

# *Data visualisatie voor sociale wetenschappen*

*Kieran Healy, bewerkt Harrie Jonkman*

*December 17th, 2017*

## *Contents*

<i>Vooraf</i>	3
<i>Wat je zult leren</i>	3
<i>Op het rechte pad</i>	3
<i>Conventies</i>	3
<i>Voordat je begint</i>	4
<i>Opstarten</i>	5
<i>Werken met RMarkdown</i>	5
<i>Werken met RStudio</i>	5
<i>Werken met R</i>	5
<i>Heb geduld met R</i>	6
<i>Je eigen data inlezen</i>	6
<i>Je eerste plot maken</i>	6
<i>Kijken naar data</i>	8
<i>Waarom naar data kijken</i>	8
<i>Wat maakt slechte figuren slecht?</i>	8
<i>Perceptie en data visualisatie</i>	8
<i>Visuele taken en grafieken decoderen</i>	8
<i>Manieren om data te representeren</i>	9
<i>Problemen van eerlijkheid en goed oordeel</i>	9
<i>Helder denken over jouw grafieken</i>	9
<i>Een plot maken</i>	11
<i>Hoe ggplot werkt</i>	11
<i>Opschonen van data</i>	11
<i>Van data naar dingen die je kunt zien</i>	11
<i>Laag voor laag opbouwen</i>	12
<i>Esthetiek tegenover vastzetten</i>	13

<i>Groep, facet en transformeren</i>	15
<i>Kleurloze data</i>	15
<i>Gegroepeerde data</i>	15
<i>In stukjes opdelen</i>	16
<i>Transformeren van data</i>	18
<i>Technieken voor samenvatten en transformeren</i>	21
<i>Tot slot</i>	23
<i>Werken met geoms</i>	24
<i>Continue variabelen via groep of categorie</i>	24
<i>Werken met modellen</i>	26
<i>Verschillende mogelijkheden ineens laten zien, met een legenda</i>	27
<i>Model objecten opschonen met Broom</i>	28
<i>Kaarten maken</i>	32
<i>Amerika in kaart brengen</i>	32
<i>Kleinere, meerdere kaarten tegelijk</i>	39
<i>Verfijnen</i>	42
<i>Kleur gebruiken in jouw voordeel</i>	44
<i>Tot slot</i>	50

## Vooraf

Met grafieken en kaarten kun je de structuur van de informatie die je hebt verzameld onderzoeken en kun je ervan leren. Goede visualisaties stellen je in staat om jouw ideeën en bevindingen beter te communiceren met anderen. Als je weet hoe je ze moet maken, kun je ook beter de grafieken van andere lezen en begrijpen.

Het doel van dit document is hier om de **ideeën** en de **methodes** van datavisualisatie op een sensibele, samenhangende en reproduceerbaar manier te introduceren. Dit document is gebaseerd op een recent document van Kieran Healy (2017) dat **Data Visualisation for Social Science. A practical introduction with R and ggplot2** heet en dat je gratis op internet vindt (<http://socviz.co/>). Hiernaast zie je de voorkant van het digitale boek afgebeeld.<sup>1</sup>

## Wat je zult leren

In deze handleiding leer je uiteindelijk vier zaken:

- Je zult de basis principes begrijpen die er zitten achter datavisualisatie;
- Je krijgt een gevoel waarom sommige grafieken en figuren goed werken en waarom andere misleidend zijn;
- Je leert verschillende soorten plots te maken in R met gebruik van het pakket ggplot2;
- Je leert ook hoe je deze plots kunt verfijnen voor een effectieve presentatie.

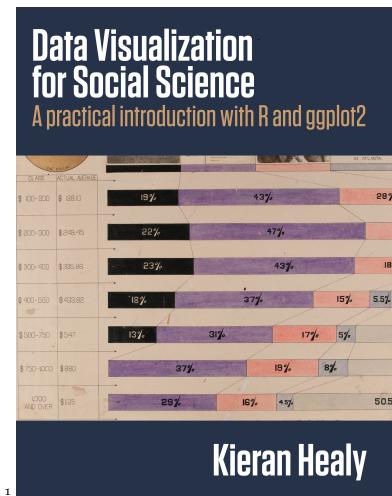
## Op het rechte pad

Wanneer je begint met programmeren in een taal zoals R kun je tegen nogal wat problemen oplopen. Het werkt niet zoals je wilt en veel zaken kunnen nogal verwarring overkomen. Toch is het zo dat je ergens moet beginnen als je ermee wilt gaan werken. Het maken van grafieken kan een goed begin zijn van programmeren omdat dit al snel mooie resultaten oplevert.

## Conventies

Je leest niet alleen gewone tekst, zoals je dat gewend bent. Je leert ook codes lezen en gebruiken die je direct in R kunt typen. Als je typt wat *mijn\_getallen* inhouden en daarna opnieuw *mijn\_getallen* typt krijg je de output:

```
mijn_getallen<-c(1,1,4,1,14,1)
mijn_getallen
```



<sup>1</sup> De voorkant van het boek

```
## [1] 1 1 4 1 14 1
```

Een groot aantal van dit soort conventies of afspraken moet je wel leren.

### *Voordat je begint*

Voordat je begint is het nodig dat je een aantal zaken installeert.

1. Je moet zorgen dat je een recente versie van R op jouw computer hebt staan. Het programma is beschikbaar voor Windows, Mac en Linux systemen.
2. Als je R hebt geïnstalleerd moet je ook RStudio installeren. Dat is een schil rond R die het werken met R makkelijk maakt.
3. Installeer ook verschillende pakketten die je nodig hebt. In het boek van Healy worden in ieder geval de volgende pakketten gebruikt:

```
my_packages <- c("tidyverse", "broom", "coefplot", "cowplot",
                 "gapminder", "GGally", "ggrepel", "ggridges", "gridExtra",
                 "interplot", "margins", "maps", "mapproj", "mapdata",
                 "MASS", "quantreg", "scales", "survey", "srvyr",
                 "viridis", "viridisLite", "devtools")
```

```
install.packages(my_packages, repos = "http://cran.rstudio.com")
```

```
devtools::install_github("kjhealy/socviz")
```

Als je de pakketten een keer hebt geïnstalleerd, hoeft dan niet nog een keer. Je moet ze wel voor gebruik steeds uit de bibliotheek halen. Het viel mij op dat tussen door ook nog een enkele keer een pakket moet worden binnengehaald. In de codes staat dan library(pakket\_x). Je moet er wel voor zorgen dat dat pakket binnen is gehaald.

## Opstarten

We willen beelden van onze data maken waar we naar kunnen kijken en waarvan we kunnen leren. R en ggplot zijn gereedschappen die we gebruiken om deze visualisaties te maken. Probeer de codes vooral steeds zelf te typen en ga ze niet copiëren en plakken. Door het zelf te doen, leer je er het meest van. Drie elementen zijn hierbij van belang: RMarkdown, RStudio en R.

## Werken met RMarkdown

Als je zaken opschrijft en jouw codes gebruikt, doe dat dan in een een teksteditor. Doe dat niet met Word maar gebruik een andere editor (ikzelf gebruik bv. TinnR hiervoor)<sup>2</sup>. Zo kun je wat je hebt gemaakt, later altijd goed reproduceren. Terwijl je ggplot leert, gaat het steeds om drie dingen:

1. **Schrijf de code** en zorg er daarbij voor dat het zo makkelijk en effectief mogelijk gaat.
2. **Kijk naar de output** wanneer je werkt, weet dan dat tussentijdse resultaten belangrijk zijn.
3. **Zet er opmerkingen bij** over wat je hebt gedaan en waarom, dat kan heel handig zijn in latere instantie.

RMarkdown is makkelijk omdat je hierbinnen de tekst en de code goed kunt combineren.

*# Een code komt dan hier te staan.*

In andere documenten heb ik hier een beschrijving van gegeven.<sup>3</sup>

## Werken met RStudio

RStudio is een goede schil om makkelijker met R te kunnen werken. Het is een geïntegreerde omgeving waarin je de code plaats, de output leest en waarbij je bij verschillende functies wordt ondersteund.<sup>4</sup>

## Werken met R

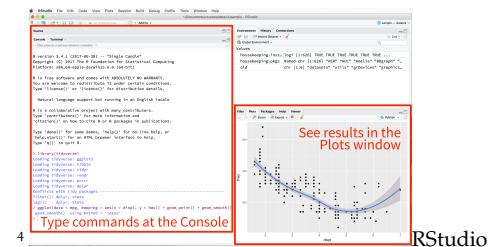
R is het basisprogramma, een programmeertaal op zich. Een aantal zaken zijn van belang:

- Wees goed georganiseerd;
- Geef alles een naam;
- Bedenk dat alles als het ware een object is;
- Die objecten moet je steeds ook een naam geven;
- Alles gaat met input, acties en output;
- Functies zijn gebundeld in libraries en daarvan zijn er onderhand zo'n 10.000. Deze libraries moet je installeren en steeds laden. ggplot2



Het logo van TinnR

<sup>3</sup> In enkele andere HH's wordt hier uitgebreid op ingegaan



RStudio

is zo'n voorbeeld van een library;

- Als je niet weet met welk object je te maken hebt, vraag naar zijn klasse zodat je weet wat je wel en niet ermee kunt doen;
- Om in een object te kijken, kun je naar de structuur vragen.

### *Heb geduld met R*

R is een programmeertaal, dat doet wat je vraagt om te doen en niet automatisch wat je wil dat het doet. Dat is nogal een subtiel verschil. Wees steeds heel geduldig met het programma. Hier kun je bijvoorbeeld op letten:

- Zorg dat de haakjes nauwkeurig worden geplaatst. Na het openen van ( sluit dan ook af met een );
- Schrijf alles helemaal uit. Als er een + aan het begin staat in plaats van een > dan is het nog niet af;
- ggplot werkt met lagen en vaak met een +. Die + moet aan het einde staan en niet aan het begin.

Dus zo:

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point()
```

en niet zo:

```
ggplot(data = mpg, aes(x = displ, y = hwy))
+ geom_point()
```

### *Je eigen data inlezen*

Data kunnen natuurlijk in allerlei vormen zijn opgeslagen en deze kun je op verschillende manieren inlezen of het nu bv. als spss, sas, stata of excel\_bestanden is opgeslagen.

Hieronder zien we bijvoorbeeld hoe we een excelsheet via internet binnenhalen:

```
# url <- "https://cdn.rawgit.com/kjhealy/viz-organdata/master/organdonation.csv"

# library(foreign)
# organs <- read_csv(file = url)
```

Hier heb je een document waarin je verder kunt lezen hoe je verschillende documenten kunt inlezen.<sup>5</sup>

### *Je eerste plot maken*

We gaan het verder hebben over het maken van grafieken. Hier zie je hoe je er een start mee kunt maken. Daarvoor wordt een dataset ge-

<sup>5</sup> Hier een verwijzing naar een document dat gaat over het inlezen van documenten

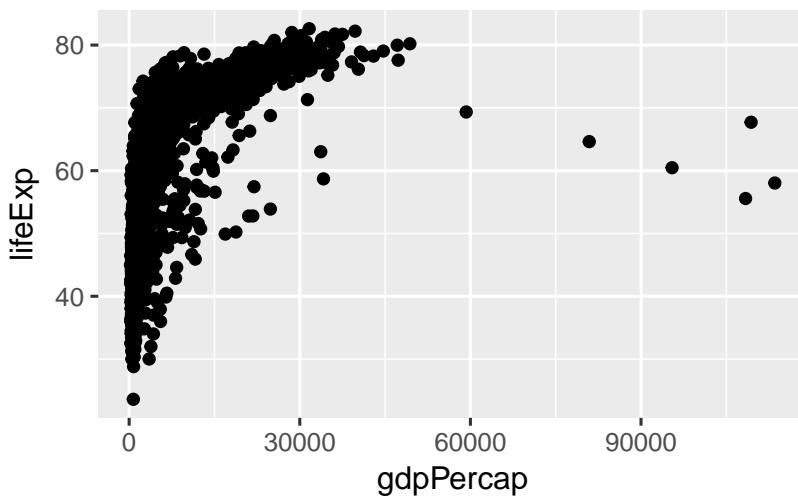
bruikt (Gapminder). We laden de library (nogmaals en niet vergeten: wel eerst installeren als je dat nog niet hebt gedaan).

```
library(gapminder)
gapminder
```

Vervolgens maken we de plot.

```
library(gapminder)
library(ggplot2)
p <- ggplot(data = gapminder,
             mapping = aes(x = gdpPercap,
                           y = lifeExp))

p + geom_point()
```



Straks gaan we uitleggen hoe het allemaal werkt. Maar eerst wat algemene informatie over het kijken naar data.

## Kijken naar data

Healy gaat uitgebreid in op het kijken naar data, wanneer er sprake is van slechte figuren en wat figuren tot goede figuren maakt. Het is zeer de moeite dit hoofdstuk over kijken door te lezen. Een taal zaken wil ik hier onder de aandacht brengen.

Datavisualisatie is een bijzonder goed middel om de data te onderzoeken, willen begrijpen en uitleggen. Maar het is ook weer geen magisch middel om de werkelijkheid te zien zoals die werkelijk is. Ook als je daar gebruik van maakt kun je mensen voor de gek houden en je kunt uiteindelijk ook jezelf ermee voor de gek houden.

## Waarom naar data kijken

Anscombe (1973) kijkt naar vier datasets met elf observaties op twee variabelen. De eigenschappen van x en y liggen niet ver van elkaar maar grafisch uitgedrukt ziet het er heel anders uit steeds.<sup>6</sup>

Een ander voorbeeld is van Jackman(1980) die refereert naar een studie waarin een negatieve associatie wordt verondersteld tussen stemmen en inkomensongelijkheid. De grafiek laat zien dat de negatieve associatie helemaal door Zuid-Afrika wordt gemaakt.<sup>7</sup>

## Wat maakt slechte figuren slecht?

Eigenlijk zijn er drie soorten problemen te onderscheiden: esthetische, substantiële en misleidend. Oftewel:

- Slechte smaak;

- Slechte data;
- Slechte perceptie.

## Perceptie en data visualisatie

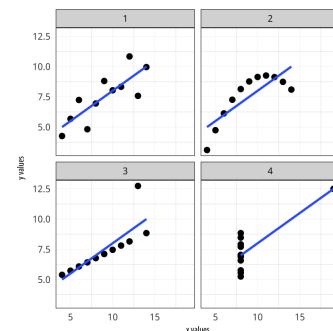
Als we naar figuren kijken, kijken we naar lijnen, vormen en kleuren. Ons visueel systeem zit zo in elkaar dat we op basis daarvan onderscheid kunnen maken en daar moeten we rekening mee houden.

Bepaalde objecten zijn in ons visuele veld makkelijker te zien dan andere objecten. Die ‘poppen op’ als het ware. Dat zorgt ervoor dat bepaalde dingen makkelijker te zien zijn dan andere zaken.

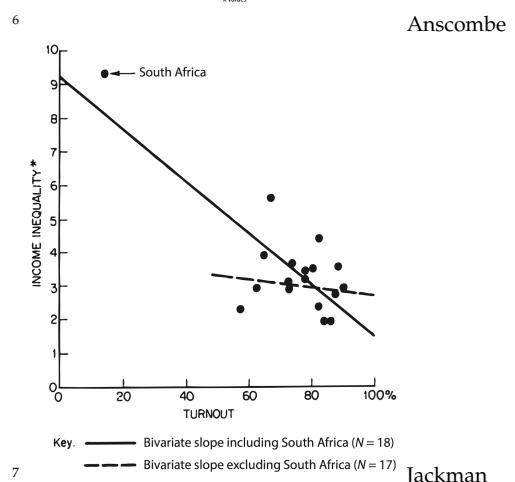
Het is ook goed om te weten welke relaties er zijn te onderscheiden en onder welke omstandigheden.

## Visuele taken en grafieken decoderen

Behalve perceptie speelt interpreteren en begrijpen van bepaalde grafieken ook een rol. Op het snijvlak gebeuren er dingen die het kijken naar de grafiek voor hen bepaalt. De datawaarden zijn in de grafiek gezet maar nu moet de kijker er ook weer waarden uithalen



Anscombe



Jackman

en de grafiek decoderen. Die moet bijvoorbeeld de relatieve positie van elementen op een algemene schaal inschatten. We kunnen bijvoorbeeld de invloed van een hoek moeilijk inschatten en daarom wordt het gebruik van staafdiagrammen ook afgeraden. En zijn er nog andere voorbeelden te noemen, bv. driedimensionale representatie van data; wij zijn niet goed in het inschatten van de z-waarde naast de x- en y-waarde.

### *Manieren om data te representeren*

Grafische elementen representeren onze data op manieren waarop we kunnen zien. Verschillende variabeleigenschappen kunnen op eenzelfde manier worden gerepresenteerd door punten bijvoorbeeld, lijnen, lijnen, vormen en dergelijke. Onze taak is om dat goed te doen en dan lopen we soms tegen problemen op.

- Het moet wel aansluiten bij de data die we hebben. Voor continue variabelen geldt soms wat anders dan voor categoriale data en daar moet je je van bewust zijn;
- Als we een visueel element hebben gekozen moeten we wel weten hoe effectief dat element is voor dat wat we willen representeren;
- Perceptuele details maken ook het succes uit;
- Als we een keuze hebben gemaakt om een variabele uit te drukken (als een positie, een lengte, een gebied, een schaduw van grijs of een kleur), hebben we een belangrijke beslissing genomen. Maar dat levert meteen weer beperkingen op waar je rekening mee moet houden.

### *Problemen van eerlijkheid en goed oordeel*

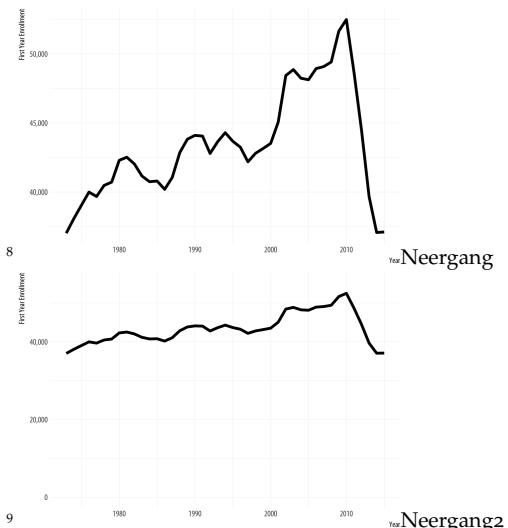
Hieronder zie data die op twee manieren zijn weergegeven. Welke is de beste? Het laat zien de ontwikkeling van studenten die zich voor rechten inschrijven. De eerstegrafiek begint bij het aantal in 1973 en eindigt in 2010.<sup>8</sup>

In de tweede grafiek wordt het aantal studenten breder afgebeeld en begint de y-as op nu<sup>9</sup>.

In het eerste geval lijkt de neergang dramatischer als de tweede. Je kunt er over discussiëren welke beter is maar de kijker zelf heeft natuurlijk ook een verantwoordelijkheid.

### *Helder denken over jouw grafieken*

Het gaat erom dat je grafieken op een eerlijke en reproduceerbare manier maakt. Met default settings kom je vaak niet ver genoeg en daarom is het goed dat je voldoende kennis en gereedschappen



hebt om het goed uit te voeren. Het maken van goede en heldere grafieken is gebaat bij een systematische manier van werken.

ggplot implementeert een grammatica van grafieken waarmee je dit werk goed kunt organiseren en je ook een goed gevoel krijgt bij de verschillende elementen. Het breekt het werk in een aantal overzichtelijke taken en structureert dit werk. De verschillende functies moet je in de vingers krijgen. Begin eenvoudig en bereidt de mogelijkheden langzaam uit. Je leert hier controle te krijgen over de verschillende elementen en je leert verschillende plot-types te maken en R helpt jou daarbij.

## Een plot maken

In dit hoofdstuk leer je ggplot te gebruiken via het maken van een aantal scatterplots. We bouwen het langzaam op zodat je begrijpt wat er gebeurt. Het zijn steeds dezelfde stappen die je moet zetten. Maar op een gegeven moment zal het ook snel gaan omdat als je weet hoe het werkt. Dan wil je ook snel meer.

## Hoe ggplot werkt

Ggplot werkt steeds op eenzelfde manier. Je start met de data en het proces eindigt met de grafiek en daartussen gaat het om een aantal opeenvolgende stappen.<sup>10</sup>

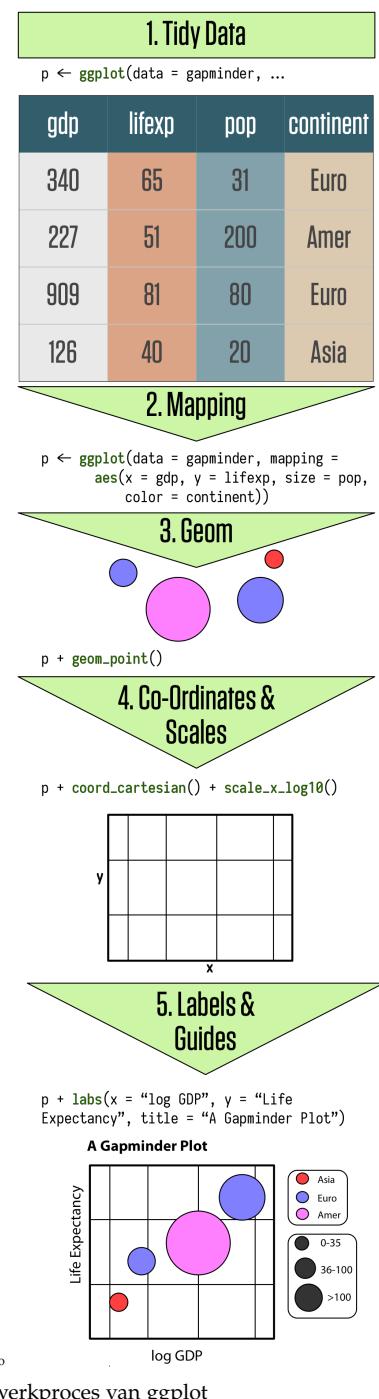
Ggplot kent een logische opbouw en de logische verbindingen tussen jouw data en de plotelementen worden ‘aesthetic mapping’ of gewoon ‘aesthetics’(aes) genoemd. Je begint altijd met het vertellen wat jouw data zijn en hoe jouw variabelen in een in de aesthetics worden geplaatst. Dan neem je dat resultaat en vertel je wat voor een soort plot je wilt hebben (bijvoorbeeld scatterplot of boxplot etc). De plot heet overal **geom** en dat combineer je met het soort plot, bv geom\_point() of geom\_boxplot(). Deze twee delen combineer je letterlijk via het “+”teken. Daarna worden op eenzelfde manier coordinaten en schalen en labels en teksten toegevoegd.

## Opschonen van data

Met opgeschoonde data werkt de opmaak van grafieken vaak beter. Soms zijn data in wijd formaat gepresenteerd. Je ziet dan het land staan en een groot aantal jaren in de kolommen er naast. Het werkt dan beter wanneer de data lang formaat staan met hetzelfde land steeds in de linker kolom en het jaar in een kolom onder elkaar. Dit is een voorbeeld. Zorg ervoor dat de data opgeschoond zijn voordat je grafieken gaan maken.

## Van data naar dingen die je kunt zien

We beginnen steeds met het schrijven van een soort basisfunctie waar we nieuwe lagen aan toevoegen en die noemen we dan bijvoorbeeld p, zoals dat hiernaast staat.



```
p ← ggplot(data= <data>,
            mapping= aes(<aesthetic> = <variable>,
                         <aesthetic> = <variable>,
                         <...> = <...>))

p + geom_<type>(<...>)+
    scale_<mapping>_<type>(<...> )+
    coord_<type>(<...> )+
    labs(<...>)
```

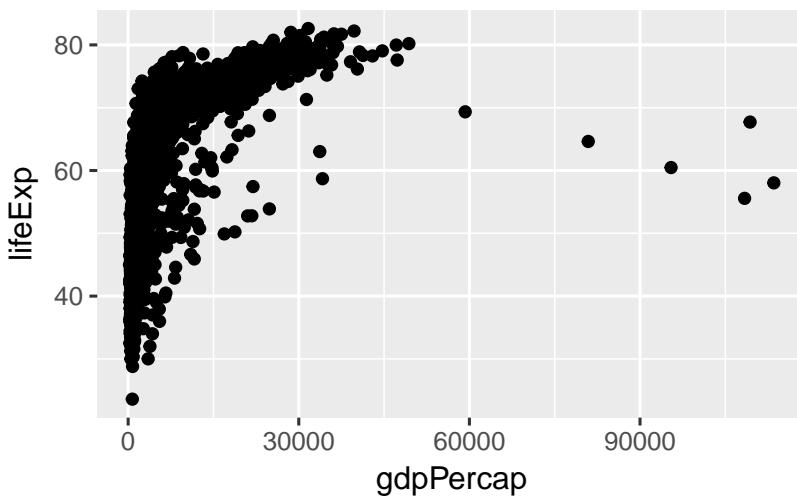
(Het

werkproces van ggplot.)

Dus eerst definieer je de p-functie en dan zeg je dat je het in een scatterplot wilt hebben afgedrukt.

```
p <- ggplot(data = gapminder,
             mapping = aes(x = gdpPercap,
                           y = lifeExp))

p + geom_point()
```



Hierboven zie je dan ook duidelijk die gelaagde opbouw.

### *Laag voor laag opbouwen*

Enkele stappen moet je steeds achter elkaar zetten:

1. Vertel de ggplot() functie wat jouw data zijn;
2. Vertel ggplot() welke relaties je wilt zien. De eerste twee stappen zetten we in een object die we p noemen;

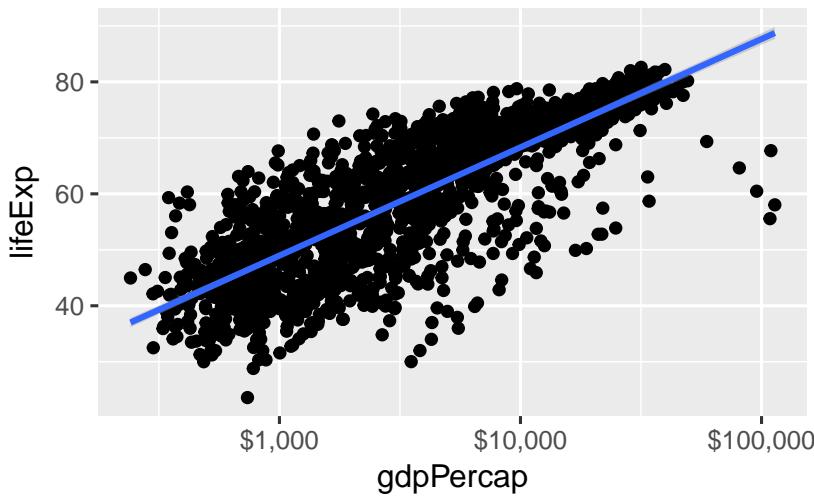
3. Vertel vervolgens aan ggplot hoe je de relaties afgedrukt wilt hebben;

4. Als je nog meer wilt, voeg er dan een nieuwe laag toe, steeds een voor een;

5. Tot slot voeg je er nog een aantal functies aan toe als schalen, labels, titels etc. Dat komt zo nog aan de orde;

Hieronder zie je een voorbeeld waar een aantal lagen op elkaar zijn geplaatst.

```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y=lifeExp))
p + geom_point() +
  geom_smooth(method="gam") +
  scale_x_log10(labels = scales::dollar)
```



### *Esthetiek tegenover vastzetten*

Je moet er voor zorgen dat de codes op de goede plaats worden gezet. Als je bijvoorbeeld de punten paars wilt maken moet je dat niet toevoegen aan mapping zoals hieronder:

```
p <- ggplot(data = gapminder,
             mapping = aes(x = gdpPercap,
                           y = lifeExp,
                           color = "purple"))

p + geom_point() +
  geom_smooth(method='loess') +
  scale_x_log10()
```

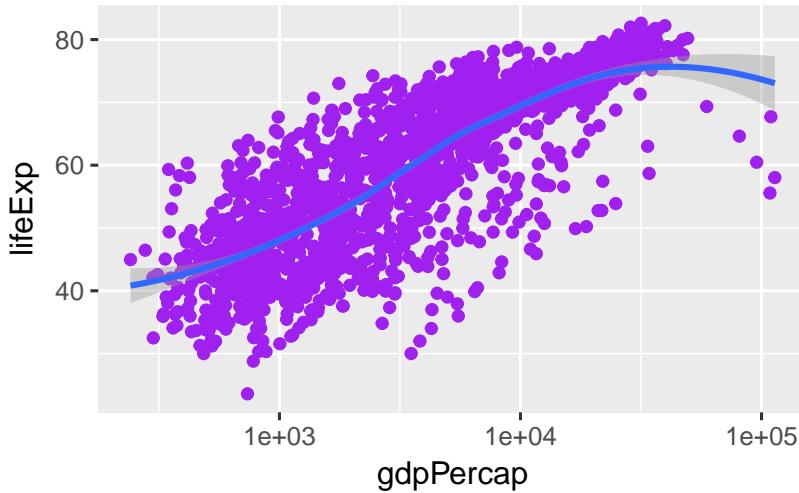
De kleur moet je toevoegen aan geom\_point zoals hier onder.

```
p <- ggplot(data = gapminder,
             mapping = aes(x = gdpPercap,
```

```

y = lifeExp))
p + geom_point(color="purple") +
  geom_smooth(method='loess') +
  scale_x_log10()

```



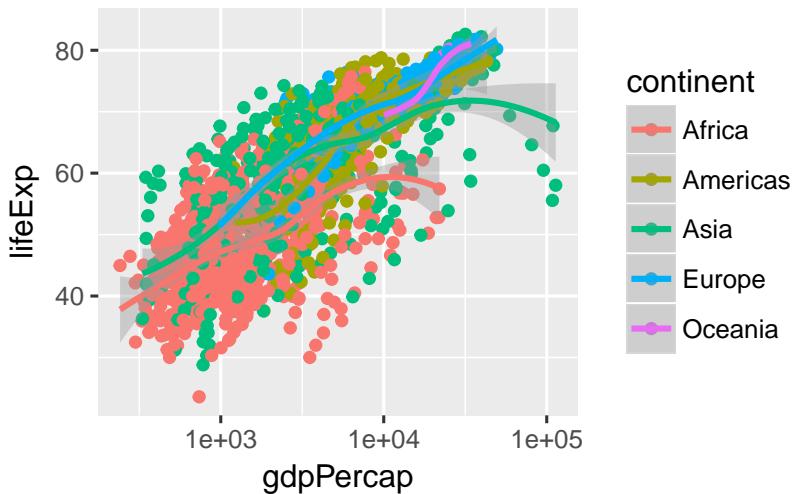
Maar we kunnen het nog veel mooier maken zoals hieronder:

```

p <- ggplot(data = gapminder,
             mapping = aes(x = gdpPercap,
                           y = lifeExp,
                           color = continent))

p + geom_point() +
  geom_smooth(method='loess') +
  scale_x_log10()

```



## *Groep, facet en transformeren*

Je kunt makkelijk tegen problemen aanlopen wanneer je ggplot gebruikt. In dit hoofdstuk gaan we in op enkele kernzaken van ggplot die wel vaker problemen veroorzaken.

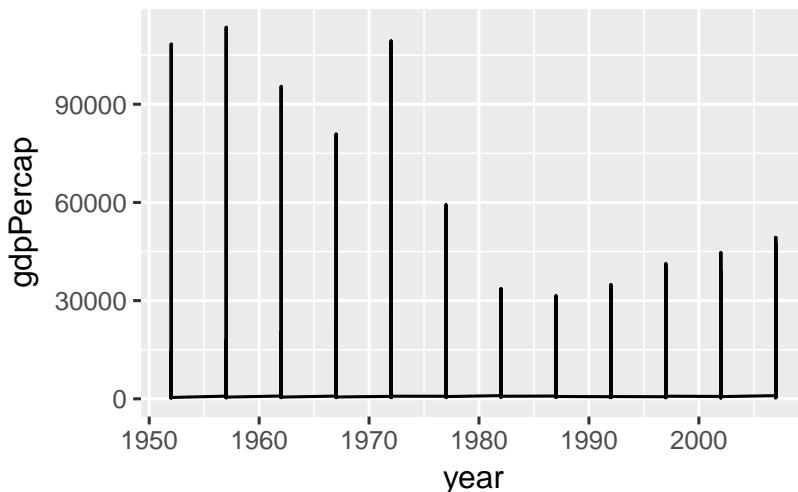
### *Kleurloze data*

Als je met ggplot werkt wil je eigenlijk iets visueels zeggen. Het kost je vaak enkele aanpassingen voordat je kunt zeggen wat je wilt zeggen. Alles wordt in stukjes opgeknippt en de fouten kunnen op verschillende plaatsen gemaakt worden. Soms zijn het algemene fouten die je maakt (een + vergeten of een haakje). Soms lukt het wel maar ziet het er niet uit. Dan heeft ggplot niet alle informatie die het nodig heeft om een mooi figuur te maken.

### *Gegroepeerde data*

Laten we nog eens naar de Gapminder dataset kijken. We maken een plot over tijd met jaar op de x-as en levensverwachting op de y-as. Dat ziet er zo'n beetje zo uit:

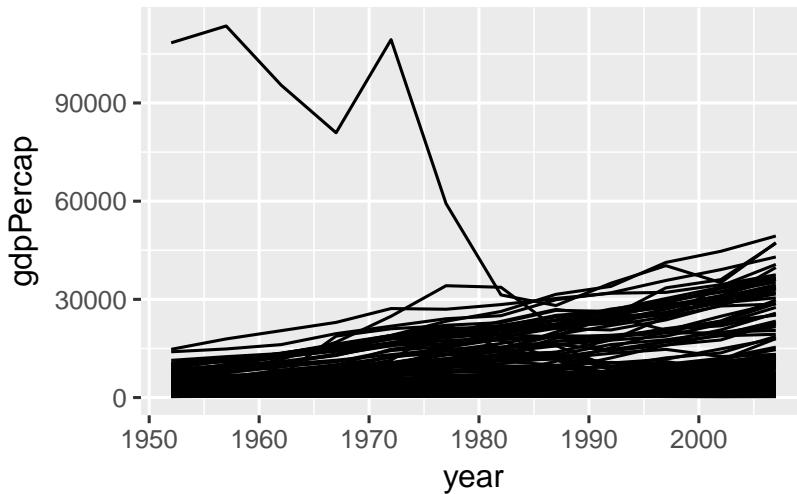
```
p <- ggplot(data = gapminder,
             mapping = aes(x = year,
                           y = gdpPercap))
p + geom_line()
```



Ondanks dat het er wel aardig uitziet, zie je ook dat de landen erin ontbreken.

```
p <- ggplot(data = gapminder,
             mapping = aes(x = year,
```

```
y = gdpPerCap))  
p + geom_line(aes(group=country))
```

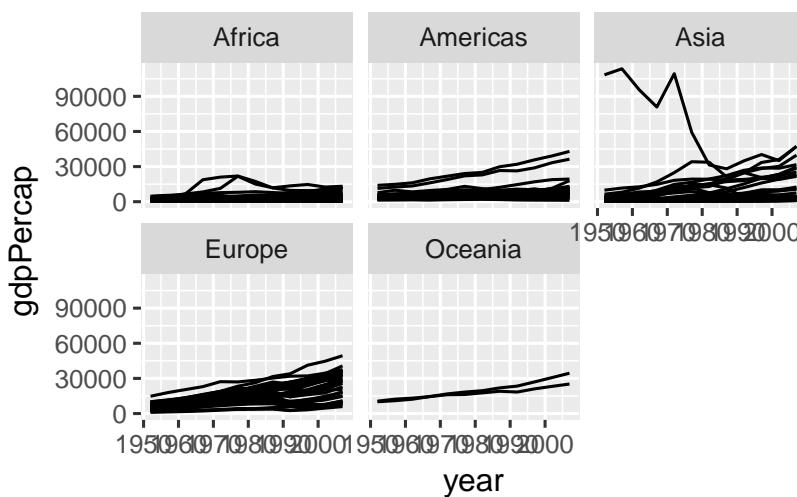


Het ziet er nog grof uit. Elk lijn staat voor een land over tijd. De uitschieter is Koeweit.

### In stukjes opdelen

Misschien kunnen we de landen wat meer groeperen en geeft dat meer orde in de chaos. We gebruiken `facet_wrap()` om het per *continent* uit te splitsen.

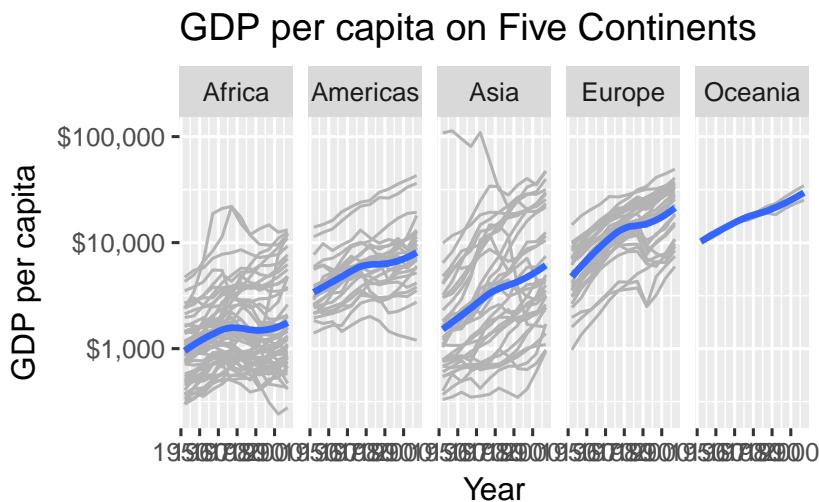
```
p <- ggplot(data = gapminder,  
             mapping = aes(x = year,  
                            y = gdpPerCap))  
p + geom_line(aes(group = country)) + facet_wrap(~ continent)
```



We kunnen het ook netjes naast elkaar zetten door `n_col` te gebruiken.

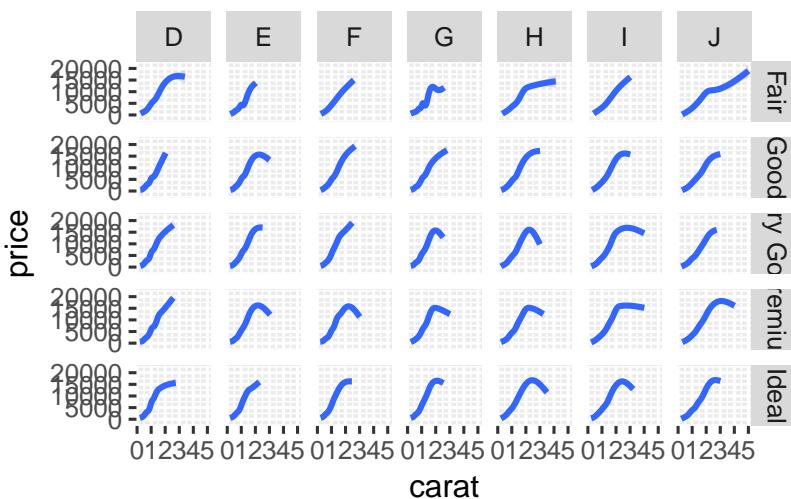
We voegen er nog enkele veranderingen aan toe zodat het wat mooier lijkt.

```
p <- ggplot(data = gapminder, mapping = aes(x = year, y = gdpPercap))
p + geom_line(color="gray70", aes(group = country)) +
  geom_smooth(size = 1.1, method = "loess", se = FALSE) +
  scale_y_log10(labels=scales::dollar) +
  facet_wrap(~ continent, ncol = 5) +
  labs(x = "Year",
       y = "GDP per capita",
       title = "GDP per capita on Five Continents")
```



```
p <- ggplot(data = diamonds, mapping = aes(x = carat, y = price))
p + geom_smooth(alpha = 0.3) + facet_grid(cut ~ color)

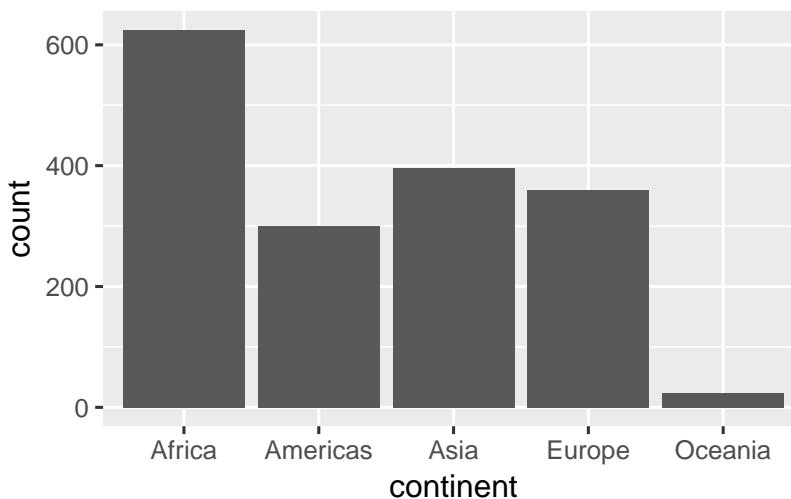
## 'geom_smooth()' using method = 'gam'
```



### *Transformeren van data*

Hierbij hebben we enkele voorbeelden gezien waarbij het commando geom\_smooth() betrokken en waarmee eigenschappen (zoals methode) konden worden toegevoegd. Met geom\_ worden statische functies uitgedrukt die niet altijd meteen helder zijn. Kijk eens hieronder waar we een staafdiagram maken.

```
p <- ggplot(data = gapminder,
             mapping = aes(x = continent))
p + geom_bar()
```



Bij het gebruik van geom\_bar moet je wel goed zicht hebben op de data en met welke variabelen je te maken hebt en hoe je die wilt afbeelden. Je hebt met categoriale data te maken (bv geordend, zoals onderwijsniveau en ongeordend zoals etniciteit, met ja en nee antwoorden en zevenpuntsschaal met het gemiddelde in het midden). Er zijn ook numerieke variabelen (zoals aantal kinderen) en die moet je dan soms weer overzichtelijk maken.

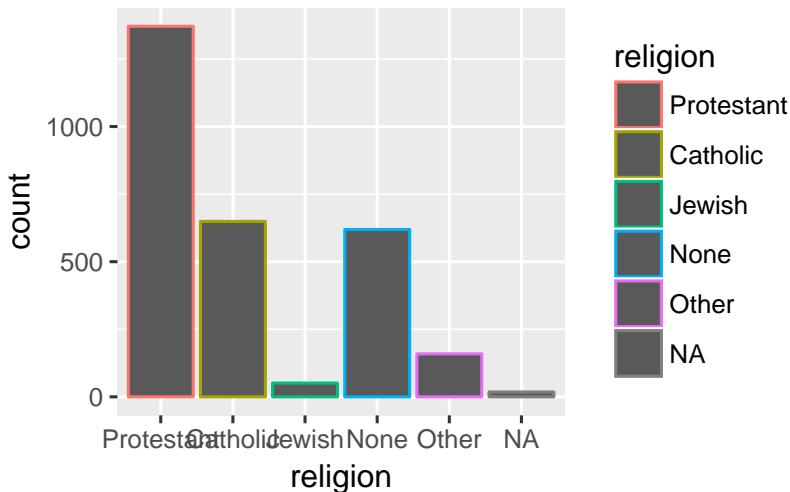
Laat ons nog eens naar een dataset kijken, dat onderdeel is van het socviz-pakket. De religie variabele drukt religieuze preferentie uit.

```
library(socviz)
table(gss_sm$religion)

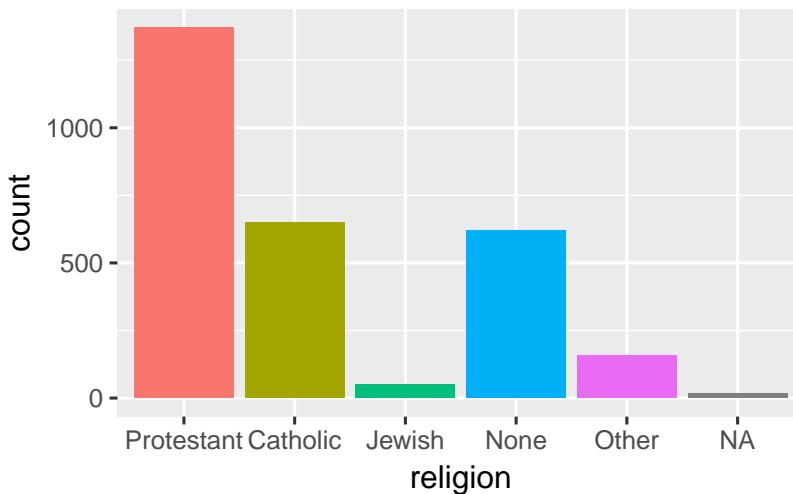
##
## Protestant    Catholic      Jewish      None
##      1371        649         51        619
##      Other
##      159
```

Om dit uit te drukken maken we een staafdiagram.

```
p <- ggplot(data = gss_sm,
             mapping = aes(x = religion, color = religion))
p + geom_bar()
```

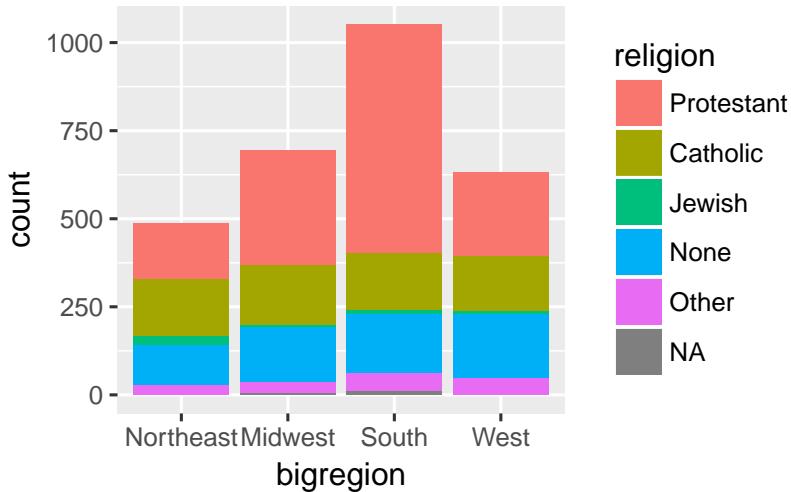


```
p <- ggplot(data = gss_sm,
             mapping = aes(x = religion, fill = religion))
p + geom_bar() + guides(fill = FALSE)
```



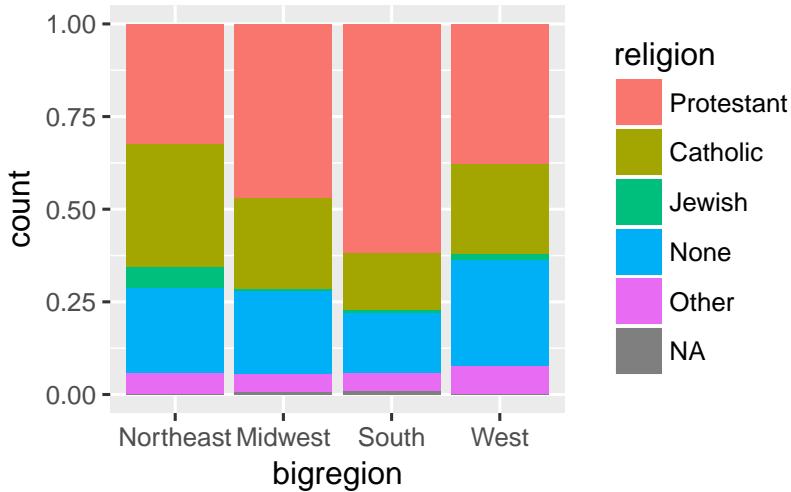
Interessanter is het om de fill-aesthetic met geom\_bar() te laten zien en daarmee de verdeling van de religies in de verschillende omgevingen.

```
p <- ggplot(data = gss_sm,
             mapping = aes(x = bigregion, fill = religion))
p + geom_bar()
```



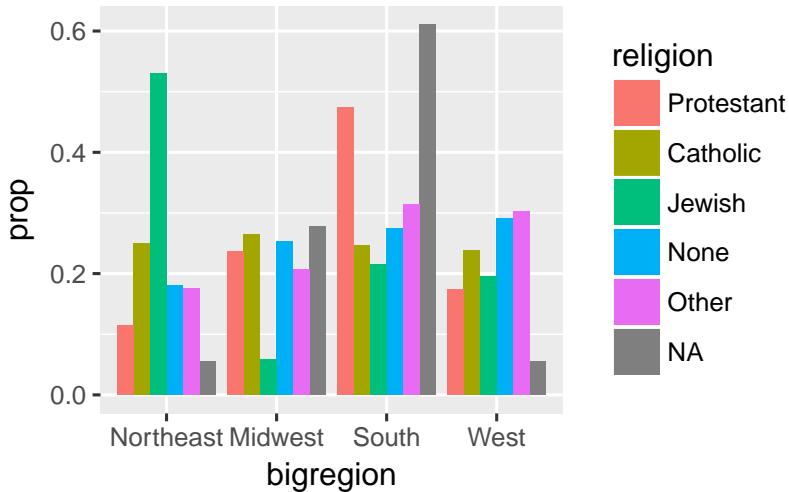
Hieronder zijn de lengtes van de staven hetzelfde en dan is het makkelijker om de verhoudingen te vergelijken.

```
p <- ggplot(data = gss_sm,
             mapping = aes(x = bigregion, fill = religion))
p + geom_bar(position = "fill")
```



Je kunt de groepen ook nog naast elkaar zetten.

```
p <- ggplot(data = gss_sm,
             mapping = aes(x = bigregion, fill = religion))
p + geom_bar(position = "dodge",
             mapping = aes(y = ..prop.., group = religion))
```



### *Technieken voor samenvatten en transformeren*

Soms is het ook handig om data te bewerken voordat je er figuren van maakt. Het pakket *dplyr* is daar heel geschikt voor.<sup>11</sup>. Via de pipeline-functie (%>%) kunnen een aantal operaties worden uitgevoerd zoals:

- Group (groeperen);

- Filter en Select (selecteren van delen van de data via de kolom, de rij of allebei);

- Mutate (nieuwe variabelen aanmaken);
- Summarize (aggregeren van de gegroepeerde data).

Deze functies gebruik je in *dplyr* via group\_by(), filter(), select(), mutate() en summarize(). Hieronder een voorbeeld waarmee je een tabel maakt.

- Je creëert een nieuw object (rel\_by\_religion) dat uitgaat van gss\_sm-data;

- De rijen worden gegroepeerd volgen bigregion;
- De tabel wordt kleiner gemaakt door drie kolommen te gebruiken (bigregion, religion en N);
- Er worden twee kolommen toegevoegd (frequentie en percentage).

<sup>11</sup> Zie voor beschrijving van dit pakket HH-11

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
```

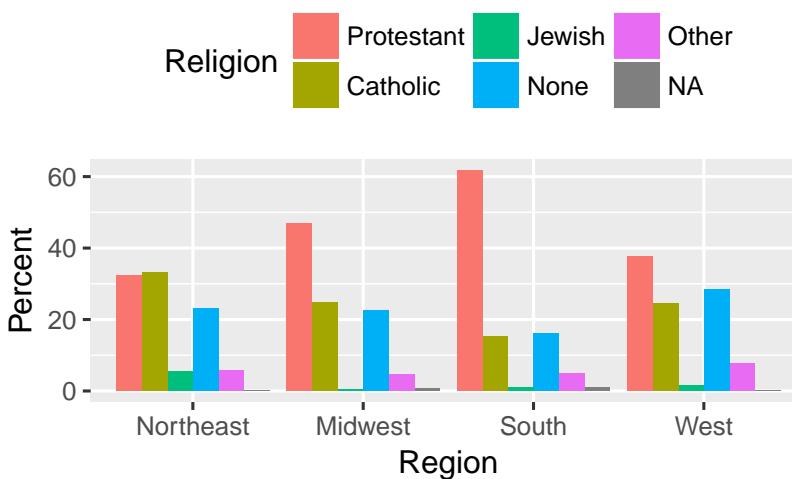
```
##  
##      intersect, setdiff, setequal, union  
  
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(N = n()) %>%  
  mutate(freq = N / sum(N),  
        pct = round((freq*100), 1))
```

Laten we het eens controleren door naar percentage te kijken en of we in de buurt van de 100 procent zitten.

```
rel_by_region %>%  
  group_by(bigregion) %>%  
  summarize(total = sum(pct))  
  
## # A tibble: 4 x 2  
##   bigregion     total  
##   <fct>       <dbl>  
## 1 Northeast    100.  
## 2 Midwest     99.9  
## 3 South        100.  
## 4 West         100.
```

Laten we kijken of we een barplot kunnen maken met de religieuze preferenties per regio.

```
p <- ggplot(rel_by_region, aes(x = bigregion, y = pct, fill = religion))  
p + geom_bar(position = "dodge", stat = "identity") +  
  labs(x = "Region", y = "Percent", fill = "Religion") +  
  theme(legend.position = "top")
```



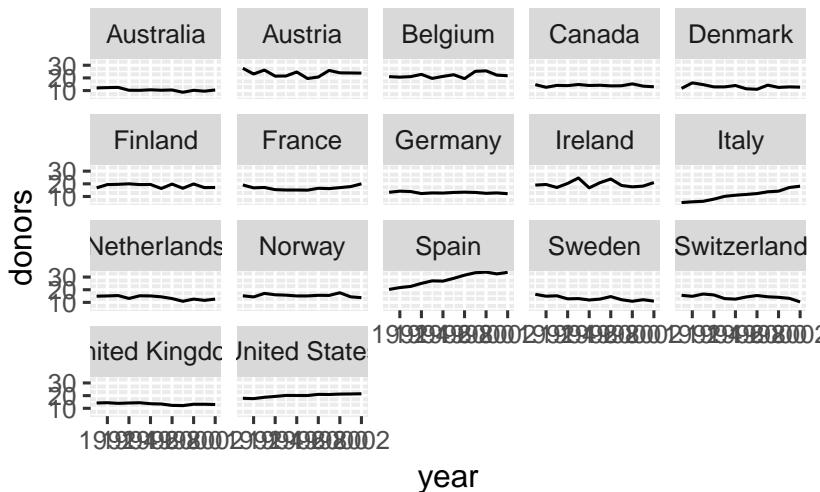
*Tot slot*

Met het transformeren moeten we wel voorzichtig zijn. Al met al heb je nu een gevoel wat de belangrijkste stappen zijn bij de visualisatie en het maken van figuren. Begin met een opgeschoonde dataset, we kunnen variabelen uitdrukken, variaties toepassen, lagen toevoegen, sommige aanvullingen maken. We weten wat we moeten doen, we weten hoe we labels en titels moeten toevoegen, evenals bijvoorbeeld subtitels. Vanuit deze basis kunnen we verder gaan en vaardigheden uitbreiden en de grafieken zo maken dat ze makkelijker te interpreteren zijn.

## Werken met geoms

Nu we een beetje door hebben hoe de structuur in elkaar zit en eenvoudige handelingen weten door te voeren, kunnen we eens kijken of we ook andere soorten plots kunnen maken en nieuwe geom\_functies kunnen gebruiken. De structuur blijft hetzelfde. Het wordt alleen allemaal wat geavanceerder. Er worden meer geoms voorgesteld waar je mee kunt werken. We leren ze beter aan te passen. We leren ook de variabelen beter te definiëren. We werken hier met een dataset over organdonatie in een aantal landen.

```
p <- ggplot(data = organdata,
             mapping = aes(x = year, y = donors))
p + geom_line(aes(group = country)) + facet_wrap(~ country)
## Warning: Removed 34 rows containing missing
## values (geom_path).
```

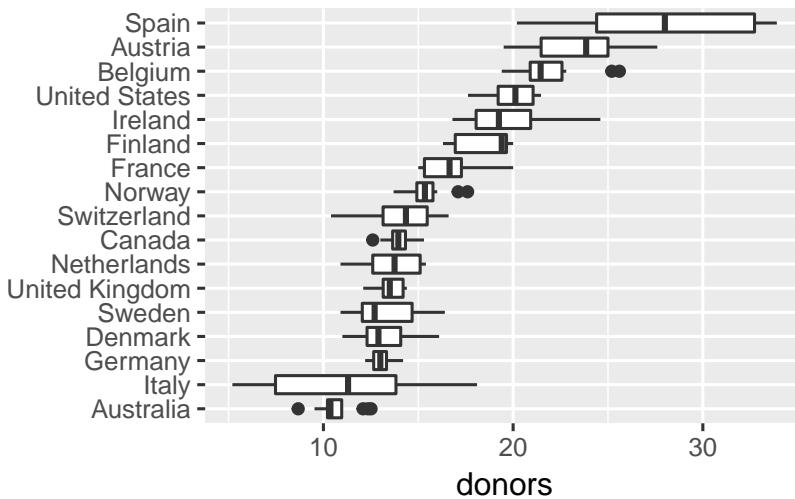


## Continue variabelen via groep of categorie

Laten we ons eerst eens concentreren op de variatie op landniveau. We herordenen het land door het gemiddelde van de donoren te nemen. Door reorder() wordt de categorie van de eerste variabele (land) geherordend op basis van het gemiddelde van donoren. Door toevoegen van na.rm=TRUE haal je de missings weg.

```
p <- ggplot(data = organdata,
             mapping = aes(x = reorder(country, donors, na.rm=TRUE),
                           y = donors))
p + geom_boxplot() +
  labs(x=NULL) +
  coord_flip()
```

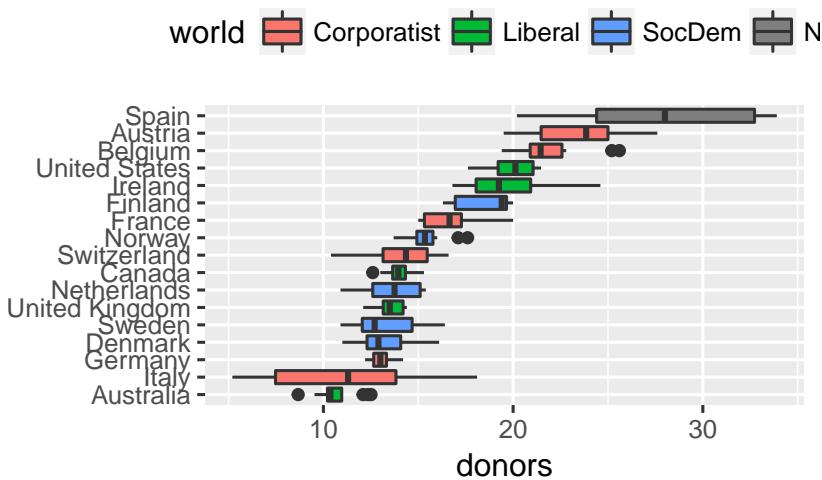
```
## Warning: Removed 34 rows containing non-finite
## values (stat_boxplot).
```



We kunnen er ook nog kleuren aan toevoegen.

```
p <- ggplot(data = organdata,
             mapping = aes(x = reorder(country, donors, na.rm=TRUE),
                           y = donors, fill = world))
p + geom_boxplot() + labs(x=NULL) +
  coord_flip() + theme(legend.position = "top")

## Warning: Removed 34 rows containing non-finite
## values (stat_boxplot).
```

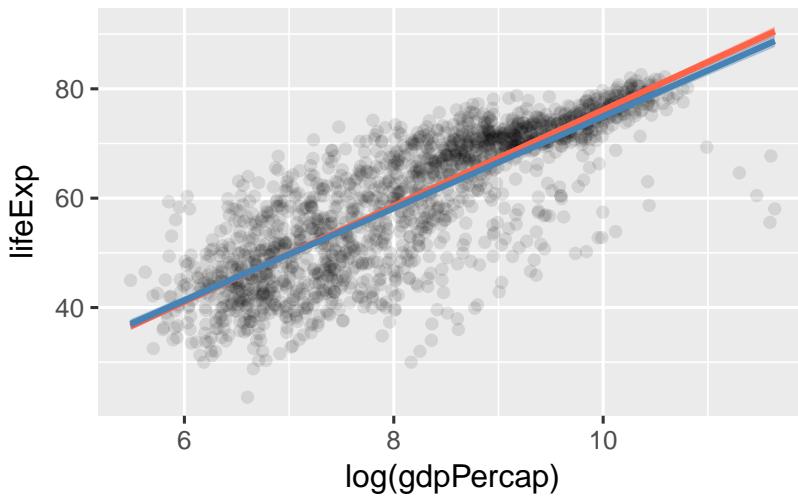


## *Werken met modellen*

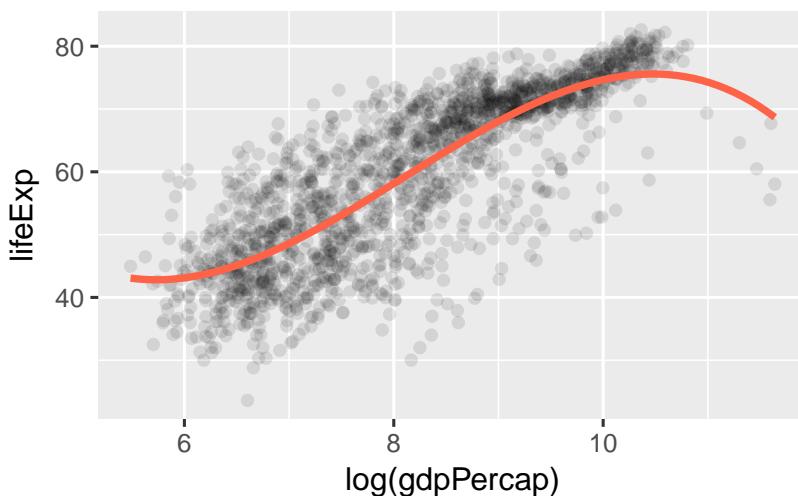
Bij datavisualisatie worden data vaak gecondenseerd en de resultaten geplot, zoals hieronder.

```
p <- ggplot(data = gapminder,
             mapping = aes(x = log(gdpPercap), y = lifeExp))

p + geom_point(alpha=0.1) +
  geom_smooth(color = "tomato", fill="tomato", method = MASS::rlm) +
  geom_smooth(color = "steelblue", fill="steelblue", method = "lm")
```



```
p + geom_point(alpha=0.1) +
  geom_smooth(color = "tomato", method = "lm", size = 1.2,
              formula = y ~ splines::bs(x, 3), se = FALSE)
```



```

p + geom_point(alpha=0.1) +
  geom_smooth(color = "tomato", size = 1.2, method = "rqss",
              lambda = 1, quantiles = c(0.20, 0.5, 0.85))

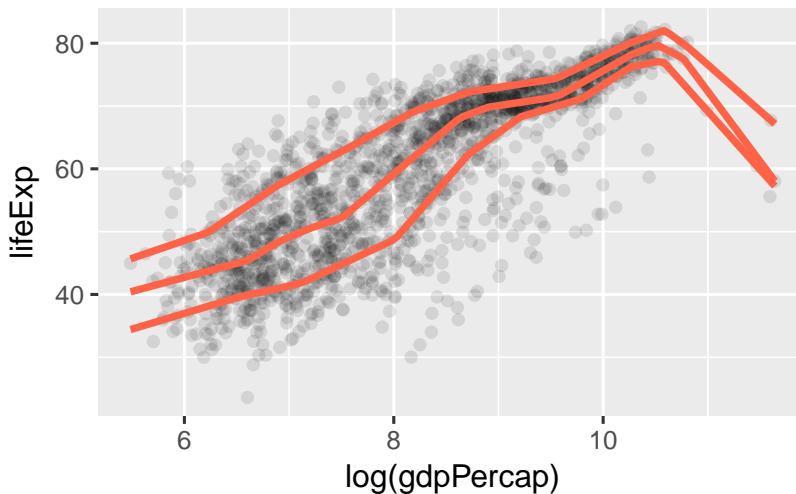
## Loading required package: SparseM

##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
## 
##     backsolve

## Smoothing formula not specified. Using: y ~ qss(x, lambda = 1)

```



Er zijn binnen ggplot ook verschillende mogelijkheden om de modellen in één keer af te drukken.

*Verschillende mogelijkheden ineens laten zien, met een legenda*

Hieronder wordt met geom\_smooth() gewerkt. Maar haal eerst RColorBrewer-pakket binnen zodat je de modellen met palette kleuren kunt afbeelden.

```

library(RColorBrewer)
model_colors <- RColorBrewer::brewer.pal(3, "Set1")
model_colors

## [1] "#E41A1C" "#377EB8" "#4DAF4A"

```

We drukken een figuur af met drie keer geom\_smooth en het onderscheid geven we met aes (color, fill) aan.

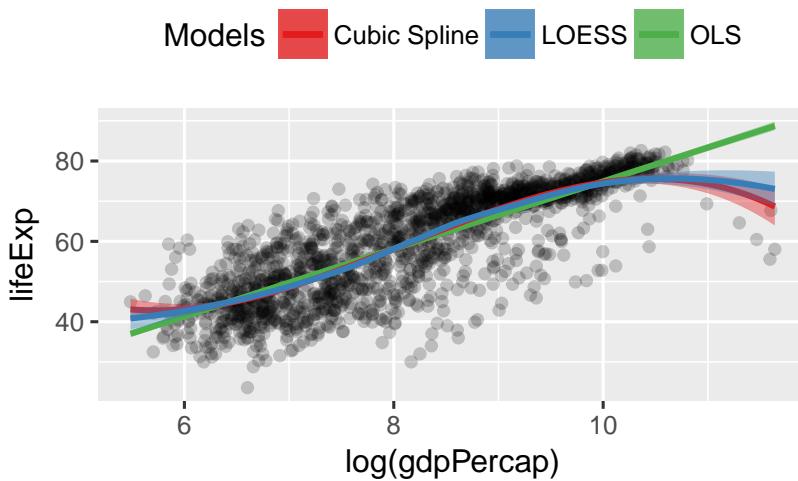
```

p0 <- ggplot(data = gapminder,
               mapping = aes(x = log(gdpPercap), y = lifeExp))

p1 <- p0 + geom_point(alpha = 0.2) +
  geom_smooth(method = "lm", aes(color = "OLS", fill = "OLS")) +
  geom_smooth(method = "lm", formula = y ~ splines::bs(x, df = 3),
              aes(color = "Cubic Spline", fill = "Cubic Spline")) +
  geom_smooth(method = "loess",
              aes(color = "LOESS", fill = "LOESS"))

p1 + scale_color_manual(name = "Models", values = model_colors) +
  scale_fill_manual(name = "Models", values = model_colors) +
  theme(legend.position = "top")

```



### *Model objecten opschonen met Broom*

Laten we eerst eens een lineaire regressie maken met Levensverwachting als uitkomst en BNP, populatie en continent als predictoren. We gebruiken nog steeds de dataset gapminder.

```

out <- lm(formula = lifeExp ~ gdpPercap + pop + continent,
           data = gapminder)

```

Hieronder zie je dan de vrij ingewikkelde uitkomsten van de regressie.

```
summary(out)
```

```
##  
## Call:
```

```

## lm(formula = lifeExp ~ gdpPercap + pop + continent, data = gapminder)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -49.161 -4.486  0.297  5.110 25.175 
##
## Coefficients:
##              Estimate Std. Error
## (Intercept) 4.781e+01 3.395e-01
## gdpPercap   4.495e-04 2.346e-05
## pop         6.570e-09 1.975e-09
## continentAmericas 1.348e+01 6.000e-01
## continentAsia    8.193e+00 5.712e-01
## continentEurope   1.747e+01 6.246e-01
## continentOceania  1.808e+01 1.782e+00
##                  t value Pr(>|t|)    
## (Intercept) 140.819 < 2e-16 ***
## gdpPercap   19.158 < 2e-16 ***
## pop         3.326 0.000901 ***
## continentAmericas 22.458 < 2e-16 ***
## continentAsia    14.342 < 2e-16 ***
## continentEurope   27.973 < 2e-16 ***
## continentOceania  10.146 < 2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.365 on 1697 degrees of freedom
## Multiple R-squared:  0.5821, Adjusted R-squared:  0.5806 
## F-statistic: 393.9 on 6 and 1697 DF,  p-value: < 2.2e-16

```

Met het pakket broom kun je de uitkomsten eenvoudiger en helderder afdrukken. Kijk maak.

```

library(broom)
out_comp <- tidy(out)

out_comp %>% round_df()

##             term estimate std.error
## 1 (Intercept) 47.81     0.34
## 2 gdpPercap   0.00     0.00
## 3 pop        0.00     0.00
## 4 continentAmericas 13.48     0.60
## 5 continentAsia    8.19     0.57

```

```

## 6   continentEurope    17.47    0.62
## 7   continentOceania   18.08    1.78
##   statistic p.value
## 1    140.82     0
## 2    19.16     0
## 3     3.33     0
## 4    22.46     0
## 5    14.34     0
## 6    27.97     0
## 7    10.15     0

```

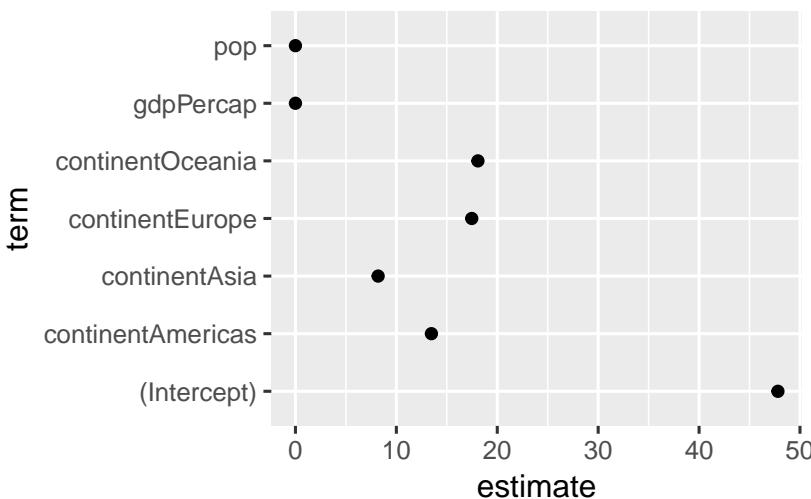
Vervolgens is het ook makkelijker om de regressie in een grafiek uit te drukken.

```

p <- ggplot(out_comp, mapping = aes(x = term,
                                      y = estimate))

p + geom_point() + coord_flip()

```



Nu nog een keer dezelfde regressie maar nu voegen we het betrouwbaarheidsinterval er aan toe (je ziet conf.low en conf.high).

```

out_conf <- tidy(out, conf.int = TRUE)
out_conf %>% round_df()

##           term estimate std.error
## 1 (Intercept)  47.81    0.34
## 2 gdpPercap    0.00    0.00
## 3 pop          0.00    0.00
## 4 continentAmericas 13.48    0.60
## 5 continentAsia    8.19    0.57
## 6 continentEurope   17.47    0.62

```

```

## 7 continentOceania    18.08     1.78
##   statistic p.value conf.low conf.high
## 1    140.82      0    47.15    48.48
## 2     19.16      0     0.00     0.00
## 3      3.33      0     0.00     0.00
## 4     22.46      0    12.30    14.65
## 5     14.34      0     7.07    9.31
## 6     27.97      0    16.25   18.70
## 7     10.15      0    14.59   21.58

out_conf <- subset(out_conf, term %in% "(Intercept)")
out_conf$nicelabs <- prefix_strip(out_conf$term, "continent")

```

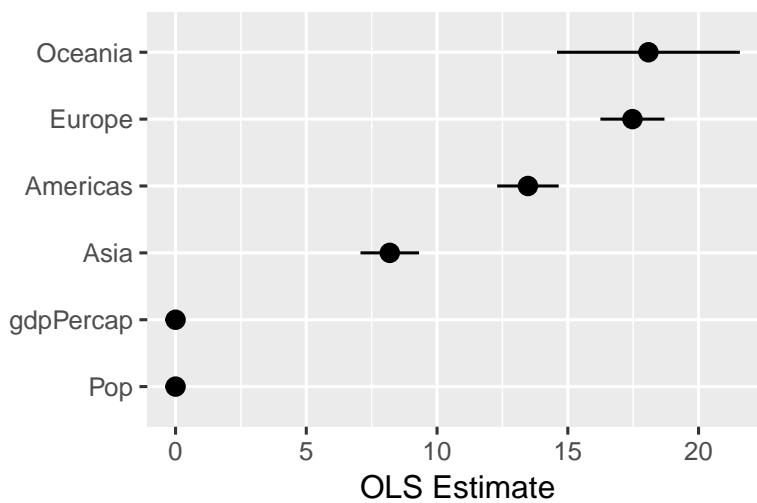
Vervolgens drukken we dit dan weer in een grafiek af met het betrouwbaarheidsinterval.

```

p <- ggplot(out_conf, mapping = aes(x = reorder(nicelabs, estimate),
                                       y = estimate, ymin = conf.low, ymax = conf.high))

p + geom_pointrange() + coord_flip() + labs(x="", y="OLS Estimate")

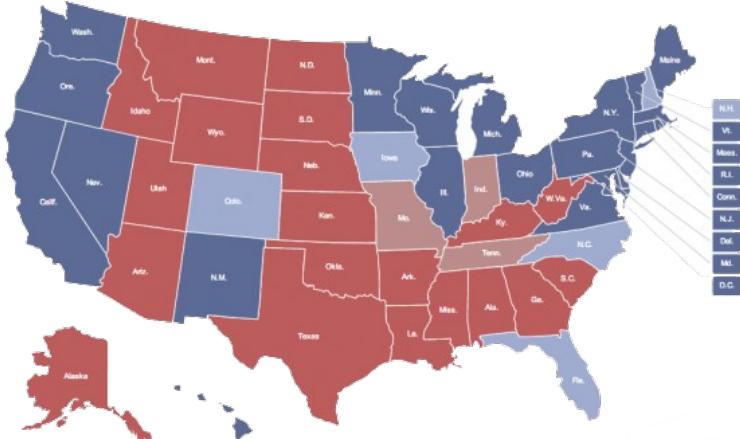
```



## Kaarten maken

Met choropleth kaarten zijn geografische regionen goed af te beelden, zeker als je ze wilt onderscheiden van een zeer beperkt aantal variabelen. Hieronder zie je een aantal verkiezingskaarten van Amerika. Het zijn de resultaten van de verkiezingen in 2012. Het zijn dezelfde data maar deze worden steeds anders afgebeeld. De onderliggende kwantiteit van interesse verschilt, de indeling van regio's kan verschillen. Uiteindelijk represeneert elke kaart wat jij wilt laten zien.

De kenmerken van de kaarten hangen mede af van het niveau waarin



je de gegevens kunt representeren.]

## Amerika in kaart brengen

Laten we de data van de 2016-presidentsverkiezingen eens gebruiken en in een grafiek zetten. Eerst maar eens kijken wat we hebben.

```
election %>% select(state, total_vote,
                      r_points, pct_trump, party, census) %>%
  sample_n(5)

## # A tibble: 5 x 6
##   state  total_vote r_points pct_trump party
##   <chr>     <dbl>     <dbl>      <dbl> <chr>
## 1 Kentu~    1924149.    29.8       62.5 Repu~
## 2 Alaba~    2123372.    27.7       62.1 Repu~
## 3 New M~    798319.     -8.22      40.0 Demo~
## 4 Arkan~    1130635.    26.9       60.6 Repu~
## 5 New J~    3906723.    -14.0      41.0 Demo~
```

Het eerste wat we dan altijd moeten doen is eerst de informatie goed krijgen en wel in de goede volgorde. We gebruiken R's-map

pakket hiervoor dat geeft ons wat informatie voordat we het figuur maken.

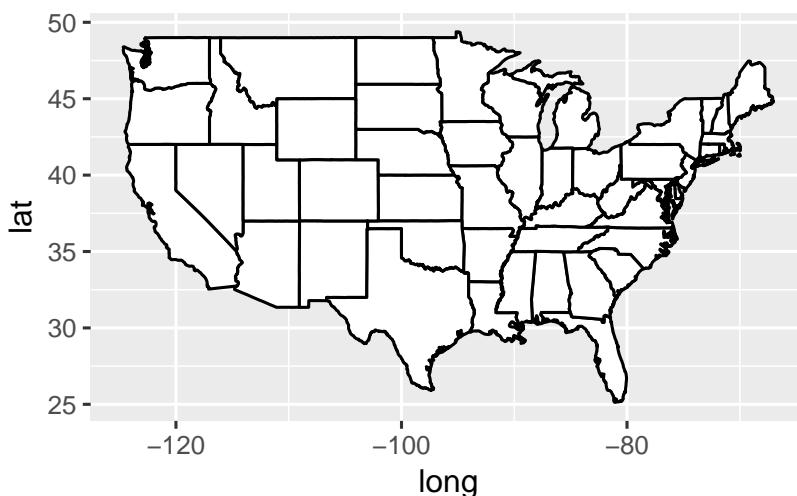
```
library(maps)
us_states <- map_data("state")
head(us_states)

##      long      lat group order  region
## 1 -87.46201 30.38968     1     1 alabama
## 2 -87.48493 30.37249     1     2 alabama
## 3 -87.52503 30.37249     1     3 alabama
## 4 -87.53076 30.33239     1     4 alabama
## 5 -87.57087 30.32665     1     5 alabama
## 6 -87.58806 30.32665     1     6 alabama
##   subregion
## 1      <NA>
## 2      <NA>
## 3      <NA>
## 4      <NA>
## 5      <NA>
## 6      <NA>
```

Dit is gewoon een dataframe met meer dan 15.000 rijen waarmee de kaart wordt gemaakt. Daar kunnen we al iets eenvoudigs van maken.

```
p <- ggplot(data = us_states,
             mapping = aes(x = long, y = lat,
                           group = group))

p + geom_polygon(fill = "white", color = "black")
```

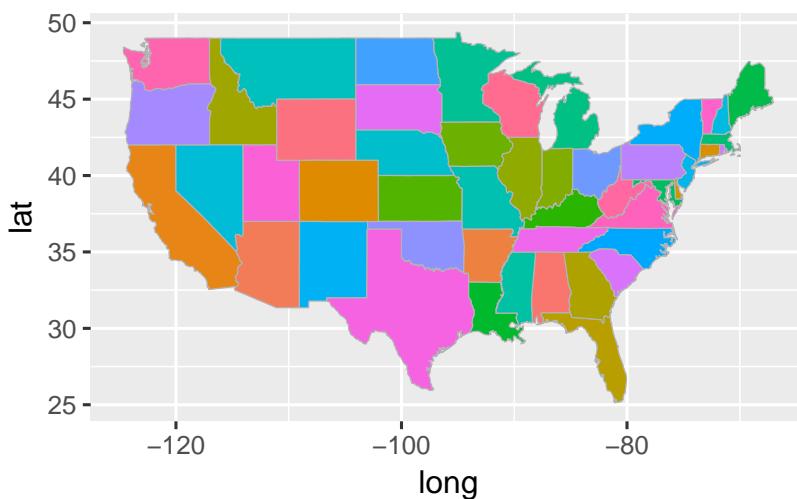


De kaart wordt gemaakt met breedte en lengtegraadpunten, die als elementen op de x en y as worden gezet. De punten worden in een kaart met elkaar verbonden.

We kunnen de staten met een kleur opvullen en staten onderscheidende kleuren geven.

```
p <- ggplot(data = us_states,
             aes(x = long, y = lat,
                 group = group, fill = region))

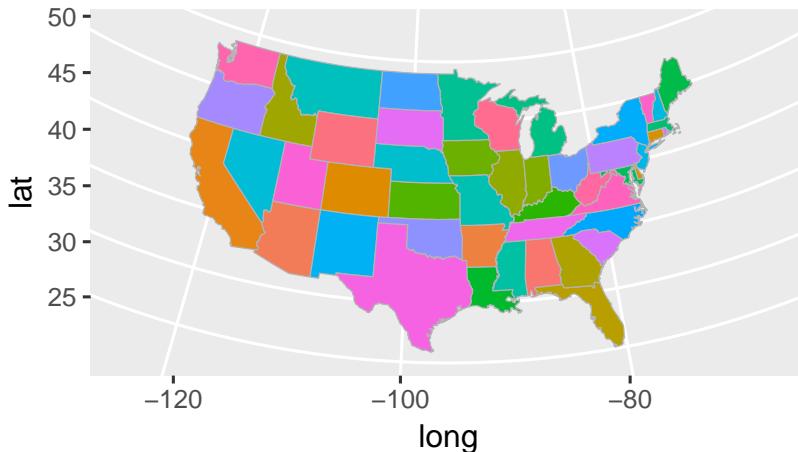
p + geom_polygon(color = "gray70", size = 0.2) + guides(fill = FALSE)
```



Vervolgens moet de kaart wat meer geprojecteerd worden en moeten de coördinaten worden aangepast zodat meer rekening wordt gehouden met de ronding van de aarde.

```
p <- ggplot(data = us_states,
             mapping = aes(x = long, y = lat,
                           group = group, fill = region))

p + geom_polygon(color = "gray70", size = 0.2) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45) +
  guides(fill = FALSE)
```



Als je de kaart hebt moet je vervolgens de data op de kaart krijgen. De eigen data moeten worden gecombineerd met de gegevens van de dataframe. Bij het combineren kunnen heel gemakkelijk fouten ontstaan. Pas hier erg goed op.

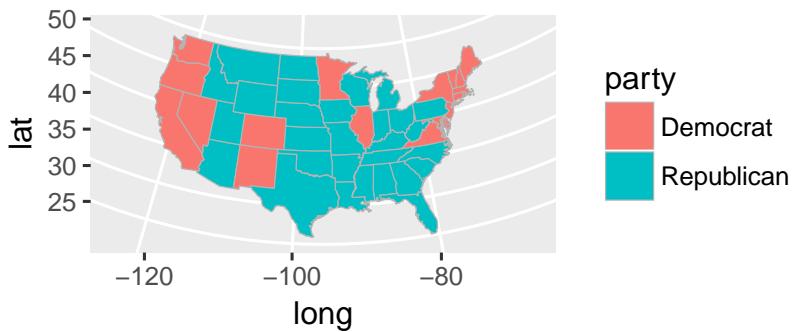
```
election$region <- tolower(election$state)
us_states_elec <- left_join(us_states, election)

## Joining, by = "region"
```

Als we bovenstaande hebben gedaan, moeten de datasets goed zijn gecombineerd en hebben we een 'gemergde'-dataset.

```
p <- ggplot(data = us_states_elec,
             aes(x = long, y = lat,
                  group = group, fill = party))

p + geom_polygon(color = "gray70", size = 0.2) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45)
```



Vervolgens gebruiken we de partijkleuren om de staten mee te vullen, zetten we de legenda er onderaan en geven we de kaart een titel mee. Vervolgens halen we ook nog de lijnen weg door een bepaald thema van kaarten te gebruiken (maar daarover straks meer).

```
library(scales)
library(maps)
p0 <- ggplot(data = us_states_elec,
               mapping = aes(x = long, y = lat,
                               group = group, fill = party))
p1 <- p0 + geom_polygon(color = "gray70", size = 0.2) +
      coord_map(projection = "albers", lat0 = 39, lat1 = 45)
p2 <- p1 + scale_fill_manual(values = party_colors) +
      labs(title = "Election Results 2016", fill = NULL)
```

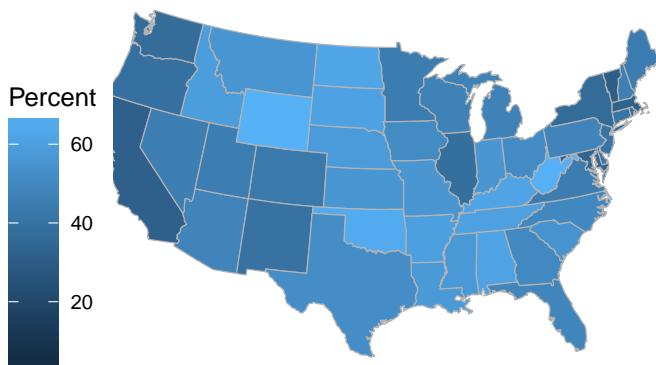
Je kunt het ook nog wat subtieler doen.

```
library(ggthemes)
p0 <- ggplot(data = us_states_elec,
               mapping = aes(x = long, y = lat, group = group, fill = pct_trump))

p1 <- p0 + geom_polygon(color = "gray70", size = 0.2) +
      coord_map(projection = "albers", lat0 = 39, lat1 = 45)

p1 + labs(title = "Trump vote") + theme_map() + labs(fill = "Percent")
```

### Trump vote



```
p2 <- p1 + scale_fill_gradient(low = "white", high = "#CB454A") +
  labs(title = "Trump vote")
# p2 + theme_map() + labs(fill = "Percent")
```

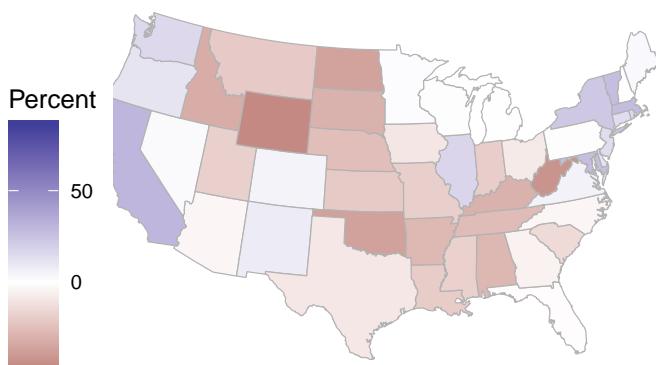
Je kunt ook een opschaling gebruiken die vanuit een middelpunt begint.

```
p0 <- ggplot(data = us_states_elec,
  mapping = aes(x = long, y = lat, group = group, fill = d_points))

p1 <- p0 + geom_polygon(color = "gray70", size = 0.2) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45)

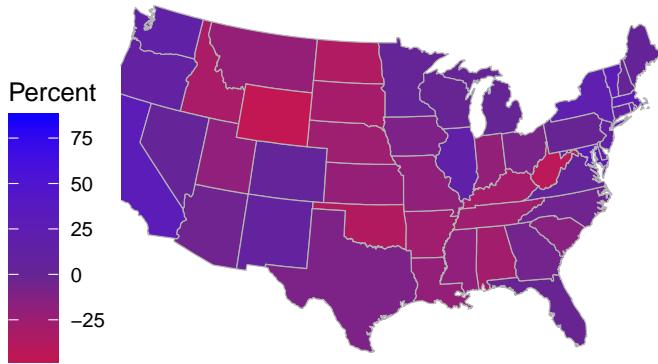
p2 <- p1 + scale_fill_gradient2() + labs(title = "Winning margins")
p2 + theme_map() + labs(fill = "Percent")
```

### Winning margins



```
p3 <- p1 + scale_fill_gradient2(low = "red", mid = scales::muted("purple"),
                                 high = "blue", breaks = c(-25, 0, 25, 50, 75)) +
  labs(title = "Winning margins")
p3 + theme_map() + labs(fill = "Percent")
```

Winning margins



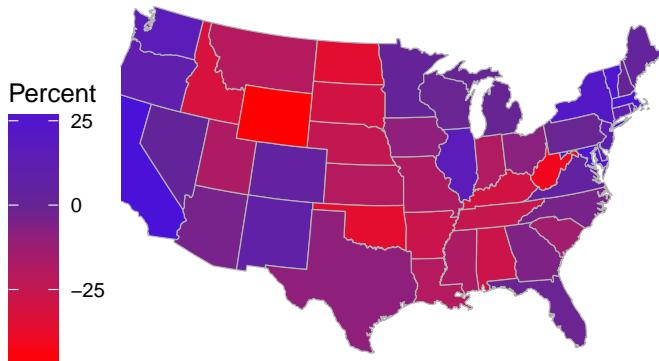
En als we nog beter met de data rekening willen houden krijgen we het volgende beeld.

```
p0 <- ggplot(data = subset(us_states_elec,
                           region %in% "district of columbia"),
              aes(x = long, y = lat, group = group, fill = d_points))

p1 <- p0 + geom_polygon(color = "gray70", size = 0.2) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45)

p2 <- p1 + scale_fill_gradient2(low = "red",
                                 mid = scales::muted("purple"),
                                 high = "blue") +
  labs(title = "Winning margins")
p2 + theme_map() + labs(fill = "Percent")
```

## Winning margins



## Kleinere, meerdere kaarten tegelijk

Soms heb je ook geografische data met herhaalde observaties over tijd. Mogelijk een bepaalde maat op land of staat-niveau die over een periode van enkele jaren is gemeten. Bijvoorbeeld data van opiaten op het niveau van Amerikaanse staten tussen 1999 en 2014.

```
head(opiates)

## # A tibble: 6 x 10
##   Year State  FIPS Deaths Population Crude
##   <int> <chr> <int> <int>     <dbl>
## 1 1999 Alaba~     1     37    4430141 0.800
## 2 1999 Alaska      2     27    624779 4.30
## 3 1999 Arizo~     4    229    5023823 4.60
## 4 1999 Arkan~     5     28    2651860 1.10
## 5 1999 Calif~     6   1474   33499204 4.40
## 6 1999 Color~     8    164    4226018 3.90
## # ... with 4 more variables: Adjusted <dbl>,
## #   Adjusted.se <dbl>, Region <ord>,
## #   Abbr <chr>
```

De Staat (State) variabelen wordt op kleine letter gezet zodat het beter werkt.

```
opiates$region <- tolower(opiates$State)
opiates_map <- left_join(us_states, opiates)

## Joining, by = "region"
```

In het opiatenbestand zit ook de Jaar(Year) variabele. Voor elk jaar maken we een aparte kaart...

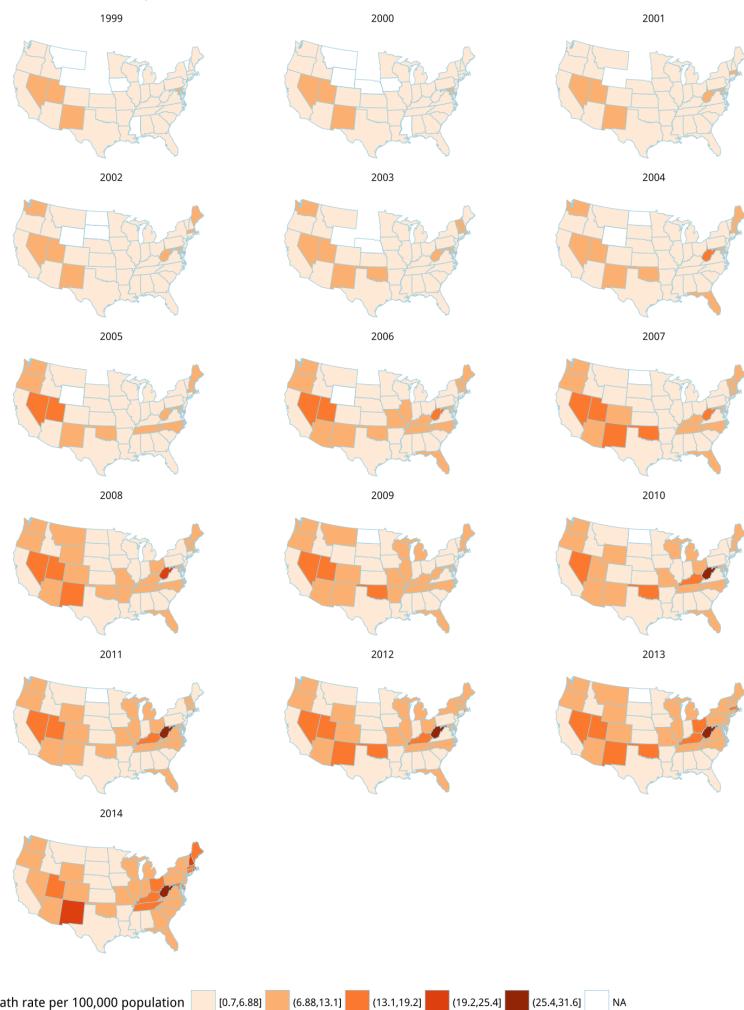
```
p0 <- ggplot(data = opiates_map,
               mapping = aes(x = long, y = lat,
                             group = group,
                             fill = cut_interval(Adjusted, n = 5)))

p1 <- p0 + geom_polygon(color = "lightblue", size = 0.2) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45)

p2 <- p1 + scale_fill_brewer(type = "seq", palette = "Oranges")

p2 + theme_map() + facet_wrap(~ Year, ncol = 3) +
  guides(fill = guide_legend(nrow = 1)) +
  theme(legend.position = "bottom",
        strip.background = element_blank()) +
  labs(fill = "Death rate per 100,000 population",
       title = "Opiate Related Deaths by State, 1999-2014")
```

Opiate Related Deaths by State, 1999-2014



^  
En ook hier is het dataframe van de staten gekoppeld aan de opiatendata.

## Verfijnen

Met de default-functies van ggplot kom je vaak al een heel eind. Alleen wanneer je de figuren wilt verfijnen of specifiek wilt aanpassen, heb je enkele functies nodig. Dat verfijnen kan ook in ggplot. Laten we eerst eens naar een nieuwe dataset kijken. Deze dataset omvat het lidmaatschap van de Amerikaanse sociologische vereniging.

```
head(asasec)
```

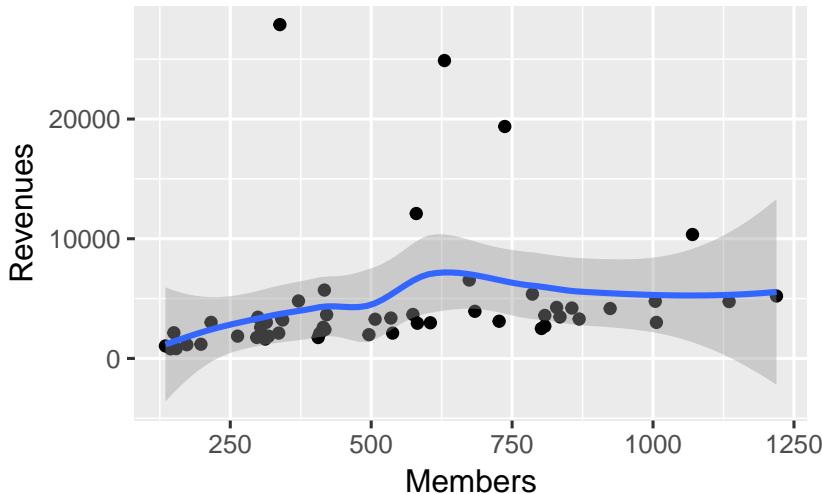
```
##                                     Section
## 1      Aging and the Life Course (018)
## 2      Alcohol, Drugs and Tobacco (030)
## 3 Altruism and Social Solidarity (047)
## 4          Animals and Society (042)
## 5          Asia/Asian America (024)
## 6          Body and Embodiment (048)
##           Sname Beginning Revenues Expenses
## 1      Aging     12752    12104    12007
## 2 Alcohol/Drugs     11933     1144      400
## 3   Altruism      1139     1862    1875
## 4     Animals       473      820    1116
## 5      Asia        9056     2116    1710
## 6      Body        3408     1618    1920
##   Ending Journal Year Members
## 1 12849      No 2005     598
## 2 12677      No 2005     301
## 3 1126       No 2005      NA
## 4 177        No 2005     209
## 5 9462       No 2005     365
## 6 3106       No 2005      NA
```

In de dataset gaat het om lidmaatschapsdata voor elke sectie over een periode van tien jaar. Laten we kijken naar de leden en de revenues in het jaar 2014.

```
p <- ggplot(data = subset(asasec, Year == 2014),
            mapping = aes(x = Members, y = Revenues, label = Sname))

p + geom_point() + geom_smooth()

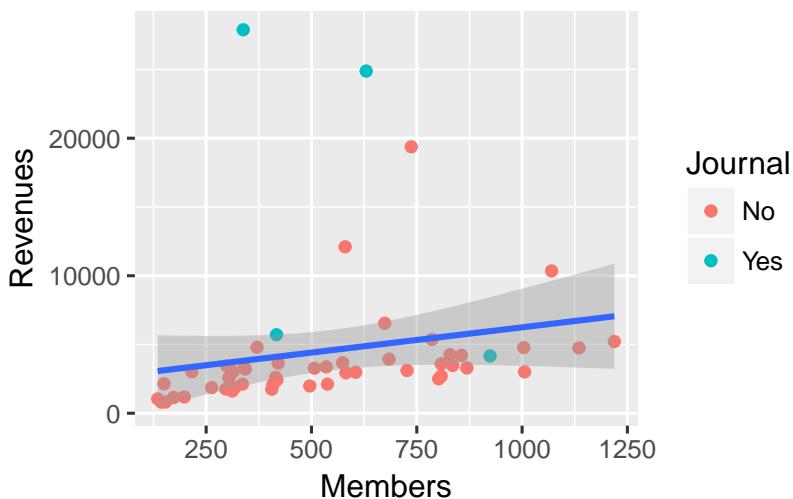
## 'geom_smooth()' using method = 'loess'
```



Om het te verfijnen kijken we eerst naar enkele uitschieters, gebruiken we een andere methode en introduceren we een derde variabele (Journal).

```
p <- ggplot(data = subset(asasec, Year == 2014),
             mapping = aes(x = Members, y = Revenues, label = Sname))

p + geom_point(mapping = aes(color = Journal)) +
  geom_smooth(method = "lm")
```



Vervolgens voegen we er ook nog wat tekst aan toe.

```
library(ggrepel)
p0 <- ggplot(data = subset(asasec, Year == 2014),
              mapping = aes(x = Members, y = Revenues, label = Sname))

p1 <- p0 + geom_smooth(method = "lm", se = FALSE, color = "gray80") +
```

```

geom_point(mapping = aes(color = Journal))

p2 <- p1 + geom_text_repel(data=subset(asasec,
                                         Year == 2014 & Revenues > 7000),
                           size = 2)

```

En nog wat aanpassingen:

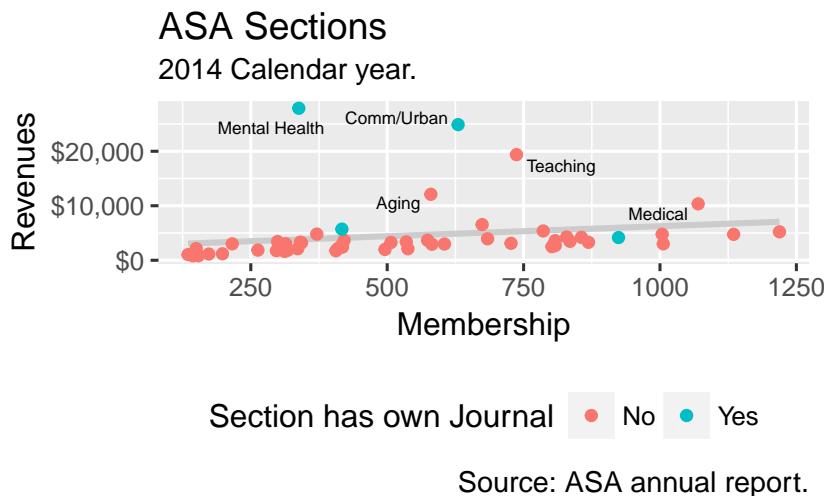
```

p3 <- p2 + labs(x="Membership",
                 y="Revenues",
                 color = "Section has own Journal",
                 title = "ASA Sections",
                 subtitle = "2014 Calendar year.",
                 caption = "Source: ASA annual report.")

p4 <- p3 + scale_y_continuous(labels = scales::dollar) +
  theme(legend.position = "bottom")

p4

```



*Kleur gebruiken in jouw voordeel*

Kleuren zijn belangrijk in het maken van figuren. Ongeordende variabelen zoals land of sexe moeten soms met kleur helder gemaakt worden. Wanneer je een geordende variabele hebt (onderwijsniveau) werkt een palette van kleur goed.



gebruik van kleuren

Het



ColorBrewer's  
sequentiele paletten



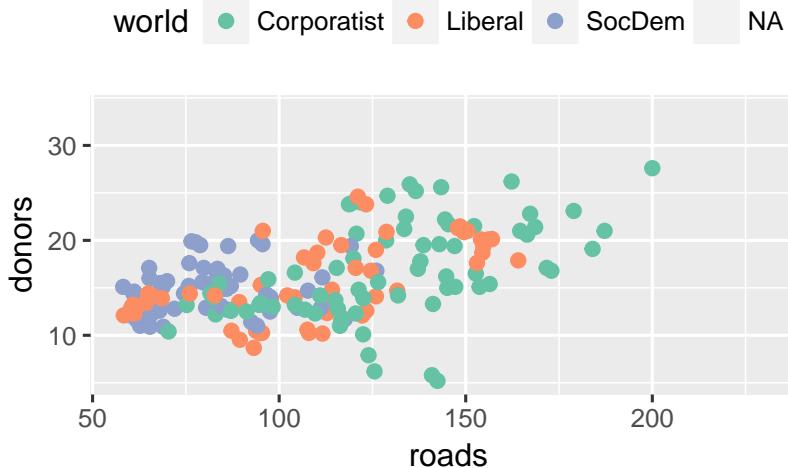
ColorBrewer's

### kwalitatieve paletten

Hieronder zie je enkele verschillende kleurspecificaties in de grafieken.

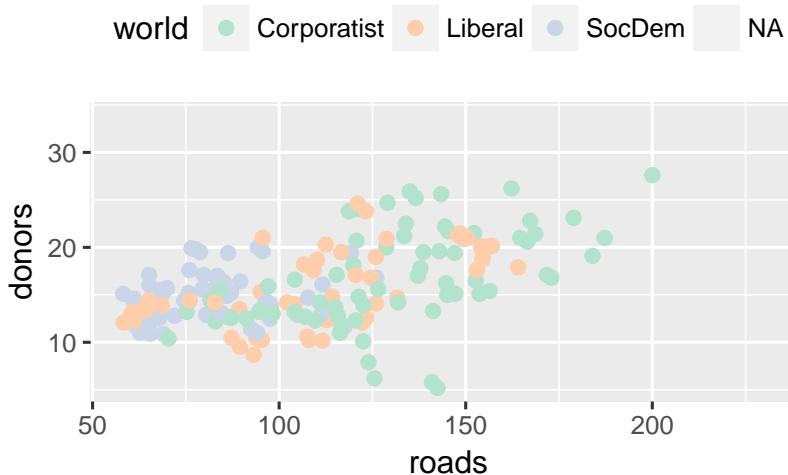
```
p <- ggplot(data = organdata,
             mapping = aes(x = roads, y = donors, color = world))
p + geom_point(size = 2) + scale_color_brewer(palette = "Set2") +
  theme(legend.position = "top")

## Warning: Removed 46 rows containing missing values
## (geom_point).
```



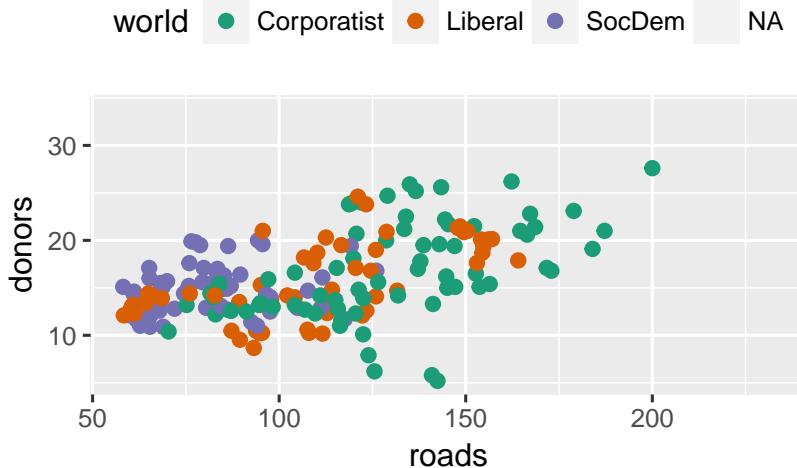
```
p + geom_point(size = 2) + scale_color_brewer(palette = "Pastel2") +
  theme(legend.position = "top")

## Warning: Removed 46 rows containing missing values
## (geom_point).
```



```
p + geom_point(size = 2) + scale_color_brewer(palette = "Dark2") +
  theme(legend.position = "top")

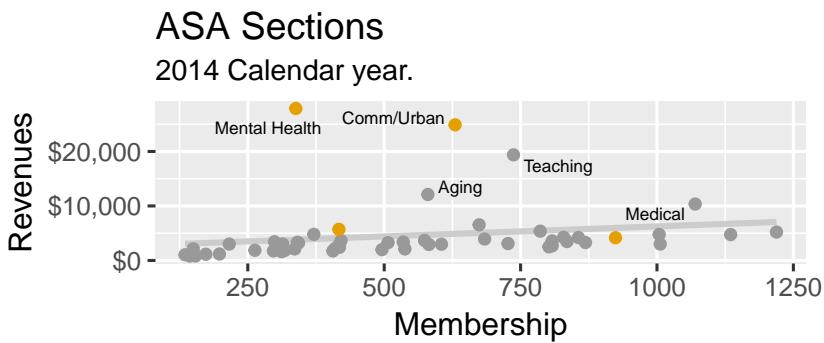
## Warning: Removed 46 rows containing missing values
## (geom_point).
```



Kleuren kun je met de hand instellen.

```
cb.palette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
                 "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

p4 + scale_color_manual(values = cb.palette)
```



Section has own Journal    ● No    ● Yes

Source: ASA annual report.

Dat instellen met de hand is soms nodig omdat het aanpast bij een partij bijvoorbeeld. Soms kun je ook de kleuren in het programma gebruiken.

## Tot slot

Dit document dat voor jou ligt laat maar een deel zien van het uitgebreide boek van Kieran Healy dat een hele goede introductie is op datavisualisatie voor de sociale wetenschappen. Sla het boek van Healy er op na en probeer verschillende grafieken te maken. Neem de tijd en probeer het rustig uit.

Aan zijn boek voegt Healy ook nog enkele Appendixen toe waarin hij ingaat op onderwerpen die met goede grafieken maken van doen hebben zoals het selecteren van elementen, opruimen van data, algemene problemen waar je tegenaan kunt lopen, functies schrijven voor herhaalde opdrachten, opslaan van de grafieken, RMarkdown en knitr, kaarten op een bepaald niveau maken en het thema (het format) waarin dit boek is gemaakt.

Healy geeft ook nog een overzicht van boeken die voor dit werk fundamenteel zijn. Hier wat boeken over R: - Garrett Grolemund and Hadley Wickham. 2016. *R for Data Science*. O'Reilly.

- Winston S. Chang. 2013. *The R Graphics Cookbook*. O'Reilly.
- Roger Peng. 2016. *R Programming for Data Science*.
- Hadley Wickham. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Second edition. Springer.

Dan wat materiaal dat ingaat op bepaalde applicaties waar in dit boek gebruik van is gemaakt: - Chris Brundson and Lex Comber.

2015. *An Introduction to R for Spatial Analysis and Mapping*. Sage.

- Peter W. Dalgaard. 2008. *Introductory Statistics with R*. 2nd Ed. Springer.

- Tilman M. Davies. 2016. *The Book of R*. No Starch Press.
- Michael Friendly and David Meyer. 2017. *Discrete Data Analysis with R*. CRC/Chapman and Hall.
- Frank E. Harrell, Jr. 2015. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. 2nd Ed. Springer.
- Kosuke Imai. 2017. *Quantitative Social Science: An introduction*. Princeton.

- W.N. Venables and B.D. Ripley. 2002. *Modern Applied Statistics with S*. 4th Ed. Springer.

- Dan wat boeken over datavisualisatie:
- Jacques Bertin. 2010 [1967]. *Semiology of Graphics*. Esri Press.
  - William S. Cleveland. 1993. *Visualizing Data*. Hobart Press.
  - William S. Cleveland. 1994. *The Elements of Graphing Data*. Revised Edition. Hobart Press.
  - Stephen Few. 2009. *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Analytics Press.
  - Tamara Munzer. 2014. *Visualization Analysis & Design*. CRC Press.

- Edward R. Tufte. 1983. *The Visual Display of Quantitative Information*. Graphics Press.

- Colin Ware. 2008. *Visual Thinking for Design*. Morgan Kaufman.

Dan zijn er websites rereleteerd aan R:

- Try R.([www.tryr.codeschool.com](http://www.tryr.codeschool.com));

- IDRE ([www.ats.ucla.edu/stat/](http://www.ats.ucla.edu/stat/));

- Flowingdata.([www.flowingdata.com](http://www.flowingdata.com));

- RStudio ([www.rstudio.com/resources/cheatsheets/](http://www.rstudio.com/resources/cheatsheets/));

- Stack Sites([www.stackoverflow.com](http://www.stackoverflow.com)).