Introductie ggplot2

A picture is worth a thousand words

Ivy Jansen, Pieter Verschelde, Thierry Onkelinx

Waarom ggplot?

- `ggplot` maakt grafieken gebaseerd op de Grammar of Graphics
  - Je geeft de gegevens mee
  - Je vertelt ggplot2 welke variabelen op de assen moeten komen
  - Je zegt welk type grafiek je wil maken (punten, lijnen, boxplot, histogram, ...)
  - En je zorgt voor de details (titels, kleuren, vormen, ...)
- De basisgrafieken zien er al heel mooi uit
- Alle soorten grafieken vertrekken van dezelfde syntax
- Grafieken worden stap voor stap opgebouwd
- Gemakkelijk om kleuren, groottes, ... te laten variëren per groep
- Automatische legende
- Mogelijk om verschillende datasets te combineren
- ...

Datasets

- `iris` data uit R
  - Opgesplitst in `irisSepal` en `irisPetal`
  - Variabelen hernoemd naar `Length` en `Width`
  - Extra variabele `Leaf.Type` aangemaakt, gelijk aan `sepal` of `petal`
  - Beide datasets onder mekaar geplakt en bewaard in `irisAll`
  - Deze 3 datasets zijn bewaard als `R` objecten, en worden ingelezen met de functie `load()`

```
load("data/mijnIris.Rdata")
```

Package `ggplot2`

- Dit package heb je al als je het package `tidyverse` geïnstalleerd hebt
- Laad het package `tidyverse` (of alleen `ggplot2`)

```
# library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------------------
## v ggplot2 3.1.0       v purrr   0.3.1
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
## -- Conflicts -------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()    masks stats::lag()
```

- Je ziet dat er een hele reeks packages geladen worden
- Er zijn 2 conflicten
  - De functie `filter()` uit het package `dplyr` overschrijft de functie `filter()` uit het `stats` package
  - De functie `lag()` uit het package `dplyr` overschrijft de functie `lag()` uit het `stats` package
  - Geen probleem, maar wees je ervan bewust wanneer je hulp zoekt over deze functies, dat je die uit het juiste package (`dplyr`) bestudeert (`R` geeft het aan als er verschillende mogelijkheden zijn)

Basis syntax `ggplot`

- Het package heet `ggplot2`, de belangrijkste functie is `ggplot()`
- Zoek hulp over `ggplot`

ggplot {ggplot2} R Documentation

Create a new ggplot

Description

`ggplot()` initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```
ggplot(data = NULL, mapping = aes(), ...,
  environment = parent.frame())
```

Arguments

`data`	Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
`mapping`	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
`...`	Other arguments passed on to methods. Not currently used.
`environment`	DEPRECATED. Used prior to tidy evaluation.

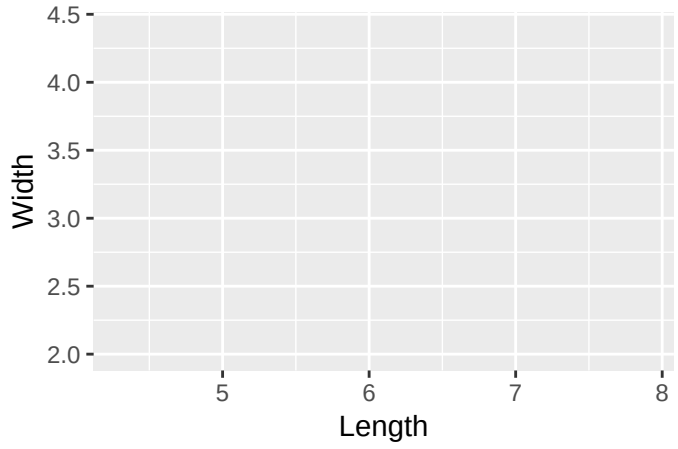
Details

`ggplot()` is used to construct the initial plot object, and is almost always followed by + to add component to the plot. There are three common ways to invoke `ggplot`:

`ggplot()` initialiseert de componenten van het `ggplot` object

- `data = NULL`: de dataset die gebruikt wordt voor de grafiek
- `mapping = aes()`: de aesthetics (*assen*), variabelen in de dataset

```
# ggplot(data = irisSepal, mapping = aes(x = Length, y = Width))
ggplot(irisSepal, aes(x = Length, y = Width))
```



- Blanco figuur
- Enkel dataset gespecifieerd, en X- en Y-as

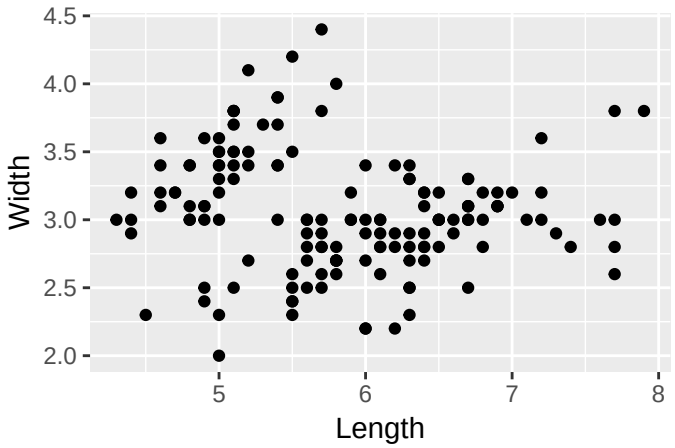
- Daarna worden er 1 of meerdere lagen met punten, lijnen,... toegevoegd volgens diezelfde X- en Y-assen, met `+ geom_xxx()`.

Lagen toevoegen

Punten

- `geom_point`

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point()
```



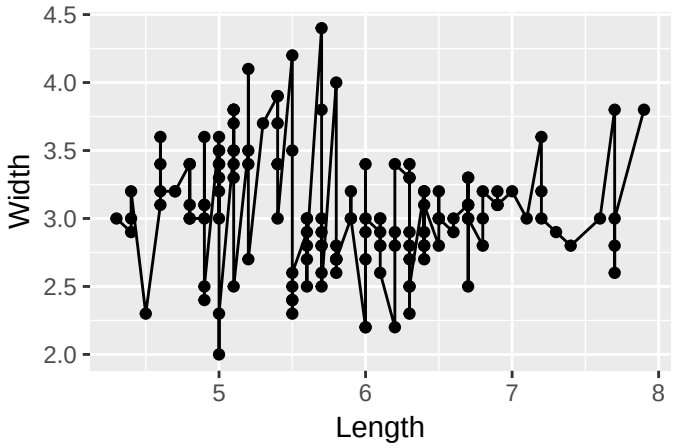
- `geom_jitter`: voegt wat ruis toe, interessant wanneer er veel punten over mekaar vallen

Lijnen

`geom_line`

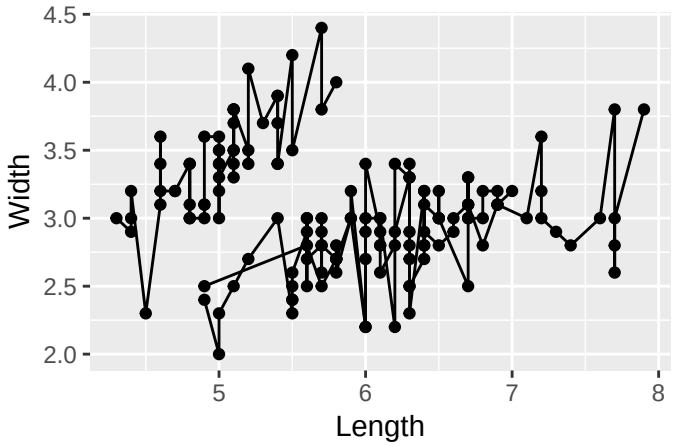
- Verbindt de punten volgens de waarden op de X-as

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point() +
  geom_line()
```



- Eventueel groeperen per soort met het `group` aesthetic

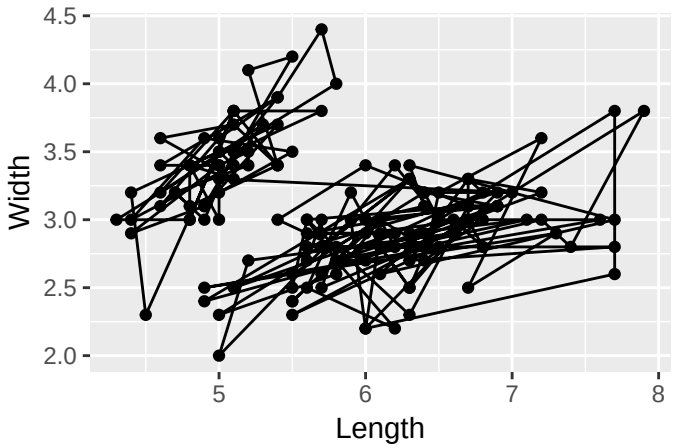
```
ggplot(irisSepal, aes(x = Length, y = Width, group = Species)) +
  geom_point() +
  geom_line()
```



`geom_path`

- Verbindt de punten volgens de volgorde in de data

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point() +
  geom_path()
```

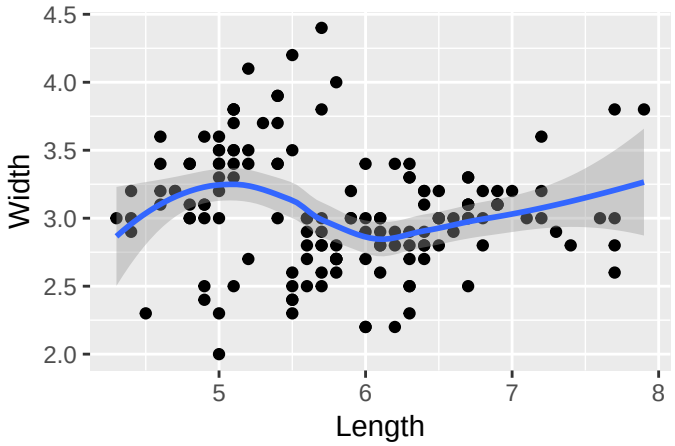


`geom_smooth`

- Smoother zonder verdere specificatie, om een patroon te herkennen in de punten

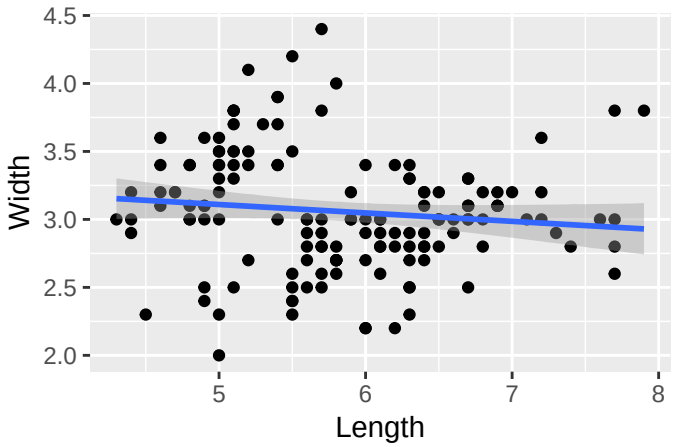
```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



- Lineaire smoother met `method = "lm"`

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point() +
  geom_smooth(method = "lm")
```



- Veel meer opties mogelijk, zie help

Andere lijnen

- `geom_hline`: horizontale lijn
- `geom_vline`: verticale lijn
- `geom_abline`: hellende lijn

Aesthetics

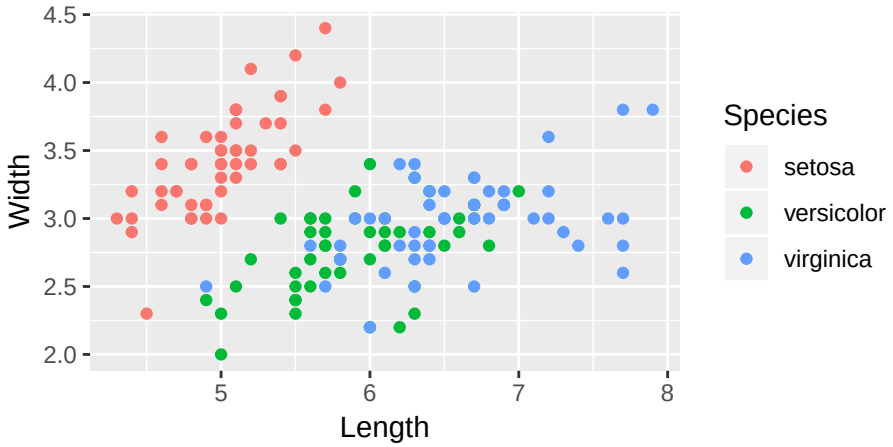
Naast de X- en Y-as kunnen we nog extra *assen* toevoegen aan de figuur:

- `color`, `shape`, `linetype`, `size`, `fill`
- Specifiëer je deze argumenten **binnen** de `aes()`, dan variëren ze volgens een bepaalde variabele (kenmerk van de observaties)
- Specifiëer je deze argumenten **buiten** de `aes()`, dan zijn het vaste kenmerken
- Dit kan voor de volledige grafiek hetzelfde, of verschillend per laag, afhankelijk in welke aesthetics `aes()` je dit argument toevoegt

`color`

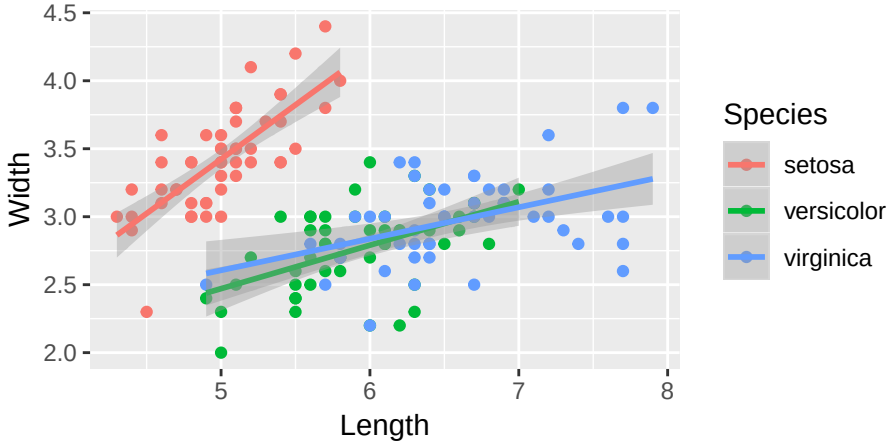
- Kleur van de punten variërend volgens soort

```
ggplot(irisSepal, aes(x = Length, y = Width, color = Species)) +
  geom_point()
```



- Kleur van de punten en lijnen variërend volgens soort

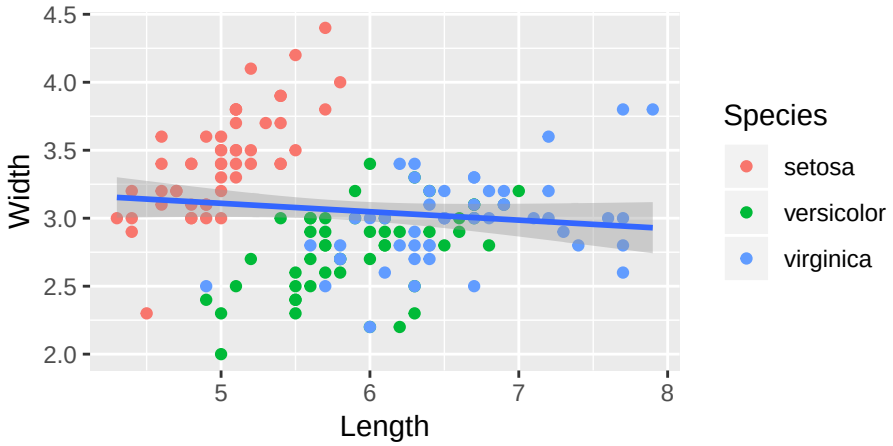
```
ggplot(irisSepal, aes(x = Length, y = Width, color = Species)) +
  geom_point() +
  geom_smooth(method = "lm")
```



  - Merk op dat we nu 3 verschillende smoothers krijgen, een per soort

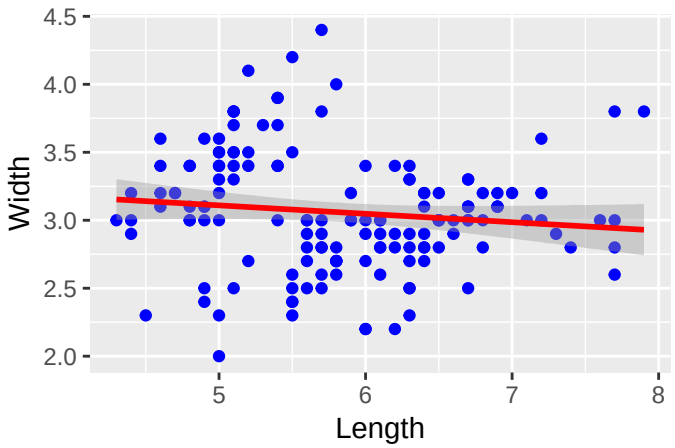
- Effect van plaatsing `color` aesthetic

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point(aes(color = Species)) +
  geom_smooth(method = "lm")
```



- Beperk het aantal kleuren (categorieën), anders zijn deze nog moeilijk te onderscheiden
- Vaste kleur, bvb blauwe punten en rode lijn

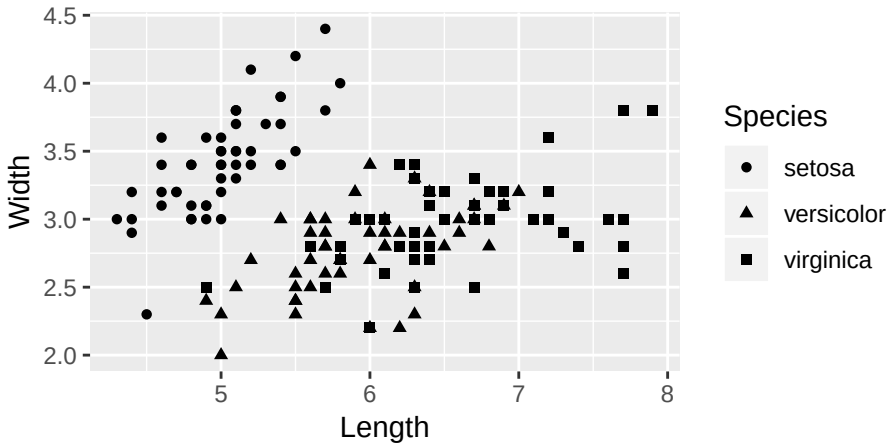
```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point(color = "blue") +
  geom_smooth(color = "red", method = "lm")
```



shape

- Symbool van de punten variërend volgens soort

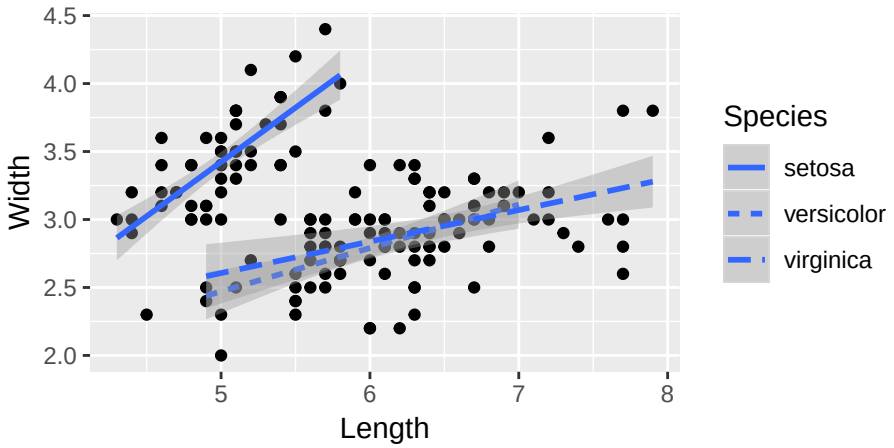
```
ggplot(irisSepal, aes(x = Length, y = Width, shape = Species)) +
  geom_point()
```



`linetype`

- Type lijn varieert volgens soort

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point() +
  geom_smooth(aes(linetype = Species), method = "lm")
```

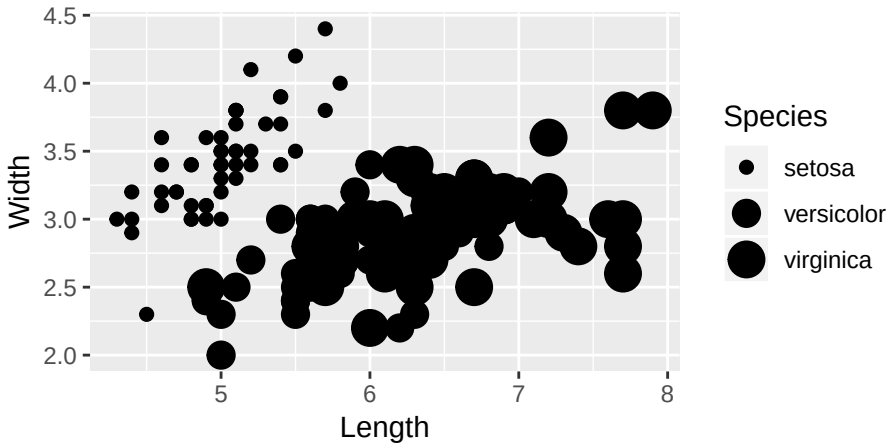


`size`

- Grootte van de punten variërend volgens soort

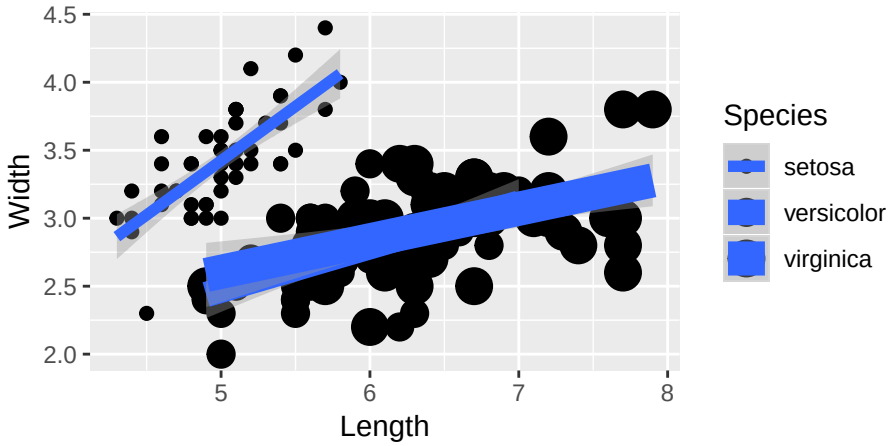
```
ggplot(irisSepal, aes(x = Length, y = Width, size = Species)) +
  geom_point()
```

```
## Warning: Using size for a discrete variable is not advised.
```



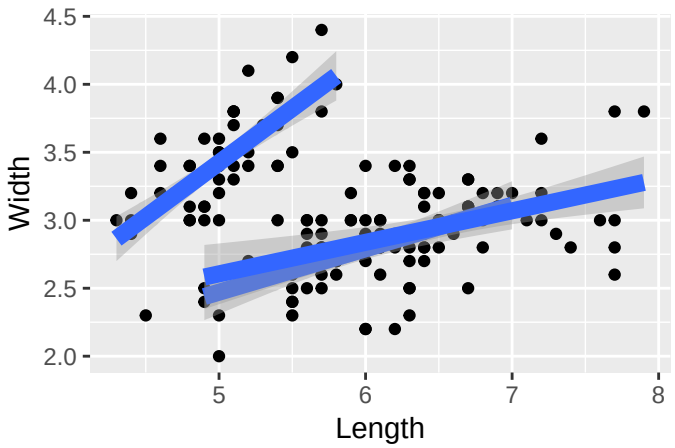
- **Waarschuwing** dat het gebruik van `size` niet aangewezen is voor een discrete variabele
- Grootte van de punten en dikte van de lijnen variërend volgens soort

```
ggplot(irisSepal, aes(x = Length, y = Width, size = Species)) +
  geom_point() +
  geom_smooth(method = "lm")
```



- Dikkere lijn, maar hetzelfde voor elke soort
  - Aangegeven door een getal, in millimeter

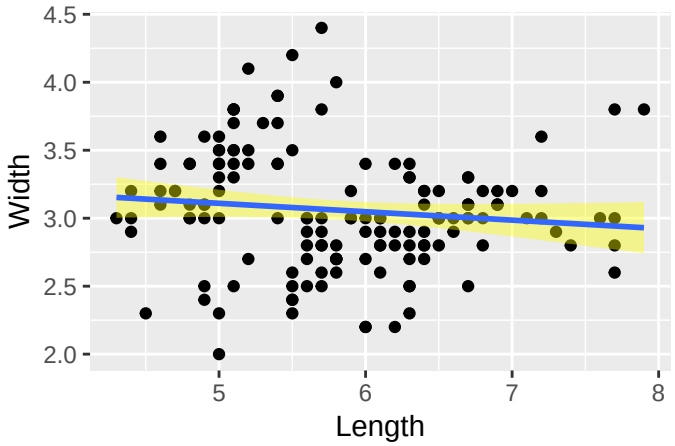
```
ggplot(irisSepal, aes(x = Length, y = Width, group = Species)) +
  geom_point() +
  geom_smooth(size = 3, method = "lm")
```



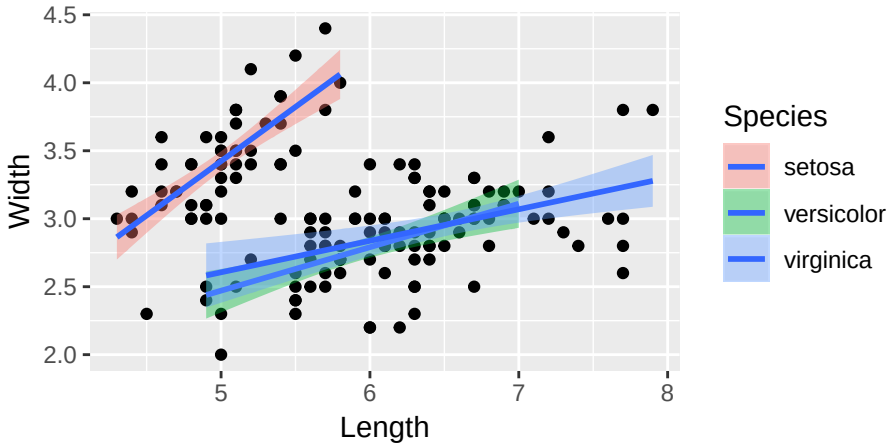
`fill`

- Hiermee kan je de kleur van het (interne) vlak bepalen
- Met `color` verander je enkel de kleur van de rand
- Punten en lijnen hebben enkel `color`, geen `fill`
- We kunnen dit wel gebruiken voor het betrouwbaarheidsinterval rond de smoother

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point() +
  geom_smooth(fill = "yellow", method = "lm")
```



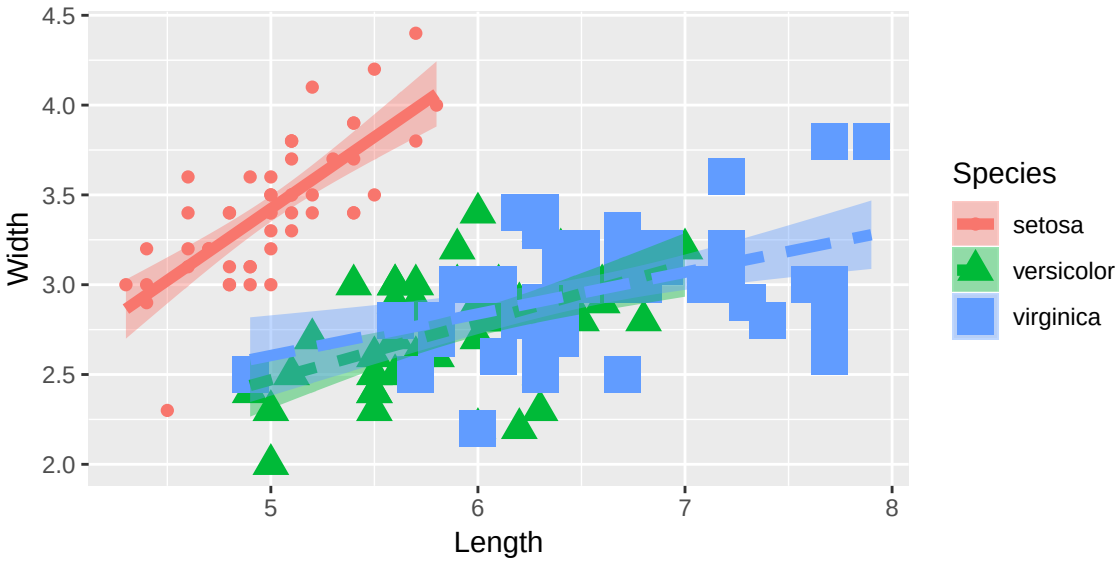

```
ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point() +
  geom_smooth(aes(fill = Species), method = "lm")
```



Combinatie van aesthetics

- Verschillende kleuren voor de punten en smoothers
- Verschillend symbool voor de punten
- Verschillende grootte voor de punten
- Verschillend lijntype
- Verschillende kleur betrouwbaarheidsinterval rond de smoothers
- Dikte van de smoothers vast op 2mm

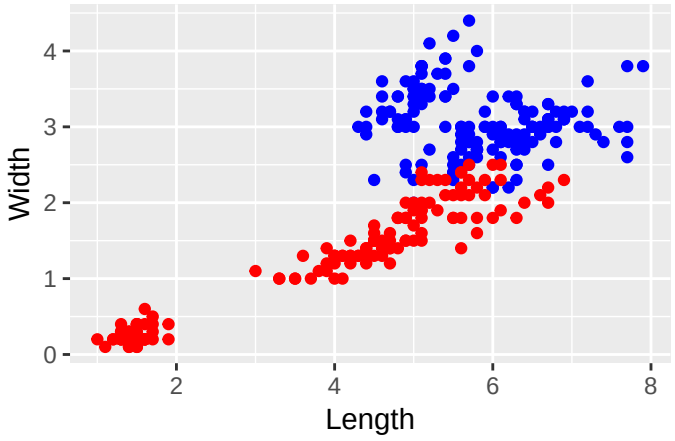
```
ggplot(irisSepal, aes(x = Length, y = Width, color = Species)) +
  geom_point(aes(shape = Species, size = Species)) +
  geom_smooth(aes(linetype = Species, fill = Species), size = 2, method = "lm")
```



Lagen met verschillende assen

- Het is mogelijk in elke laag een verschillende dataset en/of aesthetics te gebruiken
- Beperk de informatie in de `ggplot()` functie dan tot het gemeenschappelijke
- Specifieer de rest in de afzonderlijke `geom_xxx()`
  - **Opgelet**: hier moet je het data argument expliciet benoemen met `data =`
- "Stom" voorbeeld dat op een mooiere manier kan, slechts ter illustratie

```
ggplot() +
  geom_point(data = irisSepal, aes(x = Length, y = Width), color = "blue") +
  geom_point(data = irisPetal, aes(x = Length, y = Width), color = "red")
```




```
ggplot(mapping = aes(x = Length, y = Width)) +
  geom_point(data = irisSepal, color = "blue") +
  geom_point(data = irisPetal, color = "red")
```

Conclusie

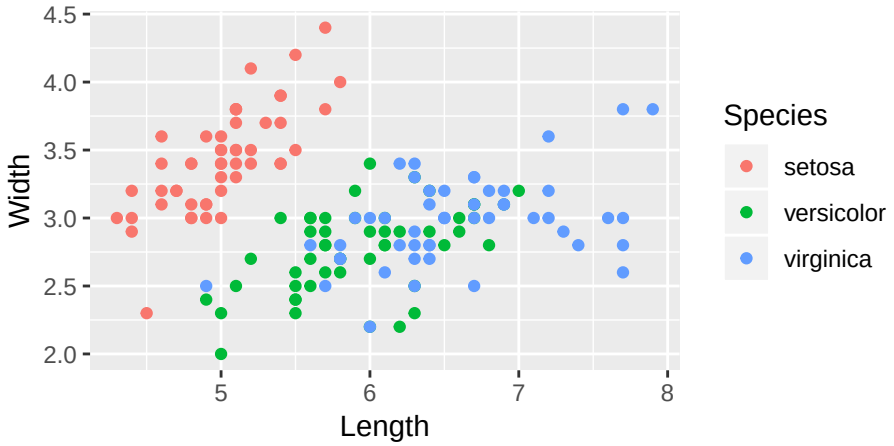
- Aesthetics in de `ggplot()` functie hebben een effect op alle lagen die toegevoegd worden
- Aesthetics in een `geom_xxx()` functie hebben enkel effect op die laag
- Aesthetics in `aes()` variëren volgens een variabele in de data
- Aesthetics buiten `aes()` zijn vast, en worden gespecifieerd met een getal of een naam
- Let ook op de aan- of aanwezigheid van een legende

  - Legende voor `color`, `shape`, `linetype`, `size`, `fill` die binnen `aes()` staan
  - Geen legende voor kleuren en groottes buiten `aes()`
- Zet in de `geom_xxx()` altijd eerst de `aes()`, en daarna pas andere opties
- Lagen worden over mekaar gelegd
  - Volgorde belangrijk voor de zichtbaarheid
  - Soms beter om de volgorde te wijzigen

Object stapsgewijs opbouwen

- Kan handig zijn om gemeenschappelijke delen niet telkens te moeten herhalen

```
p <- ggplot(irisSepal, aes(x = Length, y = Width)) +
  geom_point(aes(color = Species))
p
```



- Figuur kan dan stapsgewijs verder opgebouwd worden met geschikte lagen
  - Tonen

```
p + geom_smooth()
```

  - Bewaren in een ander object

```
p1 <- p + geom_smooth(method = "lm")
```

  - Bewaren in hetzelfde object (overschrijven)

```
p <- p + geom_smooth(aes(color = Species, fill = Species), method = "lm")
```

Facets

- Gebruik van verschillende kleuren kan verwarrend zijn, zeker als er veel punten en/of lijnen op 1 grafiek staan
- Duidelijker als gegevens over verschillende deelfiguren weergeven worden
- Opsplitsen volgens een of meerdere (categorische) variabelen
- Elke deelfiguur heeft dezelfde definitie
- `facet_wrap()`: vul het raster doorlopend
- `facet_grid()`: vul het raster zoals een tabel, waarbij in de rijen 1 variabele staat en in de kolommen een andere

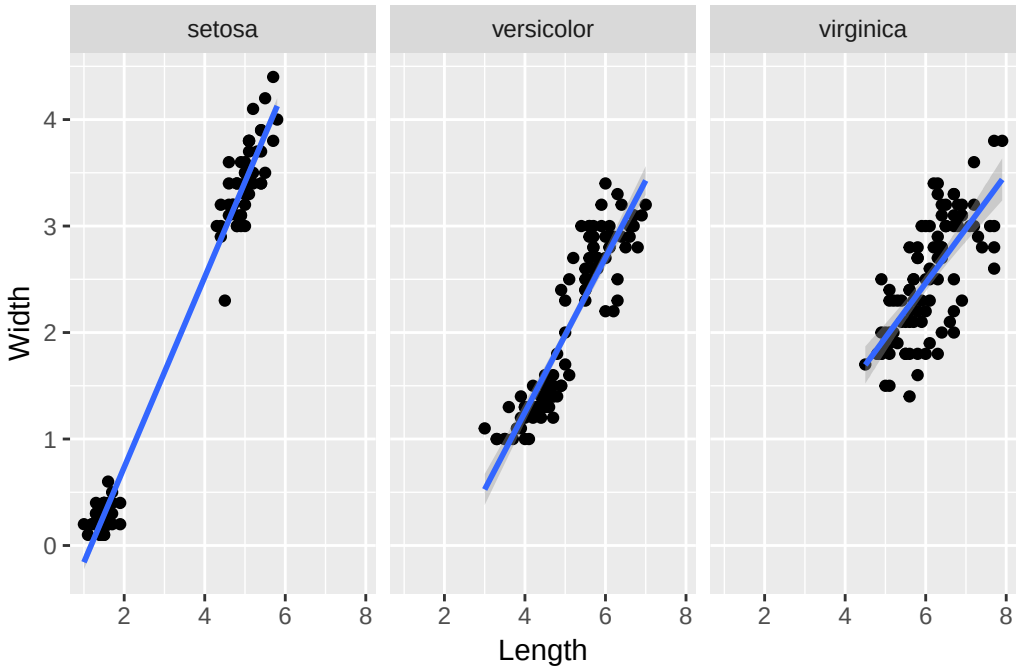
We definiëren onderstaande basisplot `p` die we vanaf nu telkens gaan aanvullen.

```
p <- ggplot(irisAll, aes(x = Length, y = Width)) +
  geom_point() +
  geom_smooth(method = "lm")
```

`facet_wrap()`

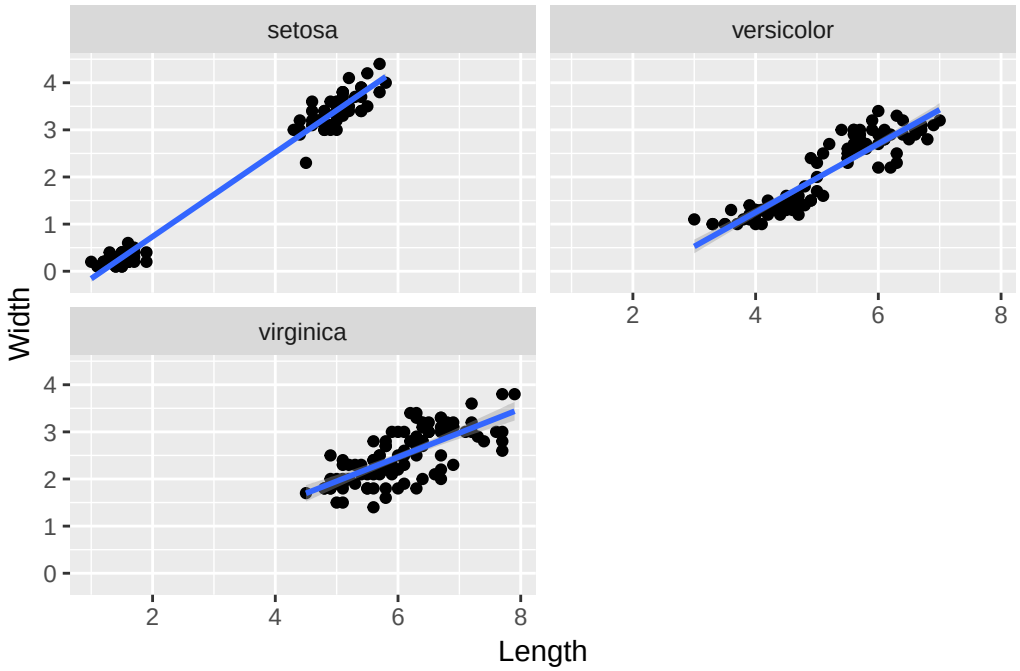
- Basisnotatie: `facet_wrap(~ NaamVariabele)`

```
p + facet_wrap(~ Species)
```



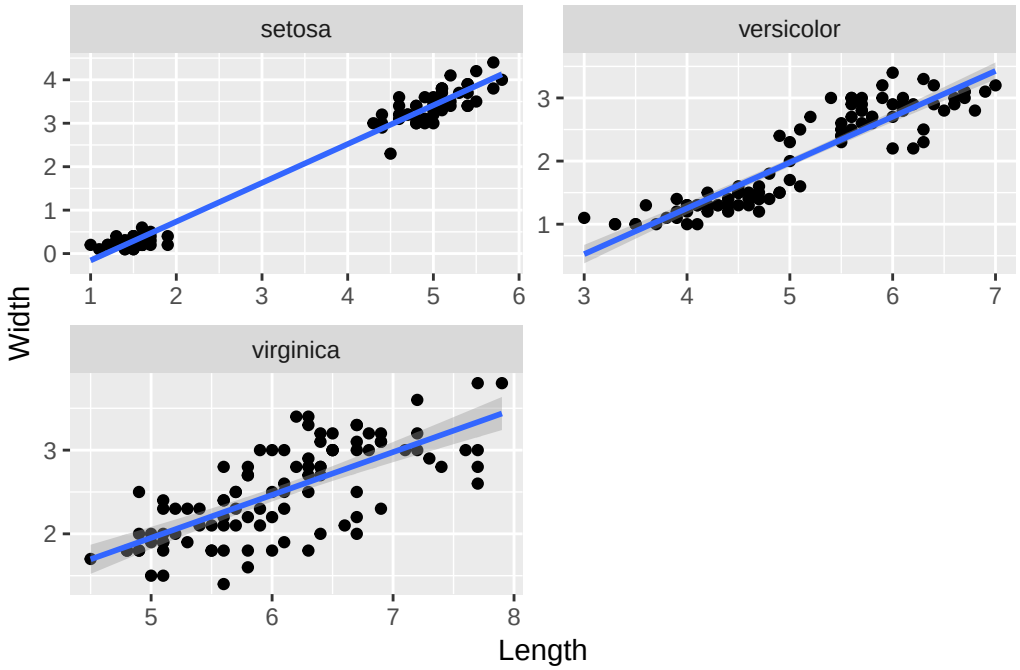
- `nrow` en `ncol`: gewenste aantal rijen en kolommen

```
p + facet_wrap(~ Species, nrow = 2)
```



- `scales`
  - default: elke subplot heeft zelfde x en y-as
  - `scales = "free_x"` elke subplot heeft aangepaste x-as
  - `scales = "free_y"` elke subplot heeft aangepaste y-as
  - `scales = "free"` elke subplot heeft aangepaste x en y-as

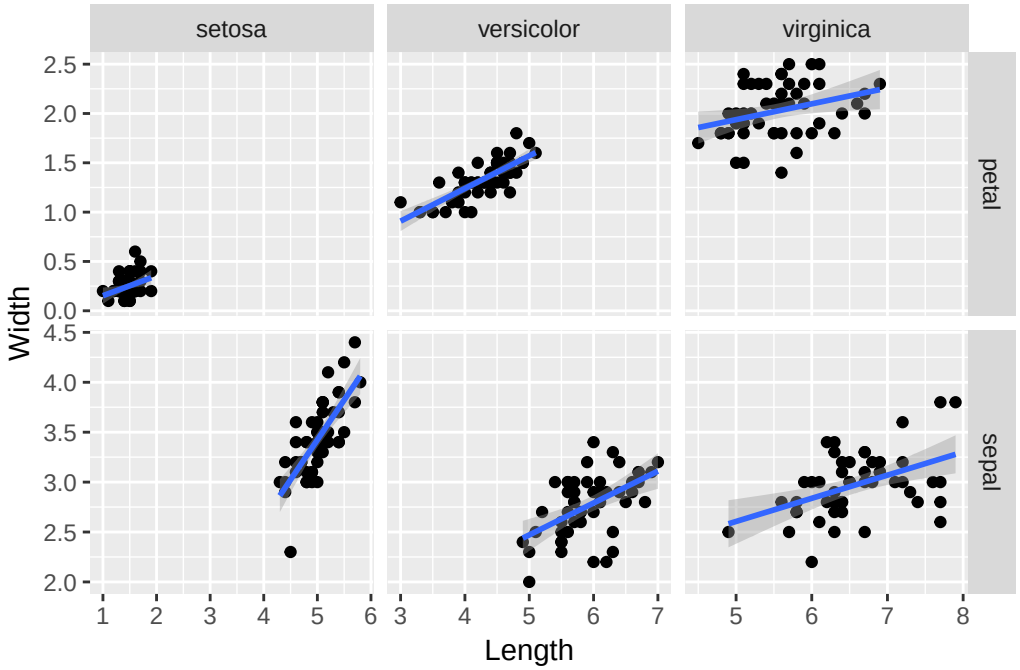
```r
p + facet_wrap(~ Species, nrow = 2, scales = "free")
```



`facet_grid()`

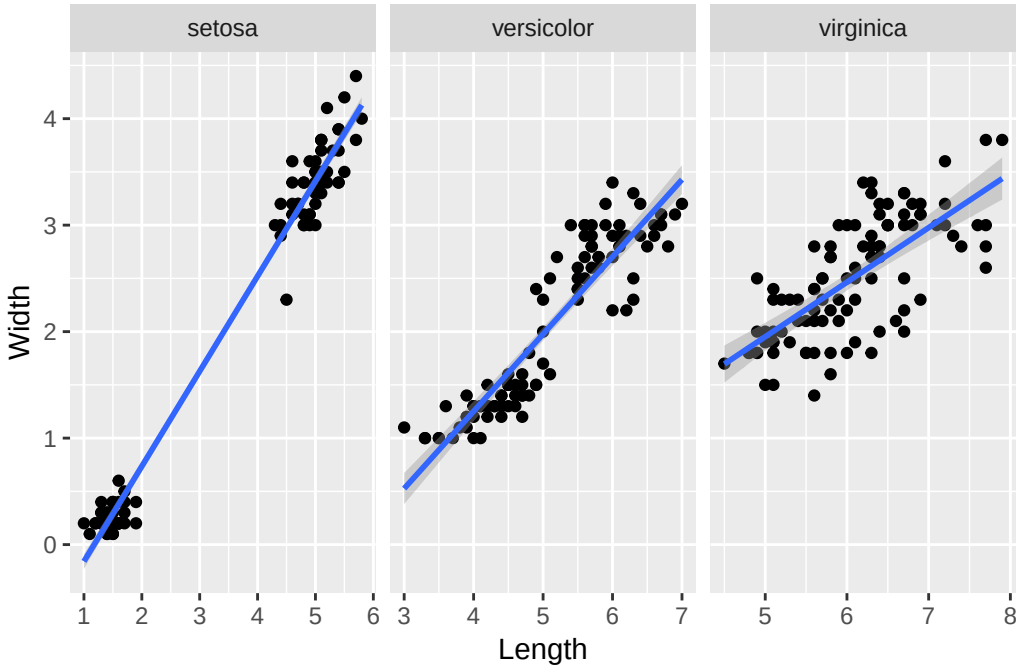
- Basisnotatie: `facet_grid(RijVariabele ~ KolomVariabele)`
- `scales`
  - default: elke subplot heeft zelfde x en y-as
  - `scales = "free_x"` elke **kolom** subplots heeft aangepaste x-as
  - `scales = "free_y"` elke **rij** subplots heeft aangepaste y-as
  - `scales = "free"` elke **kolom** en **rij** subplots heeft aangepaste x en y-as

```r
p + facet_grid(Leaf.Type ~ Species, scales = "free")
```



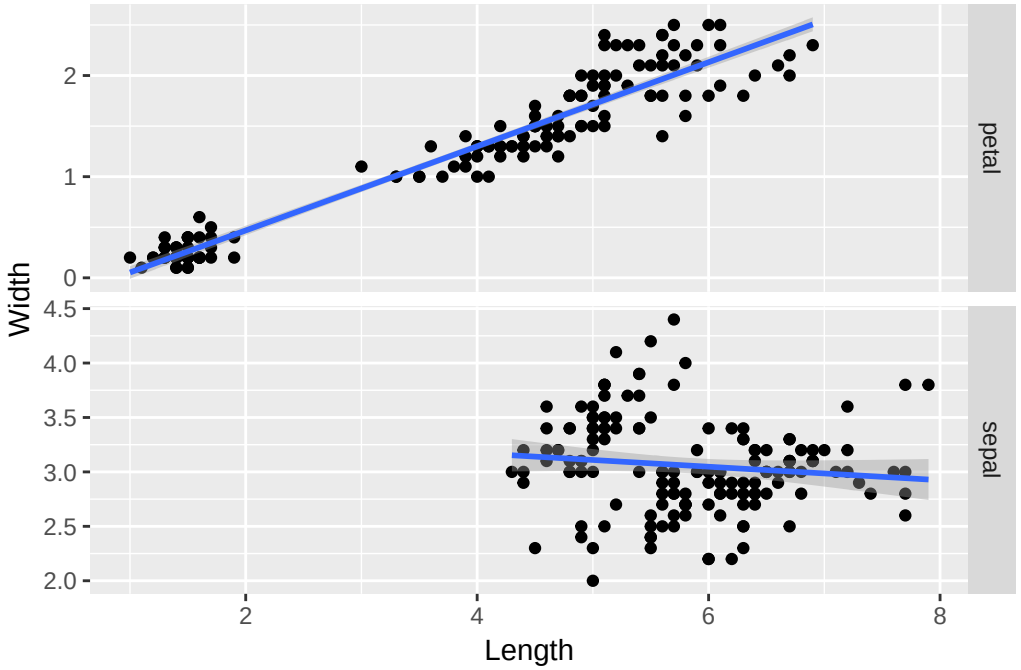
- Gebruik `facet_grid(. ~ A)` als je enkel in kolommen wilt splitsen
  - Dit geeft (bijna) hetzelfde resultaat als `facet_wrap` met `nrow = 1`

```
p + facet_grid(. ~ Species, scales = "free")
```



- Gebruik `facet_grid(A ~ .)` als je enkel in rijen wilt splitsen
  - Dit geeft (bijna) hetzelfde resultaat als `facet_wrap` met `ncol = 1`

```
p + facet_grid(Leaf.Type ~ ., scales = "free")
```



Nog enkele handige `geom_xxx()`

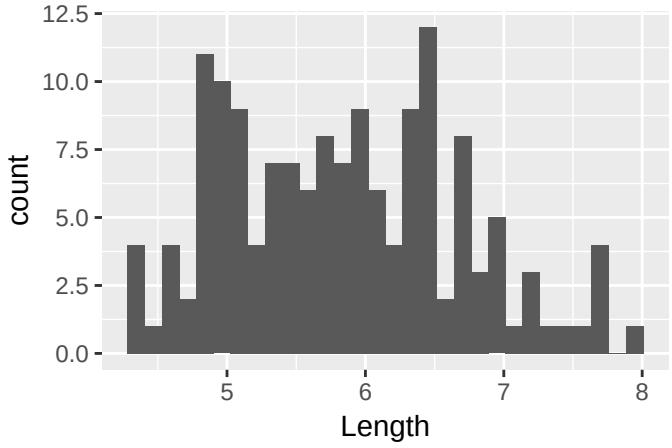
- `geom_histogram`, `geom_density`: histogram en gesmooth histogram
- `geom_bar`, `geom_col`: staafdiagram
  - `geom_bar`: hoogte proportioneel tot aantal observaties in die groep
  - `geom_col`: hoogte proportioneel tot waarde in data
- `geom_boxplot`: boxplot
- `geom_errorbar`, `geom_errorbarh`: foutenvlaggen verticaal en horizontaal

  - `geom_ribbon`: band met betrouwbaarheidsinterval
- `geom_text`: tekst labels per datapunt
- `geom_tile`, `geom_contour`: bovenaanzicht 3D oppervlak

geom_histogram

```
ggplot(irisSepal, aes(x = Length)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

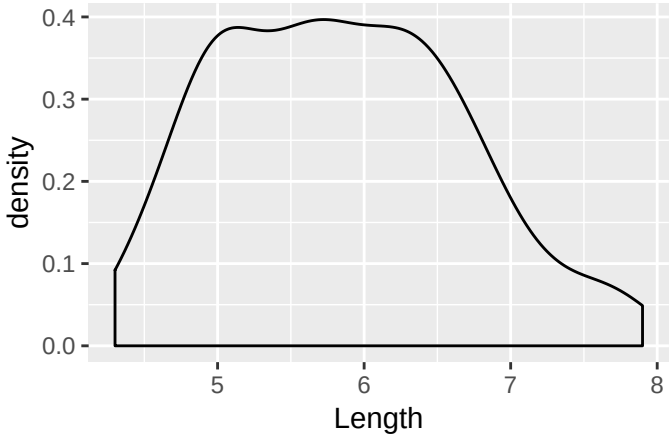


- Enkel een X-as nodig
- Waarden op de Y-as worden berekend
- Hiervoor wordt de X-as onderverdeeld in 30 `bins`, en aantallen geteld in deze intervallen
- Met `bins` of `binwidth` kan een betere keuze gemaakt worden voor deze intervallen (zie message)

```
geom_histogram(bins = 10)
geom_histogram(binwidth = 0.25)
```

geom_density

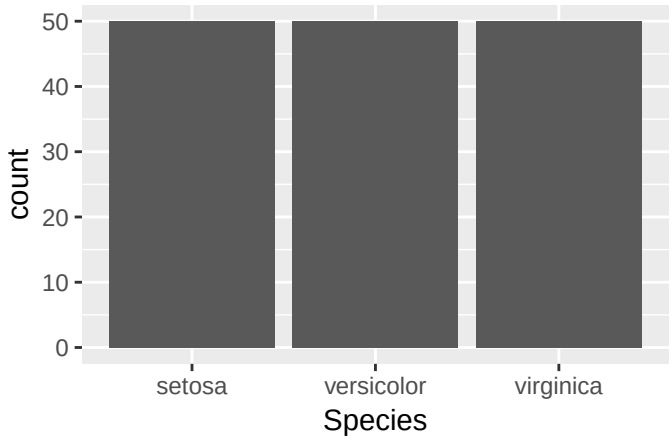
```
ggplot(irisSepal, aes(x = Length)) +
  geom_density()
```



- Enkel een X-as nodig
- Waarden op de Y-as worden berekend

geom_bar

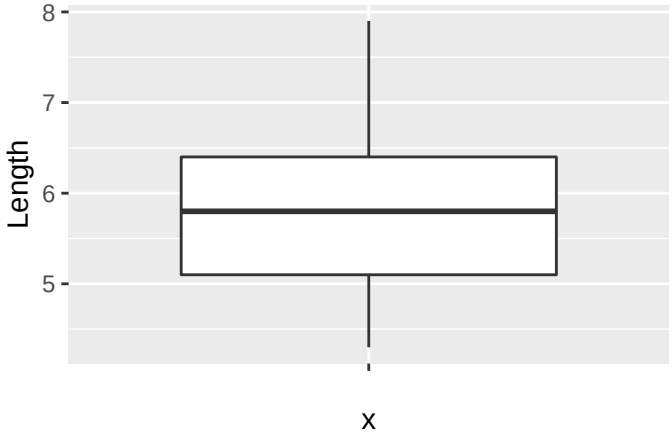
```
ggplot(irisSepal, aes(x = Species)) +
  geom_bar()
```



- Enkel een X-as nodig
- Waarden op de Y-as worden berekend

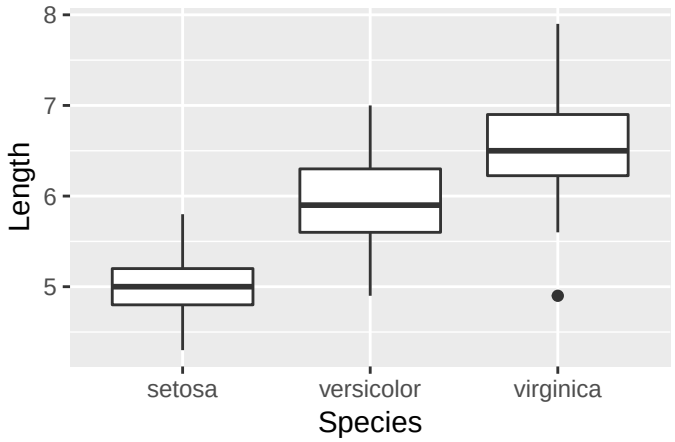
geom_boxplot

```
ggplot(irisSepal, aes(x = "", y = Length)) +
  geom_boxplot()
```



- Verplicht om X-as te definiëren
- Indien slechts 1 boxplot gewenst, dan is dit een lege character string
- Ofwel een categorische variabele om boxplot op te splitsen

```
ggplot(irisSepal, aes(x = Species, y = Length)) +
  geom_boxplot()
```

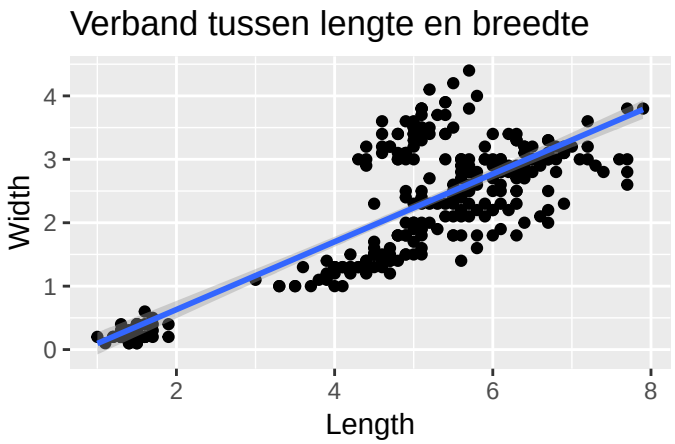


Titels aanpassen

Titel van de figuur

- Gebruik `ggtitle()` om een titel toe te voegen

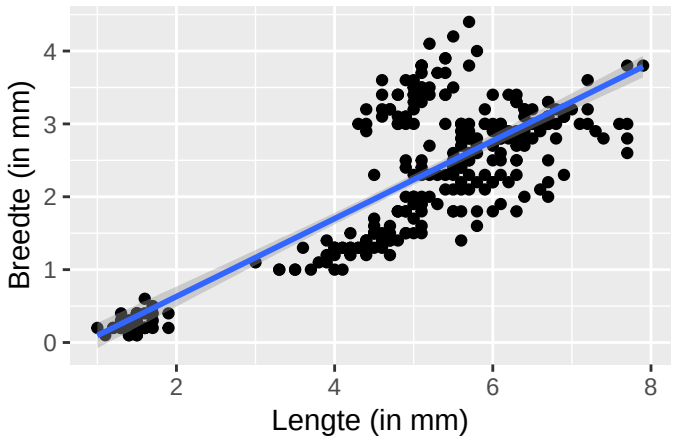
```
p + ggtitle("Verband tussen lengte en breedte")
```



Naam van de assen

- Standaard naam van de as = de naam van de variabele
  - Dus naam variabele wijzigen = naam as wijzigen
  - Mogelijkheden beperkt door eisen kolomnamen
- Alternatief: naam van de assen instellen met `xlab()` en `ylab()`

```
p + xlab("Lengte (in mm)") + ylab("Breedte (in mm)")
```



Gecombineerd in 1 functie

```
p + labs(title = "Verband tussen lengte en breedte",
         x = "Lengte (in mm)",
         y = "Breedte (in mm)")
```

Plot bewaren

- In het `plot` venster met de `Export` knop
  - Niet aan te raden, niet reproduceerbaar en je maakt fouten !!
- Ken de plot toe aan een object

```
p <- ggplot(data, aes()) + geom_xxx()
```

- Bewaar dit object met een bepaalde naam

```
ggsave(filenaam, p)
```

- Verschillende formaten door middel van het argument `device =`
  - jpeg, tiff, png, bmp, wmf, ps, pdf, ...
  - Mogelijk om al een extensie toe te voegen aan de filenaam, dan is het overbodig om een formaat mee te geven
- Extra argumenten
  - Afmetingen: `width` en `height`
  - Eenheid van de afmetingen: `units` ("in", "cm", "mm")
  - Resolutie: `dpi`
  - Plaats waar de plot bewaard moet worden, indien anders dan de working directory: `path =`

```
ggsave("Figuur_iris.png", p, path = "output/",
       width = 9, height = 6, dpi = 100)
ggsave("Figuur_iris_cm.png", p, path = "output/",
       width = 9, height = 6, units = "cm", dpi = 100)
ggsave("Figuur_iris_flou.png", p, path = "output/",
       width = 15, height = 10, dpi = 10)
```

More to learn

- Google !!!
- Cookbook for R
- ggplot2 QuickRef
- R for data science
  - Boek van Hadley Wickham en Garrett Grolemund
  - Hardcopy beschikbaar op INBO
  - Digitale versie
- Datacamp
  - (gedeeltelijk) gratis lessen (video tutorials en oefeningen)
  - Account voor 72h voor volledige toegang, daarna betalende licentie (~ €25/maand)
  - Data visualization with ggplot2
  - Grammar of Graphics
- Data Carpentry
  - Visualizing Data
- Stat 545
  - All the graph things
- Cheat Sheets
  - In RStudio onder `Help` menu
  - Online

Referenties

- R for data science
- Slides van Thierry uit 2015