Auteur: Arie Twigt

Druk: bravenewbooks.nl
Omslagontwerp en illustraties: Rosemarijn Twigt
Vormgeving binnenwerk: Arie Twigt

Data Analyseren en Programmeren met R

ISBN 9789402152067

NUR 600

Korte inhoudsopgave

Voorwoord

Allereerst, bedankt dat je dit boek erbij pakt om kennis te maken met de open source statistische programmeertaal R. Daarbij wil ik gelijk mijn vrouw bedanken die voor de illustraties heeft gezorgd in dit boek. Ze zijn handmatig met potloden op tekenpapier gemaakt en daarna met Photoshop bewerkt. Erg bedankt!

Ook wil ik mijn Sogeti collega's Dylan van Riel en Jonathan de Melker Worms bedanken voor de uitgebreide review van dit boek. Dylan leidt de Technology & Tooling groep op het gebied van Analytics binnen Sogeti, waar ook de programmeertalen R en Python worden behandeld. Jonathan is collega Data Scientist bij Sogeti die o.a. bij de Rabobank en Aegon met R gewerkt heeft. Ondanks dat hij zijn PhD moest afronden heeft hij toch tijd gehad om dit boek te reviewen.

R is de afgelopen jaren mijn stokpaardje geworden. Tijdens mijn studie *Business Administration - Information & Knowledge Management* op de Vrije Universiteit Amsterdam heb ik het vak *Business Intelligence* gekregen en kwamen er onderwerpen als **predictive analytics** (voorspellingsmodellen) en uitgebreide data-analyse aan bod. Omdat ik tussen het bestuderen van papers ook graag de praktijk in wilde duiken, ben ik hiermee naar de professor gegaan, Frans Feldberg, die mij adviseerde om eens een kijkje te nemen naar de statistische programmeertaal **R**."R? Hoe spel je dat?" "Gewoon de letter R". Uiteindelijk is dit het gesprek dat ik nu heel vaak voer met mensen aan wie ik uitleg geef over R. Zoek voor de gein maar eens op waarom de programmeertaal "R" heet. Uiteindelijk ben ik er zo enthousiast over geworden dat ik er bijna dagelijks mee experimenteerde. Uit mijn enthousiasme heb ik daar uiteindelijk een handleiding van geschreven. Dat is niet deze handleiding, maar de handleiding die ik in 2013 geschreven heb: Handleiding R. Deze handleiding is door de komst van dit boek achterhaald.

Afgelopen jaren heb ik bij verschillende klanten oplossingen met R geïmplementeerd of de klant laten kennismaken met R. Hierbij heb ik gezien dat R over het algemeen positief wordt ontvangen en zeer toepasbaar is in het bedrijfsleven. Voorbeelden zijn klanten die voor processen met complexe berekeningen overstappen van pakketten als MatLab, SPSS of Excel naar R. Het is makkelijk te implementeren, de mogelijkheden zijn eindeloos en het is gratis. Het feit dat R open source is, zorgt af en toe voor een licht drempeltje bij klanten. Gelukkig heeft Microsoft sinds 2015 R in verschillende software geïntegreerd, bijvoorbeeld op **Microsoft Azure**, **Microsoft Visual Studio** (met *R tools for Visual Studio*) en **Microsoft SQL Server Management Studio 2016**. Ook heeft Microsoft een eigen versie van R, **Microsoft R open**. Er staat je dus een mooie toekomst te wachten als je R onder knie hebt.

Nogmaals, wil ik je erg bedanken dat je dit boek erbij hebt gepakt. Met de opbouw en alle instructies wil ik de denkwijze die ik heb en de methoden die ik gebruik voor de basics naar jou overbrengen. Natuurlijk zijn er mensen die dit op een andere manier aanpakken, dit zal je naar verloop van tijd zelf ook doen als je het aardig onder de knie krijgt. In ieder geval zal dit boek je helpen met het onder de knie krijgen van de basics en zal je alle aspecten bezitten om uitgebreid data te kunnen analyseren met R. Ik ben ervan overtuigd dat je net zoveel plezier mee zult hebben dit boek te lezen en de instructies uit te voeren, als het plezier dat ik heb ervaren tijdens het samenstellen van dit boek.

Inleiding

De programmeertaal R

R wordt over het algemeen een open source statistische programmeertaal genoemd. Deze beschrijving is een hele mond vol, maar bevat wel de belangrijkste aspecten van deze programmeertaal:

- Open source;
- Statistisch;
- Programmeertaal.

Open source

R is een open source programmeertaal, dat betekent dat niemand, geen organisatie en geen persoon, de eigenaar is van R en je het dus overal gratis voor kunt gebruiken. Daarbij geeft open source aan dat de broncode waarmee de programmeertaal is opgebouwd beschikbaar is. Je kunt bijvoorbeeld bekijken met welke broncode de functies van R zijn opgebouwd. Een ander belangrijk aspect van open source in dit geval is dat je alle vrijheid hebt om onderdelen aan te passen en er extra mogelijkheden aan kunt toevoegen. De mogelijkheden hier in zijn eindeloos, in die zin dat mogelijkheden niet bewust zijn afgesloten omdat R niet is dichtgetimmerd.

Een belangrijk voorbeeld hiervan zijn de verschillende *packages* die beschikbaar zijn in R. *Packages* zijn verzamelingen van extra functies die geschreven zijn door andere ontwikkelaars en over het algemeen vrij beschikbaar worden gesteld. Deze packages maken de programmeertaal zeer krachtig, flexibel en toepasbaar voor verschillende werkzaamheden, sectoren en branches. Zo zijn er bijvoorbeeld packages die het makkelijk maken om *Twitter* data te importeren (het **twitteR** package), packages die formules van financiële calculaties bevatten (het **FinCal** package), packages waarmee je analytics kunt uitvoeren door Google Analytics data te importeren in R (het officiële **RGoogleAnalytics** package van Google) en packages om op een prachtige manier data te visualiseren (bijvoorbeeld het **ggplot2** package). Dit maakt R een Zwitsers zakmes als het gaat om werken met data en is dit voor een groot deel te danken is aan het feit dat het een open source programmeertaal is.

> ***Packages zijn goed gedocumenteerd*** *en worden beheerd in de Comprehensive R Archive Network (CRAN). Dit is ook de plek waar over het algemeen de belangrijkste packages staan en worden beheerd door de community van R. Deze packages hebben aan een aantal belangrijke criteria voldaan. De documentaties zijn over het algemeen te vinden in een overzichtelijk pdf-bestand. In deze documentatie kun je alle functies van het package bekijken, contactgegevens vinden van de ontwikkelaar en met voorbeelden zien hoe functies van het package worden toegepast. Naast CRAN, zijn er ook steeds meer packages te vinden op* ***Github****. Deze werken gewoon op dezelfde manier als packages die op CRAN staan, de installatie is alleen iets anders.*

In dit boek wordt er ook verschillende keren gebruik gemaakt van packages. Je zult zien dat packages zeer eenvoudig te installeren en te gebruiken zijn.

Statistisch

R wordt een statistische programmeertaal genoemd omdat het standaard een uitgebreide collectie van statistische functies bevat. Dit betekent absoluut niet dat R beperkt wordt tot het vakgebied statistiek of dat het minder handig is voor andere vakgebieden. De reden dat R deze statistische functies bevat is omdat het oorspronkelijk ontwikkeld is om (statistisch) onderzoek mee uit te voeren. Je kunt het hierbij vergelijken met softwarepakketten als *SPSS* en *MatLab*. Je zou op dezelfde manier kwantitatief onderzoek kunnen doen of je scriptie kunnen schrijven in R. Statistische toetsen, visualisaties of formules zijn zeer eenvoudig met functies toe te passen in R zonder dat je daar een speciaal *package* voor hoeft te installeren. Deze functies maken het data analyseren veel makkelijker. Naast de ingebakken statistische functies bevat R namelijk een breed scala aan functies om databewerkingen of analyses te doen. Bij meer uitgebreide analyses kunnen bepaalde statistische handelingen van toepassing komen. In andere programmeertalen zal je dan de formules van deze statistisch handelingen handmatig moeten programmeren, dit hoeft in R echter niet om dat zo goed als elke statistisch functie in R zit. Dit zorgt ervoor dat je een extreem krachtige tool hebt om data te analyseren, zelfs als het een stuk ingewikkelder wordt. Daarom kun je het statistische aspect van R eerder zien als een waardevolle toevoeging dan een beperking.

Programmeertaal

Het feit dat R een programmeertaal is, zorgt voor veel flexibiliteit. Het werk dat je gedaan hebt in R is in de meeste gevallen een reeks aan commando's die samengevoegd worden tot een script. In een script worden de commando's van boven naar beneden uitgevoerd, net zoals bij andere script-gebaseerde programmeertalen (*scripting languages*). Een script in R wordt een R-script genoemd en heeft de `.r` extensie (bijvoorbeeld: `analyse.r`). Dit script kun je overal mee naartoe nemen en in verschillende systemen toepassen. Zo kun je bijvoorbeeld R-code toepassen om data uit een database aan te passen, verschillende databases zijn namelijk in staat om R-code uit te voeren (bijvoorbeeld **Oracle** en **Microsoft SQL Server (vanaf 2016)**. Ook op Big Data platformen en software zoals **Hadoop** en **Spark** kun je R code schrijven om data te bewerken.

Het werk dat je maakt in R is dus op veel verschillende plekken toepasbaar en kun je eenvoudig meenemen en delen. Als je nu bijvoorbeeld Excel neemt, iets dat in Excel gemaakt is blijft in Excel. De logica en code die je in Excel gebouwd hebt kun je niet overhevelen naar een ander systeem (hooguit kun je de data delen), deze zitten namelijk in het Excel-bestand gebakken. Functies en handelingen die bouwt in R, kunnen op veel meer plekken gebruikt worden. Dit geldt trouwens voor de meeste programmeertalen, de code is los te trekken van de omgeving waar het in is gebouwd: *"Code once, use everywhere."*.

Programmeren klinkt voor sommigen misschien heel eng en code kan er soms heel ingewikkeld uitzien. Je moet je echter hierdoor niet laten afschrikken. Het is natuurlijk nodig dat je de syntax van R begrijpt en weet wanneer je welke code moet toepassen. Hier zit een leercurve in die bij R soms als vrij stijl wordt ervaren. Echter kun je deze overwinnen met genoeg oefening waardoor je een beter begrip krijgt van de syntax en de structuur waardoor je voor jezelf een eigen manier zal vinden om R goed onder de knie te krijgen. Als je hebt gewerkt met formules in Excel, kun je sommige onderdelen van die kennis meenemen

om R leren te begrijpen. Het is daarbij ook een verschil in mindset: R is geen software waarmee je op knopjes drukt en kan klikken en slepen, maar een programmeertaal waarmee aan de hand van commando's een reeks aan handelingen geeft als opdracht om door R uit te laten voeren.

R is flexibel door de vele mogelijkheden, de vele verschillende datatypes en de vele datastructuren. Maar deze flexibiliteit kan complex en ingewikkeld overkomen als je er zomaar in duikt en begint met programmeren zonder enig begrip van de structuur van R. Daarom worden er in dit boek stapje voor stapje de belangrijkste structuren en datatypes in R uitgebreid behandeld.

De structuur van R

Het is de bedoelding dat je na het lezen en het volgen van de instructies in dit boek programmeren en data analyseren met R goed onder de knie hebt gekregen. Daarom beginnen we bij het begin, namelijk de bouwstenen van de structuur in R. Zonder goede kennis van de structuur in R loop je het gevaar dat je tijdens het data analyseren in de knoop komt, bijvoorbeeld door dingen te willen doen die qua structuur helemaal niet kunnen.

Data bewerken

In bijna elk data-analyse proces moet de data eerst worden bewerkt voordat er mee geanalyseerd kan worden. De instructies geven je goede voorbeelden hoe je dit in R kunt doen. In veel gevallen kan data na het ondergaan van een aantal bewerkingen goed geanalyseerd worden. R heeft een zeer breed scala aan mogelijkheden om data te bewerken. De bewerkingen in dit boek zijn qua niveau logisch opgebouwd van eenvoudig naar geavanceerd. Daarbij zijn veel instructies en voorbeelden van bewerkingen die ik tijdens projecten bij klanten ben tegengekomen. Ik zal je dus geen nutteloze trucjes of handelingen leren, in ieder geval zo weinig mogelijk.

Data analyseren

Er zijn eindeloos veel mogelijkheden om data uitgebreid te analyseren. Zowel visueel of bijvoorbeeld statistisch. Ik heb nooit de illusie gehad om alle mogelijkheden in één boek te stoppen. Echter zit er wel een rode lijn in dit boek op het gebied van data analyseren. Onderwerpen zijn bijvoorbeeld analyses door te *slicen en dicen* zoals de Business Intelligence wereld dat noemt, statistische functies, visualisaties en speciale R-functies. Ook deze heb ik onder andere toegepast bij klanten en hebben waardevolle inzichten opgeleverd. Dit kan voor jou, of voor het bedrijf waar je voor werkt, zeer waardevol zijn.

Geen big data

Dit boek behandeld niet het onderwerp *Big Data*. Dit is data die te groot is om door een enkele CPU op je machine bewerkt te kunnen worden. Hiervoor heb je geavanceerde software nodig om meerdere computers aan elkaar te koppelen voor meer rekenkracht. Bekende software-omgevingen die dit doen zijn bijvoorbeeld **Apache Hadoop** en **Apache Spark**. Tegenwoordig ook verschillende Microsoft-producten

zoals Microsoft R server of Microsoft **Azure Batch Services**. Het leuke aan deze software is dat je er R in kunt programmeren. Hierdoor wordt het mogelijk om R-bewerkingen te doen op gigantische datasets, datasets met tientallen of zelfs honderden miljoenen aan rijen aan data. Als je R onder de knie hebt, kun je altijd naar de websites gaan die deze *Big Data* oplossingen aanbieden. Je zult de R code herkennen en daardoor de stap kunnen maken naar *Big Data*.

> *In het artikel Microsoft Azure Batch: Doorlooptijden terugbrengen van dagen naar uren (https://arietwigt.wordpress.com/2016/09/17/microsoft-azure-batch-doorlooptijden-terugbrengen-van-dagen-naar-uren/). leg ik uit hoe Microsoft Azure Batch Services in zijn werk gaat en het kunt gebruiken om meer rekenkracht te hebben voor R-calculaties.*

Geen uitgebreide statistiek of econometrie

Naast *Big Data* wordt er in dit boek ook geen uitgebreide uitleg gegeven over geavanceerde wiskundige modellen. Je krijgt echter wel voldoende informatie over de statistische functies die je kunt toepassen in R. Bijvoorbeeld het onderwerp *Lineaire regressieanalyse* in dit boek legt uitgebreid de statistische betekenis en onderbouwing van de output van de functie uit. Hiermee kun je klanten goed uitleggen wat je toepast en wat de uitkomsten daarvan zijn. Een lineaire regressie is een zeer populair voorspellingsmodel, maar slechts een van de vele mogelijkheden van voorspellende analyses. Ondanks dat is het een mooi begin en zal goede kennis van R die overgebracht wordt in dit boek je helpen als je naar meer uitgebreide wiskundige of statistische modellen wilt kijken.

Wat je leert

Dit boek kan jouw eerste kennismaking met R zijn. Als je al vaker met R hebt gewerkt, is dit boek alsnog een goede toevoeging aan jouw R-kennis. De opbouw en de structuur van R worden namelijk op een zo duidelijk en compleet mogelijke manier uitgelegd. In dit boek wordt er zoveel mogelijk een afweging gemaakt om de basics uit te leggen zonder dat het geestdodend eenvoudig en saai is. Daarbij zijn onderwerpen die te diep gaan en over het algemeen voor werkzaamheden niet relevant zijn, weggelaten. Het is de bedoeling dat je aan het einde van dit boek de belangrijkste onderdelen van de R structuur onder de knie hebt en de eigenschappen van R-objecten kent. Met deze kennis zal je aan het einde van dit boek R goed in de vingers hebben en snelheid opbouwen bij het programmeren in R. Daarbij is dit boek een mooi naslagwerk waarmee je verschillende commands kunt opzoeken. Je hoeft er namelijk niet naar te streven om de hele syntax uit je hoofd te leren, zelfs de beste programmeurs gebruiken Google.

Wat je eraan hebt

R is een waardevolle toevoeging aan je skillset en CV. Vooral bij vacatures voor een rol als bijvoorbeeld Data Scientist staat R op het lijstje van vereiste skills. Andere tools als bijvoorbeeld Python kom je ook vaak tegen. Python is ook een uitstekende tool om data te analyseren. Van oorsprong is Python meer een echte ontwikkeltaal en geeft het daarom meer mogelijkheden bij het ontwikkelen van data gedreven applicaties. Alhoewel je met R ook applicaties kunt bouwen, zou je grofweg kunnen zeggen dat je voor de beste ervaring bij het uitgebreid analyseren van data kiest voor R en als je iets wilt ontwikkelen je kiest

voor echte ontwikkeltalen zoals Python, Java of Scala. Daar staat wel tegenover dat steeds meer gerenommeerde softwarepakketten kiezen voor de integratie met R zoals bijvoorbeeld Microsoft dat doet. Zelf heb ik bijvoorbeeld een opdracht gedaan met R bij een grote verzekeraar die over het algemeen een Microsoft IT architectuur heeft. Omdat Microsoft Visual Studio en Microsoft SQL Server 2016 een uitgebreide R plugin heeft, was R de taal waarmee de uitgebreide calculaties geprogrammeerd werden. Om deze redenen is programmeren in R een waardevolle skillset waar je bij bedrijven en in de toekomst veel aan zult hebben.

> *Met R kun je ook Webapplicaties en Dashboards bouwen. Zelf heb ik op dit moment bijvoorbeeld de webapplicatie **TweetExplorer** tweetexplorer.com geschreven, waarmee Tweets eenvoudig geanalyseerd kunnen worden. Data gedreven webapplicaties kun je volledig in R schrijven in combinatie met een beetje HTML, CSS en Javascript.*

Samenvattingen en oefeningen

Ieder hoofdstuk eindigt met een samenvatting en oefeningen. Omdat het een boek is dat je kunt volgen door instructies uit te voeren in jouw eigen R-sessie, kan het afronden van een hoofdstuk langer duren dan als je het alleen leest. Een samenvatting geeft een mooie afronding van het hoofdstuk en vat mooi samen welke onderwerpen er in het hoofdstuk zijn besproken. Naast de samenvattingen heeft ieder hoofdstuk ook twee of drie oefeningen. Deze oefeningen zijn gemaakt om je los van de instructies stukken R code te schrijven waardoor je wordt gedwongen je eigen opgedane kennis toe te passen. Hiervoor kun hulp vinden in het betreffende hoofdstuk, maar voor sommige vragen moet je zelf een creatieve manier vinden of Google raadplegen. Hierdoor wordt je nog beter getraind en ervaar je de pijn die je tijdens projecten ook kan hebben als je naar een bepaalde oplossing zoekt. Mocht je er echt niet uit komen, kun je de antwoorden van de oefeningen vinden aan het einde van dit boek in het hoofdstuk *Antwoorden voor oefeningen*.

Termen

Variabele

Een variabele is een object dat een bepaalde waarde bevat. Hierbij kan een variabele alleenstaand zijn of onderdeel maken van een groter object zoals een data frame. In dat geval is een variabele een kolom in een data frame.

Object

Een object is een entiteit in R. Er zijn verschillende typen objecten in R zoals data frames (tabellen), matrixen, lijsten (lists) en vectoren. Ieder type object vervult zijn eigen functie tijdens een R-sessie. Hierbij verschillen de objecten qua mogelijkheden om deze te bewerken.

Working Directory

De *Working Directory* is een belangrijk onderdeel in R. Het is namelijk de map op jouw machine waar de R-sessie op zoek gaat naar bestanden als deze geïmporteerd moeten worden of waar R bestanden naar exporteert, bijvoorbeeld databestanden of ander r scripts. In programmeertermen kun je zeggen dat de *Working Directory* een ander woord is voor de *root directory* van de huidige R-sessie. Je kunt het `getwd()` command gebruiken om te kijken wat de *working directory* voor de huidige R-sessie is. Met het `setwd()` command kun je de *working directory* veranderen.

Project

Een project is eigenlijk geen R entiteit. Het is een object dat aangemaakt wordt door RStudio en kan daarom ook alleen door RStudio gebruikt worden. Een project is een handige *wrapper* voor de gehele omgeving van de R-sessie. Als je bijvoorbeeld bezig bent met een bepaalde analyse en je hebt hiervoor verschillende bestanden geopend, waarden aangemaakt en een working directory ingesteld, wordt dit allemaal bijgehouden en teruggehaald zodra je het project weer opent.

Functie

Een functie is een reeks aan handelingen en bewerkingen op een waarde in R. Net zoals bij andere programmeertalen zijn functies een van de belangrijkste bouwstenen van R. R komt standaard met een breed scala aan functies waarmee je uitgebreid data kunt analyseren. Daarbij kun je packages installeren die andere functies bevatten die R niet heeft. Deze packages zijn verzamelingen van functies die andere personen gemaakt hebben en beschikbaar stellen. Ten slotte kun je zelf functies schrijven en jouw verzameling aan functies samenbrengen onder een package welke je eventueel beschikbaar stelt voor andere personen. Op deze manier is de cirkel mooi rond en blijft de programmeertaal R bloeien.

Dataset

Een dataset is een verzameling gegevens die in een R-sessie kan worden ingeladen. Dit kan in de vorm van een tabel, matrix, of lijst zijn. Op deze datasets kun je bewerken of gebruiken om er verschillende functies op toe te passen. Door dit op verschillende manieren te doen, kun je verschillende inzichten uit een dataset halen.

Legenda

Commentaar

In R is het mogelijk om commentaar te geven in het script. Deze regels worden door R overgeslagen tijdens het verwerken van de code. Commentaar is een handig middel om aan te geven wat er gebeurt bij verschillende plekken in het script, voor jezelf en ook voor anderen die het script lezen.

```
# dit is commentaar
```

Output uit de R-console

In sommige voorbeelden geeft R output terug. Deze output wordt aangegeven met **##**.

```
print("Dit is output :)")


## [1] "Dit is output :)"
```

Installatie software en bestanden

R installeren

Op officiële website van R, **The Comprehensive R Archive Network**(CRAN), kun je R downloaden. Zoals je kunt zien is R voor verschillende besturingssystemen beschikbaar.

Voor Windows, kies je voor *Download R for Windows*. Op de volgende pagina kies je voor *install R for the first time*. Hiermee download je de normale *base* versie van R, de standaard R met alle standaard functies. Simpel gezegd, "hiermee download je R." Hierna volgt een wizard waarmee R wordt geïnstalleerd. Kies zo veel mogelijk voor de al standaard keuzes. Pas eventueel aan welke snelkoppelingen je wel en niet wilt hebben.

Als je een Mac OS gebruiker bent, kies je voor *Download R for (Mac) OS X*. Op de je komt hierdoor op een pagina waar verschillende download packages beschikbaar zijn, kies voor de meest recente versie. Als het package eenmaal gedownload is, kun je de eenvoudige stappen van het installatieproces volgen. De meest gebruiksvriendelijke keuzes zijn al door de installatie wizard geselecteerd.

RStudio installeren

Om RStudio te installeren ga je naar https://www.rstudio.com/products/rstudio/#Desktop. Kies voor de *Open Source Edition* door op **Download RStudio Desktop** te klikken. Hierdoor kom je op de pagina met

alle beschikbare versies van RStudio voor verschillende platformen. Kies voor het platform waar je op werkt. Volg daarna de instructies van de wizard. Kies ook hier zoveel mogelijk voor de standaard mogelijkheden.

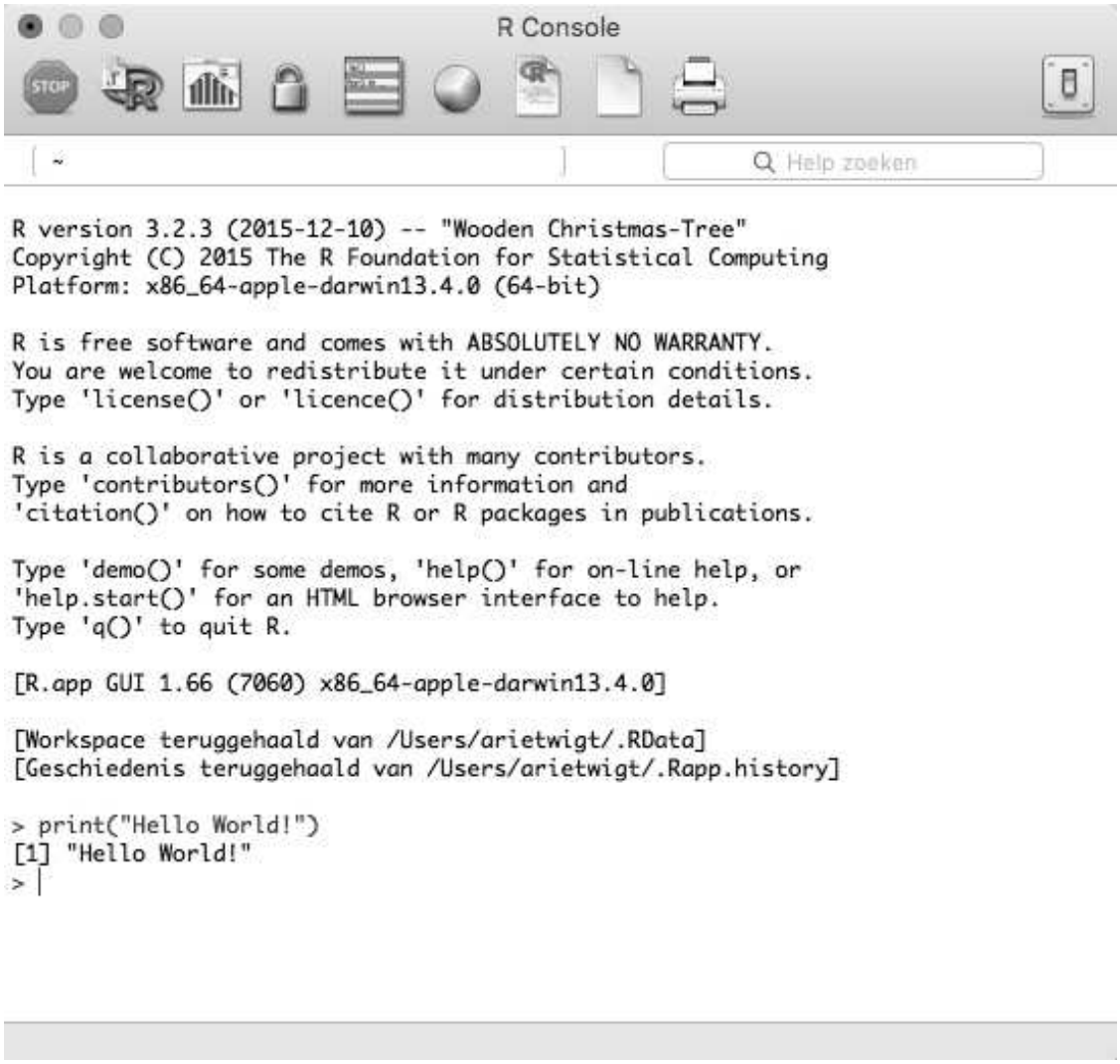
R en RStudio

Ik adviseer je om RStudio pas te installeren als R succesvol is geïnstalleerd. Het is gewoon mogelijk om RStudio te installeren zonder dat je R geïnstalleerd hebt, alleen werkt het pas als R ook geïnstalleerd is. R draait namelijk onder de motorkap van RStudio. Als je RStudio installeert nadat je R hebt geïnstalleerd, zoekt RStudio automatisch naar de locatie van R op voor jouw machine. Als dit andersom doet, is er de kans dat je dit later in RStudio zelf moet configureren.

RStudio is een zeer uitgebreide en zeer gebruiksvriendelijke ontwikkelomgeving voor R. Het geeft je als gebruiker een veel beter beeld van waar je mee bezig bent tijdens een R-sessie. Daarbij heeft RStudio veel hulpmiddelen die je kunt gebruiken door erop te klikken, terwijl je daarvoor in de normale R-commands voor zou moeten gebruiken. Ik adviseer je daarom om RStudio te gebruiken. Mocht je toch liever de normale R-omgeving gebruiken, staat niets je daarvoor in de weg. Ook dan kun je alle instructies in dit boek probleemloos volgen.

De R-console

De R-console is een venster waarin je commands kunt invoeren. Het invoeren van commands is wat we in dit geval het *programmeren* noemen. Eigenlijk spreken we pas van programmeren als we een script bouwen dat uit een verzameling commands bestaat en dit door een systeem wordt uitgevoerd. Aangezien een script bestaat uit commands en je van de commands in dit boek ook uiteindelijk je eigen script kan maken, spreken we in dit boek uiteindelijk toch van programmeren.



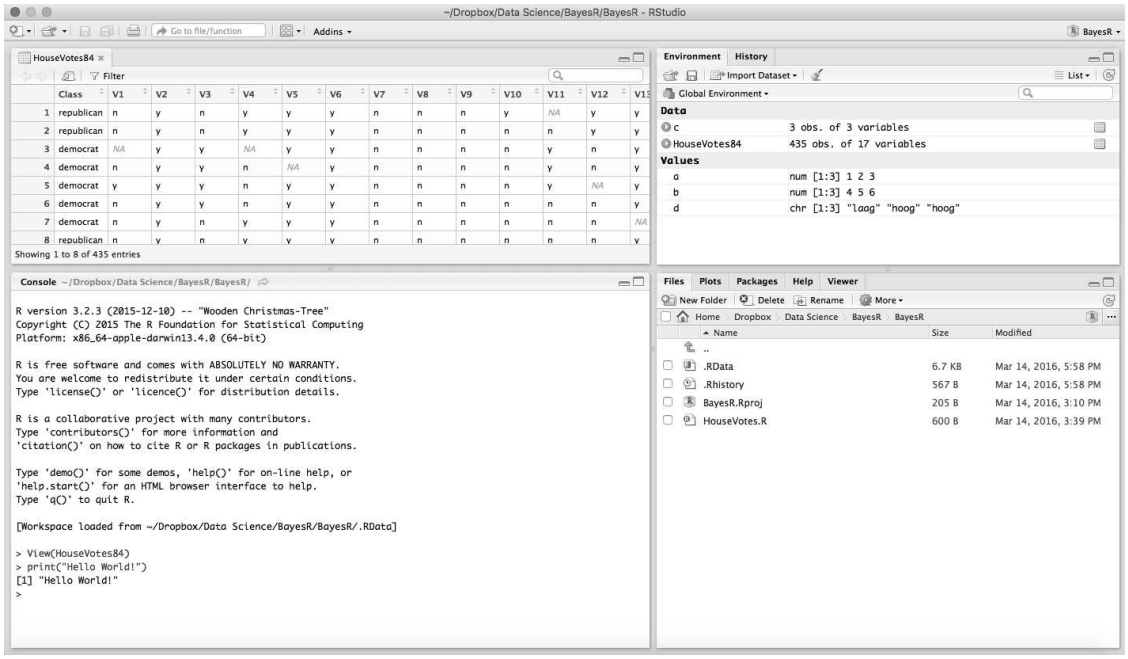
De R-console

De R-console reageert op ieder commando dat we invoeren. Grofweg doet de R-console twee dingen: opslaan en reageren. Dit wordt een *Read Evaluate Print Loop* (REPL) genoemd. Deze R-console is te zien in ***Figure 1***. Als we een waarde aan een nieuwe variabele toewijzen, slaat R deze op en geeft hij pas deze waarde terug als we deze variabele weer oproepen. Als we echter een functie gebruiken of zelf een waarde invoeren, geeft R een reactie. Deze reactie is dezelfde waarde, of de waarde na de toepassing van een functie. In de onderstaande schermafbeelding geven we de waarde `"Hello World!"` aan de `print()` functie. R past de functie toe op deze waarde en geeft het resultaat terug. We gaan hier veel dieper op in tijdens de hoofdstukken in dit boek.

De R Studio omgeving

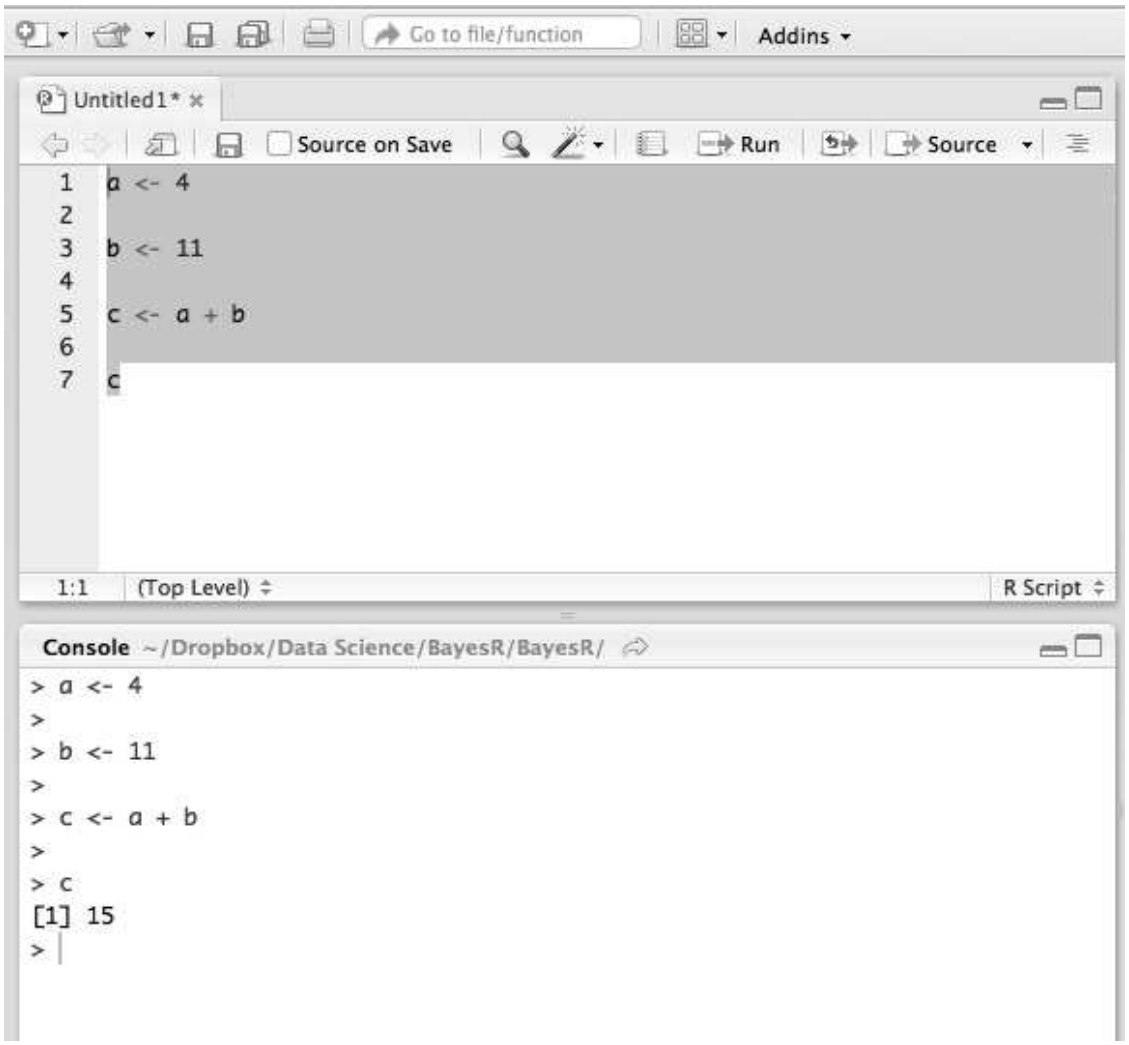
In ***Figure 2*** zie je de **RStudio IDE**. *IDE* staat voor *Intelligent Development Environment* en dat beschrijft ook precies de toegevoegde waarde van RStudio ten opzichte van de standaard R-console. De RStudio is een

stuk uitgebreider dan de R-console, maar zeker niet ingewikkeld. De extra vensters zijn namelijk visuele hulpmiddelen die eigenlijk verstopt zijn bij de normale R-console. Het venster links onderin herken je waarschijnlijk, het is namelijk exact dezelfde R-console. Deze werkt **exact** hetzelfde als de normale R-console. Een toevoeging in RStudio is dat hier de R-console *self complete* toepast op de commando's. Als je bijvoorbeeld een functie invoert of een object oproept, geeft de R-console een hint voor het commando dat je bedoelt.



De R-console

Links bovenin zie je een dataset. Omdat je logischerwijs met datasets werkt in R, wil je af en toe weten hoe de dataset eruitziet. Deze datasets worden dan op dit venster weergegeven. Daarbij zijn er handige mogelijkheden om data te sorteren en te selecteren. Dit venster is ook de plek waar jouw R scripts verschijnen. Met het knopje links bovenin (het blaadje met het plusje) kun je een nieuw R-script toevoegen. **Ik adviseer je sterk om een R script te gebruiken.** Een klein voorbeeld van een script kun je vinden in ***Figure 3***. Zo heb je altijd een overzicht van de commando's die in de R-sessie worden uitgevoerd en belangrijker nog, je kunt het opslaan als bestand en is daarmee een eindproduct van de R-sessie.



Script in R

In een script kun je de commando's intypen en laten uitvoeren in de console. Dit doe je door de cursor aan het begin van de regel van het command te plaatsen en `Ctrl + Enter` of `Cmd + Enter` (Mac) in te gebruiken. Je kunt ook de hele regel selecteren en `Ctrl + Enter` of `Cmd + Enter` gebruiken om een commando uit te voeren.

Rechts bovenin heb je het overzicht van de objecten in de huidige R-sessie. Dit zijn bijvoorbeeld variabelen met de toegewezen waarden, functies en datasets. Daarbij heb je een tab met *history*, hiermee krijg je een handig overzicht van de commando's die je in deze R-sessie hebt uitgevoerd.

Het venster rechts onderin bevat de meeste tabs van alle vensters. De *Files* tab is de verkenner waarmee je door de bestanden en mappen kunt bladeren. Het *Plots* tab is de plek waar de grafieken en andere visualisaties worden weergegeven. Dit tab opent zich automatisch zodra je in R een commando invoert om een visualisatie weer te geven. Het *packages* tab geeft een overzicht van de gedownloade en geopende packages in de huidige omgeving (jouw account op de laptop of computer). Packages zijn modules gemaakt door ontwikkelaars om extra functies toe te voegen in R, deze zullen we ook tegenkomen in dit

boek. Het *Viewer* venster is iets nieuwer dan het *Plots* venster en neemt de interactieve visualisaties zoals kaarten voor zijn rekening.

Deze vensters in RStudio geven samen een uitstekende ervaring voor het data analyseren met R. Ongetwijfeld zal je dit zelf ervaren. De zojuist behandelde onderdelen zijn slechts het topje van de ijsberg qua features in RStudio. Voor het verdere verloop van dit boek zijn de belangrijkste in ieder geval behandeld.

Datasets

Bij sommige hoofdstukken in dit boek wordt bij de voorbeelden gebruik gemaakt van databestanden. De meeste databestanden zijn al in R beschikbaar, in dit boek wordt daarbij uitgelegd hoe je deze gebruikt. Sommige databestanden worden in R geïmporteerd. Dit zijn de volgende databestanden welke op mijn blog kunt downloaden:

Databestanden:

Consumentenprijzen.csv

Projecten.csv

Projecten2.csv

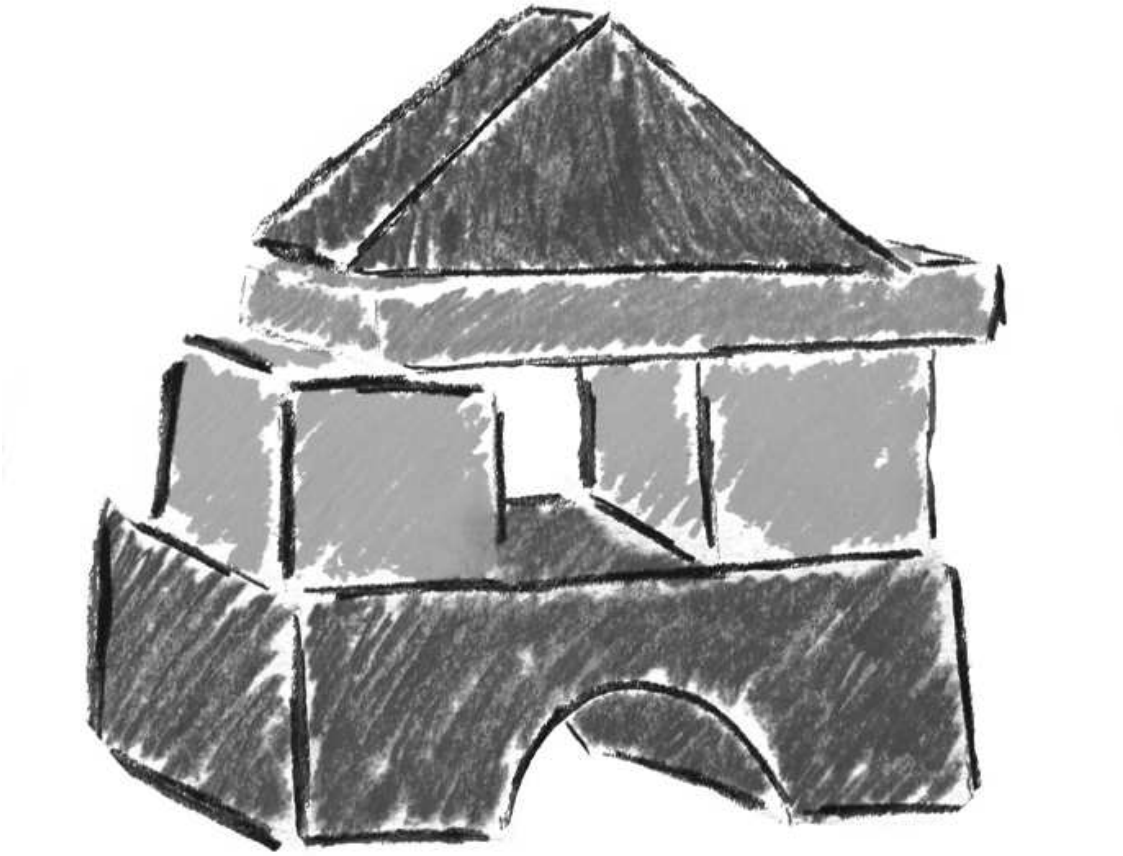
Voor een overzicht van alle datasets (ook van de oude tutorial) kun je naar de pagina van mijn oude R cursus gaan op http://r-handleiding.blogspot.nl/p/voorbeeld-databestanden-downloaden.html .

Referenties

Op sommige plekken in dit boek wordt er doorverwezen naar een bron. Dit zijn bronnen waar een bepaald onderwerp in dit boek extra wordt besproken. Het is nuttig om deze bronnen na te gaan als je, naast de rode lijn in dit boek, dieper wilt ingaan over een bepaald onderwerp. De bronnen zijn genummerd. Voor de lezers die de PDF-versie van dit boek hebben, kunnen ze eenvoudig klikken op de titel om naar de oorspronkelijke bron te gaan. Lezers van de hardcopy (Boek) versie van dit boek kunnen door middel van de nummering van bronnen het betreffende nummer opzoeken in het hoofdstuk **Referenties** dat zich aan het einde van dit boek bevindt. Achter de nummers van de bronnen staat informatie om de bron te raadplegen.

De basics van de R-syntax

In dit hoofdstuk beginnen we bij de basics van R. Namelijk het aanmaken van objecten en hier waarden aan toewijzen. Je zult zien dat R voornamelijk uit vectoren staat die een enkele of meerdere waarden bevatten. Dit kunnen getallen of andere datatypen zijn. Op deze vectoren worden toepassingen gedaan, bijvoorbeeld berekeningen of functies. Eigenlijk is dat grofweg wat je continu doet in R, waarden toewijzen aan vectoren en bewerkingen toepassen met bijvoorbeeld functies. Veel succes met het schrijven van je eerste stukjes R-code!



Variabelen aanmaken in R

R werkt over het algemeen volgens dezelfde principes als andere programmeertalen. Programmeren in R is vooral (1) variabelen aanmaken en definiëren en (2) functies toepassen op deze variabelen. Als we bijvoorbeeld een variabele willen aanmaken die de naam `"Peter"` bevat, doen we dat als volgt:

```
# Aanmaken van variabelen doe je met <- of met =

naam = "Peter"

# of
naam <- "Peter"
```

> *Je kunt `=` of `<-` gebruiken om waarden toe te wijzen in R. Deze functies doen exact hetzelfde, kies daarom wat jouw voorkeur heeft. Over het algemeen wordt er vaak voor `<-` gekozen in R. Veel mensen hebben de vraag waarom deze twee varianten bestaan als ze toch hetzelfde doen. Een artikel op Revolution Analytics heeft hier een mooie en korte uiteenzetting voor **[1] Use = or <- for assignment?**, (Revolution Analytics, 2008)*

Als we een variabele `leeftijd` willen aanmaken en daar het getal `34` aan willen toewijzen, gaat dit op dezelfde manier. Nadat de variabele is aangemaakt, kunnen we de waarde die eraan is toegewezen terugvinden door simpelweg de variabele `leeftijd` weer in te voeren. Het is namelijk één van de principes van programmeertalen om een toegewezen waarde te onthouden. Zo weet `R` in deze sessie dat aan de variabele leeftijd de numerieke waarden `34` is toegewezen.

```
leeftijd <- 34

#Geeft:
leeftijd


## [1] 34
```

Hetzelfde geldt voor het toewijzen van een waarde met een ander datatype, bijvoorbeeld een woord. We maken de variabele `woonplaats` aan die de waarde `"Almere"` bevat.

```
#of
woonplaats <- "Almere"

#Geeft:
woonplaats
```

```
## [1] "Almere"
```

> ***Alle waarden in R zijn vectoren*** *Een vector is een object dat informatie, en in het geval van R, gegevens bevat. De output de vorige voorbeelden geven getallen vóór de waarden aan ([1]). Dit geeft aan dat het de eerste waarde is uit de vector. Zo staat* `"Sassenheim"` *op plaats 1 in de vector* `woonplaats`*. Je kunt dus stellen dat alle waarden in R-vectoren zijn, zelfs als er slechts een waarde is toegewezen.*

Vectoren met meerdere waarden in R

Een vector kan ook meerdere waarden bevatten. Om meerdere waarden toe te wijzen aan een vector, gebruik je de `c()` functie. Dit kun je makkelijk onthouden door bijvoorbeeld aan de *concatenate* functie te denken van **Excel**.

We kunnen bijvoorbeeld een vector maken die drie waarden bevat:

```
# een vector 'woonplaatsen' aanmaken die 3 waarden bevat

woonplaatsen <- c("Almere", "Katwijk", "Utrecht")

# de waarden van de 'woonplaatsen' vector bekijken

woonplaatsen


## [1] "Almere"  "Katwijk" "Utrecht"
```

Zoals je ziet, heeft de vector `woonplaatsen` 3 waarden opgeslagen. Raak niet in verwarring door de `[1]` die wordt aangegeven. Dit betekent niet dat de eerste waarde uit de `woonplaatsen` vector `"Almere", "Katwijk", "Utrecht"` is, maar dat de eerste waarde op plaats `1` staat. Per 10 waarden in een vector wordt namelijk een nummering (indexing) aangegeven. Dit kun je zien als we bijvoorbeeld 12 waarden toewijzen aan een vector.

```
woonplaatsen <- c("Sassenheim", "Katwijk", "Enschede", "Groningen",
                  "Maastricht", "Rotterdam", "Amsterdam", "Leeuwarden", "Asse
n","Almere", "Utrecht", "Den Haag")

woonplaatsen


##  [1] "Sassenheim" "Katwijk"    "Enschede"   "Groningen"  "Maastricht"
##  [6] "Rotterdam"  "Amsterdam"  "Leeuwarden" "Assen"      "Almere"
## [11] "Utrecht"    "Den Haag"
```

Op deze manier kun je bijvoorbeeld makkelijk aflezen dat `"Den Haag"` op plaats 12 staat in de `woonplaatsen` vector. Deze indexering kun je ook handig als je de waarde die op een bepaalde plek staat in een vector wilt ophalen. Als we bijvoorbeeld de waarde van de `woonplaats` willen weten die op plaats `7` staat, doen we dat als volgt:

```
# de waarde ophalen die op plaats 7 staat van de 'woonplaats' vector

woonplaatsen[7]


## [1] "Amsterdam"


# de woonplaatsen ophalen die op de plaatsen 3 en 8 staan in de 'woonplaats'
vector
woonplaatsen[3:8]


## [1] "Enschede"   "Groningen"  "Maastricht" "Rotterdam"  "Amsterdam"
## [6] "Leeuwarden"
```

Let op dat de indexering van de selectie die je gemaakt hebt nu weer anders is. In dit geval staat `"Amsterdam"` namelijk op plaats 5.

> ***De index*** *van R is 1. Dit betekent dat reeksen van vectoren of andere onderdelen in R bij 1 beginnen. Dit klinkt logisch, maar het is bij programmeren meer gebruikelijk dat reeksen bij 0 beginnen. Dit worden talen genoemd met een zero based indexing. Bij onze* `woonplaasten` *vector in het voorbeeld zou in de programmeertaal Python het eerste element* `"Sassenheim"` *op plaats* `0` *staan, terwijl het bij R gewoon op plaats* `1` *staat.*

Vectoren samenvoegen

Je kunt ook waarden uit andere vectoren samenvoegen. Hiervoor voer je in plaats van de absolute waarden, de namen van de vectoren in. Dit wordt in het volgende voorbeeld gedemonstreerd.

```
# de waarde "Nina" toewijzen aan de 'naam' vector

naam <- "Nina"

# de waarde 23 toewijzen aan de 'leeftijd' vector

leeftijd <- 23

# de waarde "Amsterdam" toewijzen aan de 'woonplaats' vector
```

```
woonplaats <- "Amsterdam"

# een FALSE waarde toewijzen aan de 'rookt' vector

rookt <- FALSE

# de waarden van alle vectoren bekijken

naam


## [1] "Nina"


leeftijd


## [1] 23


woonplaats


## [1] "Amsterdam"


# de vector 'persoon' aanmaken en de waarden van de andere vectoren hier aan
toewijzen

persoon <- c(naam, leeftijd, woonplaats, rookt)

persoon


## [1] "Nina"      "23"        "Amsterdam" "FALSE"
```

Op deze manier wijs je 4 waarden toe aan de `persoon` vector.

Verschillende datatypen in R

Waarden en variabelen hebben verschillende klassen en datatypen. Voorbeelden zijn getallen (`numeric/integer`), woorden (`character/string`), boolean (`TRUE` of `FALSE`). Ook zijn er types zoals matrixen (`matrix`), data frames (`data frame`), lijsten (`list`). Deze types zijn verzamelingen van data.

Ook onze zojuist aangemaakte variabelen hebben een klasse. R heeft deze automatisch toegewezen. Met de `class()` functie kunnen we er achter komen welk datatype een variabele is.

```
naam <- "Geertje"

class(naam)


## [1] "character"


class(leeftijd)


## [1] "numeric"
```

De klasse van de variabele `naam` is een `character` en `leeftijd` is `numeric`. Hierbij kun je het `character` datatype vergelijken met het `string` datatype die je in andere programmeertalen tegenkomt. Het `numeric` datatype in R kun je het best vergelijken met het `float` datatype die je in andere programmeertalen tegenkomt.

Automatische definitie van datatypen in R

Zoals je misschien hebt gemerkt, hoef je in R niet aan te geven welke klasse een variabele moet krijgen. Dit doet R automatisch. Bij een numerieke waarde weet R namelijk dat het om een `numeric` klasse gaat, net als voor bijvoorbeeld een woord of een naam automatisch het `character` datatype wordt toegewezen.

> *Bij andere programmeertalen zoals Java en C# is het nodig om aan te geven welk datatype de nieuwe variabele moet hebben, bijvoorbeeld:*

C#: `static string naam = "Arie";`

R: `naam <- "Arie"` of `naam = "Arie"`.

Datatypes van variabelen in R veranderen

Het is handig dat R automatisch de klasse definieert. Echter ben je soms niet tevreden met het datatype dat R automatisch toewijst. Daarom kun je in R ook zeer eenvoudig het datatype veranderen. Als wij bijvoorbeeld de variabele leeftijd willen veranderen van `numeric` naar `character`, gebruiken wij de `as.character`-functie.

```
# de variabele "leeftijd" definiëren
```

```
leeftijd <- 26
class(leeftijd)


## [1] "numeric"


#een variabele "leeftijd2" aanmaken die met "string" als datatype

leeftijd2 <- as.character(leeftijd)
leeftijd2


## [1] "26"


class(leeftijd2)


## [1] "character"
```

Echter kunnen niet alle variabelen naar de gewenste datatype geconverteerd worden. Zo kun je bijvoorbeeld een woord niet omzetten naar een numerieke waarde.

```
naam <- "David"

# een nieuwe variabele "naam2" aanmaken met een naam geconverteerd naar numer
iek

naam2 <- as.numeric(naam)


## Warning: NAs introduced by coercion


class(naam2)


## [1] "numeric"


naam2


## [1] NA
```