Introductie R

Michiel Beelaerts

2024-04-07

Table of contents

1 Welkom!

Welkom!

In dit leerpad zal je kennismaken met R en RStudio.

Op de volgende pagina's vliegen we erin met de vraag wat R nu eigenlijk is, maar eerst vind je hieronder een korte gebruiksaanwijzing die je wegwijs maakt in alle kaders, kleuren en knoppen die je onderweg zal tegenkomen. Als je dit leerpad voor het eerst bezoekt kan het nuttig zijn om dit snel even te doorlopen.

Terzijde, we raden sterk aan om deze site altijd te raadplegen met browsers Mozilla Firefox of Google Chrome.

Gebruiksaanwijzing

Laten we eerst eens demonstreren hoe de rest van dit leerpad eruit ziet. Je zal snel merken dat de pagina's een speciale layout hebben met veel kleuren en kaders. Die layout is niet willekeurig, maar dient om alles overzichtelijker te maken.

Het belangrijkste dat je zal zien is een onderscheid tussen R-code en R-output. In R zal je voortdurend instructies geven aan je computer om berekeningen te maken. Die instructies zal je geven in de vorm van **code**. Je computer gaat aan de slag met die instructies en geeft je het resultaat. Dat is de **output**.

R-code

R-code vind je terug in een lichtblauw kader. Deze kaders bevatten instructies die je kan kopiëren met het icoontje dat rechts in de lichtblauwe balk verschijnt wanneer je erover zweeft met je muis. Vervolgens kan je die code plakken in een R-script (later meer over "scripts") en uitvoeren. Maak je vooral nog geen zorgen om wat de onderstaande code precies betekent!

```
head(mtcars)
```

Met een #-symbool is het mogelijk om R-code van wat commentaar te voorzien. Alle woorden, cijfers en symbolen die na het #-symbool staan worden genegeerd door R. Zulke commentaar zullen we op deze site gebruiken om kort toe te lichten wat we proberen te bereiken met een bepaald stukje code. (Het is trouwens een uitstekend idee om dat ook in je eigen code regelmatig te doen!)

```
head(mtcars) # we willen de eerste 6 rijen van de dataset 'mtcars' zien
```

Soms vind je korte stukjes R-code ook middenin een stuk tekst. Die code zie je tegen een lichtgrijze achtergrond en in een ander lettertype, zoals `mean(mtcars$mpg)`.

R-output

De output (zeg maar: het resultaat) van de code verschijnt in de console van RStudio (later meer daarover). Zulke output wordt op deze site altijd tegen een oranje achtergrond getoond. Je ziet ook een `[1]` staan. Dit getal tussen vierkante haakjes geeft gewoon aan dat hier het eerste element van de output staat.

```
[1] 21.0 21.0 22.8 21.4 18.7 18.1
```

Vaak zal je de code en de bijhorende output meteen onder elkaar zien. De code in het blauwe kader hieronder levert je de output in het oranje kader op.

```
head(mtcars$mpg) # toon de eerste 6 elementen van de variabele genaamd 'mpg' uit de dataset
```

```
[1] 21.0 21.0 22.8 21.4 18.7 18.1
```

Navigeren

Aan de linkerkant vind je een inhoudstafel van het volledige leerpad. Je kan een onderdeel laden door op de titel te klikken.

Het zoekveld net boven de inhoudstafel kan je gebruiken om snel terug te vinden waar een bepaald onderwerp wordt besproken.

Aan de rechterkant zie je een korte inhoud van wat er allemaal te vinden is in het onderdeel dat momenteel open staat. Ook hier kan je op een titel klikken om snel naar dat deeltje te gaan.

Helemaal onderaan vind je ten slotte pijlen waarmee je naar het vorige of volgende onderdeel kan navigeren.

Andere onderdelen

Links

Hier en daar staat een externe link naar een andere website. Dergelijke links zijn in blauwe letters aangeduid, zoals deze link naar de UGent-website. Links openen vanzelf in een nieuw tabblad.

Vraag en antwoord

In de loop van de tekst stellen we je af en toe een vraag die je kan gebruiken als zelftest (en om wakker te blijven). Om het antwoord te weten klik je op de knop zoals hieronder. Er klapt dan een extra stukje open met het antwoord en eventueel wat uitleg.

Klik hier om het antwoord te zien

Het antwoord.

Voetnoten

Op sommige plaatsen is het nodig om wat uitleg apart te plaatsen, in een voetnoot. Dit wordt aangeduid zoals bij het getalletje op het einde van deze zin[1]. Klikken op dat getalletje brengt je naar de voetnoot onderaan de pagina.

Samenvatting

Helemaal onderaan sommige pagina's vind je een groen kader. Daarin vind je een mini-samenvatting van de pagina of soms een voorbeschouwing op de volgende pagina waarin er zal worden voortgebouwd op wat je al hebt geleerd.

Samenvatting…

… of soms een "brug" naar de volgende pagina.

[1] Voetnoot. Klik op de pijl op het einde om terug te keren.

2 Wat is R?

R is niet alleen de 18e letter van het alfabet, maar ook de naam van een boeiend stuk software waarmee je zal leren werken in de loop van je opleiding. Maar wat is het juist en waarvoor dient het?

2.1 R is een hulpmiddel bij onderzoek

De vraag die je je waarschijnlijk al sinds het begin van je opleiding stelt is: "zijn studenten van de Faculteit Psychologie & Pedagogische Wetenschappen (FPPW) slimmer dan andere studenten?"

Nieuwsgierige wetenschapper die je bent, beslis je om een antwoord op die vraag te zoeken. Hoe pak je dat aan?

Een goed begin zou zijn om data te verzamelen. Je laat bijvoorbeeld enkele honderden FPPW'ers en enkele honderden andere studenten een intelligentietest afleggen. Dat levert je een reeks scores op.

Wat nu? Staren naar de tabellen met data zal je niet veel dichter bij een antwoord brengen. Je zal iets moeten doen met de data.

Een logische werkwijze zou kunnen zijn:

de gemiddelde score berekenen van de FPPW'ers enerzijds en van de andere studenten anderzijds

het verschil tussen die gemiddeldes berekenen

uitzoeken of het verschil tussen de gemiddeldes betekent dat je kan concluderen dat FPPW'ers inderdaad slimmer zijn dan andere studenten

De drie stappen hierboven houden allemaal in dat je berekeningen zal moeten uitvoeren met je data. Dat kan je in principe met een doorsnee rekenmachine, maar je zal snel merken dat je op die manier wel extreem veel werk zal hebben en dat de kans op fouten ook erg groot is.

R dient om dat soort berekeningen makkelijker te maken. Je kan R beschouwen als een soort rekenmachine die specifiek is ontworpen voor onderzoekers die dingen willen berekenen zoals in de drie stappen hierboven. Met R kan dat snel, makkelijk en foutloos.

Hoe je zulke berekeningen kan uitvoeren in R? In het volgende stukje wordt een eerste tip van de sluier gelicht.

2.2 R is een programmeertaal

In veel wetenschappelijk onderzoek moet je dus berekeningen uitvoeren met data. R is speciaal ontworpen met dat doel voor ogen.

De manier waarop je dat doet in R is door instructies (of "commando's") te typen, die je computer opdragen om iets met de data te doen. Om ervoor te zorgen dat je computer doet wat jij wil, moet je die instructies geven in een taal die de computer kan begrijpen: de programmeertaal R.

In het voorbeeld van daarnet wilde je drie berekeningen laten uitvoeren.

de gemiddelde intelligentiescore berekenen van de FPPW'ers enerzijds en van de andere studenten anderzijds

het verschil tussen die gemiddeldes berekenen

uitzoeken of het verschil tussen de gemiddeldes betekent dat je kan concluderen dat FPPW'ers inderdaad slimmer zijn dan andere studenten

Elk van die drie berekeningen kan je laten doen door een stukje "code", dat wil zeggen een zinnetje in de programmeertaal R.

We zullen je hier nog niet meteen overdonderen met allerlei R-code. Dat zal je later wel nog genoeg leren. Maar om het toch een beetje concreet te maken: als de intelligentiescores van de FPPW-studenten in het object `scores_fppw` zitten, dan kan je het gemiddelde berekenen met de volgende code:

```
mean(scores_fppw)
```

Zo makkelijk kan het zijn!

2.3 “RStudio”?

R kan je vergelijken met een motor met wielen eronder: het is in essentie het enige wat je nodig hebt om van punt A naar punt B te raken. Het is “bruikbaar”.

“Bruikbaar” is nog niet hetzelfde als “gebruiksvriendelijk”. Je reis van punt A naar punt B zal geen gezellig ritje zijn als je enkel over een motor met wielen beschikt.

Dat is waarom je RStudio zal gebruiken. Het is een computerprogramma dat het werken met R een stuk makkelijker en aangenamer maakt. Het maakt het bijvoorbeeld makkelijker om overzicht te houden over je code en om stukken code op te slaan zodat je er later verder aan kan werken.

R en RStudio zal je afzonderlijk moeten downloaden en installeren op je computer. Hoe je dat kan doen leer je in het volgende hoofdstuk.

3 R en RStudio installeren

Om alle mogelijkheden van R te kunnen gebruiken moet je natuurlijk eerst enkele zaken installeren op je eigen computer. Hoe je dat kan doen zullen we in deze module stap voor stap tonen. Je zal zien dat het niet zo moeilijk is. Je computer doet heel veel werk automatisch.

Je zal eerst R moeten installeren. Daarna volgt de installatie van RStudio.

Het proces van installeren is lichtjes verschillend voor Windows en voor macOS (Apple). Klik op het pijltje naar rechts tot je op de juiste pagina terechtkomt of klik in de inhoudstafel aan de linkerkant.

3.1 Windows

3.1.1 R installeren voor Windows

In het filmpje hieronder wordt voorgetoond hoe je R kan downloaden en installeren voor Windows. De verschillende stappen die je moet doorlopen zijn:

Surf naar de website www.r-project.org

Klik door naar de pagina “Download”

Kies een mirror, bij voorkeur uit België (tenzij je je in een ander land bevindt)

Klik op de link “Download R for Windows”

Klik op “base”

Klik op de link “Download R 4.0.3 for Windows”. De cijfers kunnen verschillend zijn, afhankelijk van wat de laatste nieuwe versie is.

R is nu een programma op je computer dat je kan openen. Op de afbeelding hieronder zie je hoe dat eruit ziet: héél basic en ouderwets. Het lijkt niets meer dan een paar regeltjes tekst onder elkaar waar je ook zelf wat in kan typen. In principe zou je hiermee aan de slag kunnen gaan, maar dit is toch vooral iets voor de liefhebbers van extreme soberheid.

R is niet ontworpen als een gebruiksvriendelijk "product". Om het werken met R aangenamer én efficiënter te maken, zal je RStudio gebruiken. Zo meteen kom je te weten hoe je RStudio kan downloaden en installeren.

3.1.2 RStudio installeren voor Windows

R kan je zien als een motor met wielen eronder. Het is genoeg om je van punt A naar punt B te brengen. Maar met een motor alleen ben je niet zo veel. Je hebt ook de rest van de auto nodig om comfortabel te kunnen reizen. Dat is ongeveer wat RStudio doet: het maakt werken met R vlot, comfortabel en gebruiksvriendelijk.

In het filmpje verderop zie je hoe je het programma "RStudio" kan downloaden. De stappen die je moet doorlopen zijn:

Zoek via Google naar "download RStudio".

Het eerste zoekresultaat, van de website rstudio.com, is meteen de juiste link.

Kies voor de gratis versie en klik op "Download".

Op deze pagina zie je opnieuw een downloadknop. Daarmee start je het installatieproces.

Je zal enkele keren op "Ok" en "Volgende" moeten klikken en ten slotte op "Voltooien".

3.2 macOS

3.2.1 R installeren voor macOS

In de video hieronder tot minuut 2:09 kan je zien hoe het installeren van R verloopt op een computer met het besturingssysteem macOS. Alle laptops en vaste PC's van het merk Apple gebruiken dit besturingssysteem.

De stappen die je zal moeten doorlopen zijn:

Surf naar de website www.r-project.org

Klik door naar de pagina "Download"

Kies een mirror, bij voorkeur uit België (tenzij je je in een ander land bevindt)

Klik op de link "Download R for (Mac) OS X"

Klik op de link die eindigt op ".pkg" en sla het bestand op

Open het bestand dat je net hebt opgeslagen en doorloop het installatieproces.

In de video vanaf minuut 1:54 zag je al heel kort hoe R eruit ziet: héél erg basic en ouderwets. Het lijkt niets meer dan een paar regeltjes tekst onder elkaar waar je ook zelf wat in kan typen. In principe zou je hiermee aan de slag kunnen gaan, maar dit is toch vooral iets voor de liefhebbers van extreme soberheid.

R is niet ontworpen als een gebruiksvriendelijk "product". Om de ervaring van het werken met R aangenamer én efficiënter te maken, zal je RStudio gebruiken. Zo meteen leer je hoe je RStudio kan downloaden en installeren.

3.2.2 RStudio installeren voor macOS

In de video hieronder (van minuut 2:09 tot 3:53) zie je hoe de installatie van RStudio verloopt op een computer met het besturingssysteem macOS. De verschillende stappen die je moet doorlopen zijn:

Zoek de woorden “download RStudio” via Google

De eerste link in de lijst zou meteen de juiste moeten zijn. Deze link brengt je naar de website rstudio.com

Scroll een beetje naar beneden en kies RStudio Desktop Free. Klik op de blauwe downloadknop.

Klik op de downloadknop met de tekst “Download RStudio for Mac”.

Open het bestand dat je net hebt gedownload en doorloop het installatieproces op je computer.

Klaar voor gebruik!

Mooi zo! RStudio is nu een programma op je computer dat je kan openen, net zoals alle andere programma’s, zoals Word of Powerpoint. Telkens als je met R wil werken zal je nu RStudio openen.

3.3 Versies van R

R is software die voortdurend evolueert. Regelmatig wordt een nieuwe versie uitgebracht. Soms zijn er belangrijke verschillen tussen de versies, waardoor het belangrijk is om te weten in welke versie je aan het werken bent.

Je kan aflezen welke versie van R je gebruikt bij het opstarten van RStudio. In de console zie je een versienummer. In de afbeelding hieronder is het versie 4.0.4.

Je kan de versie ook te weten komen door het commando `version` uit te voeren. Bijna helemaal onderaan vind je vervolgens de huidige versie.

Is je versie van R verouderd? Je kan altijd de nieuwe versie downloaden van de website (www.r-project.org). De installatie gebeurt net zoals beschreven staat eerder in deze module. De oude versie die je voordien had zal gewoon worden vervangen.

4 Instructies geven in R

Tip: Het is een goed idee om zelf RStudio te openen en zoveel mogelijk actief mee te doen!

In dit hoofdstuk leer je hoe RStudio eruit ziet en krijg je een idee van wat je ermee kan doen. Het belangrijkste is dat je voor het eerst zal zien hoe je instructies (ook wel "commando's" genoemd) kan geven.

R is een programmeertaal. Dat betekent dat je commando's zal typen "ergens" in RStudio. Waar ergens?

Als je RStudio voor het eerst opent, ziet het er ongeveer uit zoals op de afbeelding hieronder. Er zijn enkele deelvensters.

Aan de linkerkant (in het lichtblauw hieronder) zie je de console. Dit is het hart van R. Het is de plaats waar jouw commando's geïnterpreteerd worden en waar je het "antwoord" van R of de "output" zal kunnen zien.

Het is mogelijk om commando's te typen in de console. Typ bijvoorbeeld de berekening 5+3 in de console en druk op Enter.

Je ziet dat het commando inderdaad wordt uitgevoerd: je krijgt output terug van R. Dit werkt wel, maar regel per regel commando's typen in de console is echt niet handig. Dat zal je heel snel merken wanneer je iets complexere code moet schrijven.

Een veel beter alternatief is werken met scripts. Een script open je door te klikken op het witgroene icoontje linksboven (zie hieronder) en vervolgens op "R Script" te klikken. Je kan het ook helemaal bovenaan via het menu File > New File > R Script.

Er opent een nieuw deelvenster en de console wordt wat kleiner.

Zo'n script is een eenvoudig tekstbestand dat in RStudio wordt geopend. Code die je in het script plaatst, kan je nu makkelijk laten uitvoeren door de console. Dit wordt gedemonstreerd in het filmpje. Uitleg in tekstvorm vind je onder het filmpje.

Om één regel code uit te voeren klik je met je muis in de regel code die je wil uitvoeren. Druk nu tegelijk op de knoppen CTRL en Enter op je toetsenbord (voor Mac: Command en Enter). Op deze manier wordt één regel van je code uitgevoerd.

Wil je meerdere regels code in één keer uitvoeren, selecteer dan alle gewenste regels door te slepen met je muis en klik vervolgens op CTRL en Enter (voor Mac: Command en Enter).

Voila!

Wat is nu eigenlijk het voordeel van te werken met een script? De code zelf verdwijnt niet van zodra je ze uitvoert in de console. Daardoor kan je die bv. verschillende keren na elkaar uitvoeren en aanpassen. Het voordeel van zulke scripts is nu misschien nog niet 100% duidelijk, maar geloof me, het is véél handiger dan code rechtstreeks in de console te typen, zeker wanneer de code wat langer en ingewikkelder is.

Proficiat!

Je kan nu commando's ingeven in R. De wereld ligt aan je voeten!

5 Data in R

R dient om met data te werken. Hoe ziet die data er dan uit? Hoe kan je ermee aan de slag?

5.1 Een eerste object maken

In de video wordt getoond hoe je je eerste object in R kan maken. Onder de video vind je de bijhorende uitleg in tekstvorm. Probeer het zeker ook zelf eens uit in RStudio!

In R zal je heel vaak met objecten werken. Laten we eerst kijken naar hoe je zulke objecten kan creëren.

Je kan gewoon een naam verzinnen voor het object dat je wil aanmaken en die naam in R typen. Je zal ook meteen iets "in" je object willen stoppen. Dat kan je doen met het symbool `<-`. Dit symbool noemen we de toekenningsoperator. Het bestaat uit twee tekens, namelijk `<` en `-`.

Je code ziet er dan bijvoorbeeld als volgt uit:

```
mijn_leeftijd <- 19
```

Ter herinnering: code uitvoeren doe je door tegelijk CTRL en Enter (Windows) of Command en Enter (Mac) in te tikken.

Wanneer je deze code uitvoert, maak je een object genaamd `mijn_leeftijd`. Dit object bevat nu één waarde, namelijk 19. In RStudio zie je dit object nu ook verschijnen in de lijst rechtsboven.

Je kan steeds opvragen wat de inhoud van een object is door de naam van het object te typen en deze code uit te voeren:

```
mijn_leeftijd
```

In de console zie je verschijnen:

```
[1] 19
```

Gelukt! Ben je al van je stoel gevallen van de mogelijkheden van R? Neen? Ok, dat is begrijpelijk, maar je kan er natuurlijk veel meer mee. In een volgend voorbeeld gaan we een stapje verder: een object met de naam `leeftijden` aanmaken dat meerdere gegevens bevat. Dit zijn bijvoorbeeld leeftijden van vijf leden van een basketbalteam.

Zo'n reeks van gegevens wordt ook een vector genoemd. Die kan je maken met de functie `c()`, zoals in dit voorbeeld:

```
leeftijden <- c(19, 22, 18, 22, 24)
```

De functie `c()` zorgt ervoor dat alles wat tussen haakjes staat als één samenhangend object wordt beschouwd door R.

Ook de inhoud van dit object kan je makkelijk terug oproepen.

```
leeftijden
```

```
[1] 19 22 18 22 24
```

Het object `leeftijden` bevat inderdaad niet één getal, maar een reeks getallen. Deze getallen staan trouwens in een vaste volgorde.

Let erop dat R hoofdlettergevoelig is. Wanneer je het object `leeftijden` wil oproepen, mag je het dus niet schrijven met hoofdletter L. Doe je dat toch, dan krijg je een foutmelding `object 'Leeftijden' not found`. Dat komt omdat het object `Leeftijden` gewoon niet bestaat.

5.2 Een vector met namen

Een vector kan ook gegevens van het nominale meetniveau bevatten. Hoe je zo'n vector kan maken zie je in het filmpje. Eronder vind je de bijhorende uitleg in tekstvorm.

De term "data" of "gegevens" betekent niet automatisch dat het om kwantitatieve gegevens gaat. Het is even goed mogelijk om een vector met gegevens van het nominale meetniveau aan te maken. In het voorbeeld van het basketbalteam zou het bijvoorbeeld om de voornamen van de basketbalspelers kunnen gaan. Het enige verschil met de vector die hun leeftijden bevat, is dat de gegevens nu tussen aanhalingstekens moeten staan.

```
voornamen <- c("Laura", "Danira", "Stefanie", "Leyla", "Chelsy")
```

Het is belangrijk om te beseffen dat de vijf namen een vaste plaats hebben gekregen in de vector (net zoals eerder in de vector `leeftijden`). Laura staat op de eerste plaats, Danira op de tweede plaats, enzoverder.

Het is ook mogelijk om de namen afzonderlijk op te roepen. Dat kan door de naam van de vector te combineren met vierkante haakjes [ ], om bijvoorbeeld de derde voornaam uit de vector op te roepen:

```
voornamen[3]
```

```
[1] "Stefanie"
```

Je hebt dus een selectie gemaakt van enkel het derde element uit de vector. Wat zou volgens jou de output zijn als je het volgende commando zou ingeven?

```
voornamen[5]
```

Klik hier om het antwoord te zien

Het 5e element uit de vector voornamen is “Chelsy”.

En wat met het volgende commando? Begrijp je welke selectie hier wordt gemaakt? Voer het commando zelf uit in RStudio en vergelijk de output met de vector `voornamen`.

```
voornamen[1:3]
```

Klik hier om het antwoord te zien

Met dit commando selecteer je de elementen 1 tot en met 3 uit de vector. Hier zijn dat dus “Laura”, “Danira” en “Stefanie”.

5.3 Rugnummers

Verder wil je ook de rugnummers van de speelsters bijhouden. Dat werkt opnieuw zoals voorheen: met de functie `c()` stop je de rugnummers in een vector.

Even goed nadenken: wat is het meetniveau van de variabele rugnummers? Rugnummers zijn getallen, dus je zou denken dat dit precies werkt zoals in het voorbeeld met de leeftijden. Maar rugnummers zijn verschillend van leeftijden van mensen. Het zijn variabelen van het nominale meetniveau.

Als je (verkeerdelijk) de volgende code zou invoeren, dan denkt R dat deze vector gegevens bevat van interval- of rationiveau.

```
rugnummers_fout <- c(4, 12, 7, 76, 33)
```

Waarom is dat nu zo verkeerd? Omdat het lijkt alsof je uit deze vector informatie zou kunnen halen die er eigenlijk niet in zit. Bijvoorbeeld, uit de vector `rugnummers_fout` zou je het verschil kunnen berekenen tussen rugnummers van verschillende speelsters. Of je zou het gemiddelde rugnummer kunnen berekenen. Die berekeningen hebben echter geen betekenis.

Om dergelijke verwarring te voorkomen moet je de rugnummers in een vector stoppen zoals je ook de voornamen van de speelsters in een vector hebt opgeslagen: met dubbele aanhalingstekens. Op die manier weet R dat dit een nominale variabele is.

```
rugnummers <- c("4", "12", "7", "76", "33")
```

Nu zal R niet toelaten dat je bijvoorbeeld het verschil berekent tussen twee rugnummers. Je zal een foutmelding krijgen. En dat is terecht.

```
rugnummers[4]-rugnummers[2]
```

```
Error in rugnummers[4] - rugnummers[2] :
  non-numeric argument to binary operator
```

5.4 Een dataframe creëren

Bekijk de video voor een demonstratie. Onder de video vind je de uitleg in tekstvorm.

Om deze module af te ronden zullen we de vectoren die we hebben aangemaakt, samenbrengen in een dataframe. Wat is dat nu weer?

Een dataframe is niet meer dan een verzameling van vectoren van dezelfde lengte, die ook inhoudelijk bij elkaar horen. De vectoren die je eerder al hebt aangemaakt, kunnen als voorbeeld dienen. Voorlopig zijn dat drie losse vectoren. R weet niet dat ze bij elkaar horen.

```
voornamen <- c("Laura", "Danira", "Stefanie", "Leyla", "Chelsy")

leeftijden <- c(19, 22, 18, 22, 24)

rugnummers <- c("4", "12", "7", "76", "33")
```

Het dataframe maken doe je met de functie `data.frame()`. De verschillende vectoren die je in het dataframe wil stoppen zet je simpelweg tussen de haakjes. Je kan het object meteen ook een naam geven.

```
mijn_team <- data.frame(voornamen, leeftijden, rugnummers)
```

De functie `data.frame()` is een manier om aan R te zeggen dat de eerste waarde uit de vector voornamen hoort bij de eerste waarde uit de vector `leeftijd` en bij de eerste waarde uit de vector `rugnummers`. Hetzelfde geldt voor de tweede, derde,... waarde uit alle drie de vectoren. Natuurlijk is het heel belangrijk dat de volgorde van de data in de drie vectoren correct is: Laura is 19 jaar oud en heeft rugnummer 4, Danira is 22 en heeft rugnummer 12, enzovoort. Als dat niet zo is, dan zal je dataframe niet meer zijn dan een verwarrende hoop gegevens zonder structuur.

Vergeet niet dat R hoofdlettergevoelig is. Schrijf de naam van de functie `data.frame()` dus niet als `Data.frame()`. Anders krijg je een foutmelding:

```
mijn_team <- Data.frame(voornamen, leeftijden, rugnummers)
```

```
Error in Data.frame(voornamen, leeftijden, rugnummers) :
  could not find function "Data.frame"
```

Aan de hand van de zelfgekozen naam `mijn_team` kan je het object makkelijk weer oproepen:

```
mijn_team
```

```
  voornamen leeftijden rugnummers
1     Laura         19          4
2    Danira         22         12
3  Stefanie         18          7
4     Leyla         22         76
5    Chelsy         24         33
```

Het dataframe is dus een grotere structuur die meerdere vectoren bevat. Met het symbool $ kan je de afzonderlijke vectoren in het dataframe oproepen. Dat gaat bijvoorbeeld zo:

```
mijn_team$voornamen
```

```
[1] "Laura"    "Danira"   "Stefanie" "Leyla"    "Chelsy"
```

Uit deze vector kan je dan weer de individuele elementen selecteren met vierkante haakjes, zoals je eerder al hebt gedaan:

```
mijn_team$voornamen[4]
```

```
[1] "Leyla"
```

Met dat laatste commando zeg je eigenlijk "geef me het vierde element uit de vector `voornamen` uit het dataframe `mijn_team`".

Wat zou het volgende commando opleveren als output? Denk even na en voer het commando dan uit in RStudio.

```
mijn_team$leeftijden[1]
```

Klik hier om het antwoord te zien

```
[1] 19
```

Het dataframe is dus een grotere structuur met verschillende vectoren. Het is een soort rechthoekig "blok", met rijen en kolommen in een vaste volgorde. Je kan uit dit dataframe ook op de volgende manier onderdelen selecteren:

```
mijn_team[2,3]
```

Met die code selecteer je het element op de tweede rij, uit de derde kolom. Kijk even terug naar het dataframe en zoek op welk element dit is. Voer de code uit. Klopt de output met wat je dacht?

Klik hier om het antwoord te zien

```
[1] "12"
```

5.5 Slot

In deze module heb je als een echte tovenaar nieuwe objecten gecreëerd uit het niets!

Je hebt die objecten bovendien een inhoud gegeven. Je weet dat die objecten ook vectoren kunnen zijn. Je kan die zelf aanmaken met de functie `c()`.

Ten slotte kan je verschillende vectoren verzamelen in een dataframe met -niet zo verrassende functie `data.frame()`.

6 Eerste berekeningen in R

Misschien herinner je je nog dat in het begin van dit leerpad werd gezegd dat R dient om allerlei berekeningen te maken. Eindelijk zijn we zover om dit voor het eerst te demonstreren!

6.1 Bewerkingen met getallen

R kan je onder meer gebruiken voor eenvoudige wiskundige bewerkingen. Bekijk de video voor een demonstratie. Onder de video vind je de bijhorende uitleg in tekstvorm.

R is een programmeertaal. Dat betekent dat het een taal is waarmee je instructies geeft aan je computer.

Niet alle instructies geven het gewenste resultaat. R kan geen boterhammen voor je smeren of je kot opkuisen, dus het heeft weinig zin om dat te vragen aan R.

Wat kan je dan wel met R? R is een hulpmiddel dat je helpt om onderzoeksvragen te beantwoorden. En dus bevat R allerlei mogelijkheden (echt serieus, héél veel mogelijkheden!) om berekeningen te maken die daarbij helpen.

Een aantal van de commando's om zulke berekeningen te maken zal je leren kennen in de loop van je vakken. Voor deze introductie beginnen we bij het begin: eenvoudige wiskundige bewerkingen als optellen en aftrekken, vermenigvuldigen en delen. Er wordt soms gezegd dat R een veredelde rekenmachine is. En inderdaad, je kan R gebruiken als een gewone rekenmachine.

Tip: probeer de commando's die je op deze pagina tegenkomt ook zelf eens uit in RStudio. Dat is belangrijk om het allemaal wat in de vingers te krijgen.

6.1.1 Getallen

Je kan getallen optellen.

```
4+3
```

```
[1] 7
```

Het maakt voor R trouwens niet uit of je spaties tussen de tekens zet. Om de leesbaarheid van je code te verbeteren is het wel een goed idee om regelmatig spaties te gebruiken.

```
4 + 3
```

```
[1] 7
```

```
4          +          3
```

```
[1] 7
```

Je kan een getal van een ander getal aftrekken.

```
5-2
```

```
[1] 3
```

Twee getallen vermenigvuldigen kan ook. Je gebruikt daarvoor een asterisk `*`.

```
6*3
```

```
[1] 18
```

Je voelt het al komen: je kan ook een getal delen door een ander getal. Merk op dat de output "1 komma 5" geschreven wordt met een punt, dus niet "1,5". Ook als je zelf commando's schrijft, moet je een punt gebruiken om kommagetallen te typen.

```
12/8
```

```
[1] 1.5
```

En ook getallen tot een macht verheffen kan natuurlijk. 2 tot de 5e macht verheffen doe je zo:

```
2^5
```

```
[1] 32
```

Deze bewerkingen kan je combineren, eventueel ook met haakjes erbij. De gebruikelijke volgorde van bewerkingen is hier van toepassing.

```
10 / 6 - 7
```

```
[1] -5.333333
```

```r
10 / (6 - 7)
```

```
[1] -10
```

```r
(11 - 4) * 4 / 2 + (1 / 3)
```

```
[1] 14.33333
```

6.1.2 Objecten

Al onder de indruk van de kracht van R? Waarschijnlijk niet. Dat was natuurlijk nog heel basic. Wat kan er verder nog? Het is ook mogelijk om te rekenen met objecten die getallen bevatten, bijvoorbeeld zo:

```r
getal1 <- 8
getal2 <- 10
```

Het object `getal1` bevat de waarde 8 terwijl object `getal2` de waarde 10 bevat. Net als voorheen kan je wiskundige bewerkingen uitvoeren, maar nu door de objectnamen te gebruiken in plaats van de getallen zelf.

```r
getal1 + getal2
```

```
[1] 18
```

```
(getal2 / getal1) - getal1
```

```
[1] -6.75
```

```
20 - getal1
```

```
[1] 12
```

That's it?

Allemaal goed en wel, maar tot hier toe heb je nog geen duidelijk antwoord op de vraag waarom R handiger is dan een gewone rekenmachine. Zo meteen zal dat duidelijker beginnen worden.

6.2 Bewerkingen met vectoren

Bekijk het filmpje om te zien hoe je wiskundige bewerkingen kan uitvoeren met vectoren in R. Onder de video vind je de bijhorende uitleg in tekstvorm.

In R kan je rekenen met ganse vectoren. Nu zal je pas voor het eerst beginnen merken hoe handig R eigenlijk is.

6.2.1 Voorbeeld 1

Bijvoorbeeld, je wil de evolutie van het gewicht van baby's onderzoeken.

Je weegt baby's bij de geboorte. Die gegevens (in kilogram) sla je op in een vector:

```
gewicht_geboorte <- c(3.411, 2.965, 3.215, 3.009, 3.228)
```

Vervolgens meet je dezelfde baby's opnieuw, 6 maanden na de geboorte. Ook deze gegevens sla je – in dezelfde volgorde! – op in een vector.

```
gewicht_6mnd <- c(5.033, 5.011, 5.501, 5.120, 4.963)
```

Een logische vraag zou nu kunnen zijn hoeveel elke baby is bijgekomen 6 maanden na de geboorte. Je zou natuurlijk telkens de waarden voor het gewicht van de individuele baby's kunnen aftrekken:

$5.033 - 3.411$ voor de eerste baby,

$5.011 - 2.965$ voor de tweede baby,

enzovoort voor de andere baby's.

Dat is natuurlijk heel tijdrovend en je loopt bovendien het risico dat je fouten maakt. Voor een vijftal baby's valt het nog mee, maar in echt onderzoek zal je vaak veel meer dan 5 metingen uitvoeren.

In R hoef je gelukkig niet zoveel werk te doen. In plaats daarvan kan je eenvoudigweg de twee vectoren van elkaar aftrekken. Dit verschil steek je in een nieuw object, bijvoorbeeld met de naam `gewicht_evolutie`.

```
gewicht_evolutie <- gewicht_6mnd - gewicht_geboorte
```

Wanneer je dit nieuwe object oproept, zie je een vector die het verschil in gewicht bevat van elke baby tussen de geboorte en 6 maanden na de geboorte. Veel eenvoudiger, niet?

```
gewicht_evolutie
```

```
[1] 1.622 2.046 2.286 2.111 1.735
```

R is dan wel een "soort rekenmachine", dit voorbeeld demonstreert voor het eerst dat R toch een pak meer kan dan de doorsnee rekenmachine op je smartphone.

6.2.2 Voorbeeld 2

Laten we nog een tweede voorbeeld bekijken van een berekening met vectoren: Je kan een vector vermenigvuldigen met een getal. Dit kan bijvoorbeeld handig zijn om de gegevens te veranderen van eenheid. Bijvoorbeeld, je beschikt over gegevens in meter, maar je wil die liever in centimeter.

```
data_meter <- c(1.55, 1.79, 1.99, 1.91, 1.65, 1.81, 1.49, 1.74, 1.79)
```

```
data_centimeter <- 100 * data_meter

data_centimeter
```

```
[1] 155 179 199 191 165 181 149 174 179
```

Natuurlijk kan je ook vectoren optellen, vermenigvuldigen, delen,...

7 Eenvoudige functies

Je weet al hoe je allerlei berekeningen kan uitvoeren, zowel met getallen als met vectoren. Daarmee kan je heel veel gedaan krijgen, maar sommige berekeningen zouden wel eens heel omslachtig kunnen zijn. Gelukkig bestaan er ook functies. Dat zijn hulpmiddelen die je leven (in R) veel makkelijker maken.

7.1 Functies voor vectoren

In het kader van een onderzoek heb je 10 mensen bevraagd en gegevens verzameld over hun persoonlijke maandelijkse inkomen (in euro). Je stopt die gegevens in een vector `inkomen`.

```
inkomen <- c(1850, 1722, 2319, 4480, 1398, 849, 1679, 1612, 3938, 2405)
```

Nu wil je graag het gemiddelde berekenen van die 10 inkomens. Dat kan je natuurlijk doen door alle inkomens op te tellen en te delen door 10:

```
(1850 + 1722 + 2319 + 4480 + 1398 + 849 + 1679 + 1612 + 3938 + 2405) / 10
```

```
[1] 2225.2
```

Je krijgt wel het juiste antwoord, maar zelfs voor 10 inkomens is het al een hele klus om op deze manier te werken. In echt onderzoek zal je vaak nog veel meer dan 10 observaties maken. Voor 100 of 1000 inkomens zal je veel te veel werk hebben als je de berekening van het gemiddelde op de bovenstaande manier aanpakt. Je zal dus een betere methode nodig hebben.

Hiervoor kan je een functie gebruiken. Een functie kan je zien als een verkorte manier om een gewenst resultaat te krijgen. Bijna elke functie heeft een input nodig. De functie doet iets met die input en geeft je een output terug.

In dit voorbeeld wens je het gemiddelde te berekenen van een reeks gegevens. R weet niet uit zichzelf van welke gegevens jij het gemiddelde wil weten. Jij moet dat aan R vertellen. Hier zijn de gegevens in de vector `inkomen` de input. Je geeft die input aan de functie door die tussen de haakjes te plaatsen, zoals hieronder. De vector `inkomen` is nu een "argument" van de functie `mean()`. De output is het gemiddelde.

```
mean(inkomen)
```

```
[1] 2225.2
```

Op een heel gelijkaardige manier kan je nog allerlei andere zaken te weten komen over de vector `inkomen`. Je geeft telkens de vector als argument aan een bepaalde functie.

De mediaan berekenen:

```
median(inkomen)
```

```
[1] 1786
```

De som van alle waarden in de vector:

```
sum(inkomen)
```

```
[1] 22252
```

De grootste waarde uit de vector:

```
max(inkomen)
```

```
[1] 4480
```

De kleinste waarde uit de vector:

```
min(inkomen)
```

```
[1] 849
```

En het aantal waarden in de vector:

```
length(inkomen)
```

```
[1] 10
```

Je kan met deze functies ook onmiddellijk berekeningen maken. Voer bijvoorbeeld eens de volgende code uit:

```
sum(inkomen) / length(inkomen)
```

```
[1] 2225.2
```

Wat is de output van deze code? Herken je dit getal? Begrijp je wat er is gebeurd?

Klik hier om het antwoord te zien

Dit getal is het gemiddelde dat je eerder al hebt berekend met de functie `mean()`. Het is logisch dat dit dezelfde waarde is, want met sum(inkomen) bereken je de som van alle waarden in de vector. Dat totaal deel je vervolgens door de lengte van de vector, met andere woorden door het aantal observaties. Dit zijn gewoon de stappen in het berekenen van het gemiddelde.

7.2 Functies voor dataframes

Functies kunnen ook gebruikt worden om info te halen uit een dataframe in plaats van een vector. Misschien heb je behalve het inkomen ook gevraagd naar hoeveel andere gezinsleden in het gezin leven van de bevraagde personen. Die gegevens zitten in de vector **gezinsleden**.

```
inkomen <- c(1850, 1722, 2319, 4480, 1398, 849, 1679, 1612, 3938, 2405)

gezinsleden <- c(2, 3, 0, 1, 1, 0, 2, 2, 1, 0)
```

De vectoren **gezinsleden** en **inkomen** kan je samenvoegen tot een dataframe, zoals je eerder al hebt gedaan:

```
mijn_data <- data.frame(inkomen, gezinsleden)
```

De functie **dim()** vertelt je hoeveel observaties en hoeveel variabelen er in het dataframe zitten. Nu geef je het dataframe als argument aan de functie:

```
dim(mijn_data)
```

```
[1] 10  2
```

De output die je kreeg, leert je dat er 10 observaties zijn van 2 variabelen. Is dat wat je had verwacht?

Er bestaan nog andere handige functies zoals **dim()** die dienen om overzicht te geven over je data, zonder er berekeningen mee uit te voeren.

Om de eerste 6 observaties uit een dataframe te bekijken kan je de functie **head()** gebruiken.

```
head(mijn_data)
```

```
  inkomen gezinsleden
1    1850           2
2    1722           3
3    2319           0
4    4480           1
5    1398           1
6     849           0
```

Misschien wil je enkel de eerste 3 rijen van het dataframe zien, in plaats van de eerste 6 rijen. Dat kan je aanpassen in je commando. Je gebruikt dezelfde functie maar geeft er een extra argument aan.

```
head(mijn_data, n=3)
```

```
  inkomen gezinsleden
1    1850           2
2    1722           3
3    2319           0
```

Met het argument `n` geef je aan hoeveel rijen van het dataframe je wil zien. Standaard zijn het er 6, maar je kan het dus aanpassen. Als je de argumenten in de juiste volgorde zet, mag je het deel `n=` ook weglaten. Het eerstvolgende stukje code zal dus ook werken, maar het tweede niet:

```
head(mijn_data, 3)
```

```
  inkomen gezinsleden
1    1850           2
2    1722           3
3    2319           0
```

```
head(3, mijn_data)
```

```
Error in checkHT(n, dx <- dim(x)) :
  invalid 'n' - must have length one when dim(x) is NULL, got 2
```

Als je vragen hebt over een functie en/of de argumenten die je eraan kan geven, dan kan je altijd meer informatie krijgen door de functie te laten voorafgaan door een vraagteken. Je ziet dan rechtsonder in RStudio een pagina verschijnen met uitleg. Daarin staat bijvoorbeeld welke argumenten allemaal aan een functie kunnen worden gegeven.

```
?head()
```

Een interessante functie om overzicht te krijgen over je data is `str()`. Deze functie geeft je een lijst met alle variabelen die in het dataframe zitten, en meteen zie je erbij om welk type variabele het gaat.

```
str(mijn_data)
```

```
'data.frame':   10 obs. of  2 variables:
 $ inkomen    : num  1850 1722 2319 4480 1398 ...
 $ gezinsleden: num  2 3 0 1 1 0 2 2 1 0
```

Het woordje `num` naast de variabelen `inkomen` en `gezinsleden` geeft aan dat het allebei numerieke variabelen zijn.

Leren werken met functies is misschien niet altijd makkelijk in het begin, maar het went snel en het kan zelfs leuk zijn. Onthoud vooral dat functies dienen om je leven makkelijker te maken: minder werk, minder code typen, minder kans op fouten,…

8 Je werk opslaan

Je hebt net, na 20 keer proberen, een moeilijke baas verslagen in een computerspel. Of je hebt zojuist uren gezwoegd aan een presentatie of paper voor school. Wat wil je in zo'n situatie zeker niet voorhebben? Juist, dat er iets fout loopt met je computer en dat je al je voortgang kwijt bent.

In RStudio is dat natuurlijk ook zo. Daarom is het belangrijk om je werk regelmatig op te slaan. Zo kan je de volgende keer gewoon verdergaan waar je was gestopt. Hier zal je leren hoe je je werk kan opslaan in RStudio.

De uitleg is lichtjes verschillend voor Windows en macOS, dus kies de juiste pagina in het overzicht aan de linkerkant of klik op het pijltje naar rechts tot je op de juiste pagina terechtkomt. Onderaan elke pagina vind je een samenvatting.

8.1 Windows

8.1.1 Een script opslaan

In de afbeelding hieronder vind je een script waar je lang aan hebt gezwoegd. Tijd om je voortgang eens op te slaan. Hoe kan je dat doen? Je hebt waarschijnlijk wel al eens een Word-document of een Powerpoint-presentatie gemaakt en opgeslagen. Een script opslaan werkt op dezelfde manier.

Een script opslaan doe je met het blauwe icoontje aan de bovenkant van het script. Op de afbeelding hieronder staat het aangeduid met de rode pijl. Een andere mogelijkheid is tegelijk de toetsen CTRL en S indrukken.

Kies een map op je computer waar je het script wil opslaan. Hier kiezen we voor C:/Gebruikers/michi/Documenten/UGent. Geef je script een herkenbare naam, zoals hier "basketbal.R". De extensie ".R" wijst erop dat het om een R-script gaat.

Als je succesvol hebt opgeslagen, wordt het icoontje bijna onzichtbaar, zoals hieronder. Dat wil zeggen dat al je werk is opgeslagen.

Als je nu opnieuw een wijziging aanbrengt in je R-code, dan merk je dat het icoontje opnieuw zichtbaar wordt. Logisch, want er zijn nieuwe wijzigingen in je code die kunnen worden opgeslagen.

Als je nu opnieuw opslaat, dan wordt het vorige bestand gewoon overschreven. Wil je je script toch op een andere locatie opslaan, dan kan dat via File > Save as…

8.1.2 Een dataframe opslaan

Bekijk de onderstaande video. Onder de video vind je dezelfde uitleg in tekstvorm.

Behalve een script kan je ook een dataframe opslaan. Je gebruikt daarvoor de functie `write.csv()`.

Bekijk de onderstaande code. Er worden drie vectoren aangemaakt: `voornamen`, `leeftijden` en `rugnummers`.

```
voornamen <- c("Laura", "Danira", "Stefanie", "Leyla", "Chelsy")

leeftijden <- c(19, 22, 18, 22, 24)

rugnummers <- c("4", "12", "7", "76", "33")
```

Deze vectoren worden nu samengebracht in een dataframe genaamd `mijn_team`:

```
mijn_team <- data.frame(voornamen, leeftijden, rugnummers)
```

Je hebt dus data van drie variabelen samengebracht in één object: `mijn_team`. Nu wil je dit object in zijn geheel opslaan op je computer. Dat kan met volgende code:

```
write.csv(mijn_team, file = "mijn_eerste_dataframe.csv", row.names = FALSE)
```

Deze functie maakt een csv-bestand[1] aan op je computer. Dat is een type bestand dat perfect geschikt is om onderzoeksgegevens te bevatten.

Het eerste argument van de functie write.csv() bepaalt wat er in het csv-bestand moet terechtkomen.

Het tweede argument legt de naam vast van het nieuwe bestand.

Het laatste argument is voorlopig minder belangrijk.

[1] 'csv' staat voor 'comma-separated value'.

De vraag is: waar is dat csv-bestand nu eigenlijk opgeslagen? Als je het later opnieuw nodig hebt, in welke map op je computer moet je dan gaan zoeken?

Dat kan je achterhalen met de functie `getwd()`. Die functie geeft je de huidige werkmap of working directory. Dat is een map op je computer die gelinkt is met je activiteiten in RStudio. Heel wat functies maken gebruik van die map om

ofwel zaken op te slaan vanuit RStudio: bijvoorbeeld een dataframe opslaan met de functie `write.csv()`

ofwel omgekeerd, bestanden in te lezen naar RStudio: bijvoorbeeld een dataframe inlezen met de functie `read.csv()`

Je huidige werkmap is bijvoorbeeld:

```
getwd()
```

```
[1] "C:/Gebruikers/michi/Documenten"
```

Dat is misschien niet de map waar je het dataframe wil bewaren[2]. De vraag die waarschijnlijk op je lippen brandt (of niet, soms?) is: Hoe kan je de working directory aanpassen? Dat bekijken we in het volgende onderdeel.

8.1.3 Een `working directory` kiezen

De functie `write.csv()` slaat je dataframe op in de huidige working directory. Om te achterhalen welke map op dit moment je working directory is, kan je volgend commando gebruiken:

```
getwd()
```

[2]Het is trouwens ook niet noodzakelijk dezelfde locatie als waar je eerder het script hebt opgeslagen, namelijk C:/Gebruikers/michi/Documenten/UGent

Je krijgt dan een locatie op je computer als output, bijvoorbeeld:

```
[1] "C:/Gebruikers/michi/Documenten"
```

RStudio heeft die map automatisch gekozen, maar misschien wil je je dataframe helemaal niet op die plaats opslaan. Dan moet je de working directory veranderen. De makkelijkste manier om dat te doen is door bovenaan te klikken op Session. Ga vervolgens naar Set Working Directory en klik op Choose Directory, zoals je in de afbeelding hieronder kan zien.

In het venster dat verschijnt kies je een bestaande map, of maak je een nieuwe, die vanaf nu als working directory zal dienen.

Dubbelcheck even of de working directory inderdaad aangepast is:

```
getwd()
```

```
[1] "C:/Gebruikers/michi/Documenten/Statistiek/Werken met R"
```

Gelukt! Dat is de makkelijkste manier om je working directory te veranderen, maar er is een vervelend nadeel aan verbonden. Als je later opnieuw het script opent in R, dan zal de working directory opnieuw veranderd zijn naar de oude, automatisch gekozen map (in ons geval was dat C:/Gebruikers/michi/Documenten).

Ambetant! Wat nu? Is daar een goede oplossing voor? Dat leer je in het volgende onderdeel.

8.1.4 De functie `setwd()` gebruiken

Pro tip: wanneer je een nieuw script start, volg dan altijd eerst de onderstaande 5 stappen.

1. Kies een working directory via Session > Set Working Directory > Choose Directory.

2. Voer het commando `getwd()` uit.

3. In de console verschijnt nu de locatie van je working directory. Selecteer die locatie, rechtsklik erop en kopieer.

4. Typ nu helemaal bovenaan in je script `setwd()`

5. Plaats tussen de haakjes van de functie `setwd()` de locatie die je in stap 3 hebt gekopieerd. Vergeet niet om de locatie ook tussen aanhalingstekens te zetten. Sla je script op.

Het kan een beetje omslachtig lijken, maar je raakt het snel gewoon om altijd een script te beginnen met de functie `setwd()`.

Het voordeel aan deze manier van werken is dat je script nu bijna vanzelf gelinkt zal zijn aan de gewenste map op je computer. Het enige wat je moet doen is de regel code

```
setwd("C:/Users/MijnNaam/Documents/Statistiek/Werken met R")
```

uitvoeren wanneer je het script opent om erin verder te werken. Wanneer je nu bijvoorbeeld een dataframe wil opslaan met de functie `write.csv()` dan zal dit bestand automatisch in de working directory terechtkomen. Er zijn nog heel wat andere functies behalve `write.csv()` waarvoor dat handig is.

Nog een laatste opmerking: het is meestal een goed idee om je script zelf ook op te slaan in diezelfde map, de working directory. Om het allemaal simpel en overzichtelijk te houden…

8.1.5 Samengevat

Je werk opslaan in R: Easy peasy, right? Maar mocht het toch een beetje verwarrend zijn, vatten we hier nog eens kort samen.

Hoe kan je een script opslaan?

Dit werkt gelijkaardig aan bijvoorbeeld een Word-document opslaan: wanneer je een nieuw script voor de eerste keer opslaat, moet je een map kiezen. Vanaf dan kan je gewoon opslaan met het blauwe diskette-icoontje of met de toetsen CTRL + S. Dit zal altijd op dezelfde locatie gebeuren en het vorige bestand overschrijven. (Wil je je script toch op een andere locatie opslaan, dan kan je onder File kiezen voor Save as …)

Een dataframe opslaan met de functie `write.csv()`

Als je onderzoeksdata in een dataframe hebt samengebracht, kan het handig zijn om dat object (het dataframe dus) op je computer op te slaan. Dat kan je in je script doen met de functie `write.csv()`, bijvoorbeeld zo:

```
write.csv(mijn_df, file = "mijn_eerste_dataframe.csv", row.names = FALSE)
```

Je hebt hier geen locatie (map op je computer) gekozen om het csv-bestand in te creëren. Het bestand wordt automatisch aangemaakt in je working directory (zie hieronder).

Een working directory vastleggen met de functie `setwd()`

Een working directory of werkmap is een locatie op je computer die gelinkt is met je activiteiten in RStudio. Heel wat functies maken gebruik van die map om

ofwel zaken op te slaan vanuit RStudio: bijvoorbeeld een dataframe opslaan met de functie `write.csv()`

ofwel omgekeerd, bestanden in te lezen naar RStudio: bijvoorbeeld een dataframe inlezen met de functie `read.csv()`

Een working directory vastleggen kan je met de functie `setwd()`, bijvoorbeeld:

```
setwd("C:/Users/MijnNaam/Documents/Statistiek/Werken met R")
```

Het is een goede gewoonte om een script altijd te starten met de functie `setwd()`.

8.2 macOS

8.2.1 Een script opslaan

In de afbeelding hieronder vind je een script waar je lang aan hebt gezwoegd. Tijd om je voortgang eens op te slaan. Hoe kan je dat doen?

Een script opslaan doe je met het blauwe icoontje aan de bovenkant van het script. Op de afbeelding hieronder staat het aangeduid met de rode pijl. Het kan ook via File > Save.

Kies een map op je computer waar je het script wil opslaan. Hier kiezen we bijvoorbeeld voor /Users/jouwnaam/STATISTIEK. Geef je script een herkenbare naam.

Als je succesvol hebt opgeslagen, wordt het icoontje bijna onzichtbaar, zoals hieronder. Dat wil zeggen dat al je werk opgeslagen is.

Als je nu opnieuw een wijziging aanbrengt in je R-code, dan merk je dat het icoontje opnieuw zichtbaar wordt. Logisch, want er zijn nieuwe wijzigingen in je code die kunnen worden opgeslagen.

8.2.2 Een dataframe opslaan

Bekijk de onderstaande video. Onder de video vind je dezelfde uitleg in tekstvorm.

Behalve een script kan je ook een dataframe opslaan. Je gebruikt daarvoor de functie `write.csv()`.

Bekijk de onderstaande code. Er worden drie vectoren aangemaakt: voornamen, leeftijden en rugnummers.

```
voornamen <- c("Laura", "Danira", "Stefanie", "Leyla", "Chelsy")

leeftijden <- c(19, 22, 18, 22, 24)

rugnummers <- c("4", "12", "7", "76", "33")
```

Deze vectoren worden samengebracht in een dataframe genaamd `mijn_team`:

```
mijn_df <- data.frame(voornamen, leeftijden, rugnummers)
```

Je hebt dus data van drie variabelen samengebracht in één object: `mijn_team`. Nu wil je dit object in zijn geheel opslaan op je computer. Dat kan met volgende code:

```
write.csv(mijn_team, file = "basketbal.csv", row.names = FALSE)
```

Deze functie maakt een csv-bestand[3] aan op je computer. Dat is een type bestand dat perfect geschikt is om onderzoeksgegevens te bevatten.

Het eerste argument van de functie `write.csv()` bepaalt wat er in het csv-bestand moet terechtkomen.

Het tweede argument legt de naam vast van het nieuwe bestand.

Het laatste argument is hier minder belangrijk.

Grote vraag is: waar is dat csv-bestand nu eigenlijk opgeslagen? Als je het later opnieuw nodig hebt, in welke map op je computer moet je dan gaan zoeken?

Dat kan je achterhalen met de functie `getwd()`. Die functie vertelt je de locatie op je computer waarmee je R-sessie gelinkt is. Die noemen we de werkmap of working directory.

```
getwd()
```

```
[1] "/Users/jouwnaam/Documents"
```

[3] 'csv' staat voor 'comma-separated value'.

Dat is misschien niet de locatie waar je het dataframe wilde opslaan[4]. In dat geval zal je de working directory willen veranderen.

De vraag die waarschijnlijk op je lippen brandt (of niet, soms?) is: Hoe kan je die working directory aanpassen? Meer daarover in het volgende onderdeel!

8.2.3 Een `working directory` kiezen

Om te achterhalen welke map op dit moment je working directory is, kan je volgend commando gebruiken:

```
getwd()
```

Je krijgt dan een locatie op je computer als output, bijvoorbeeld:

```
[1] "/Users/jouwnaam/Documents"
```

RStudio heeft die map automatisch gekozen, maar misschien wil je je huidige script helemaal niet op die plaats opslaan. Dan moet je je working directory veranderen. De makkelijkste manier om dat te doen is door bovenaan te klikken op Session. Ga vervolgens naar Set Working Directory en klik op Choose Directory, zoals je in de afbeelding hieronder kan zien.

In het venster dat nu verschijnt kies je een bestaande map, of maak je een nieuwe, die vanaf nu als working directory zal dienen.

[4] Het is trouwens ook niet noodzakelijk dezelfde locatie als waar je eerder het script hebt opgeslagen, namelijk /Users/jouwnaam/STATISTIEK

Dubbelcheck even of de working directory inderdaad aangepast is:

```
getwd()
```

```
[1] "/Users/jouwnaam/Statistiek"
```

Gelukt!

Dat is de makkelijkste manier om je working directory te veranderen, maar er is een vervelend nadeel aan verbonden. Als je later opnieuw het script opent in R, dan zal de working directory opnieuw veranderd zijn naar de oude, automatisch gekozen map (in ons geval was dat /Users/jouwnaam/Documents).

Ambetant! Wat nu? Is daar een goede oplossing voor? Dat leer je in het volgende onderdeel.

8.2.4 De functie `setwd()` gebruiken

Pro tip: wanneer je een nieuw script start, volg dan altijd eerst de onderstaande 5 stappen.

1. Kies een working directory via Session > Set Working Directory > Choose Directory.

2. Voer het commando `getwd()` uit.

3. In de console verschijnt nu de locatie van je working directory. Selecteer die locatie, rechtsklik erop en kopieer.

4. Typ nu helemaal bovenaan in je script `setwd()`.

5. Plaats tussen de haakjes van de functie `setwd()` de locatie die je in stap 3 hebt gekopieerd. Vergeet niet om de locatie ook tussen aanhalingstekens te zetten.

Het kan een beetje omslachtig lijken, maar je raakt het snel gewoon om altijd een script te beginnen met de functie `setwd()`.

Het voordeel aan deze manier van werken is dat je script nu bijna vanzelf gelinkt zal zijn aan de gewenste map op je computer. Het enige wat je moet doen is de code

```
setwd("/Users/jouwnaam/STATISTIEK")
```

uitvoeren wanneer je het script opent om erin verder te werken. Wanneer je nu bijvoorbeeld een dataframe wil opslaan met de functie `write.csv()` dan zal dit bestand automatisch in de working directory terechtkomen. Er zijn nog heel wat functies behalve `write.csv()` waarvoor dat handig is.

Nog een laatste opmerking: het is meestal een goed idee om je script ook op te slaan in diezelfde map, de working directory. Kwestie van het allemaal simpel en overzichtelijk te houden…

8.2.5 Samengevat

Je werk opslaan in R: Easy peasy, right? Maar mocht het toch een beetje verwarrend zijn, vatten we hier nog eens kort samen.

Hoe kan je een script opslaan?

Dit werkt gelijkaardig aan bijvoorbeeld een Word-document opslaan: wanneer je een nieuw script voor de eerste keer opslaat, moet je een map kiezen. Vanaf dan kan je gewoon opslaan met het blauwe diskette-icoontje of via File > Save. Dit zal altijd op dezelfde locatie gebeuren en het vorige bestand overschrijven. (Wil je je script toch op een andere locatie opslaan, dan kan je onder File kiezen voor Save as …)

Een dataframe opslaan met de functie `write.csv()`

Als je je onderzoeksdata in een dataframe hebt samengebracht, kan het handig zijn om dat object (het dataframe dus) op je computer op te slaan. Dat kan je in je script doen met de functie `write.csv()`, bijvoorbeeld zo:

```
write.csv(mijn_df, file = "mijn_eerste_dataframe.csv", row.names = FALSE)
```

Je hebt hier geen locatie (map op je computer) gekozen om het csv-bestand in te creëren. Het bestand wordt automatisch aangemaakt in je working directory (zie hieronder).

Een working directory vastleggen met de functie `setwd()`

Een working directory of werkmap is een locatie op je computer die gelinkt is met je activiteiten in RStudio. Heel wat functies maken gebruik van die map om

ofwel zaken op te slaan vanuit RStudio: bijvoorbeeld een dataframe opslaan met de functie `write.csv()`

ofwel omgekeerd, bestanden in te lezen naar RStudio: bijvoorbeeld een dataframe inlezen met de functie `read.csv()`

Een working directory vastleggen kan je met de functie `setwd()`, bijvoorbeeld:

```
setwd("/Users/jouwnaam/Statistiek")
```

Het is een goede gewoonte om een script altijd te starten met de functie `setwd()`.

9 Data importeren

Moet het nog herhaald worden? Je zal in R heel vaak met data werken. Dat wist je natuurlijk al. Maar hoe kan je die data in RStudio inladen, zodat je ermee aan de slag kan?

Data zullen vaak opgeslagen zijn in een csv-bestand. Dat is een specifiek bestandstype dat heel geschikt is om onderzoeksdata te bevatten. In R bestaat een handige functie om zo'n csv-bestand in te laden: `read.csv()`. Met deze functie kan je data inladen…

…vanop een locatie op je computer (ook "lokaal" genoemd)

…vanop een locatie op het internet

Laten we een voorbeeld van elk geval bekijken. Het is lichtjes verschillend voor Windows en Mac OS, dus kies in het overzicht links voor de juiste pagina of klik door via het pijltje rechts tot je op de juiste pagina landt.

9.1 Windows

9.1.1 Lokale data inladen

In deze situatie heb je een csv-bestand ergens op je computer opgeslagen. In een ideale wereld bevindt dit bestand zich in je working directory.

Ter herinnering, de working directory is een locatie op je computer die in verbinding staat met RStudio. Om te achterhalen wat op dit moment je working directory is gebruik je de functie `getwd()`.

```
getwd()
```

```
[1] "C:/Gebruikers/michi/Documenten/Statistiek/Werken met R"
```

Als zich in die map een csv-bestand bevindt, dan kan je het oproepen door simpelweg de naam van het bestand als argument te geven aan de functie `read.csv()`. Het is vaak een goed idee om dit bestand bij het inlezen meteen in een nieuw object te stoppen, hier `basketbal_dataframe` genaamd.

```
basketbal_dataframe <- read.csv("basketbal.csv")
```

Inspecteer de eerste 6 rijen van het object dat je net hebt gemaakt.

```
head(basketbal_dataframe)
```

```
  voornamen leeftijden rugnummers
1     Laura         19          4
2    Danira         22         12
3  Stefanie         18          7
4     Leyla         22         76
5    Chelsy         24         33
```

Als het bestand dat je wil inlezen zich niet in je working directory bevindt, maar ergens anders op je computer, dan moet je de locatie[1] meegeven als argument aan de functie `read.csv()`. Bijvoorbeeld:

```
deelnemers_dataframe <- read.csv("C:/Users/mbeelaer/Documents/UGent/biostats.csv")
```

Bekijk opnieuw de eerste 6 rijen ter controle.

```
head(deelnemers_dataframe)
```

[1] Zorg ervoor dat je enkel / gebruikt in de maplocatie, geen \.

```
  Name      Sex Age Height..in. Weight..lbs.
1 Alex        M  41          74          170
2 Bert        M  42          68          166
3 Carl        M  32          70          155
4 Dave        M  39          72          167
5 Elly        F  30          66          124
6 Fran        F  33          66          115
```

Dit werkt ook prima, maar het is wat omslachtiger, omdat je eerst de locatie van het bestand moet achterhalen en vervolgens tussen de haakjes van de functie `read.csv()` plakken.

9.1.2 Data van het internet inladen

Iemand heeft een csv-bestand gemaakt over de films waarin Robert De Niro meespeelt en heeft die data ter beschikking gesteld op het internet. Het bestand is te vinden op het volgende internetadres (ook "url" genoemd):

https://people.sc.fsu.edu/~jburkardt/data/csv/deniro.csv

De data in dit bestand kan je rechtstreeks in RStudio inladen. Dit doe je door de url als argument aan de functie `read.csv()` te geven:

```
film_data <- read.csv("https://people.sc.fsu.edu/~jburkardt/data/csv/deniro.csv")
```

Om de data te inspecteren kan je de functies `head()` en `str()` gebruiken.

```
head(film_data)
```

```
  Year Score                 Title
1 1968    86             Greetings
2 1970    17           Bloody Mama
3 1970    73              Hi, Mom!
4 1971    40           Born to Win
5 1973    98          Mean Streets
6 1973    88  Bang the Drum Slowly
```

```
str(film_data)
```

```
'data.frame':   87 obs. of  3 variables:
 $ Year : int  1968 1970 1970 1971 1973 1973 1974 1976 1976 1977 ...
 $ Score: int  86 17 73 40 98 88 97 41 99 47 ...
 $ Title: chr  " Greetings" " Bloody Mama" " Hi, Mom!" " Born to Win" ...
```

De data lijken goed ingeladen! Kan je achterhalen hoeveel films in de dataset zijn opgenomen?

Klik hier om het antwoord te zien

Er zijn 87 "observations" van 3 variabelen in het dataframe. Dat wil zeggen dat jaartal, score en titel van 87 films werden bijgehouden.

9.2 macOS

9.2.1 Lokale data inladen

In deze situatie heb je een csv-bestand ergens op je computer opgeslagen. In het eenvoudigste geval bevindt dit bestand zich in je working directory.

Ter herinnering, de working directory is een locatie op je computer die in verbinding staat met RStudio. Om te achterhalen wat op dit moment je working directory is gebruik je de functie `getwd()`.

```
getwd()
```

```
[1] "/Users/jouwnaam/Statistiek"
```

Als zich in die map een csv-bestand bevindt, dan kan je het oproepen door simpelweg de naam van het bestand als argument te geven aan de functie `read.csv()`. Het is vaak een goed idee om dit bestand bij het inlezen meteen in een nieuw object te stoppen, hier `basketbal_dataframe` genaamd.

```
basketbal_dataframe <- read.csv("basketbal.csv")
```

Inspecteer de eerste 6 rijen van het dataframe met de functie `head()`.

```
head(basketbal_dataframe)
```

```
  voornamen leeftijden rugnummers
1     Laura         19          4
2    Danira         22         12
3  Stefanie         18          7
4     Leyla         22         76
5    Chelsy         24         33
```

Als het bestand dat je wil inlezen zich niet in je working directory bevindt, dan moet je de locatie[2] meegeven als argument aan de functie `read.csv()`. Bijvoorbeeld:

```
deelnemers_dataframe <- read.csv("/Users/jouwnaam/Documents/biostats.csv")
```

Bekijk opnieuw de eerste 6 rijen ter controle.

[2] Zorg ervoor dat je enkel / gebruikt in de maplocatie, geen \.

```
head(deelnemers_dataframe)
```

```
  Name      Sex Age Height..in. Weight..lbs.
1 Alex        M  41          74          170
2 Bert        M  42          68          166
3 Carl        M  32          70          155
4 Dave        M  39          72          167
5 Elly        F  30          66          124
6 Fran        F  33          66          115
```

Dit werkt ook prima, maar het is wat omslachtiger, omdat je eerst de locatie van het bestand moet achterhalen en vervolgens tussen de haakjes van de functie `read.csv()` plakken.

9.2.2 Data van het internet inladen

Iemand heeft een csv-bestand gemaakt over de films waarin Robert De Niro meespeelt en heeft die data ter beschikking gesteld op het internet. Het bestand is te vinden op het volgende internetadres (ook "url" genoemd):

https://people.sc.fsu.edu/~jburkardt/data/csv/deniro.csv

Deze data kan je rechtstreeks in RStudio inladen. Dit doe je door de url als argument aan de functie `read.csv()` te geven. Het is best om dit bestand meteen in een object te stoppen, hier `film_data` genoemd:

```
film_data <- read.csv("https://people.sc.fsu.edu/~jburkardt/data/csv/deniro.csv")
```

Om de data te inspecteren kan je de functies `head()` en `str()` gebruiken.

```
head(film_data)
```

```
  Year Score                 Title
1 1968    86             Greetings
2 1970    17           Bloody Mama
3 1970    73              Hi, Mom!
4 1971    40           Born to Win
5 1973    98          Mean Streets
6 1973    88  Bang the Drum Slowly
```

```
str(film_data)
```

```
'data.frame':   87 obs. of  3 variables:
 $ Year : int  1968 1970 1970 1971 1973 1973 1974 1976 1976 1977 ...
 $ Score: int  86 17 73 40 98 88 97 41 99 47 ...
 $ Title: chr  " Greetings" " Bloody Mama" " Hi, Mom!" " Born to Win" ...
```

De data lijken goed ingeladen! Kan je achterhalen hoeveel films in de dataset zijn opgenomen?

Klik hier om het antwoord te zien

Er zijn 87 “observations” van 3 variabelen in het dataframe. Dat wil zeggen dat jaartal, score en titel van 87 films werden bijgehouden.

10 Meer functies via “packages”

Wanneer je R voor het eerst installeert, dan krijg je een basispakket dat toepasselijk “base R” heet. In dat basispakket worden al heel wat functies meegeleverd. Met die functies kan je allerlei taken laten uitvoeren met een minimum aan code.

Zo bevat “base R” een functie om het gemiddelde te berekenen van een reeks getallen. De vector `sport` bevat het aantal uur dat een groep leerlingen per week aan sport besteedt. De functie `mean()` geeft je heel snel het gemiddelde aantal uur voor die klas.

```
sport <- c(6, 0, 7, 8, 3, 0, 3, 3, 7, 2, 6, 5, 3, 4, 7, 3, 7, 3, 6, 0)

mean(sport)
```

```
[1] 4.15
```

Eén van de mooie aspecten van R is dat iedereen nieuwe functies kan maken en die beschikbaar kan stellen voor alle gebruikers wereldwijd. Die extra functies worden gebundeld in zogenaamde packages. Op die manier zijn er duizenden uitbreidingen ontstaan van het basispakket van R. Zo heeft iemand allerlei functies gemaakt die helpen bij het visueel voorstellen van data. Die functies zijn gebundeld in een package genaamd `ggplot2`.

Als je functies uit packages wil gebruiken, moet je die eerst installeren en vervolgens laden.

Eerst het installeren. Dat is ongelooflijk eenvoudig. Voor bijvoorbeeld het package `lavaan`, typ je gewoon de onderstaande code in R.

```
install.packages("lavaan")
```

Het package wordt nu automatisch gedownload en geïnstalleerd. Let erop dat je de naam van het package tussen aanhalingstekens plaatst. Het installeren van een package hoef je maar één keer te doen.

Om de functies uit het package te gebruiken in je code, moet je het laden met de functie `library()`. Hierbij is het goed mogelijk dat er één of meer 'warnings' verschijnen in de console. Dat is meestal geen enkel probleem. Een 'warning' geeft gewoon informatie die je als gebruiker misschien zal interesseren. Het betekent niet meteen dat er iets fout is gegaan. Dat is anders dan een 'error'. Als je dat ziet staan is er wel een probleem en zal het package niet naar behoren werken.

```
library(lavaan)
```

Het laden van een package moet je wél elke keer doen dat je RStudio start (in tegenstelling tot het installeren). Daarom is het een goed idee om een script altijd te starten met het laden van de packages die je van plan bent te gebruiken. Wil je bijvoorbeeld de packages `lavaan` en `ggplot2` gebruiken, plaats dan bovenaan in je script de code:

```
library(lavaan)
library(ggplot2)
```

Voer deze twee regels code uit elke keer dat je het script opent. Zo beschik je meteen over alle functies uit die packages.

Twee functies om te onthouden

Om nieuwe packages te kunnen gebruiken moet je maar twee functies onthouden:

`install.packages()` moet je éénmalig uitvoeren.

`library()` moet je uitvoeren elke keer wanneer je de functies uit een package wil gebruiken.