CHESTER ISMAY, BEWERKT DOOR HARRIE JONKMAN

GEWEND RAKEN AAN R, RSTUDIO EN R MARKDOWN

Contents

1 Introductie 5

2 Waarom R? 7

3 R en RStudio Basis 11

3.1 Waarom R? 11

3.2 Wat is RStudio? 12

3.3 Werken in RStudio Server 13

3.4 RStudio Layout 15

4 R Markdown 19

4.1 Fouten oplossen in een R Markdown file 19

4.2 De componenten van een R Markdown File 20

4.3 R Markdown Chunk Opties 24

4.4 Algemene richtlijnen voor het schrijven van R Markdown Files 25

4.5 Help -> Cheatsheets 25

5 R Analyse met gebruikmaking van R Markdown 27

5.1 Een beginnende werkwijze 27

5.2 R gebruiken met de periodieke table dataset 27

5.3 Gemiddelde, mediaan, standaard deviatie, samenvatting in vijf, distributie 27

5.4 Aanvullende inhoud die gedekt moet worden 27

5.5 R Markdown templates 27

6 Algemene fouten in R detecteren 29

7 Bibliography 31

1
Introductie

Dit boekje is ontworpen om nieuwe gebruikers van R, RStudio en R Markdown te laten wennen met enkele introductiestappen die nodig zijn om met je eigen reproduceerbaar onderzoek te beginnen. Het geeft ook een overzicht van veel algemene R fouten (en wat ze in leken termen betekenen). Hieronder ook veel screenshots en GIFs, maar als meer verklaring of uitleg nodig is of wat ook wat het boek betreft, alsjeblieft laat dat via GitHub hier weten hier of email Chester of mij met een referentie naar de fout/gebied waar meer begeleiding nodig is. Push verzoeken met betrekking tot typos of verbeteringen zijn ook zeer welkom.

Dit boek evolueert en zal indien nodig worden bijgesteld gebaseerd op feedback. Controleer de datum aan het begin van dit hoofdstuk om te zien wanneer het boek voor het laatst is ververst. In aanvulling, elk hoofdstuk laat de datum van de laatste update zien.

2
Waarom R?

Als R en programmeren helemaal nieuw voor jou is, kan het zijn dat je je een beetje ongerust maakt. Je bent niet gewend om commando's te typen die de computer vertellen wat er moet gebeuren. Je bent meer gewend aan drop-down menu's en andere grafische interfaces voor de gebruiker die je in staat stellen te doen wat je wilt doen. Dus waarom stappen dan zoveel bedrijven, hogescholen/universiteiten en individuen met allerlei disciplinaire achtergronden over op het gebruiken van R?

Op deze vragen zijn vele antwoorden te geven, maar de meest belangrijke voor ons nu zijn:

1. R is gratis. RStudio is gratis.

   Een van de belangrijkste voordelen voor ons wat het werken met R en RStudio is dat ze allebei gratis te gebruiken zijn. R is een open-source programmeertaal die de laatste jaren ontzettend sterk gegroeid. Dagelijks versterken ontwikkelaars de functionaliteit van het programma en de pakketten die worden opgeleverd. Waar andere meer commerciele pakketten zijn blijven steken in de donkere jaren (de jaren negentig, bijvoorbeeld) van ontwikkeling en ontzettend duur kunnen zijn in het gebruik, blijft R een gratis alternatief dat gebruikers op allerlei niveaus in staat stelt om bij te dragen.

   RStudio is a grafische gebruikersinterface die iemand iemand in staat stelt om met een R code te werken en de resultaten van die code op een eenvoudige manier te zien. Ook RStudio kan ook gratis worden binnengehaald om mee te werken.

2. Analyses die in R zijn gedaan zijn reproduceerbaar.

   Omdat veel wetenschappelijke velden zich bewegen naar meer reproduceerbare analyses, zijn de erkende wijs- en klik systemen eigenlijk een hindernis geworden voor dat proces. Als je de analyses opnieuw moet uitvoeren met die systemen, zul je heel voorzichtig jouw analyses, tekst en plots moeten kopieren en plakken in jouw teksteditiing-programma, helemaal het begin tot en met het einde. Iedereen die weet heeft van dit soort kopieren- en plakken weet dat het vatbaar is voor fouten en ongelovelijk langdradig kan zijn.

   Als je de werkwijze volgt die in dit boekje wordt beschreven, zullen jouw analyses reproduceerbaar worden zodat jij je geen zorgen meer hoeft te maken over deze kopieer- en plakzaken. Zoals je nu misschien wel vermoedt, is het veel beter als je in staat bent om jouw code en data inputs te definieren en jouw analyses opnieuw te draaien dan je zorgen

te maken over of jouw resultaten handmatig zijn over te zetten van het ene naar het andere programma. Reproduceerbaarheid helpt jou als programmeur ook omdat je vooral met jezelf te maken hebt als je over een langere periode aan een project werkt. In plaats van heel zorgvuldig alle stappen van het proces zorgvuldig op te schrijven opdat je de goede drop-down optie kiest, wordt jouw hele code opgeslagen.

3. R gebruiken maakt samenwerking makkelijker.

   Dit helpt jou ook bij samenwerking omdat, zoals je later zult zien, een R Markdown-code, die de hele analyse, documentatie, commentaar en code bevat, kan worden gedeeld met anderen. Dit reduceert tijd die nodig is om met anderen te werken en beperkt kansen op fouten die gemaakt worden wanneer met wijs- en kliksystemen wordt gewerkt. De mantra hier is **Zeg Nee tegen tegen Kopieren en Plakken!**, goed voor jouw eigen gezondheid en in het belang van de wetenschap.

4. Antwoorden vinden op vragen is veel makkelijker.

   Als je ooit wat te doen hebt gehad met software, weet je hoe moeilijk het is om antwoorden te vinden op jouw vragen. “Hoe kan ik dit proces iemand anders duidelijk maken? Moet ik screenshots maken? Moet ik echt de IT-afdeling bellen en uren wachten tot iemand een antwoord geeft?” R is een programmeertaal en daarom is het veel makkelijker (na wat praktijkoefeningen) om Google of Stack Overflow te gebruiken om antwoorden te vinden op jouw vragen. Je zult verbaasd zijn over hoeveel gebruikers dit soort fouten zijn tegengekomen die jij zult zien als je begint.

   Ik vraag Google vaak(bijna op dagelijkse basis)zaken als “Hoe maak ik een side-by-side boxplot in R gekleurd door een derde variabele?”. Je zult beter worden in het werken met R door anderen hulp te vragen en vragen te beantwoorden die anderen hebben. In aanvulling, Hoofdstuk 6 beschrijft veel voorkomende fouten en hoe je ze kunt oplossen.

5. Door het programmeren heen ploeteren helpt jou het te leren.

   We weten allemaal dat leren niet makkelijk is. Heb je moeite om een lijst met meer dan 10 stappen die je moet volgen te onthouden of zo? Heb je moeite om steeds maar weer te bedenken wat de volgende stappen in het proces zijn? Dat is heel normaal vooral als je de procedure een tijd lang niet hebt doorlopen. Het leren volgen van een procedure is makkelijk op de korte termijn, maar kan heel frusterend zijn omdat je het niet op de langere termijn vasthoudt. Programmeren (als het goed wordt gedaan) ondersteunt lange termijn denken in plaats van korte termijn oplossingen.

   Een ongemakkelijke zaak die we vaak voor een gegeven aannemen is dat onze hersenen ons altijd de makkelijke weg laten kiezen. Als je echt iets wilt leren te doen (zoals het programmeren met R), moet je je bij tijd en wijle gefrustreerd voelen. Iets leren is vaak aan frustratie gekoppeld. (We zijn geneigd al deze frustratie te vergeten en alleen maar denken aan waar we nu zijn.) R frustreert mij nog steeds met regelmaat, maar ik groei met de praktijkervaringen en ik zie nu graag uitdagingen. Hadley Wickham heeft dit fenomeen aardig omvat in de proloog van het boek “Hands-On Programming with R” (Grolemund, 2014):

   > Als je leert programmeren, zul je gefrustreerd raken. Je leert een nieuwe taal en het duurt een tijd voordat het allemaal vloeiend gaat. Maar frustratie is niet alleen iets heel natuurlijks,

> het is eigenlijk een positief teken waar je naar uit zou moeten kijken. Frustratie is de manier waarop jouw hersenen lui worden; ze proberen jou te laten stoppen en iets makkelijks of voor het plezier te laten doen. Als je fysiek fitter wilt worden, moet je je lichaam uitdagen zelfs als het klaagt. Als je beter wilt worden in programmeren, dan moet je je hersenen uitdagen. Herken deze frustratie en zie het als iets positiefs: het rekt jezelf uit. Jezelf elke dag wat uitdagen en je zult snel een programmeur met vertrouwen zijn.

Last updated: By chester on Saturday, July 29, 2017 14:11:02 PDT

3
R en RStudio Basis

3.1 Waarom R?

In het vorige hoofdstuk besprak ik de vele redenen waarom jij je analyses (vooral data-analyses) met R moet doen. Als je dat hoofdstuk hebt overgeslagen in de hoop snel wat van R te leren, adviseer ik er naar terug te gaan en het zorgvuldig over te lezen. Als je met R begint is het vooral belangrijk dat introductiehoofdstuk van tijd tot tijd terug te lezen.

3.1.1 R's begin

R is gemaakt door een groep statistici die een alternatief wilden voor de kostbare opties. Omdat het door statistici is gemaakt (in plaats van door computerwetenschappers) betekent dat R wat eigenzinnige aspecten heeft die wat tijd kunnen kosten om aan gewend te raken. We zullen zien dat er veel pakketten zijn ontworpen om jou hierbij te helpen en dat je geen geavanceerde opleidingen nodig hebt om snel met R te kunnen werken.

Terug naar de ontwikkeling van R. R is gemaakt door **R**oss Ihaka en **R**obert Gentleman uit Nieuw Zeeland die aan de Universiteit van Auckland werkten. Het is een spin-off van de S programmeertaal en genoemd naar de eerste letters van de namen van deze ontwikkelaars (zoals je in de benadrukking hierboven kunt zien). De eerste ideeen om R te ontwikkelen ontstonden in 1992 en de eerste versie van R kwam in 1994 uit. Je kunt veel meer over de achtergronden van R, zijn kenmerken en de verbinding met de S-taal op de Wikipedia pagina.

3.1.2 R pakketten

Toen ik student was op de Northern Arizona Universiteit leerde ik in 2007 voor het eerst R te gebruiken van Dr. Philip Turk. Toen kon ik niet vermoeden dat het aantal R-gebruikers zo zou stijgen zoals we dat vanaf 2011 hebben gezien. Ik had nooit gedacht dat studenten die een introductiecursus statistiek volgen zouden worden aangemoedigd om R te gebruiken.

In 2007 was het vooral nog esoterisch en een lastige taal die door statistici werd gebruikt voor analyses. Om gewend te raken aan het maken van plots en het werken met data was vooral lastig voor hen die weinig tot geen programmeerervaring hadden. Maar wat veranderde er nu sinds 2007 aan het leren van R?

Ik geloof dat een van grootste ontwikkelingen het verschijnen van pakketten was die het werken met R voor beginners eenvoudiger maken. Er verschenen pakketten door R-gebruikers

die de de functionaliteit van de basis installatie versterkten. Pakketten gemaakt door Hadley Wickham en anderen hebben de laatste jaren sterk aan de mogelijkheden van R bijgedragen, terwijl ook het beginnen met R werden versimpeld. De Wikipedia pagina vermeldde (january 2016) dat er rond de 7800 aanvullende R pakketten beschikbaar zijn.[1]

Een andere belangrijke ontwikkeling is de grafische interface voor gebruikers die `RStudio` wordt genoemd en het pakket `rmarkdown` dat door mensen van RStudio, Inc. is gemaakt. We zullen het nog hebben over `rmarkdown` (ook wel R Markdown genoemd) in een apart hoofdstuk 4. Eerst hebben we het over RStudio.

[1] Je ziet, bijvoorbeeld, hoe je deze pakketten moet downloaden via `install.packages("dplyr")` en ze in jouw R werk omgeving moet laden via `library("dplyr")`in hoofdstuk 5.

3.2 Wat is RStudio?

RStudio is een krachtig, gratis, open-source integreerde omgeving voor R. Die ontwikkeling begon in 2010 en de eerste beta-release verscheen february 2011. Het is beschikbaar in twee edities: RStudio Desktop en RStudio Server. Dit boekje zal de nadruk leggen op RStudio Server, maar beide versies zijn vrijwel hetzelfde om mee te werken.

Je vindt instructies om R en RStudio binnen te halen hieronder, zowel voor Windows en Mac machines. Als je RStudio Server gebruikt, zal jouw leraar, professor of mensen van de IT afdeling deze stappen voor jouw zetten. Op de RStudio Server log je via een web browser op een account in de cloud in. Voor beginners zitten er veel voordelen aan het gebruik van RStudio Server waaronder het delen van R projecten die je steunen met feedback en het oplossen van fouten. Installeren van software zorgt zo voor zijn problemen en kan hoofdpijn veroorzaken en daar heb je niks mee te maken als je RStudio Server gebruikt.

Opmerking voor de geoefende gebruiker: Je kunt ook je eigen RStudio Server installeren voor rond de $5 per maand via Digital Ocean. Instructies om dat te doen kun je vinden via Dean Attali hier en op de Digital Ocean site hier.

Nadat je een paar maanden met RStudio Server hebt gewerkt, wordt het aangeraden om RStudio Desktop op jouw computer binnen te halen. De instructies om dat te doen vind je hier beneden.

3.2.1 R en RStudio Desktop installeren

Het is belangrijk op te merken dat je RStudio Desktop niet kunt installeren zonder R te installeren omdat RStudio R nodig heeft om te kunnen draaien. Een stap-voor-stap richtlijn om R en RStudio Desktop te installeren met screenshots kun je vinden

- hier voor de Mac en
- hier voor een PC.

Als je PDF documenten wilt maken (die een download van *LaTeX* nodig heeft) kun je enkele van de latere stappen van de installatie overslaan. Het wordt aangeraden dat je **HTML** als de 'Default Output Format' voor R Markdown instelt. Daarover vind je meer in Hoofdstuk 4.

3.3 Werken in RStudio Server

3.3.1 Inloggen en het beginscherm

Met RStudio Server kun je via het web analyses in R draaien. Dat betekent dat je alleen maar een internetverbinding nodig hebt en een webbrowser om je analyses te kunnen draaien. Jouw leraar of ondersteuner zal jou de link naar het internetadres van jouw RStudio Server geven. Nadat je de link het ingevuld, zul je een pagina zien die hier op lijkt:

```
https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/screenshots/server_login.png
```

Nadat je hebt ingelogd met jouw wachtwoord en password, ziet het beeld er een beetje zo uit.

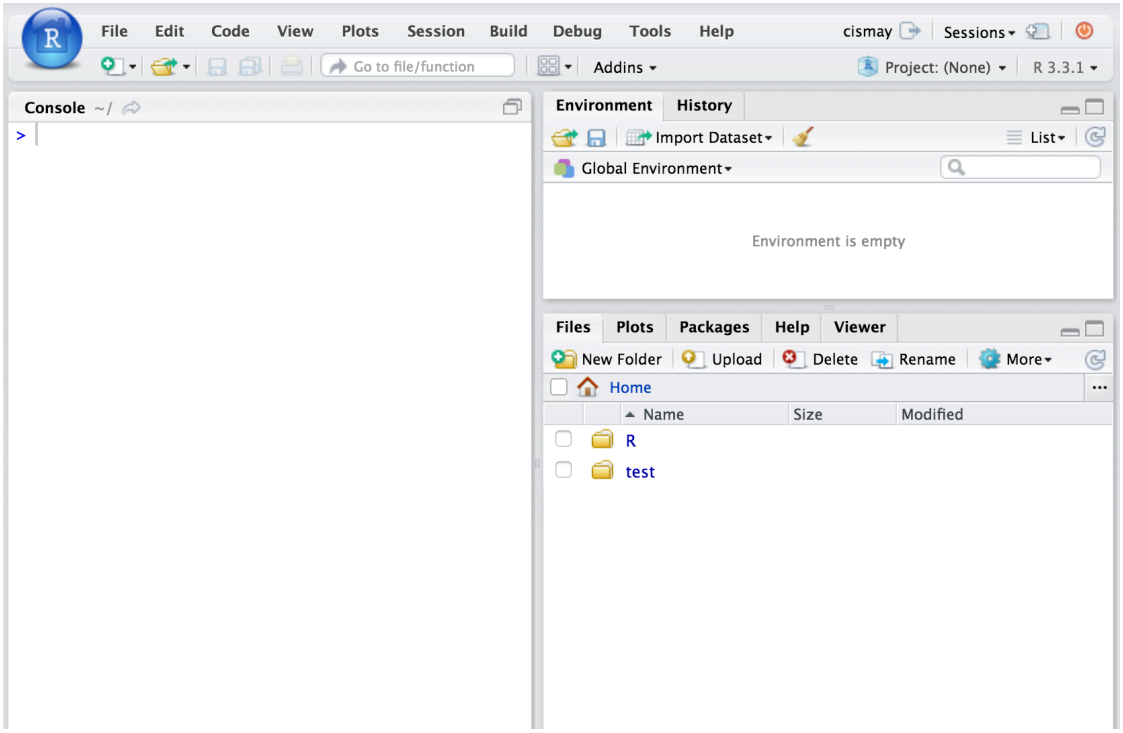

Figure 3.1: Beginpagina voor RStudio Server

Een screenshot van RStudio Desktop ziet er zoiets uit:

Zoals je ziet zijn ze vrijwel identiek. Daardoor wordt het overstappen van de een naar de andere RStudio set-up heel eenvoudig. Uitleg over de drie verschillende RStudio schermen (dat zullen er weldra vier worden) en hun betekenis komt aan de orde in hoofdstuk 4. Je zult zien dat veel van wat volgt ook betrekking heeft op RStudio Desktop (behalve het Shared Projects kenmerk). Maar het is altijd aan te raden om een RStudio project te maken ongeacht of je in cloud of lokaal werkt.

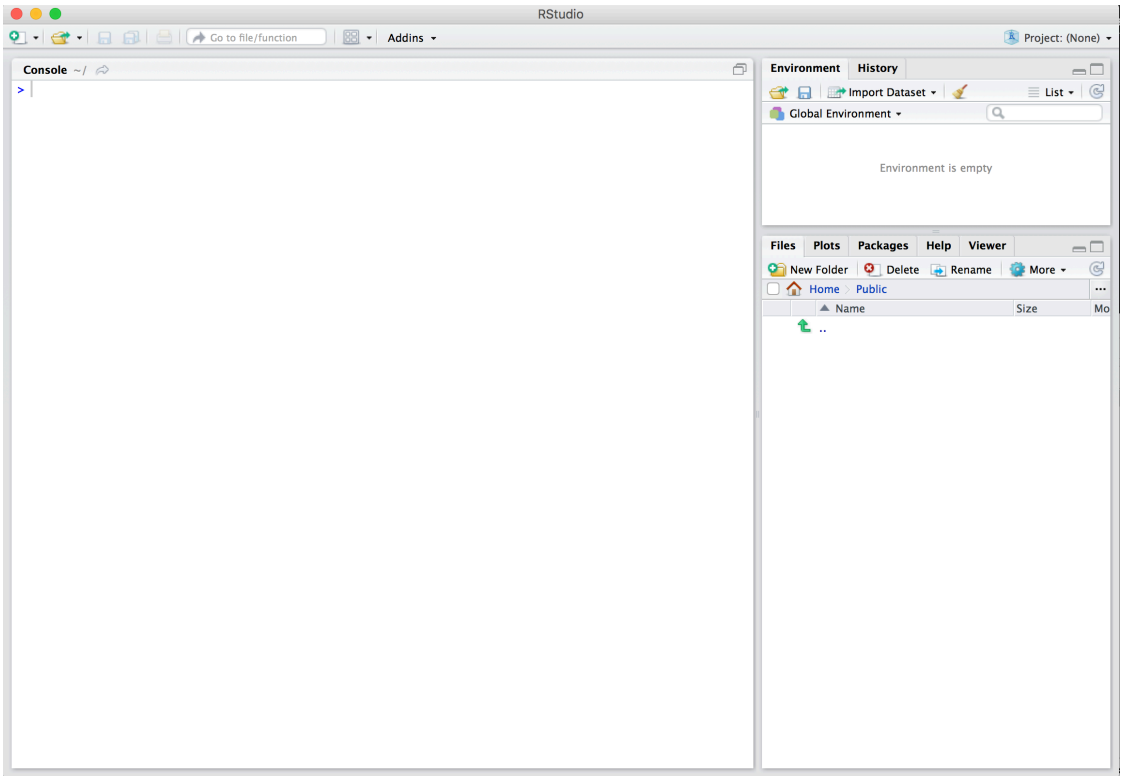

Figure 3.2: Beginpagina voor RStudio Desktop

3.3.2 Basis Workflow met RStudio

Een goede methode om een nieuw project met R-code te starten is om met een nieuw RStudio project te beginnen en daarin te werken. RStudio project files hebben de extensie `.Rproj` en slaan metadata op die te maken hebben met de documenten die je hebt opgeslagen en informatie over de R omgeving waarin je werkt. Meer informatie over RStudio projecten is beschikbaar via RStudio, Inc. hier.

Als je huiswerk of labopdrachten deelt met jouw instructeur kan het bijvoorbeeld handig zijn om een RStudio project te starten, dit te delen met jouw instructeur en dan nieuwe folders aan te maken voor elk lab. We zullen dat voorbeeld hieronder volgen.

Het gifbestand hieronder laat je zien hoe je een nieuw RStudio project start dat `initial` heet en tevens jouw eerste R Markdown file. Let op dat dat je een beschrijving ziet over welke versie van R draait op jouw initiele login zoals het gif in het scherm laat zien.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/proj_rmd.gif`

We hebben hier onze **first_rmarkdown.Rmd** file opzet.

3.3.3 Projecten op RStudio Server Pro delen

Je zult nu een voorbeeld zien hoe je een project met een ander deelt. Dit zal jou in staat stellen om jou en mensen waarmee je samenwerkt (andere studenten, jouw instructeur, etc.)

op de **Rmd** op hetzelfde moment. Dat is hetzelfde als op hetzelfde moment in een Google Doc te werken als iemand anders.

RStudio Server komt in een aantal verschillende formats en je zult ervoor moeten zorgen dat jij (of jouw IT medewerker) RStudio Server Pro hebt geinstalleerd om Shared Projects te kunnen gebruiken. Meer informatie over RStudio, Inc. vind je op: hier. Hieronder is een gif-bestand voorbeeld om dit **initial.Rproj** project file te delen met een andere gebruiker van RStudio Server.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/share_proj.gif`

We kunnen nu samen aan dit project werken, de files op de gemeenschappelijke folder opslaan waar **initial.Rproj** huizen en ons commentaar typen en de code in **first_rmarkdown.Rmd** plaatsen of binnen andere files.

In Hoofdstuk 4, zul je zien waarom het aangeraden wordt om te werken in R Markdown files en je zult ook sommige voorbeelden zien van hoe R met R Markdown werkt.

3.4 RStudio Layout

Misschien word je aanvankelijk wel een beetje overspoeld door al die verschillende panelen en tabs die beschikbaar zijn in RStudio. Je zult binnenkort wel van deze layout gaan houden maar, zoals met alle dingen, kost het jou wat tijd om eraan gewend te raken. We zullen met het linksboven paneel beginnen, doorgaan naar het paneel rechtsboven, dan naar het linksonder-paneel en tot slotte naar het onderste rechter paneel. Deze panelen kunnen geheel worden aangepast maar het wordt beginners aangeraden om dingen in het begin in de standaard layout te houden.

3.4.1 Code Editor / View Window

Aannemelijk dat het paneel waar je de meeste tijd doorbrengt het paneel links boven is waar je de meeste tijd doorbrengt. Bij de eerste keer inloggen was het er nog niet, maar het kwam naar voren toe we het **first_rmarkdown.Rmd** maakten. Dit paneel dient als een plaats waar je inhoud en de inhoud van de files en objecten in R kunt zien. In het GIF-bestand hieronder, kun je zien dat we de tekst in de file kunnen aanpassen en de file dan kunnen opslaan.

Let op dat de tab de kleur van de filename zal veranderen van zwart naar red en er verschijnt een asterisk na de filename. Dit is om jou eraan te herinneren dat de file niet is opgeslagen. Je zult je de gewoonte eigen moeten maken om files regelmatig op te slaan. Je kunt verschillende tabs open hebben staan en verschillende files in dit paneel kunnen zien. Je zult ook zien dat je met `View` datasets in dit paneel kunt zien en daar komen we in Hoofstuk **??** op terug.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/top_left_pane.gif`

Misschien is het nog niet helemaal duidelijk wat deze veranderingen te weeg brengen. Je zult op de **Knit HTML** knop drukken boven het linker boven paneel om al jouw tekst, code en output samen te brengen. We zijn er ook nog niet helemaal!

3.4.2 Omgeving / Geschiedenis (Environment / History)

Het volgende paneel is standaard dat van de **Environment** tab en een **History** tab. Om een gevoel te krijgen wat deze panelen bieden moeten we ook de linksonder **Console** tab gebruiken. Ik zal je laten zien hoe verschillende objecten in R zijn te gebruiken via **Console**. Eerst zul je zien dat **Environment** tab ons vertelt dat de "Environment" leeg is. Als je op de **History** tab drukt, zul je een blank scherm zien met enkele iconen. We zullen hier niet alle knoppen uitleggen, maar je bent misschien aangemoedigd om het uit te proberen en op ze te klikken om een gevoel te krijgen van wat ze doen. Als ik de code in de **Console** tik kijk hoe de **Environment** en de **History** tabs veranderen.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/env_history.gif`

Wat de **Console** betreft, kun je aan een plaats denken waarbinnen je wat kunt spelen. Het is, zeg maar, jouw R zandbak. Je kunt jouw code uitproberen om er zeker van te zijn dat het werkt en het dan als je tevreden bent jouw tekst in de **Rmd** file plakken in een chunkvorm. We zullen hier meer voorbeelden van zien in de hoofdstukken die nog komen.

Let altijd op dat als je de code in de console zet, zoals ik deed met `sum_1_2`, deze het resultaat zal laten zien. Je hebt ook de assignment operator gezien die als `<-` werd geschreven. Je kunt dit lezen als dat de inhoud aan de rechterkant in een object geplaatst is en genoemd naar wat er aan de linker kant staat. Bijvoorbeeld, `num1` is de naam van een object dat de waarde 7 opslaat.

Een krachtig kenmerk van de R `sum_1_2` is naar voren gekomen in de `sum_1_2 <- sum(num1, num2)` regel. `sum` is een functie. Functies worden bij naam genoemd en dan volgt een haakje, hun argumenten verdeelt door komma's en dan weer door een haakje afgesloten. Als we verder gaan zul je meer voorbeelden hiervan zien. Natuurlijk zullen we R voor meer gebruiken dan voor een simpele rekenmachine zoals hier maar dit geeft jou een idee wat de **Environment** en **History** tabs opslaan.

3.4.3 Console

Je zult regelmatig de **Console** gebruiken als een manier om jouw werk te controleren of om na te denken over hoe je een probleem gaat oplossen met het gebruik van R. Voordat RStudio uitkwam, hadden R-gebruikers een scherm zoals de **Console** waar ze hun commando's in konden zetten en ze in andere schermen de resultaten konden zien. We zullen zien dat de **Console** en de **Code Editor/View Window** ons in staat stellen alles van onze code op te slaan in een file en dan deze code "runnen" door de **Console** om te controleren of het werkt. Het voorbeeld GIF-bestand hieronder laat zien hoe dit gedaan kan worden.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/console_code.gif`

Regels om objecten een naam te geven

Het is een goede gewoonte om variabelen een naam te geven die correspondeert met waar ze voor staan. Als je twee cijfersommen op elkaar deelt, kun je een naam als `ratio_van_sommen` nemen om naar dat object te verwijzen. R heeft enkele restricties op wat in de namen van objecten kan worden meegenomen:

1. Object namen kunnen niet met een getal beginnen.
2. Object namen kunnen niet symbolen omvatten die in de wiskunde gebruikt worden of andere handelingen die in R gebruikt kunnen worden. Deze symbolen omvatten onder anderen `$`, `@`, `!`, `^`, `+`, `-`, `/` en `*`.

Een andere belangrijke eigenschap van R is dat het lettertype gevoelig is. Je zult zien wat dit betekent in het GIF bestand hier beneden dat ook voorbeelden laat zien van invalide objectnamen. Let op dat we blijven werken binnen de R chunk zoals we dat in de laatste GIF deden. Je zult zien dat deze code chunks een aardige manier opleveren om onze analyses te kunnen vinden.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/naming_objects.gif`

Het viel je allicht op dat R enkele checks uitvoert en jou attendeert op enkele mogelijke fouten door een red X te plaatsen aan de links van de coderegel met een fout.Het zal niet alle fouten laten zien, maar het kan behulpzaam zijn. Let ook op dat `Naam`, `naam`, en `nAam` naar drie verschillende waarden verwijzen.

Iets anders dat moet worden opgemerkt is dat je getallen in een object kunt opslaan met een gegeven naam zoals we eerder hebben gedaan met `num1`. Als we een karakter string willen opslaan zoals “Chester”, kunnen we de naam van het object aan de linker kant en `<-` de string in aanhalingstekens aan de rechterkant `<-` plaatsen. Er zijn meer complexe objecttypes zoals je zult zien in hoofdstuk 5, maar het is altijd belangrijk te denken aan het verschil tussen een getal en een karakter in R.

Het is ook een goed idee om niet de namen van de functies te gebruiken die in R zijn ingebouwd. Het kan zijn dat je de optelling van twee getallen `sum` wilt noemen en R staat dat toe, maar het wordt ten **HOOGSTE** aangeraden dat je meer beschrijvende namen gebruikt en niet kiest voor objectnamen die hetzelfde zijn als gebruikelijke R functies. Iets als `sum_densities` is beter en minder gebruikelijk als naam van een functie.

3.4.4 De `help` functie (`?`)

Natuurlijk, je zult niet alle namen van de ingebouwde functies kennen voor dat je ermee werkt, maar het is iets waar je aan moet denken. Als je je ooit afvraagt of een functie ingebouwd is of in een pakket zit dat je gebruikt, kun je de `?` functie in de **Console** gebruiken om dat te controleren. Enkele voorbeelden worden hieronder als een GIF getoond.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/help_func.gif`

3.4.5 Het paneel rechtsonder

Het paneel rechtsonder in RStudio bevat de meeste standaard tabs en is een geschikte plaats om een varieteit van allerlei informatie over jouw RStudio project en zijn files.

Files

De meest linker toont jou de file en folder structuur. Dit laat jou zien waar de files opgeslagen worden, hoe ze genoemd worden en andere folders die er in jouw project folder mogen

zitten. Dit is hetzelfde als naar **Mijn Computer** gaan op een PC of de **Finder** openen op een PC. Tegelijk geeft dit de file en de directory structuur van ofwel in de cloud voor RStudio Server of op jouw lokale machine voor RStudio Desktop.

Plots / Viewer

Je zult duidelijke voorbeelden zien van wat de **Plots** en **Viewer** tabs bieden in Hoofdstuk 4. Als je wilt zal **Plots** jou de grafieken/figuren laten zien die jouw R code heeft gegenereerd. De **Viewer** tab kan jou het HTML file resultaat laten zien dat gemaakt is via een R Markdown **Knit**.

Pakketten

Je kunt een gevoel krijgen voor welke pakketten zijn gedownload op jouw computer/jouw cloud server door op de **Packages** tab te klikken. Je kunt ook zien welke pakketten in jouw omgeving zijn gedownload door te kijken of er een check-mark is aangeklikt naast de pakketnaam. Let op dat je misschien niet alle pakketten hebt gedownload op jouw machine zoals ik hier beneden in het GIF-bestand. Dat is prima. Het is alleen maar een voorbeeld van wat je mag verwachten.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/package_tab.gif`

Jou zal ook de **Beschrijving** van het pakket opvallen evenals het **Versie** nummer hier. Pakketten worden regelmatig ververst en verbeterd, dus dit is een manier om te controleren of je de meest recente versie hebt van een pakket. Onthoud dat hiervoor wordt gezorgd als je op een RStudio Server installatie werkt. Als je op een RStudio Desktop werkt, dan vind je de **Install** en **Update** allicht handig voor het binnenhalen of het verversen van je huidige geinstalleerde pakketten.

Help

We zagen ook een voorbeeld van het gebruik van **Help** tab als we de `?` functie intikken. Zo vind je documentatie over de R functies, datasets en pakketten. Als je een code ziet waar je niet helemaal zeker over bent, dan is het vaak goed een vraagteken te plaatsen gevolgd door de naam en te bekijken of de ingebouwde documentatie jou uit de brand kan helpen.

Voor het laatst geupdated:

```
## [1] "By chester on Saturday, July 29, 2017 14:11:03 PDT"
```

4
R Markdown

R Markdown biedt u de mogelijkheid om een rijk, volledig gedocumenteerd reproduceerbaar analyse uit te voeren. Het stelt de gebruiker in staat een enkele file samen te stellen met daarin al het commentaar, R code en de metadata die nodig zijn om de analyse van begin tot en met het einde te reproduceren. R Markdown kan chunks in de R code verwerken tegelijk met Markdown tekst. Dit kan samen komen tot een mooi HTML, PDF of Wordfile zonder dat kennis nodig is van HTML of LaTeX-code of zonder dat men lang bezig is om het in een goed Microsoft Word DOCX file te krijgen.

Een R Markdown file kan veel verschillende formats genereren en dit alles kan gedaan worden in een enkelvoudige tekstfile met een klein beetje opmaak. Ik denk dat je behoorlijk verrast zult zijn over hoe eenvoudig het is om een Markdown document te schrijven na jouw eerste kennismakingen.

4.1 Fouten oplossen in een R Markdown file

Nu gaan we terug naar de R Markdown file. Dit document hebben we in Hoofdstuk 3 gemaakt en heet **first_rmarkdown.Rmd**. We weten dat we enkele fouten hebben achterlaten in de variabelen hier en terwijl het misschien gek lijkt om de fouten te laten zien die we daar hebben gemaakt, laat het iemand die hier nieuw is goed zien met welke fouten we in het begin te maken kunnen hebben. We zien wat er gebeurt als we op de **Knit HTML** knop drukken met deze fouten. Dan schonen we de code op en zien wat het resultaat is van **Knit**.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/rmd_errors.gif`

Als je voor het eerst een R Markdown file maakt, is hier een basistemplate met code en tekst die er voor jou ingezet zijn. Dit om jou een gevoel te geven van hoe jij je eigen R Markdown file kunt maken met jouw eigen R code en jouw eigen commentaar. Wij hebben hier wat van die code aangepast. Ik besloot om alle regels in de `cars` chunkcode weg te halen zelfs al komen de fouten niet voor in de objectsdefinieringen waar namen in waren opgeslagen. Wij zien dat er een HTML-file in het **Viewer** paneel is gemaakt omdat **View in Panel** was geselecteerd.

Als je over de **Including Plots** tekst kijkt kun je verbaasd zijn als je ziet dat er geen plot in de R Markdown file verschijnt, maar in de HTML file is er een scatterplot die de verschil-

lende temperatuur- en drukwaarden aangeeft. Dit is iets waar eerder op is gewezen. R Markdown draait de opgeslagen code in R chunks en plaatst de output in HTML (of PDF of DOCX, etc.) formaat.

Je kunt ook zien dat de tekst verschijnt als commentaar voor en na de R code. Je zult snel begrijpen waarom de test "Inclusief Plots" text "Including Plots" zoveel groter is dan de andere tekst.

Belangrijke opmerking: Onthoud dat de R code die je wilt draaien in een chunk moet worden opgeslagen (in de goede volgorde) voor jouw analyse om reproduceerbaar te zijn en ook voor jou zelf om geen fouten te ontvangen als je **Knit**. Het is makkelijk om veel werk in de R **Console** te doen en dan te vergeten dat werk in een chunk aan jouw **Rmd** file te plaatsen. Dit is waarschijnlijk de eerste fout die je zult zien als je begint te werken in RStudio. Een voorbeeld van deze fout vind je in een GIF file hieronder.

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/forget_copy.gif`

De `object niet gevonden`-fouten samen met spellingsfouten en niet afgeronde R code segmenten zijn de meest voorkomende fouten in R. Je vindt hier meer over in Hoofdstuk 6.

4.2 De componenten van een R Markdown File

4.2.1 YAML

Het bovenste deel van de file wordt de YAML-kop genoemd. YAML betekent "YAML Ain't Markup Language" en de website `http://yaml.org` definieert er het volgende statement over:

> YAML is een menselijke dataserialisatie standaard voor alle programmeertalen.

Essentieel, YAML slaat de metadata op die voor het document nodig zijn. Een voorbeeld van zo'n YAML kop komt van onze **first_rmarkdown.Rmd** file:

```
---
title: "First RMarkdown"
author: "Chester Ismay"
output: html_document
---
```

Er zijn verschillende velden die kunnen worden aangepast in de YAML kop. Het belangrijkste om op te merken zijn de drie streepje waarmee de YAML kop begint en eindig. Inspringen speelt ook een belangrijke rol bij YAML dus wees zorgvuldig bij het uitlijnen van de tekst.

4.2.2 Koppen

`https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/headers.gif`

Je kunt verschillende grootes maken door simpel een of meer `#` toe te voegen aan het begin van de tekst waar je kop wilt plaatsen.

4.2.3 Vet

Als je een hash-tag in de tekst van jouw R Markdown document ziet, weet je dat dit zal corresponderen met de grotere vette tekst[1] die de start van een sectie van jouw document aangeven.

Dit is een van de aardige kenmerken van R Markdown. Je kunt simpel naar de kale tekst kijken en weten dat het in de 'knitted' document terug zal komen. We kunnen ook verschillende stijlen van nadrukken van woorden, frases of zinnen toevoegen door ze met symbolen te omringen. Hieronder zijn enkele voorbeelden.

[1] Tenzij je een vierde, vijfde of zesde niveau kop wilt, maar dat komt niet zo vaak voor.

https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/emphasis.gif

Je begint te zien hoe makkelijk het is om je output aan te passen. We zullen nu manieren bespreken om links naar URLs toe te voegen, geordende en ongeordende lijsten en andere veel gebruikte Markdown kenmerken te maken.

4.2.4 Linken

Om een URL-link toe te voegen, kun je de naam van de link of resultaat van de HTML file resulting HTML binnn zulke `[ ]` haakjes plaatsen en link zelf binnen zulke `( )` haakjes rechts er direct naast zonder tussenruimte.

https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/links.gif

4.2.5 Opsommingen

Het GIF-bestand hieronder laat het proces zien om geordende en ongeordende opsommingen te maken.

https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/lists.gif

Let op dat alleen cijfers nodig zijn zoals we zagen bij het opsommen van "Opwarmen van eten" met een "1." Wij kunnen ook ongeordende en geordende opsommingen krijgen door de tekst met twee spaties in te laten springen.

In veel van de voorbeelden die volgen, zul je de actuele tekst zien die je getypt hebt in jouw R Markdown document, opgelicht met een grijze achtergrond en ook de resultaten van die tekst er direct onder.

```
1. Opstaan
  - Uit bed stappen
1. Eten opwarmen
  - Keukendeur openen
  - Bord uit de kast pakken
2. Koffie maken
  i. Water opwarmen
```

```
  ii. Bonen malen
3. Ontbijt maken

  We kunnen hier een paragraaf (of twee) beschrijven over hoe we een ontbijt maken.
  Als we een paragraaf wat laten inspringen en een nieuwe regel maken newline, zal het onder
  het item inspringen.
```

1. Opstaan
   - Uit bed stappen
2. Eten opwarmen
   - Keukendeur openen
   - Bord uit de kast pakken
3. Koffie maken
   i. Water opwarmen
   ii. Bonen malen
4. Ontbijt maken

4.2.6 Gevarieerd Markdown

Regels afbreken / wit ruimte

Regels afbreken in combinatie met wit ruimte zijn ontzettend belangrijke onderdelen in Markdown. Ze geven heel vaak de start van een nieuwe paragraaf aan.

```
Hier is een voorbeeld van een tekst met alleen een regelbreuk.
Je mag verwachten dat deze regel in een nieuwe paragraaf verschijnt maar dat gebeurt niet.
```

Hier is een voorbeeld van een tekst met alleen een regelbreuk. Je mag verwachten dat deze regel in een nieuwe paragraaf verschijnt maar dat gebeurt niet.

Om een nieuwe paragraaf te starten, moet je ervan verzekerd zijn dat er wit ruimte bestaat tussen de twee paragrafen:

```
Hier is een voorbeeld van een tekst met alleen een regelbreuk.

Je mag verwachten dat deze regel in een nieuwe paragraaf verschijnt maar dat gebeurt niet.
```

Hier is een voorbeeld van een tekst met alleen een regelbreuk.

Je mag verwachten dat deze regel in een nieuwe paragraaf verschijnt maar dat gebeurt niet.

Horizontale regels

Een andere bruikbare manier om verschillende delen van jouw analyse te scheiden is via het gebruik van horizontale regels. Deze kunnen makkelijk worden toegevoegd door het plaatsen van de asterixs achter elkaar (of drie scheidingsstreepjes)

```
***
```

```
---
```

Aanhalingstekens

Als je iemand graag wilt aanhalen of een ingesprongen tekstblok wilt maken, kun je dat makkelijk doen door een `>` te plaatsen aan het begin van een passage:

```
> Reproduceerbaar onderzoek is het idee dat data analyses, en meer in het algemeen,
wetenschappelijke claims, met hun data en software code zodat anderen de bevindingen kunnen
verifieren en erop kunnen voortbouwen. - Roger Peng
```

Tekst commentaar

Op bepaalde momenten wil je commentaar op de tekst geven maar dan binnen een R Markdown document. Bijvoorbeeld je hebt iets geschreven maar dat wil je niet in het 'gebreide' document, althans je bent niet zeker of je het helemaal weg wilt gooien. In dat geval hoef je alleen maar een blok tekst te plaatsen binnen `<!-- -->` zoals hieronder te zien is:

```
<!--
Ik wil deze tekst bewaren voor later en wil deze nog niet weggooien.
-->
```

Deze tekst zie je dan niet terug in het boek.

Vergelijkingen

Als je mooie mathematische formules in jouw document wilt opnemen, kun je ze tussen twee dollartekens plaatsen:

```
$y = mx + b$
```

$y = mx + b$

4.2.7 R Chunks

Het duurde even om bij het beste deel van R Markdown aan te komen: de toevoeging van R code in het document en het meenemen van die code in de resultaten van het 'gebreide' document. Je hebt al wat R chunks gezien in de R Markdown file tot nu toe. Aan een aantal eigenschappen moet je in ieder geval gewend raken:

- Ze beginnen en eindigen altijd met drie van deze tekens (die zitten vaak links boven op het toetsenbord) ```` ``` ````.
- Na deze tekens, zie je dat R chunks beginnen met `{r}`, soms met bepaalde namen of andere chunk opties toegevoegd en dan eindigt die eerste regel met een `}`.

- De regels die binnen de tekens vallen is een normale R code die je kunt runnen in de R Console.

Laten we eens naar een voorbeeld kijken of en naar onze eigen R chunks in onze **first_rmarkdown.Rmd** file:

https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/rchunks.gif

Dit voorbeeld laat jou twee verschillende manieren zien om een vector van waarden te maken. Verdere discussie hierover zie je in Hoofdstuk 5. Je ziet dat de code automatisch runt als we op de **Knit HTML** knop drukken met de output in de gebreide file.

4.2.8 R code in de tekst

We hebben gezien dat we een R code kunnen gebruiken en dat kan binnen een R chunk code geplaatst tussen drie van die tekens aan het begin en het einde.

https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/inliner.gif

4.2.9 Code benadrukken

Het is goed om je de gewoonte eigen te maken om de R code en de objecten daarin te benadrukken en zo te onderscheiden van de rest van de tekst. Dat kan door zo'n enkelvoudig teken ('backtick') zoals bij `one_value`.

4.3 R Markdown Chunk Opties

De R chunk opties die je waarschijnlijk het meeste gebruikt zijn `echo`, `eval` en `include`. Als standaard is elk van deze opties ingesteld als `TRUE`.

- `echo` geeft aan of de code die de resultaten laat zien moeten worden geproduceerd voor de corresponderende R output.
- `eval` specificeert of de code moet worden geevalueerd of afgedrukt zonder output.
- `include` specificeert of de code en zijn output zullen worden opgenomen in het 'gebreide' document. Als het op `FALSE` is gezet draait de code maar niets van de code wordt in het document opnomen.

https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/chunkops.gif

https://raw.githubusercontent.com/ismayc/rbasics-book/gh-pages/gifs/chunkops2.gif

4.4 Algemene richtlijnen voor het schrijven van R Markdown Files

Witte ruimte is jouw vriend. Je moet altijd voor een wit ruimte zorgen tussen R chunks en jouw Markdown tekst. Het maakt jouw document veel beter leesbaar en zo maak je minder potentiele fouten. Zorg ook voor een witregel tussen de kop en jouw koppen en paragrafen.

Commentaar is altijd goed. Leg uit wat je bedoelt en wat je ideeen zijn waar je dat kunt. Onthoud dat jij zelf eigenlijk de beste partner bent voor de tijd dat je onderweg bent. Wees daarom aardig voor jezelf en leg uit wat je bedoelt zodat je het kunt onthouden!

Onthoud dat de Console en de R Markdown omgevingen (als je 'breit') niet met elkaar interacteren. Dat dwingt jou ertoe dat jij alleen de code in jouw R chunk plaats die precies de resultaten produceert die je met anderen wilt delen. Zadel jouw document niet op met extra output. Je moet beknopt en duidelijk zijn in wat je precies doet.

Die chunk opties kunnen jouw documenten echt mooier maken en je kunt het afstemmen of wat jij jouw lezer wilt laten zien. Je kunt meer informaties vinden over alle beschikbare R chunk opties hier.

4.5 Help -> Cheatsheets

RStudio heeft enkele hele behulpzame cheatsheets die als goede referenties voor veel teken die je binnen RStudio wilt uitvoeren. Er zijn goede PDF versies te vinden en dan moet je naar **Help -> Cheatsheets** binnen RStudio gaan.

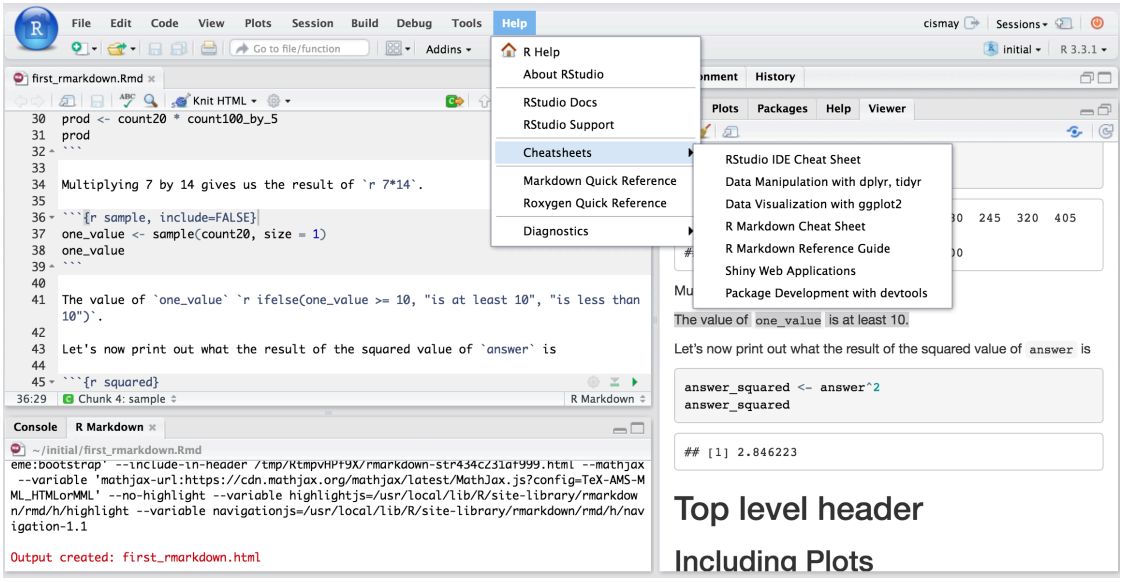

Figure 4.1: RStudio Cheatsheets Screenshot

Last updated:

```
## [1] "By chester on Saturday, July 29, 2017 14:11:03 PDT"
```

5
R Analyse met gebruikmaking van R Markdown

5.1 Een beginnende werkwijze

- "File organisatie en namen geven zijn krachtige wapens tegen chaos." - Jenny Bryan

5.2 R gebruiken met de periodieke table dataset

- De `library` functie

5.3 Gemiddelde, mediaan, standaard deviatie, samenvatting in vijf, distributie

5.4 Aanvullende inhoud die gedekt moet worden

- data structuren (vectors, lists, data frames, matrices)
- Vector operaties
- indexing/subsetting
- functies (default arguments)
- Hoofdletter of niet doet er toe in R!
- Waarom sommige argumenten citaten nodig hebben en andere niet?

5.5 R Markdown templates

Voor het laatst ververst:

```
## [1] "By chester on Saturday, July 29, 2017 14:11:03 PDT"
```

6
Algemene fouten in R detecteren

https://github.com/noamross/zero-dependency-problems/blob/master/misc/stack-overflow-common-r-errors.md

http://blog.revolutionanalytics.com/2015/03/the-most-common-r-error-messages.html

Voor het laatst ververst:

```
## [1] "By chester on Saturday, July 29, 2017 14:11:03 PDT"
```

7
Bibliography

Grolemund, G. (2014). *Hands-On Programming with R.* O'Reilly.