Les 1 - Introductie R (1)

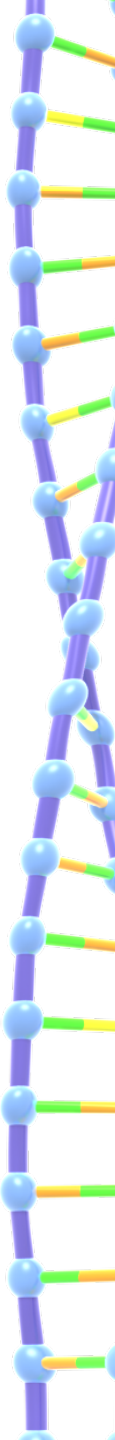
Emile Apol

Patrick Deelen

Hanze University Groningen
Applied Sciences
Institute for
Life Science & Technology

OVERZICHT STATISTIEK 3

- Leerdoelen
  - Werken met R
  - Statistische testen om microarray studies te analyseren
  - Visualisatie van microarray data
  - Multiple hypothesis testing
  - Clustering methoden
  - Studielast: 84 uur
  - Waarvan 63 uur zelfstudie



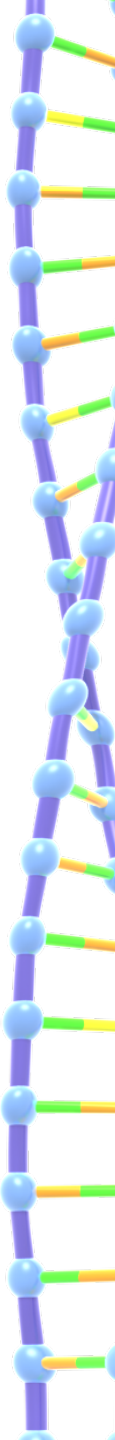
Beoordeling

- Eindcijfer
  - Bonusopdrachten (Black Board) 25%
  - Tentamen 75%
- Weging:
  - Bonus opdrachten cijfer: B
  - Tentamen cijfer: T
  - Eindcijfer C:

```
if(B >= T)
  C <- (B + 3*T)/4
else
  C <- T
```

Les 1

- R
- RStudio
- Programmeer regels
- Basis R
- R datatypen
  - Vector
  - Array
  - Matrix



R

- Programmeertaal, staat los van statistiek
- Rekenen vindt plaats op vectoren
- Veel statistiek packages beschikbaar
- Ook heel veel bioinformatica packages

- R (vanaf 1997) is voortgekomen uit S (AT & T, 1970s)

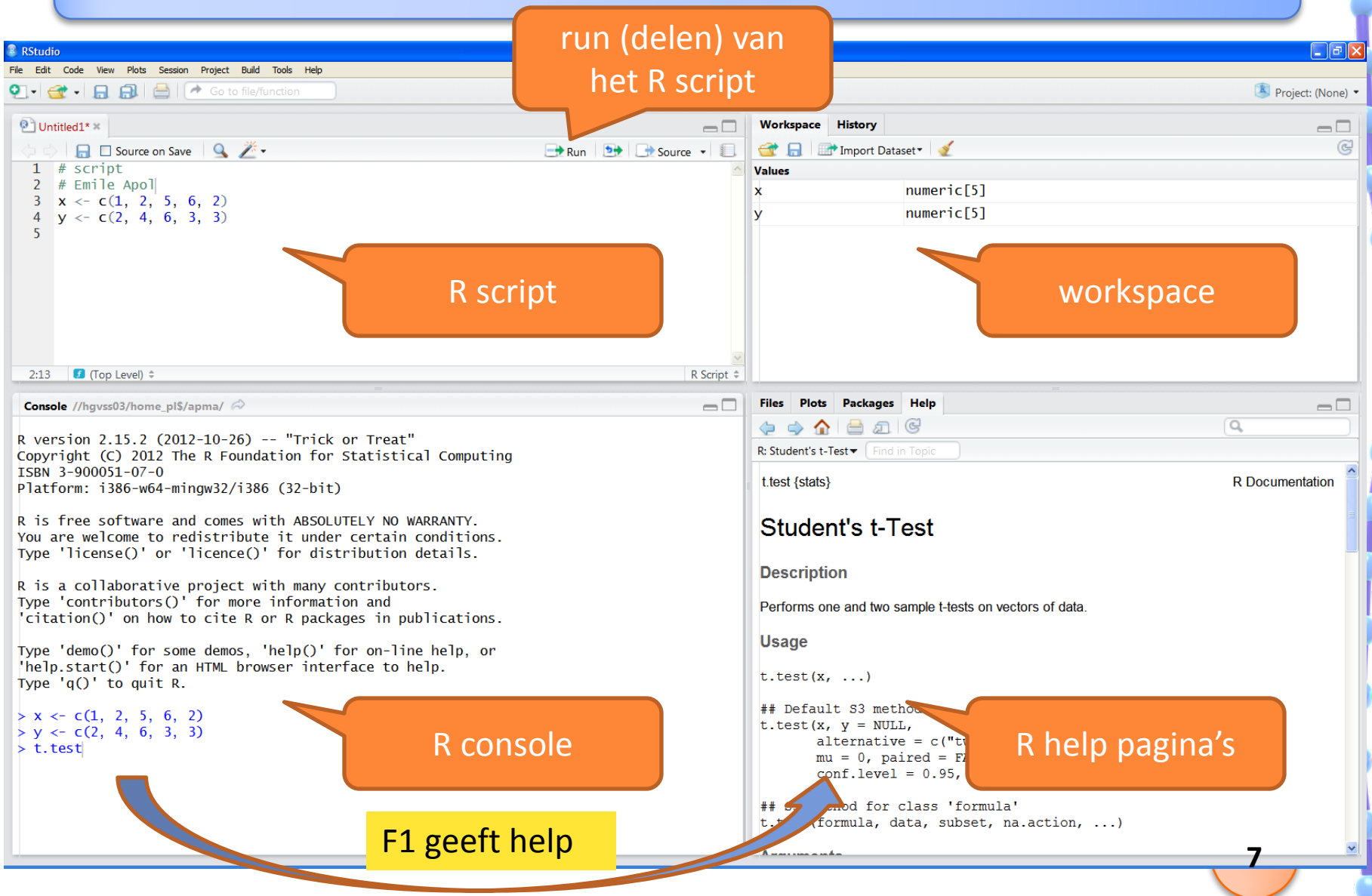
http://www.r-project.org/

RSTUDIO

- Integrated Development Environment (IDE)
- Makkelijker met R werken
- Zowel onder Linux als Windows en ook OS-X

- R console
- R script editor
- Workspace/History
- Help/Plots/Packages/Files

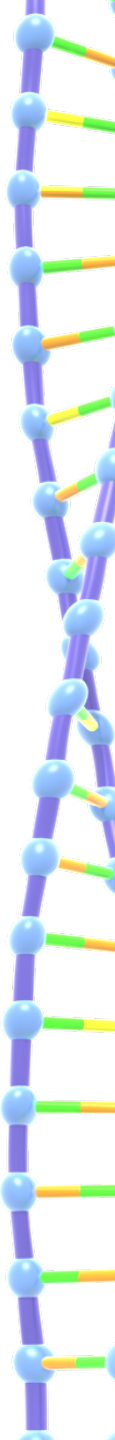
RSTUDIO


run (delen) van het R script
RStudio
File Edit Code View Plots Session Project Build Tools Help
Project: (None)
Untitled1*
Source on Save
Run
Source
1 # script
2 # Emile Apol
3 x <- c(1, 2, 5, 6, 2)
4 y <- c(2, 4, 6, 3, 3)
5
R script
2:13
(Top Level)
R Script
Workspace
History
Import Dataset
Values
x numeric[5]
y numeric[5]
workspace
Console //hgvss03/home_pl$/apma/
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-w64-mingw32/i386 (32-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.
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.
> x <- c(1, 2, 5, 6, 2)
> y <- c(2, 4, 6, 3, 3)
> t.test
R console
F1 geeft help
Files Plots Packages Help
R: Student's t-Test
Find in Topic
t.test {stats}
R Documentation
Student's t-Test
Description
Performs one and two sample t-tests on vectors of data.
Usage
t.test(x, ...)
R help pagina's

R Programmeerstijl regels 1/2

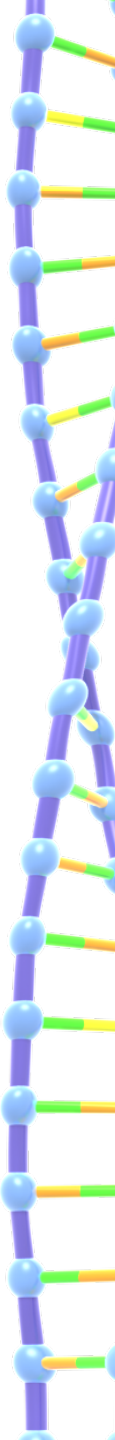
- Onderstaande regels zijn van toepassing op alle code die jullie moeten inleveren.

1. Namen van variabelen beginnen met een kleine letter
2. Nieuwe woorden in een variabele naam beginnen met een hoofdletter (of gescheiden door punt)
3. Bij afkortingen alleen hoofdletter voor eerste letter in variabele naam
4. Variabele namen en commentaar in code is altijd in het Engels
5. Variabele namen moeten beschrijvend zijn.



R Programmeerstijl regels 2/2

6. Voor functie namen gelden de zelfde regels als bij variabelen.
7. Spaties voor en na een operator
8. 1 spatie na een komma
9. Inspringen binnen { } blokken
10. Gebruik waar mogelijk **`apply`** in plaats van for loops
11. Functies binnen een **`apply`** altijd apart gedefinieerde functie



VOORBEELD R SOURCE DOCUMENT 1/2

```
#Naam
#Email adres

#Short description of script

###############
#    Libs     #
###############

library(Biobase)
```

VOORBEELD R SOURCE DOCUMENT 2/2

```
##############
# Functions #
##############

#function description
sd <- function(……

##############
#   Code    #
##############

#inline comment
```

R TOEWIJZEN VARIABELEN

- `x <- 1`
- `1 >- x`
- `x = 1`
- `assign("x", 1)`

- Bovenstaande commando's doen precies hetzelfde
- Conventie voor R is gebruik van **<-**

DATA TYPES

- Veel gebruikte data types of modes in R zijn:
  - integer
  - double of numeric
  - complex
  - logical (**TRUE** = **T**, **FALSE** = **F**)
  - character
  - list
- Er zijn er nog meer, zie **`?mode`** in R

OPERATOREN

- Rekenen: **+** **-** ***** **/** **^**
- Logica
  - **&** # and
  - **|** # or
  - **!** # not
- Vergelijkingen
  - **<** **<=** **>** **>=** **==**

Ook: xor( ), &&, ||

Precedence operatoren

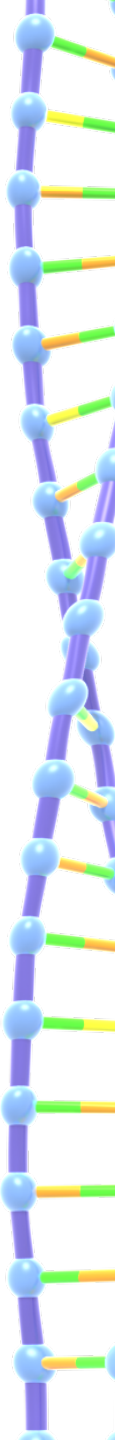
Operator	Beschrijving
$	List element extraction
[[[	Vector and list element extraction
^	Exponentiation
-	Unary minus
:	Sequence generation
* /	Multiply and divide
+ -	Addition and substraction
< > <= >= == !=	Comparison operators
!	Logical negation
& &&	Logical and
| ||	Logical or
~	Formula
<- -> =	Assignment

VECTOREN

- Wat is een vector?
  - Een reeks getallen
- Hoe kan je vector wiskundig weer geven?
  - $a = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$
- Hoe geef je elementen van een vector aan?
  - Met subscripting
  - a_1

Vectoren in R

- Bestaat uit 1 type variabele
  - Bv: integer
- Vector is het basis type in R
  - I.p.v. een scalar
  - Een enkele waarde is een vector met een lengte van 1



Maken van een vector

- Met de functie "**c**"
  - `a <- c(1, 2, 4)`
- Met de colon operator "**:**"
  - `a <- 1:30`
  - `a <- 30:1`
- Met de functie "**rep**"
  - `a <- rep(1:3, times = 3)`
  - `a <- rep(1:3, 3)`
- Met de functie "**seq**"
  - `a <- seq(from = 1, to = 3, by = .2)`
  - `a <- seq(1, 2, 0.2)`

REP( )

- Twee verschillende manieren van de functie **`rep( )`**:
  - **`rep(c(1, 2, 3), times = 2)`**
  - **`rep(c(1, 2, 3), 2)`**

  geeft: 1, 2, 3, 1, 2, 3
  - **`rep(c(1, 2, 3), each = 2)`**

  geeft: 1, 1, 2, 2, 3, 3

VECTOR REKENEN

- R kan rekenen met vectoren
- Stel je wil alle waarden uit vector **a** optellen bij waarden uit vector b en opslaan in vector **c**

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

- Zonder rekenen op vector zou je een for loop moeten gebruiken en steeds 2 waarden bij elkaar optellen
- In R met vector rekenen kan je het volgende doen:
  - **c <- a + b**

VECTOREN EN VERGELIJKINGEN

- Stel je wil weten welke waarden in **a** kleiner of gelijk zijn aan de waarden in **b** en dit opslaan in **c**
  - $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \leq \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$
- In R kan ook dit heel makkelijk en zonder loop:
  - **c <- a < b**
- Vector **c** zou nu in dit voorbeeld 3 keer een TRUE of FALSE bevatten.
  - Een waarde voor iedere vergelijking bv: $a_1 \leq b_1$

VECTOREN EN LOGISCHE OPERATOREN

- Stel ik wil weten welke elementen uit vector **a** kleiner zijn dan in vector **b** en groter zijn dan 4
  - **c <- a < b**
    d <- a > 4
- Vector **c** is true voor alle waarden van **a** die kleiner zijn dan **b**
- Vector **d** is true voor alle waarden **a** die groter zijn dan 4
- We kunnen nu heel makkelijk bepalen wanneer aan beide voorwaarden voldaan wordt en dit opslaan in **e**
  - **e <- c & d**

VECTOR ELEMENTEN SELECTEREN

- Elementen selecteren ofwel “subsetting”
- Kan met behulp van de subscripts
  - `a[1] #selecteert element 1`
  - `a[c(1, 6, 7)] #element 1, 6 en 7`
  - Let op: R telt vanaf 1, dus element “0” bestaat niet
- Kan ook met logische vector
  - `a <- 1:3`
  - `b <- c(TRUE, FALSE, TRUE)`
  - `a[b] #selecteert element 1 en 3`
- Een subset van een vector is ook een vector

LOGISCHE VECTOREN

- Een logische vector bevat TRUE's en FALSE's:
  - `a <- c(TRUE, TRUE, FALSE, TRUE)`
  - `a <- c(T, T, F, T)`
- Je kunt ook met logische vectoren rekenen, want
  - `TRUE = T = 1`
  - `FALSE = F = 0`
- Dus
  - `sum(a)  # 3, want 3 x TRUE`

SUBSETTING

- In R kun je heel gemakkelijk elementen uit een vector halen zonder for loop of if statements:
  - `a <- c(1, 2, 5, 3, 6, 8)`
- Selecteer alle elementen uit **a** die groter zijn dan 3:
  - `a[a > 3]  # 5, 6, 8`

Arrays

- Kan bestaan uit meerdere dimensies
- Een array met 2 dimensies is een matrix
- Een array met 1 dimensie is vergelijkbaar met een vector
  - Let op, er zijn een paar uitzonderingen
- Een array heeft een dimensie vector die beschrijft hoe groot hij is.
- Een array bestaat ook altijd uit 1 type variabele

ARRAYS GEBRUIKEN

- Een array maken met 3 dimensies
  - `a <- array(data = 1:16, dim = c(2,4,2))`
- `dim(a) #geeft grootte van dimensies`
- `a[1,1,1] #selecteert element dat in alle dimensies op 1 staat`
- `a[1, , ] #selecteert alle elementen uit dimensie "2" en "3" die in dimensie "1" op positie 1 staan`

Matrix

- Een matrix is een rechthoekige verzameling van waarden
- Een waarde in een matrix wordt een element genoemd
- Wiskundige weergave van een matrix
  - $a = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$
- Matrix elementen kan je ook weer geven m.b.v. subscripting
  - $a_{2,3}$ (heeft in onze voorbeeld matrix waarde: 6)

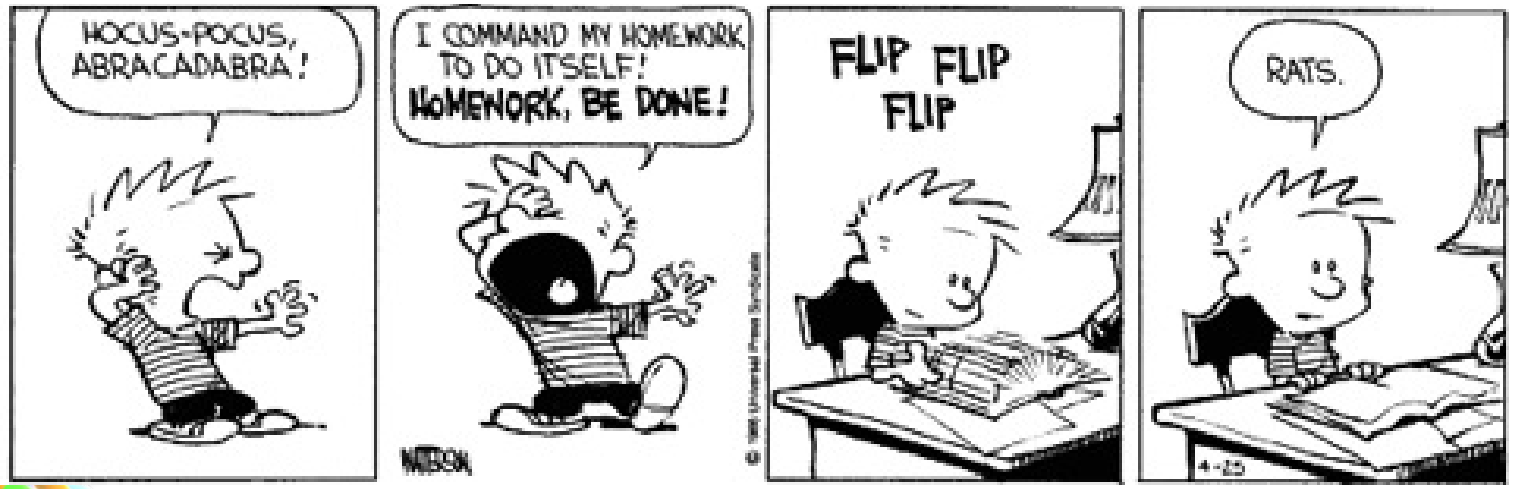
Matrix in R

- Een matrix in R is een array met 2 dimensies:
  - `a <- array(data = 1:6, dim=c(2,3))`
  - `a <- matrix(data = 1:6, nrow = 2, ncol = 3)`
  - `rownames(a) <- c("r1", "r2")`
  - `colnames(a) <- c("c1", "c2", "c3")`
  - `a[2,3] # rij 2 kolom 3`
  - `a[ ,3] # kolom 3`

OVERIGE R / RSTUDIO COMMANDO'S

- afsluiten R console:
  - **`q()`**
- help over functie:
  - **`help(functie)`**
  - **`?functie`**
  - tab # in RStudio: extra info over argumenten
  - F1 # in RStudio: R help pagina
- laden van een library:
  - **`library(MASS)`**
- variabele(n) verwijderen:
  - **`rm(x, y, z, myData)`**

Jullie kunnen nu de opdrachten van les 1 maken



Hanze University Groningen
Applied Sciences
Institute for
Life Science & Technology