Introductie R en RStudio

Ivy Jansen, Pieter Verschelde, Thierry Onkelinx

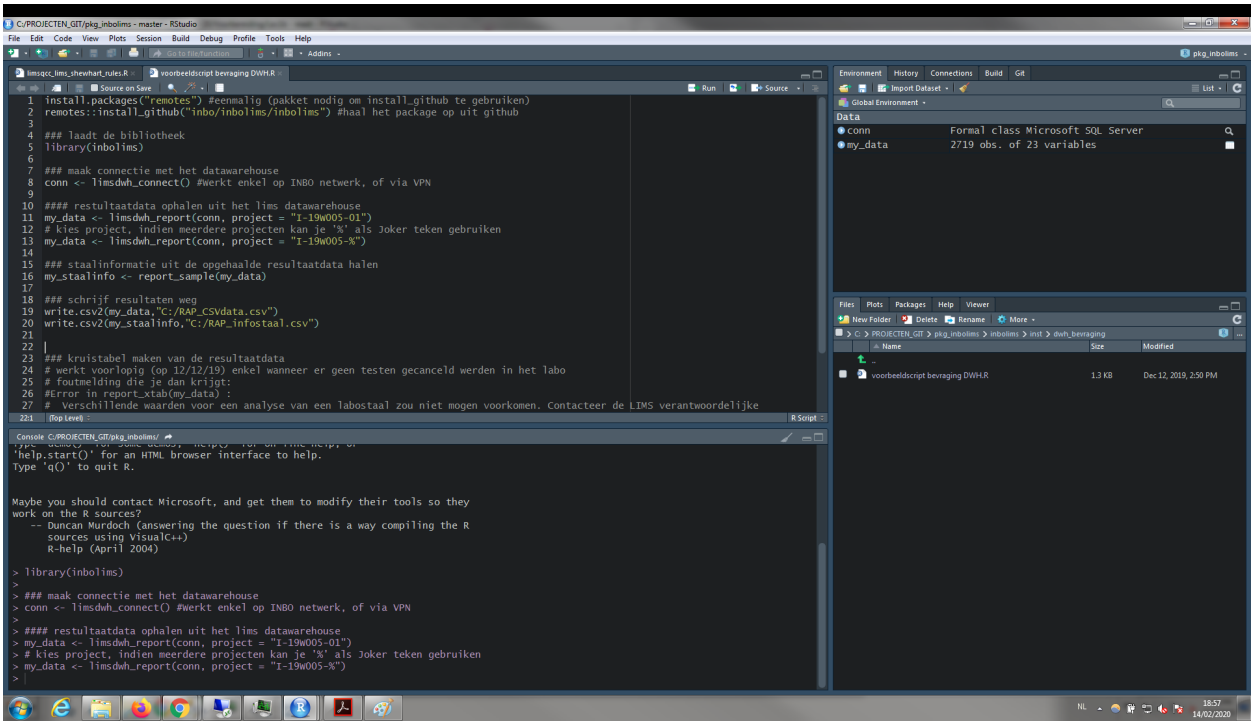
R vs Rstudio

- R
  - Scripting taal waarmee je gegevens (statistisch) kan verwerken
  - De software R is de motor voor de gegevensverwerking
  - Het programma R zelf beschikt over een rudimentaire GUI (Graphical User Interface)
    - Zelden rechtstreeks gebruikt
    - Meestal via een uitgebreide interface die R oproept, zoals RStudio
- Waarom R?
  - Open Source en gratis
  - Softwarepakket speciaal ontworpen voor data verwerking, maar daarnaast ook heel flexibel
  - Krachtige motor voor visualisaties
  - Veel tools voor data verkenning en kneding ("wrangling")
  - Van zeer eenvoudige tot heel complexe modellen kunnen gemakkelijk gecodeerd worden
  - Zowat iedere nieuw gepubliceerde techniek is onmiddellijk in R beschikbaar
  - Kan gemakkelijk met andere programma's interfacen zoals QGIS, STAN, MARK, INLA
- Hoe werk je met R?
  - Schrijf een script
  - Voer de commando's van het script uit
  - Bewaar de resultaten en figuren die bekomen worden
- RStudio
  - Interface (toolbox) rond R
  - Om het gebruik van R sterk te vereenvoudigen
  - Meest gebruikte interface voor R, sterk aanbevolen door BMK
  - Andere toolboxen als TINN-R, Eclipse, Emacs, . . . hebben niet de kracht van RStudio
- Wat te doen na een eerste installatie of een upgrade van R en RStudio?
  - zie informatie op de tutorials website van INBO
  - Specifiek voor R en Rstudio kan je de links hieronder vinden (aangeraden, niet verplicht)
    - RStudio
    - R

Opbouw van de les

- Les 1: Basisprincipes Rstudio en R
    1. Rstudio leren gebruiken
    2. Basis R code
    3. Oefening / Huistaak
    4. Data inlezen (mogelijks wordt dit deels of in het gehele verschoven naar les 2)

Basisscherm



Rstudio bestaat uit een menu bovenaan en bestaat standaard over 4 hoofdpanelen. Je kan de layout, de kleuren, enz... allemaal aanpassen.

1. Titelbalk (helemaal bovenaan):
    - balk zoals in de meeste programma's om te knippen/plakken, nieuwe bestanden te maken, opties te wijzigen, ...
2. Scriptvenster (Linksboven): Alleen zichtbaar al je een script hebt openstaan
3. Console output (Linksonder): Hierin worden de R commando's uitgevoerd en krijg je standaard de output
4. R omgeving (Rechtsboven): Bestaat uit verschillende tabbladen
    1. Environment: Bevat alle objecten die actief zijn in de sessie
    2. History: Bevat de historiek van je commando's die je hebt uitgevoerd
    3. Connections / Git / Build: Enkel zichtbaar als je effectief deze nodig hebt
5. Varia omgeving (Rechtsonder): Bevat veel verschillende types info
    1. Files: Toont de fysieke bestanden op je harde schijf (een soort verkenner)

2. Plots: Als je figuren maakt, worden deze hier getoond
3. Packages: overzicht van alle geïnstalleerde en geladen packages
4. Help: Help voor R functies
5. Andere tabbladen zoals viewer

R console en eerste korte introductie in de R taal

R kan je gebruiken van een simpele rekenmachine tot een uitgebreide set aan commando's en functies. Dit gebeurt door commando's zoals hieronder te typen in de R console.

R als rekenmachine

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

Nieuwe objecten creëren

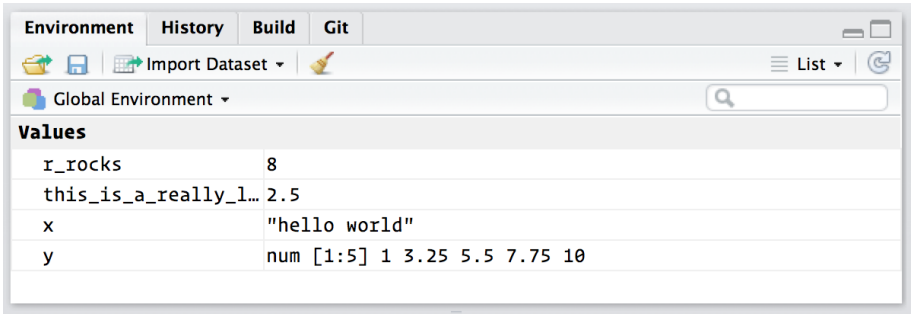
```
x <- 3 * 4
y <- sqrt(169)
z <- (x > y)
naam <- "Ivy Jansen"
```

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

```
object_name <- value
```

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

Environment paneel



- De environment geeft een overzicht van de objecten die gekend zijn

  - Voor simpele objecten zie je direct de waarde
  - Voor data en lijsten krijg je de dimensies te zien
    * Je kan klikken op het object en dan wordt dit getoond in een datavenster, waarin je kan filteren, sorteren, . . . , zonder impact op het object zelf
    * Je kan op het blauwe pijltje klikken om het object uit te vouwen en meer in detail te zien

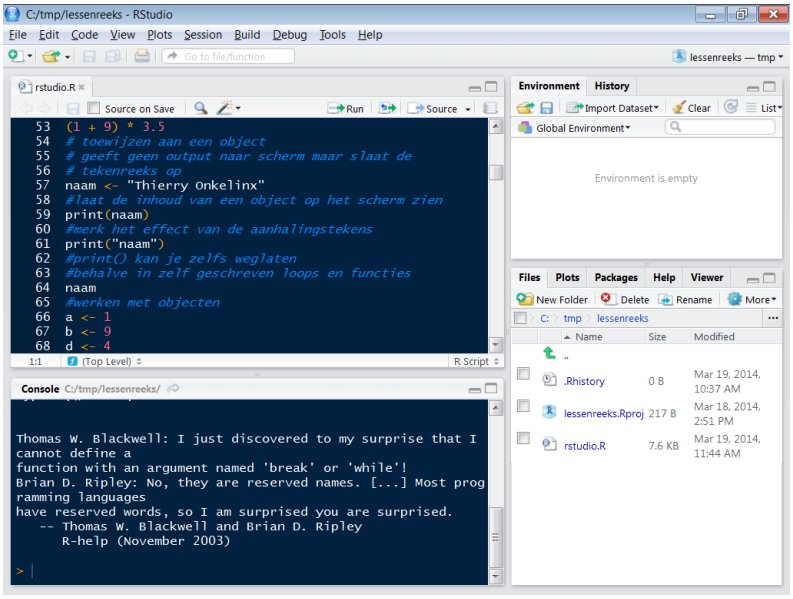
Scripts paneel

Wat is een script

- Eenvoudig tekstbestand met .R extensie
- Bevat een set van R commando's in een logische volgorde
  - Programma's als Rstudio zullen de commando's kleuren naargelang ze commentaar, functies, . . . zijn
- Gemakkelijk commentaar toe te voegen
  -

    ### is het commentaarsymbool. Alles op dezelfde regel na het hekje wordt niet als code aanzien, maar als commentaar

  - met de snelkoppeling CTRL-SHIFT-C kan je verschillende lijnen tegelijk uitcommentariëren of terug actief maken
- Principe
  - Alle code in een script
  - De code moet je zelf nog naar de console, voordat R aan de berekeningen begint
    * 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



Basisstructuur van een script

1. Laden van packages en scripts
2. Data inlezen
3. Data formatteren en selecteren
4. Exploratieve analyse (eenvoudige tabellen en figuren)
5. Data dieper onderzoeken en samenvatten
6. Statistische analyses
7. Valideren resultaten
8. Resultaten rapporteren (mooie en doordachte tabellen en figuren)

Let op: Een script wordt altijd van boven naar onder uitgevoerd. Dus als je iets nodig hebt van bovenstaande regels, moet je die eerst uitvoeren.

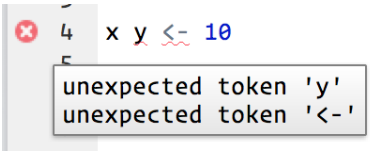
Meestal zal je je code over verschillende scripts verspreiden. Deze zijn niet afhankelijk van elkaar, dus zal je moeten afdwingen dat de nodige scripts eerst uitgevoerd zijn

- Je kan dit manueel
- Je kan je scripts een logische volgorde geven, zoals een nummering
- Je kan bovenaan een script `source("pad_naar_script_dat_eerst_uitgevoerd_wordt")` zetten

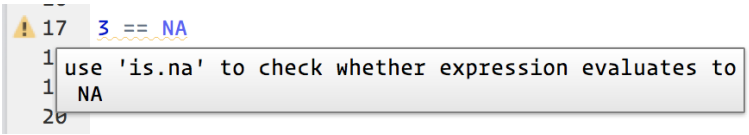
Handige weetjes i.v.m. scripts

- RStudio beschikt over *syntax highlighting* (code kleuren) een *syntax controle* (codeerfouten opsporen voor je de code uitvoert)

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

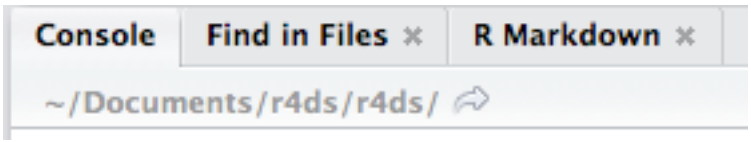


- Ook mogelijke problemen worden aangegeven



- 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?

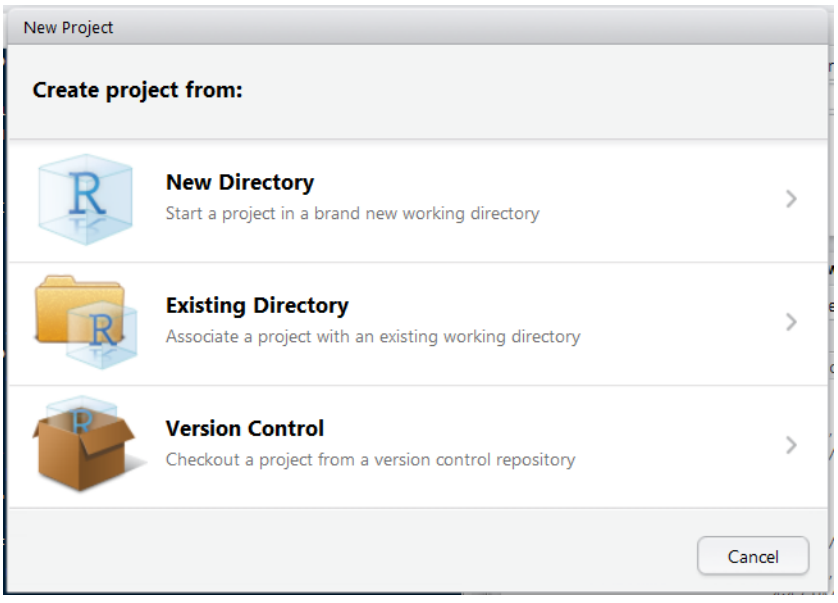


- Working directory
  - `getwd()`
  - `setwd("/path/to/my/CoolProject")`
  - `Files` venster –> `More` –> `Set As Working Directory`
  - Bovenaan in de console kan je ook je huidige werkdirectory terugvinden
- Gebruik van paden
  - Om data in te lezen
  - Om resultaten weg te schrijven
  - Keuze tussen 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
  - Dit is niets exotisch:
  - er wordt een tekstbestandje aangemaakt met de extensie `.Rproj` waarop je kan in de windows verkenner klikken om het project te openen
  - daarnaast wordt de map `.Rproj.user` aangemaakt die heel wat info bijhoudt
  - Wij raden aan zoveel mogelijk met projecten te werken!
- 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 al houdt niets je tegen om projecten binnen projecten te gebruiken
- 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

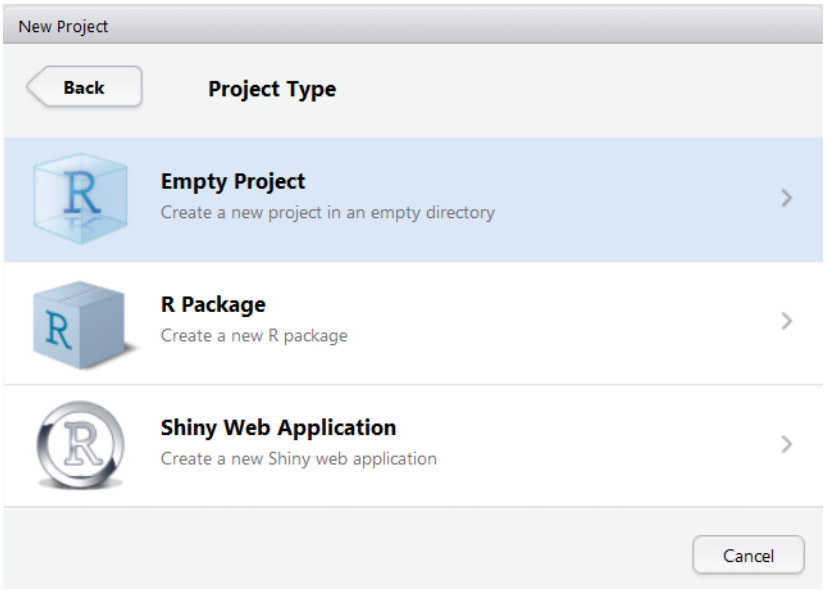


- **New directory**:

  - 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

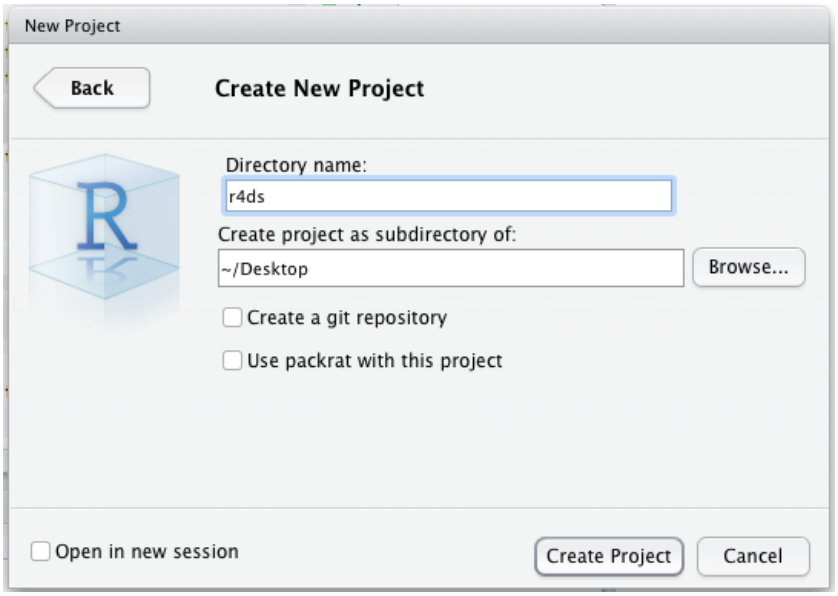
- **Version control**:
  - Start een project met versiebeheer
  - Geavanceerde versie van *track changes* in Word
  - Behoorlijk geavanceerd en buiten scope van deze cursus



- **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*

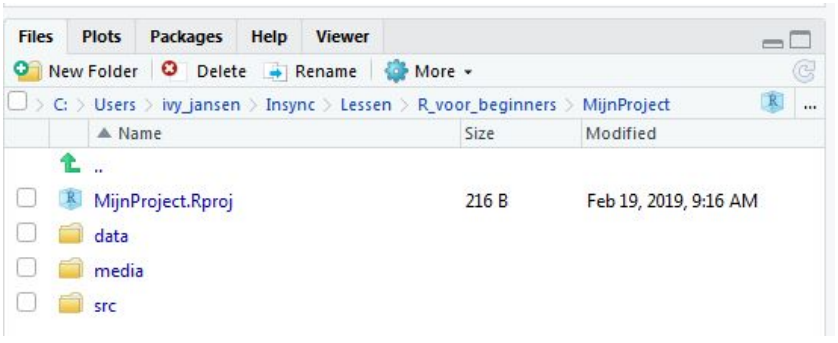
- **R Package**
  - Voor wie zelf code wil documenteren onder de vorm van een R package
  - Buiten de scope van deze cursus
  - Slides van workshop beschikbaar bij BMK

- **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 of ./figuren ./tabellen
  - Alle figuren en dergelijke
- ./src of ./R
  - Alle scripts
- Nieuwe mappen gemakkelijk aan te maken via `Files` venster



Packages (libraries)

- 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")`

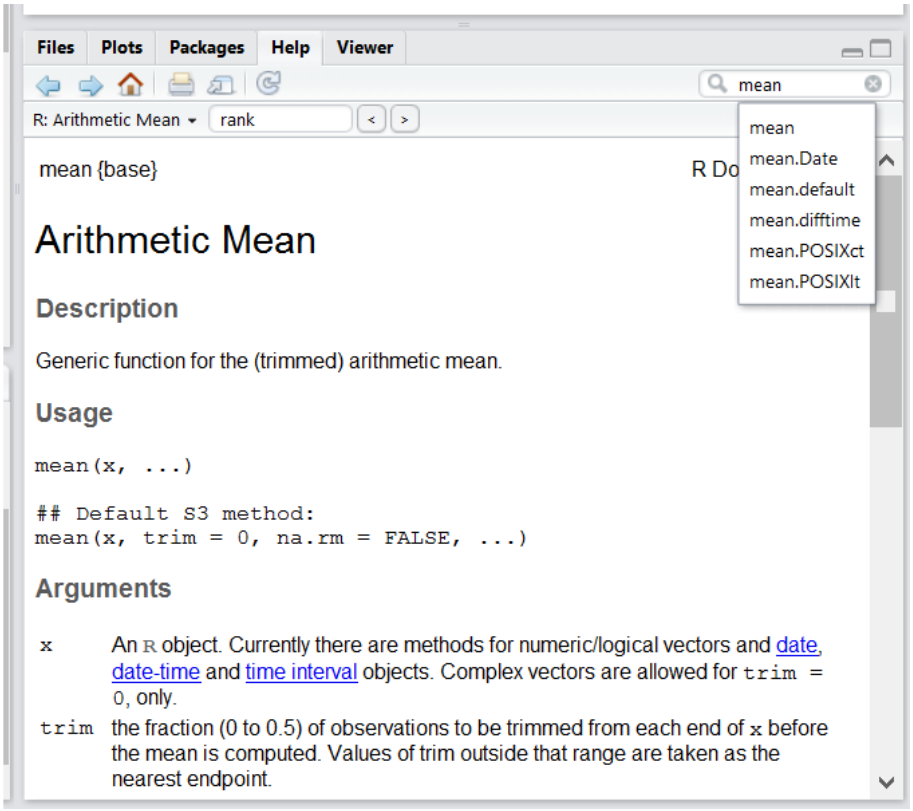
- Kan ook via het `Packages` venster –> `Install`
- Zorg dat je verbonden bent met het internet !!!
- Je kan de functies, objecten en help files van een package pas gebruiken nadat het geladen is met het commando `library()`. Zet dit commando steeds bovenaan je R-script, zodat alle packages die nodig 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
- Packages gebruiken heel vaak andere packages die moeten geïnstalleerd worden (dependency). Dus een installatie kan wel wat duren.

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 of heel specifiek 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
  - R zonder google is quasi onmogelijk, maar via google vind je heel veel antwoorden vrij snel
- 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`

R code basis

Eerder hebben we al getoond hoe je R als rekenmachine kan gebruiken en objecten kan maken en eenvoudige functies uitvoeren. Hier gaan we nu op verder.

Je zal zelden code rechtstreeks in de console intypen, maar zal daarvoor meestal een script gebruiken, waar je dan stukken code uit selecteert en laat runnen in de R console (ctrl + ENTER).

Objectnamen

In tegenstelling tot veel andere programma's werkt R steeds met objecten die je manipuleert en bevraagt. Dus als je een berekening uitvoert en bewaart in een object, zal je dat object moeten tonen om het resultaat te zien. Vaak gebeurt dit via `print` of `summary`. Een object kan gaan van een enkel getal, tot een heuse lijst van statistische modellen.

Niet alle objectnamen zijn zomaar toegestaan:

- Objectnamen moeten 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)
```

Rstudio 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.

R packages

Veel functies bevinden zich in R packages of libraries. om deze functies te kunnen gebruiken moet je de library eenmalig installeren en bij iedere nieuwe R sessie deze laden als je deze nodig hebt. Later gaan we hier veel dieper op in.

```
install.packages(tidyverse) #eenmalig

tekstje <- "van deze string wil ik het aantal characters tellen"
str_count(tekstje) #werkt niet, de functie is nog niet gekend

library(tidyverse) #telkens je een nieuwe R sessie start
str_count(tekstje) #nu werkt het wel
```

Data Types

R gebruikt verschillende datatypes om gegevens te gebruiken. Het gegevenstype kan je via `class(objectnaam)` oproepen.

Enkelvoudige types

1. Numerieke waarden (geen quotes)
    - gehele getalen (integer): 3, 5, -17
    - kommagetallen (numeric, float, double): 3.00, 3.14, -7.52, 3.4e08
    - logische waarden (logical):
        - waar: TRUE, T, 1
        - onwaar: FALSE, F, 0
    - Datum/Datumtijd: '2019-04-01 13:13:13' of idem 1554117193
2. Tekstuele waarden (altijd in quotes)
    - string: betekenisloze waarden: "ik beteken helemaal niets"
    - factor: categorische variabele: "appel", "peer", "banaan"
    - ordered: geordende categorieën: "weinig", "middelmatig", "veel"
    - Quotes zijn essentieel. Zonder quotes denkt R dat het gaat over de objecten appel, peer, weinig, veel. Aangezien deze niet bestaan in het geheugen van R, zal je een foutmelding krijgen.
3. Speciale waarden (geen quotes)
    - NA: ontbrekende waarde
    - NaN: ongeldige waarde
    - Inf: 1

```
waarde1 <- 3
waarde2 <- 5
waarde3 <- 7
waarde3
```

```
## [1] 7
```

```
naam1 <- "categorie1"
naam2 <- "categorie2"
naam3 <- "categorie3"
naam3
```

```
## [1] "categorie3"
```

```
opmerking <- "Dit is een simpele dataset"
```

Array (of vector)

- Een reeks van enkelvoudige waarden van hetzelfde type
- Een kolom in een tabel
- Kan meerdere dimensies hebben: bijvoorbeeld een matrix
- een rij cijfers wordt met c(elem1, elem2, elem3) gecodeerd

```
waardekolom <- c(waarde1, waarde2, waarde3)
waardekolom
```

```
## [1] 3 5 7
```

```
waardenrange <- 10:1
waardenrange
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
waardenrange2 <- seq(from = 1, to = 20, by = 2)
waardenrange2
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

```
namenkolom <- c(naam1, naam2, naam3)
namenkolom
```

```
## [1] "categorie1" "categorie2" "categorie3"
```

```
meerderedimensies <- matrix(1:9, ncol = 3)
meerderedimensies
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Dataset (data.frame, tibble)

- Lijst van Verschillende vectoren (kolommen) met hetzelfde aantal elementen
- Kolommen hebben allemaal een (logische) naam
- tibble is een speciaal soort dataframe van het tidyverse package – Wordt altijd mooi geprint in de Console (in tegenstelling tot grote dataframes) – Belangrijkste informatie wordt getoond

```
dataset <- data.frame(naam = namenkolom,
waarde = waardekolom)
dataset
```

```
##         naam waarde
## 1 categorie1      3
## 2 categorie2      5
## 3 categorie3      7
```

Lijst

- Verzameling die al het voorgaande kan bevatten, alsook andere lijsten

```
lijst <- list(datasetnaam = dataset,
beschrijving = opmerking)
lijst
```

```
## $datasetnaam
##         naam waarde
## 1 categorie1      3
## 2 categorie2      5
## 3 categorie3      7
##
## $beschrijving
## [1] "Dit is een simpele dataset"
```

Elementen selecteren in een vector, dataframe en lijst

Vector

Selecteren in een vector/array:

- Elementen selecteren met indices tussen vierkante haken [], dimensies gescheiden door een komma
- Negatieve indices: alles behalve die elementen
- Indices herhalen om bepaalde elementen meer te laten voorkomen
- bij meerdimesionele arrays kan je een dimensie leeglaten, dan worden alle elementen gekozen
  - dus `matrixnaam[1,]` zal alle elementen uit de eerste rij tonen
  - R wil automatisch vereenvoudigen, dus als je wil dat `matrixnaam[1,]` een matrix blijft en geen vector wordt, kan je `drop = FALSE` gebruiken

```
waardekolom[2]
```

```
## [1] 5
```

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

```
## [1] "categorie3" "categorie2"
```

```
meerderedimensies[1, 2]
```

```
## [1] 4
```

```
waardekolom[-1]
```

```
## [1] 5 7
```

```r
namenkolom[-c(2, 4)]
```

```
## [1] "categorie1" "categorie3"
```

```r
meerderedimensies[-3, -2]
```

```
##      [,1] [,2]
## [1,]    1    7
## [2,]    2    8
```

```r
meerderedimensies[1, ]   #selecteer hele eerste rij  --> vereenvoudigd tot vector
```

```
## [1] 1 4 7
```

```r
class(meerderedimensies[1, ] )
```

```
## [1] "integer"
```

```r
meerderedimensies[1, , drop = FALSE] #selecteer hele eerste rij --> blijf matrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
```

```r
class(meerderedimensies[1, , drop = FALSE] )
```

```
## [1] "matrix" "array"
```

```r
waardekolom[c(1, 2, 3, 2, 1, 2)]
```

```
## [1] 3 5 7 5 3 5
```

Dataframe

- Elementen selecteren zoals in een meerdimensionale vector, of uit een kolom zoals bij vectoren
- Kolommen selecteren met $ of [[]], dit wordt dan een vector
- Kolommen selecteren met enkele haken resulteert normaal gezien in een data.frame met enkel die kolommen, tenzij R kan vereenvoudigen, wat je kan vermijden door `drop = FALSE` te gebruiken

```r
dataset$waarde
```

```
## [1] 3 5 7
```

```r
dataset[[2]]
```

```
## [1] 3 5 7
```

```r
dataset[2]
```

```
##   waarde
## 1      3
## 2      5
## 3      7
```

```r
dataset["waarde"]    #blijft data.frame
```

```
##   waarde
## 1      3
## 2      5
## 3      7
```

```r
dataset[["waarde"]]  #conversie naar vector
```

```
## [1] 3 5 7
```

```r
dataset[3, 2]
```

```
## [1] 7
```

```r
dataset[3, 2, drop = FALSE]
```

```
##   waarde
## 3      7
```

```r
class(dataset[3, 2, drop = FALSE])
```

```
## [1] "data.frame"
```

```r
dataset$waarde[2]
```

```
## [1] 5
```

Lijsten

- Deelverzameling selecteren met $ of [[]]
- Indien je enkele haken gebruikt [], dan zal het resultaat opnieuw een lijst zijn met enkel de gekozen elementen
- Daarna verdere selecties zoals bij vectoren of dataframes

```r
lijst["datasetnaam"]
```

```
## $datasetnaam
##         naam waarde
## 1 categorie1      3
## 2 categorie2      5
## 3 categorie3      7
```

```
class(lijst["datasetnaam"])
```

```
## [1] "list"
```

```
lijst[2]
```

```
## $beschrijving
## [1] "Dit is een simpele dataset"
```

```
lijst$datasetnaam
```

```
##         naam waarde
## 1 categorie1      3
## 2 categorie2      5
## 3 categorie3      7
```

```
class(lijst$datasetnaam)
```

```
## [1] "data.frame"
```

```
lijst[[2]]
```

```
## [1] "Dit is een simpele dataset"
```

```
lijst$datasetnaam$waarde[2]
```

```
## [1] 5
```

```
lijst[[1]][[2]][[2]]
```

```
## [1] 5
```

Inspecteren van vectoren en data.frames

- Structuur van een object en zijn elementen: `str()`
- Datatype van een object: `class()`
- Samenvatting van een object: `summary()`
- Meer informatie over factoren: `levels()`, `labels()`
- Aantal elementen in een vector: `length()`
- Aantal rijen en/of kolommen in een dataframe: `nrow()`, `ncol()`, `dim()`
- Rijnamen van een dataframe: `rownames()`
- Kolomnamen van een dataframe: `colnames()` of `names()`
- Eerste n regels van een dataframe: `head()`
- Laatste n regels van een dataframe: `tail()`
- Een volledig overzicht van een dataframe: `View()` (hoofdletter V)

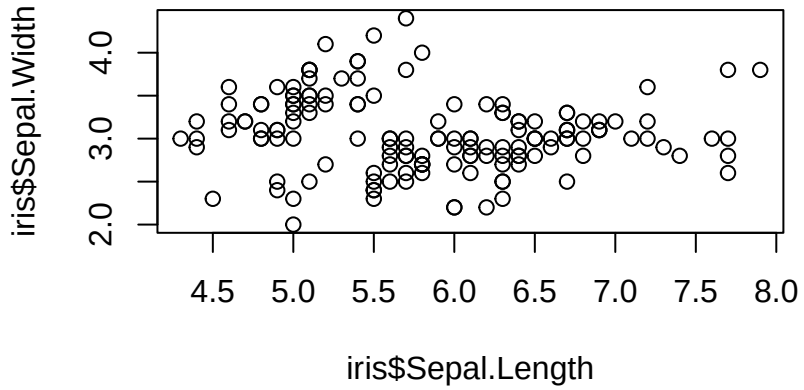
Wees kritisch bij het evalueren

- Juiste datatypes
- Correct aantal NA waarden
- Realistische waarden voor min, max, mean, . . .

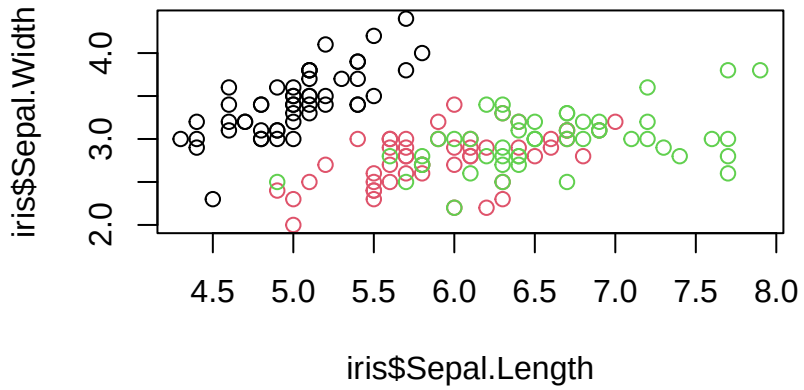
Plotjes

De iris datase is een dataset die met R meegeleverd wordt. Onderstaande plotfuncties zijn de basis R plots, maar in de volgende les van de cursus, wordt een ander type figuren (ggplot) gebruikt die standaard heel wat mooiere resultaten geven en veel mogelijkheden bieden zonder te ingewikkelde codering.

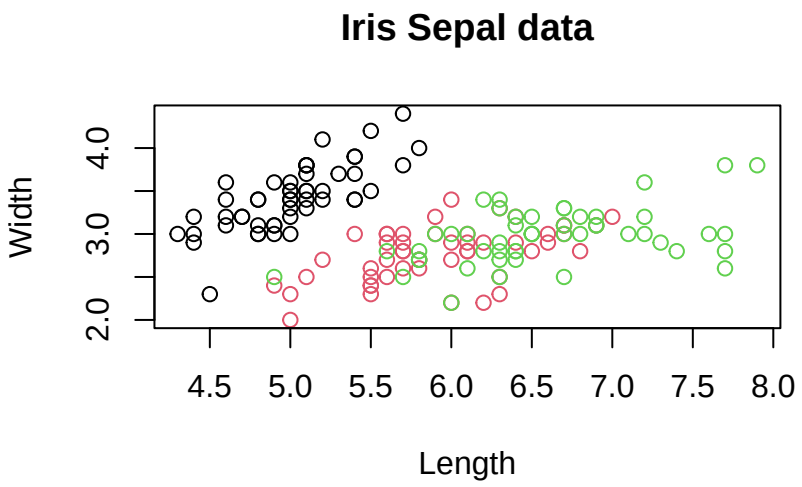
```
plot(iris$Sepal.Length, iris$Sepal.Width)
```



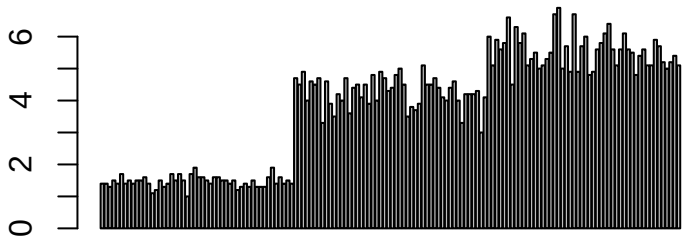

```
plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Species)
```




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




```
barplot(iris$Petal.Length)
```



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
  - Veel functies voorzien de parameter `na.rm = TRUE` welke de bereking maakt met de missings eruit gegooid.
- Functies om te kunnen omgaan met missing data
  - is.na()
  - na.omit()
  - complete.cases()

```r
heights <- c(1, 2, 4, 4, NA, 6, 8)
mean(heights)
```

```
## [1] NA
```

```r
max(heights)
```

```
## [1] NA
```

```r
mean(heights, na.rm = TRUE)
```

```
## [1] 4.166667
```

```r
max(heights, na.rm = TRUE)
```

```
## [1] 8
```

```r
!is.na(heights)
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

```r
na.omit(heights)
```

```
## [1] 1 2 4 4 6 8
## attr(,"na.action")
## [1] 5
## attr(,"class")
## [1] "omit"
```

```r
complete.cases(heights)
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

```r
heights[!is.na(heights)]
```

```
## [1] 1 2 4 4 6 8
```

Meldingen in de Console

De (rode) meldingen in de Console kunnen 3 verschillende dingen betekenen:

- Foutmelding
  - Begint met Error
  - R stopt
  - Moet opgelost worden alvorens verder te kunnen

  - Gevaar: Als een object reeds bestaat, en er komt een foutmelding, dan blijft de oude waarde in het object bestaan

- Waarschuwing
  - Begint met Warning
  - R gaat gewoon verder
  - Zeker nakijken en indien nodig aanpassen
  - Kan de reden aangeven waarom rare resultaten verkregen worden
- Mededeling
  - Begint niet met Error of Warning
  - Geeft meer informatie (vooral bij laden van packages)
  - Niet nodig iets aan te passen

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