkt niet altijd, omdat de foutmelding heel generiek kan zijn
  - Voeg de naam van de functie of het package toe in de zoekopdracht
- http://stackoverflow.com/questions/tagged/r
  - Gebruik de tag `[r]`
  - Uitdaging is om de juiste woorden te gebruiken in de zoekopdracht
- Ik krijg van iemand een script (of open een "oud" script van mezelf) en weet niet (meer) wat een bepaalde functie doet
  - Zet de cursor op die functie (of selecteer de functie) en druk `F1`

Vectoren

- Meest voorkomend en elementair gegevenstype in R
- Een reeks waarden
- Gebruik de functie `c()` om een reeks waarden toe te kennen aan een vector

```
weight_g <- c(50, 60, 65, 82)
weight_g
animals <- c("muis", "rat", "hond", "kat")
animals
```

- Aanhalingstekens rond tekst zijn essentieel. Zonder aanhalingstekens denkt R dat het gaat over de objecten `muis`, `rat`, `hond` en `kat`. Aangezien deze niet bestaan in het geheugen van R, zal je een foutmelding krijgen
- Er zijn ook nog functies die vectoren creëren

```
Tot10 <- seq(1, 10)
Tot10
Vanaf10 <- 10:1
Vanaf10
```

Inhoud van een vector inspecteren

- Hoeveel elementen zitten er in de vector? `length()`
- Datatype van de vector (alle elementen moeten van hetzelfde datatype zijn): `class()`
- Structuur van een object en zijn elementen: `str()`

Subsetting vectors

- Een of meerdere waarden van een vector selecteren = een of meer indices tussen vierkante haken

```
animals[2]
animals[c(3, 2)]
```

- Negatieve indices: alles behalve die elementen

```
animals[-1]
animals[-c(2, 4)]
```

- Indices herhalen om bepaalde elementen meer te laten voorkomen

```
more_animals <- animals[c(1, 2, 3, 2, 1, 4)]
more_animals
```

- In R beginnen indices bij 1

Simpele plot van 2 vectoren

```
x <- c(2,  9,  6,  3,  6,  5,  7,  0, 10,  4)
y <- c(4, 25, 18,  6, 16, 14, 20, -2, 28,  9)
plot(x, y)
```

Belangrijkste datatypes

- `"integer"` voor gehele getallen
- `"numeric"` voor reële getallen
- `"character"`
- `"logical"` voor `TRUE` en `FALSE`
- `"factor"` voor een (al dan niet geordende) categorische variabele
- `"Date"` voor een datum met of zonder tijd

```
animals_factor <- as.factor(animals)
class(animals_factor)
levels(animals_factor)
labels(animals_factor)
str(animals_factor)
```

Dataframes

- Vectoren combineren

```
EigenData <- data.frame(
  D = x,
  E = y,
  e = c("A", "B", "C", "B", "C", "A", "A", "A", "C", "C")
)
EigenData
```

- Standaard dataframes in R

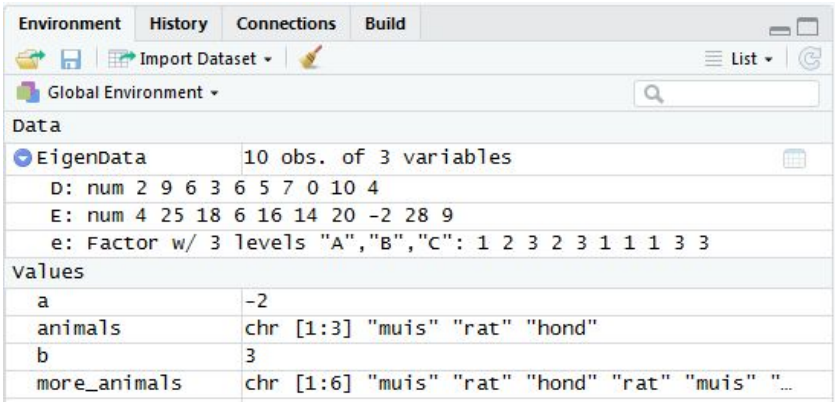
```
iris
```

- Dataframes inlezen vanuit een bestand (Excel, tekst, csv,. . .)
  - Zie verder

Eigenschappen van dataframes

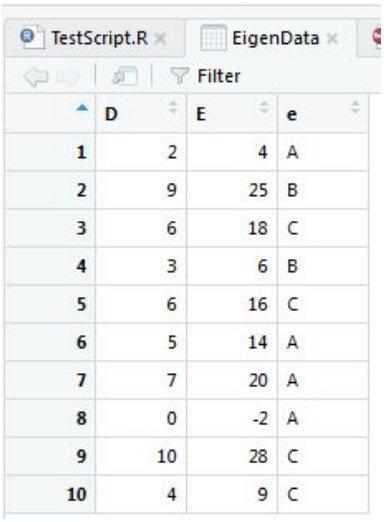
- Aantal rijen `nrow()`
- Aantal kolommen `ncol()`

- Aantal rijen en kolommen `dim()`
- Rijnamen `rownames()`
- Kolomnamen `colnames()` of `names()`
- Structuur `str()`



Dataframe bekijken

- Een volledig overzicht `View()`



	D	E	e
1	2	4	A
2	9	25	B
3	6	18	C
4	3	6	B
5	6	16	C
6	5	14	A
7	7	20	A
8	0	-2	A
9	10	28	C
10	4	9	C

- Eerste n regels `head()`
- Laatste n regels `tail()`
- Samenvatting `summary()`
- Variabele (= kolom) selecteren
  - Met vierkante haken `EigenData[, 3]`
  - Met een dollar-teken en de variabele naam `EigenData$e`
  - Dit wordt dan een vector
- **Wees kritisch** bij het evalueren
  - Juiste datatypes
  - Correct aantal NA waarden

  - Realistische waarden voor min, max, mean, ...

```
class(iris)
str(iris)
summary(iris)
iris[, 4]
iris$Species
```

Missing data

- Ontbrekende waarden codeer je als `NA`
  - -9999 = *recipe for disaster*
- Bewerkingen op getallen
  - Meeste functies geven `NA` als resultaat wanneer er missing data aanwezig zijn
  - Extra moeilijkheid !!!
  - Argument `na.rm = TRUE` toevoegen aan de functie

```
heights <- c(1, 2, 4, 4, NA, 6, 8)
mean(heights)
max(heights)
mean(heights, na.rm = TRUE)
max(heights, na.rm = TRUE)
```

- Functies om te kunnen omgaan met missing data
  - `is.na()`
  - `na.omit()`
  - `complete.cases()`

```
!is.na(heights)
na.omit(heights)
complete.cases(heights)
heights[!is.na(heights)]
```

Tibbles

- Speciaal soort dataframe van het `tidyverse` package
- Wordt altijd mooi geprint in de Console (in tegenstelling tot grote dataframes)
- Belangrijkste informatie wordt getoond

```
iris
library(tidyverse)
IRIS <- as_tibble(iris)
IRIS
class(IRIS)
```

Plotjes

```
plot(iris$Sepal.Length, iris$Sepal.Width)
plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Species)
plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Species,
     main = "Iris Sepal data", xlab = "Length", ylab = "Width")
barplot(iris$Petal.Length)
```

Tips & tricks, shortcuts

- ALT + "-" : maak een toekenningspijltje " `<-` "
- TAB : vraag R om het commando aan te vullen
- ↑ (in Console) : roep het vorige commando terug op
- CTRL + ENTER : voer het commando uit waar de cursor staat (niet nodig om te selecteren)
- CTRL + SHIFT + ENTER : voer alle commando's uit
- CTRL + S : save script
- CTRL + SHIFT + S : source het script (alle commando's uitvoeren)
- CTRL + Z : undo
- CTRL + SHIFT + Z : redo
- CTRL + L : clear console
- CTRL + SHIFT + F10 : restart R
- CTRL + SHIFT + C : verander een regel code in een commentaar regel (#) of omgekeerd

More to learn

- R for data science
  - Boek van Hadley Wickham en Garrett Grolemund
  - Hardcopy beschikbaar op INBO
  - Digitale versie
- Datacamp
  - (gedeeltelijk) gratis lessen (video tutorials en oefeningen)
  - Account voor 72h voor volledige toegang, daarna betalende licentie (~ €25/maand)
  - Introduction to R
  - Importing data in R (part 1)
    - readr
    - readxl
  - Gewone tutorial Quick-R
- Data Carpentry
  - Data Carpentry is a non-profit organization that develops and provides data skills training to researchers
  - Building communities teaching universal data literacy
  - Lessen voor ecologen
- Stat 545
  - Topic list
- Cheat Sheets
  - In RStudio onder `Help` menu
  - Online

Referenties

- R for data science
- Slides van Thierry uit 2015