# Hoorcollege 3

# Introductie tot ggplot2
# Misleiden met visualisaties


Informatie-uitwisseling: informatievisualisatie

**Jeroen Ooge**

Universiteit Utrecht

# Overzicht hoorcolleges

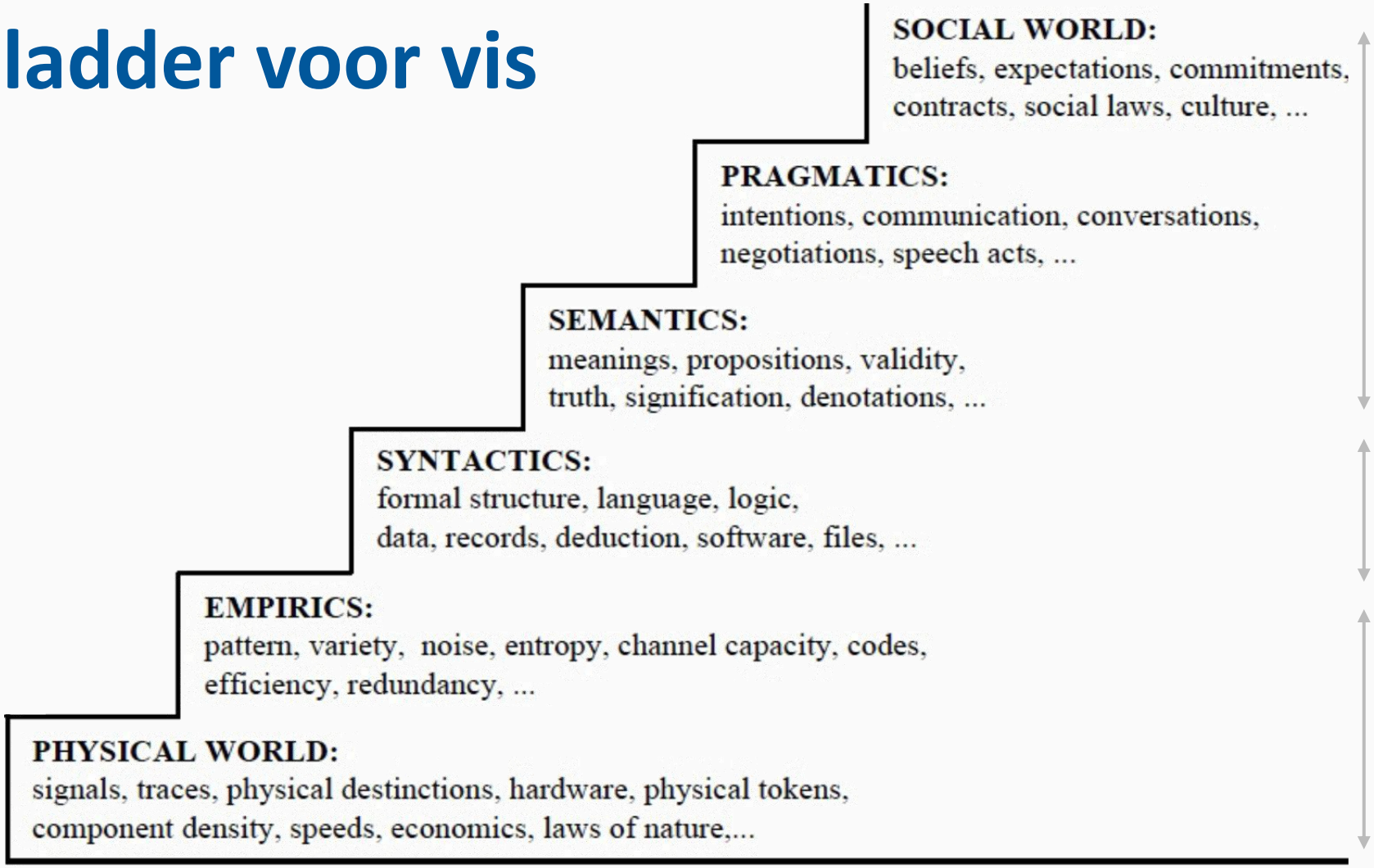

**Hoorcollege 1**
Algemene principes
en vis-types

**Hoorcollege 2**
Theoretische
fundamenten van vis


**Hoorcollege 3**
Inleiding ggplot2 +
hoe vis kan misleiden

**Hoorcollege 4**
Vis in rapporten +
interactieve vis

# De semiotische ladder voor vis




communicatie met vis

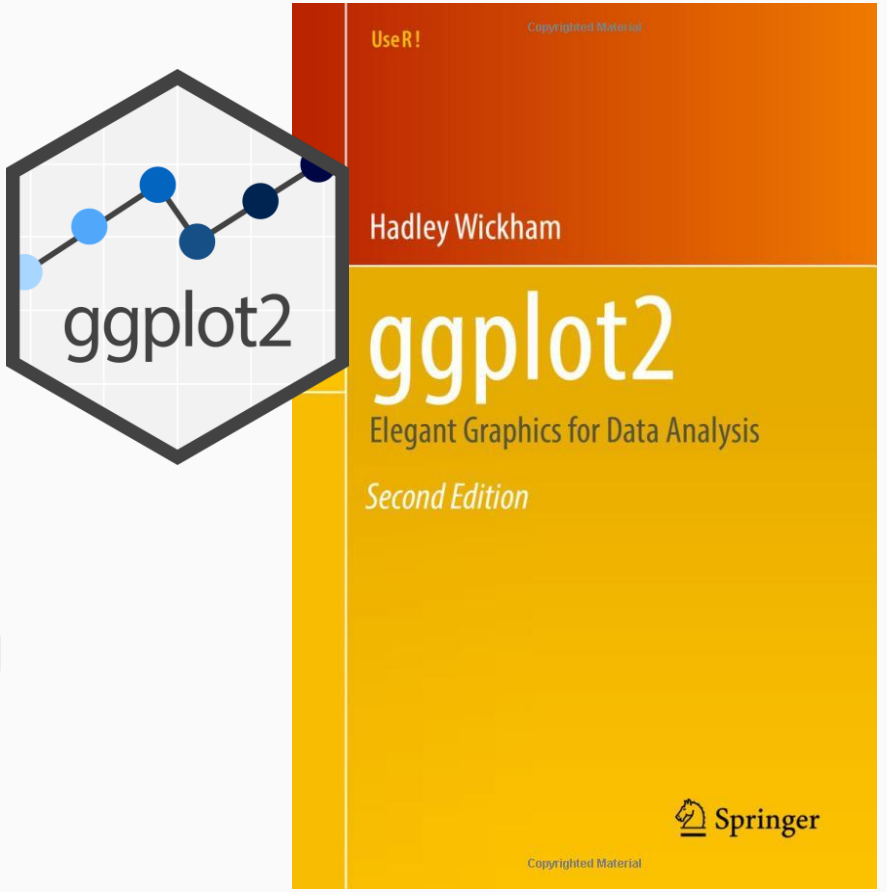
vertaling naar vis

data verzamelen

Hoorcollege 3

# Introductie tot ggplot2
# Misleiden met visualisaties

1. Introductie tot ggplot2
2. Misleiden met visualisaties
   a) Slecht ontwerp
   b) Dubieuze data
   c) Te weinig data
   d) Verborgen of onduidelijke onzekerheid
   e) Suggestieve patronen



# Enkele bekende vis-tools


meer interactie,
meer flexibiliteit,
stijlere leercurve

# Datastructuren in R

Tibbles (voorheen dataframes) zijn speciale tabellen

```
mpg
#> # A tibble: 234 × 11
#>   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
#>   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
#> 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa…
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
#> 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
#> 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
#> 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
#> # ℹ 228 more rows
```

kolomnamen (attributen)

kolomtypes (character, integer, double, list, logical…)

rijen met data (items)

**Tip**: hervorm de data die je wilt visualiseren tot 1 tibble

Er is veel te leren over *data wrangling* in R, bv. de package dplyr




Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org
https://dplyr.tidyverse.org

# Data visualization with ggplot2 : : CHEATSHEET

## Basics

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.


To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.


Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>)),
  stat = <STAT>, position = <POSITION>) +                  Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

**ggplot(**data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

**last_plot()** Returns the last plot.

**ggsave(**"plot.png", width = 5, height = 5**)** Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

## Aes

Common aesthetic values.

**color** and **fill** - string ("red", "#RRGGBB")

**linetype** - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

**size** - integer (in mm for size of points and text)

**linewidth** - integer (in mm for widths of lines)

**shape** - integer/shape name or a single character ("a")


posit®

## Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

### GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

**a + geom_blank()** and **a + expand_limits()**
Ensure limits include values across all plots.

**b + geom_curve(**aes(yend = lat + 1, xend = long + 1), curvature = 1**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

**a + geom_path(**lineend = "butt", linejoin = "round", linemitre = 1**)**
x, y, alpha, color, group, linetype, size

**a + geom_polygon(**aes(alpha = 50)**)** - x, y, alpha, color, fill, group, subgroup, linetype, size

**b + geom_rect(**aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

**a + geom_ribbon(**aes(ymin = unemploy - 900, ymax = unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

### LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

**b + geom_abline(**aes(intercept = 0, slope = 1)**)**
**b + geom_hline(**aes(yintercept = lat)**)**
**b + geom_vline(**aes(xintercept = long)**)**

**b + geom_segment(**aes(yend = lat + 1, xend = long + 1)**)**
**b + geom_spoke(**aes(angle = 1:1155, radius = 1)**)**

### ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

**c + geom_area(**stat = "bin"**)**
x, y, alpha, color, fill, linetype, size

**c + geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

**c + geom_dotplot()**
x, y, alpha, color, fill

**c + geom_freqpoly()**
x, y, alpha, color, group, linetype, size

**c + geom_histogram(**binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight

**c2 + geom_qq(**aes(sample = hwy)**)**
x, y, alpha, color, fill, linetype, size, weight

### discrete

d <- ggplot(mpg, aes(fl))

**d + geom_bar()**
x, alpha, color, fill, linetype, size, weight

### TWO VARIABLES

#### both continuous

e <- ggplot(mpg, aes(cty, hwy))

**e + geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

**e + geom_point()**
x, y, alpha, color, fill, shape, size, stroke

**e + geom_quantile()**
x, y, alpha, color, group, linetype, size, weight

**e + geom_rug(**sides = "bl"**)**
x, y, alpha, color, linetype, size

**e + geom_smooth(**method = lm**)**
x, y, alpha, color, fill, group, linetype, size, weight

**e + geom_text(**aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

#### one discrete, one continuous

f <- ggplot(mpg, aes(class, hwy))

**f + geom_col()**
x, y, alpha, color, fill, group, linetype, size

**f + geom_boxplot()**
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

**f + geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill, group

**f + geom_violin(**scale = "area"**)**
x, y, alpha, color, fill, group, linetype, size, weight

#### both discrete

g <- ggplot(diamonds, aes(cut, color))

**g + geom_count()**
x, y, alpha, color, fill, shape, size, stroke

**e + geom_jitter(**height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

### THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

**l + geom_contour(**aes(z = z)**)**
x, y, z, alpha, color, group, linetype, size, weight

**l + geom_contour_filled(**aes(fill = z)**)**
x, y, alpha, color, fill, group, linetype, size, subgroup

**l + geom_raster(**aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE**)**
x, y, alpha, fill

**l + geom_tile(**aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size, width

#### continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

**h + geom_bin2d(**binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

**h + geom_density_2d()**
x, y, alpha, color, group, linetype, size

**h + geom_hex()**
x, y, alpha, color, fill, size

#### continuous function

i <- ggplot(economics, aes(date, unemploy))

**i + geom_area()**
x, y, alpha, color, fill, linetype, size

**i + geom_line()**
x, y, alpha, color, group, linetype, size

**i + geom_step(**direction = "hv"**)**
x, y, alpha, color, group, linetype, size

#### visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

**j + geom_crossbar(**fatten = 2**)** - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

**j + geom_errorbar()** - x, ymax, ymin, alpha, color, group, linetype, size, width
Also **geom_errorbarh()**.

**j + geom_linerange()**
x, ymin, ymax, alpha, color, group, linetype, size

**j + geom_pointrange()** - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

#### maps

Draw the appropriate geometric object depending on the simple features present in the data. aes() arguments: map_id, alpha, color, fill, linetype, linewidth.

nc <- **sf::st_read(**system.file("shape/nc.shp", package = "sf"**)**)

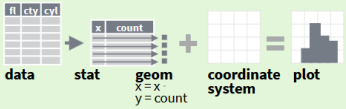
ggplot(nc) +
**geom_sf(**aes(fill = AREA)**)**



https://rstudio.github.io/cheatsheets/data-visualization.pdf

# Stats An alternative way to build a layer.

A stat builds new variables to plot (e.g., count, prop).




Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function).
Use **after_stat(name)** syntax to map the stat variable **name** to an aesthetic.

geom to use | stat function | geommappings

**i + stat_density_2d(**aes(fill = after_stat(level)), geom = "polygon"**)**

variable created by stat

**c + stat_bin(**binwidth = 1, boundary = 10**)**
**x, y** | count, ncount, density, ndensity

**c + stat_count(**width = 1**) x, y** | count, prop

**c + stat_density(**adjust = 1, kernel = "gaussian"**)**
**x, y** | count, density, scaled

**e + stat_bin_2d(**bins = 30, drop = T**)**
**x, y, fill** | count, density

**e + stat_bin_hex(**bins = 30**) x, y, fill** | count, density

**e + stat_density_2d(**contour = TRUE, n = 100**)**
**x, y, color, size** | level

**e + stat_ellipse(**level = 0.95, segments = 51, type = "t"**)**

**l + stat_contour(**aes(z = z)**) x, y, z, order** | level

**l + stat_summary_hex(**aes(z = z), bins = 30, fun = max**)**
**x, y, z, fill** | value

**l + stat_summary_2d(**aes(z = z), bins = 30, fun = mean**)**
**x, y, z, fill** | value

**f + stat_boxplot(**coef = 1.5**)**
**x, y** | lower, middle, upper, width , ymin, ymax

**f + stat_ydensity(**kernel = "gaussian", scale = "area"**) x, y** | density, scaled, count, n, violinwidth, width

**e + stat_ecdf(**n = 40**) x, y** | x, y

**e + stat_quantile(**quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq"**) x, y** | quantile

**e + stat_smooth(**method = "lm", formula = y ~ x, se = T, level = 0.95**) x, y** | se, x, y, ymin, ymax

**ggplot() + xlim(**-5, 5**) + stat_function(**fun = dnorm, n = 20, geom = "point"**) x** | x, y

**ggplot() + stat_qq(**aes(sample = 1:100)**)**
**x, y, sample** | sample, theoretical

**e + stat_sum() x, y, size** | n, prop

**e + stat_summary(**fun.data = "mean_cl_boot"**)**

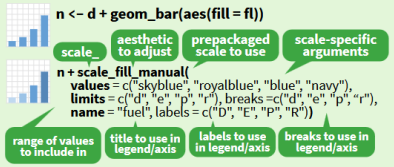
**h + stat_summary_bin(**fun = "mean", geom = "bar"**)**

**e + stat_identity()**

**e + stat_unique()**

# Scales Override defaults with **scales** package.

**Scales** map data values to the visual values of an aesthetic. To change a mapping, add a new scale.



**n <- d + geom_bar(aes(fill = fl))**

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

**n + scale_fill_manual(**
**values** = c("skyblue", "royalblue", "blue", "navy"),
**limits** = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
**name** = "fuel", labels = c("D", "E", "P", "R"))

range of values to include in | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

## GENERAL PURPOSE SCALES

Use with most aesthetics

**scale_*_continuous()** - Map cont' values to visual ones.
**scale_*_discrete()** - Map discrete values to visual ones.
**scale_*_binned()** - Map continuous values to discrete bins.
**scale_*_identity()** - Use data values as visual ones.
**scale_*_manual(**values = c()**)** - Map discrete values to manually chosen visual ones.
**scale_*_date(**date_labels = "%m/%d"), date_breaks = "2 weeks"**)** - Treat data values as dates.
**scale_*_datetime()** - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.

## X & Y LOCATION SCALES

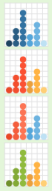
Use with x or y aesthetics (x shown here)

**scale_x_log10()** - Plot x on log10 scale.
**scale_x_reverse()** - Reverse the direction of the x axis.
**scale_x_sqrt()** - Plot x on square root scale.

## COLOR AND FILL SCALES (DISCRETE)

**n + scale_fill_brewer(**palette = "Blues"**)**
For palette choices:
RColorBrewer::display.brewer.all()
**n + scale_fill_grey(**start = 0.2, end = 0.8, na.value = "red"**)**

## COLOR AND FILL SCALES (CONTINUOUS)



o <- c + geom_dotplot(aes(fill = x))
**o + scale_fill_distiller(**palette = "Blues"**)**
**o + scale_fill_gradient(**low="red", high="yellow"**)**
**o + scale_fill_gradient2(**low = "red", high = "blue", mid = "white", midpoint = 25**)**
**o + scale_fill_gradientn(**colors = topo.colors(6)**)**
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

## SHAPE AND SIZE SCALES

p <- e + geom_point(aes(shape = fl, size = cyl))

**p + scale_shape() + scale_size()**
**p + scale_shape_manual(**values = c(3:7)**)**
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
**p + scale_radius(**range = c(1,6)**)**
**p + scale_size_area(**max_size = 6**)**

# Coordinate Systems

r <- d + geom_bar()

**r + coord_cartesian(**xlim = c(0, 5)**)** - xlim, ylim
The default cartesian coordinate system.

**r + coord_fixed(**ratio = 1/2**)**
ratio, xlim, ylim - Cartesian coordinates with fixed aspect ratio between x and y units.

**r + coord_flip()**
Flip cartesian coordinates by switching x and y aesthetic mappings.

**r + coord_polar(**theta = "x", direction=1**)**
theta, start, direction - Polar coordinates.

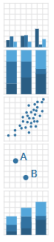
**r + coord_trans(**y = "sqrt"**)** - x, y, xlim, ylim
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

**π + coord_sf()** - xlim, ylim, crs. Ensures all layers use a common Coordinate Reference System.

# Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))



**s + geom_bar(position = "dodge")**
Arrange elements side by side.
**s + geom_bar(position = "fill")**
Stack elements on top of one another, normalize height.
**e + geom_point(position = "jitter")**
Add random noise to X and Y position of each element to avoid overplotting.
**e + geom_label(position = "nudge")**
Nudge labels away from points.
**s + geom_bar(position = "stack")**
Stack elements on top of one another.

Each position adjustment can be recast as a function with manual **width** and **height** arguments:
s + geom_bar(position = position_dodge(width = 1))

# Themes

**r + theme_bw()**
White background with grid lines.
**r + theme_gray()**
Grey background (default theme).
**r + theme_dark()**
Dark for contrast.
**r + theme_classic()**
**r + theme_light()**
**r + theme_linedraw()**
**r + theme_minimal()**
Minimal theme.
**r + theme_void()**
Empty theme.

**r + theme()** Customize aspects of the theme such as axis, legend, panel, and facet properties.
r + labs(title = "Title") + theme(plot.title.position = "plot")
r + theme(panel.background = element_rect(fill = "blue"))

# Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()



**t + facet_grid(. ~ fl)**
Facet into columns based on fl.

**t + facet_grid(year ~ .)**
Facet into rows based on year.

**t + facet_grid(year ~ fl)**
Facet into both rows and columns.

**t + facet_wrap(~ fl)**
Wrap facets into a rectangular layout.

Set **scales** to let axis limits vary across facets.

**t + facet_grid(drv ~ fl, scales = "free")**
x and y axis limits adjust to individual facets:
**"free_x"** - x axis limits adjust
**"free_y"** - y axis limits adjust

Set **labeller** to adjust facet label:

**t + facet_grid(. ~ fl, labeller = label_both)**

| fl: c | fl: d | fl: e | fl: p | fl: r |
|---|---|---|---|---|

**t + facet_grid(fl ~ ., labeller = label_bquote(**alpha ^ .(fl)**))**

$\alpha^{c}$ $\alpha^{d}$ $\alpha^{e}$ $\alpha^{p}$ $\alpha^{r}$

# Labels and Legends

Use **labs()** to label the elements of your plot.

**t + labs(x** = "New x axis label", **y** = "New y axis label",
**title** ="Add a title above the plot",
**subtitle** = "Add a subtitle below title",
**caption** = "Add a caption below plot",
**alt** = "Add alt text to the plot",
<AES> = "New <AES> legend title"**)**

**t + annotate(**geom = "text", x = 8, y = 9, label = "A"**)**
Places a geom with manually selected aesthetics.

**p + guides(**x = guide_axis(n.dodge = 2)**)** Avoid crowded or overlapping labels with guide_axis(n.dodge or angle).

**n + guides(**fill = "none"**)** Set legend type for each aesthetic: colorbar, legend, or none (no legend).

**n + theme(**legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or "right".

**n + scale_fill_discrete(**name = "Title", labels = c("A", "B", "C", "D", "E"**)**
Set legend title and labels with a scale function.

# Zooming

**Without clipping** (preferred):
**t + coord_cartesian(**xlim = c(0, 100), ylim = c(10, 20)**)**

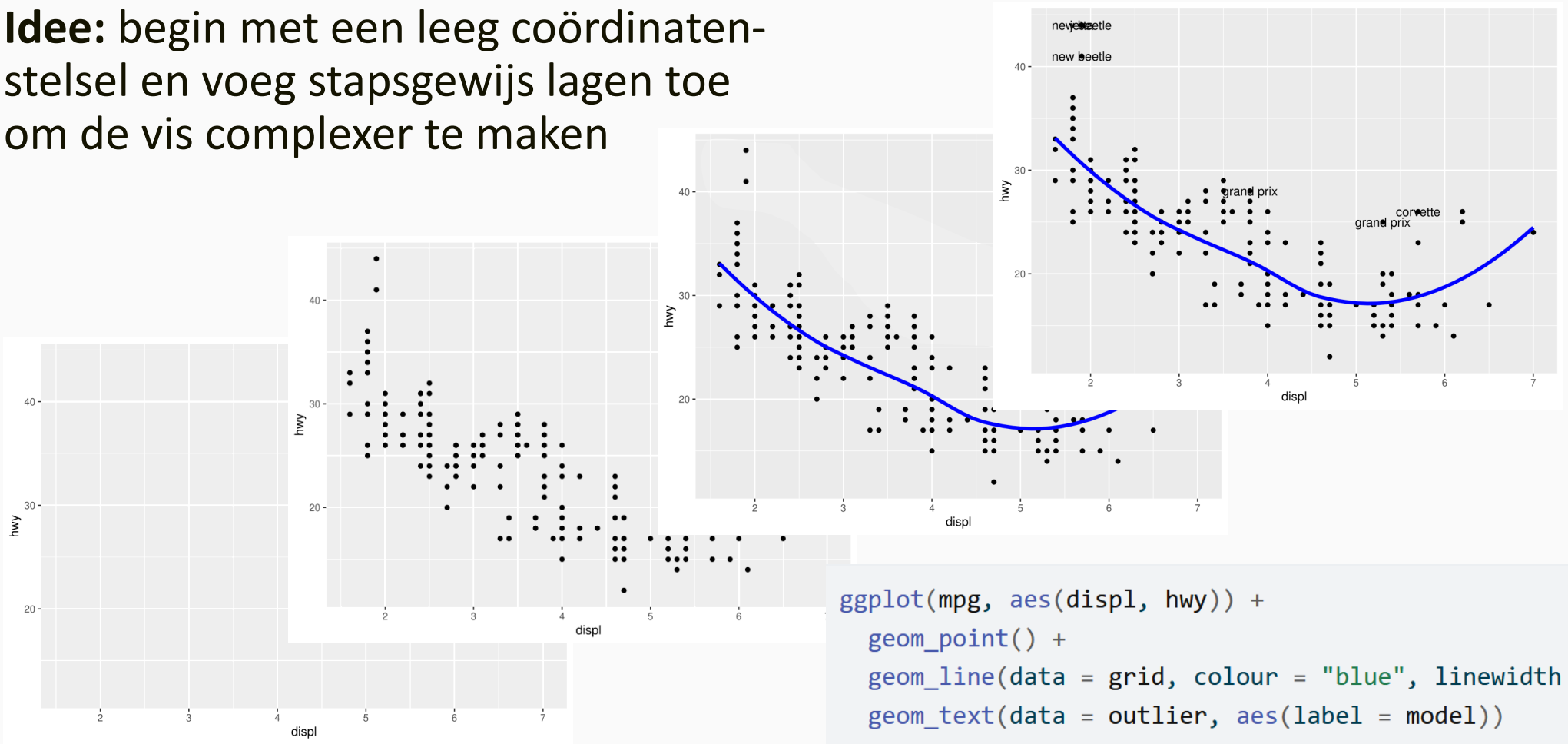
**With clipping** (removes unseen data points):
**t + xlim(**0, 100**) + ylim(**10, 20**)**
**t + scale_x_continuous(**limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**



https://rstudio.github.io/cheatsheets/data-visualization.pdf

# Basics: laag per laag een vis maken

**Idee:** begin met een leeg coördinaten-stelsel en voeg stapsgewijs lagen toe om de vis complexer te maken

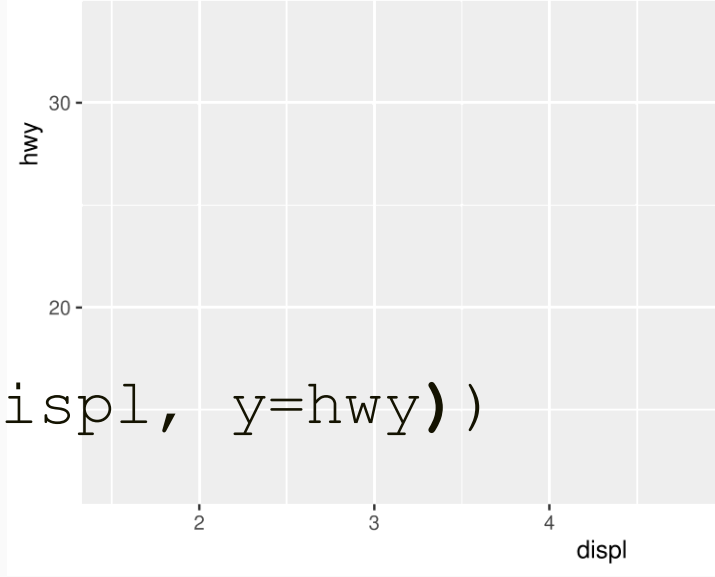



```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_line(data = grid, colour = "blue", linewidth = 1.5) +
  geom_text(data = outlier, aes(label = model))
```



Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/layers

# Basics: laag per laag een vis maken

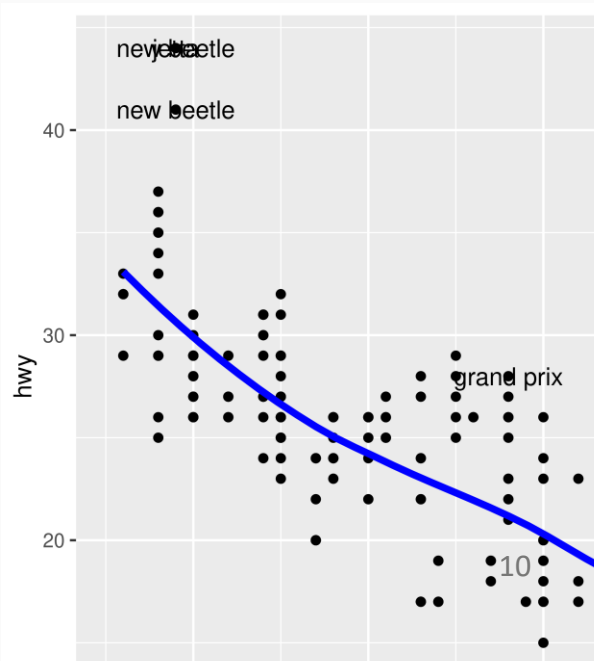



Voluit staat hier: `ggplot(data=mpg, mapping=aes(x=displ, y=hwy))`
De “aesthetics” mappen de data op visuele elementen

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_line(data = grid, colour = "blue", linewidth = 1.5) +
  geom_text(data = outlier, aes(label = model))
```

Als de data of mapping ontbreken, dan worden die van de eerste laag gebruikt

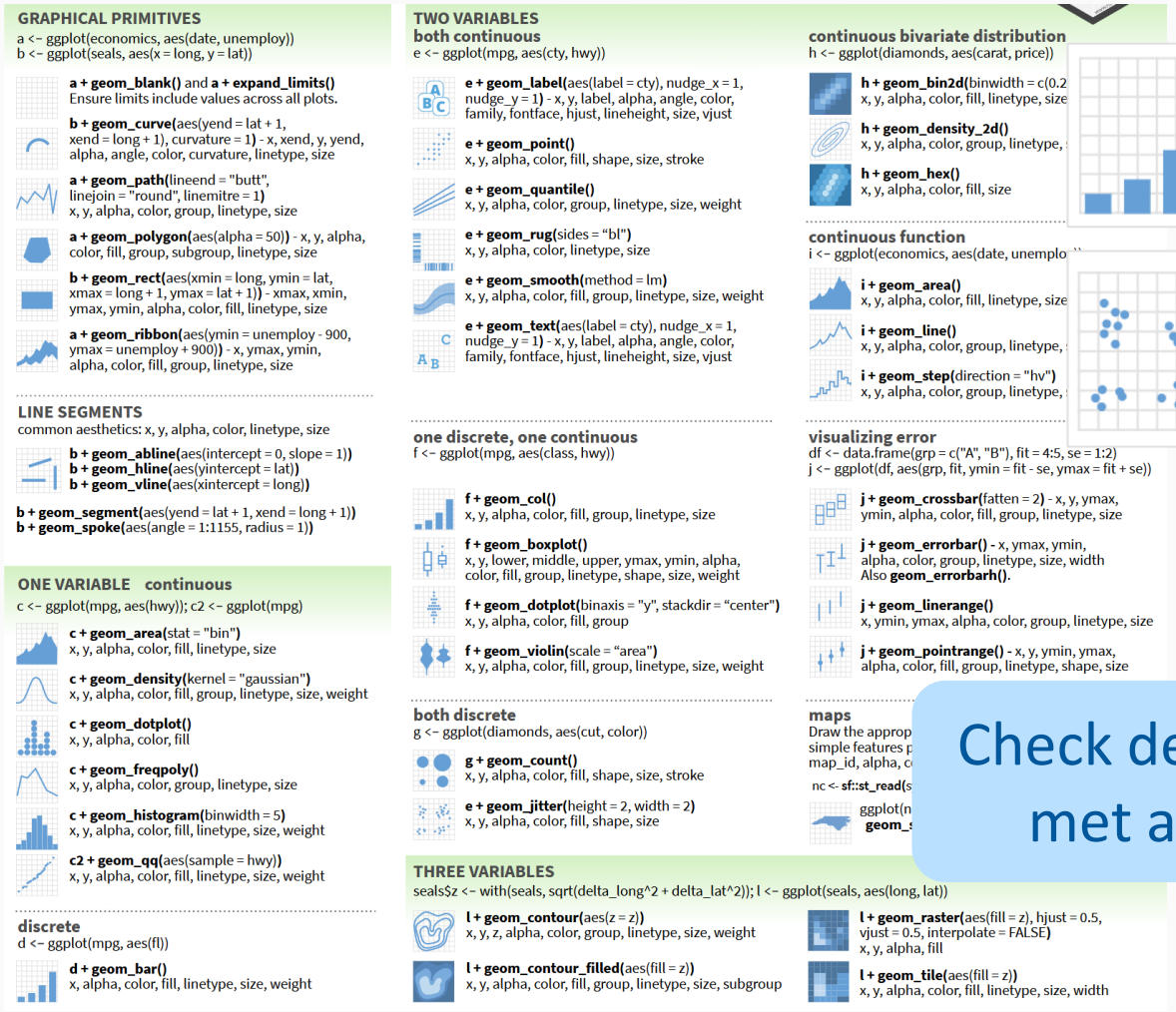
De data en mapping kunnen in elke laag verschillen






Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/layers

# Geoms: encodeer data met “geometries”




**d + geom_bar()**
x, alpha, color, fill, linetype, size, weight

**e + geom_jitter(**height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

Check de cheatsheet met alle geoms

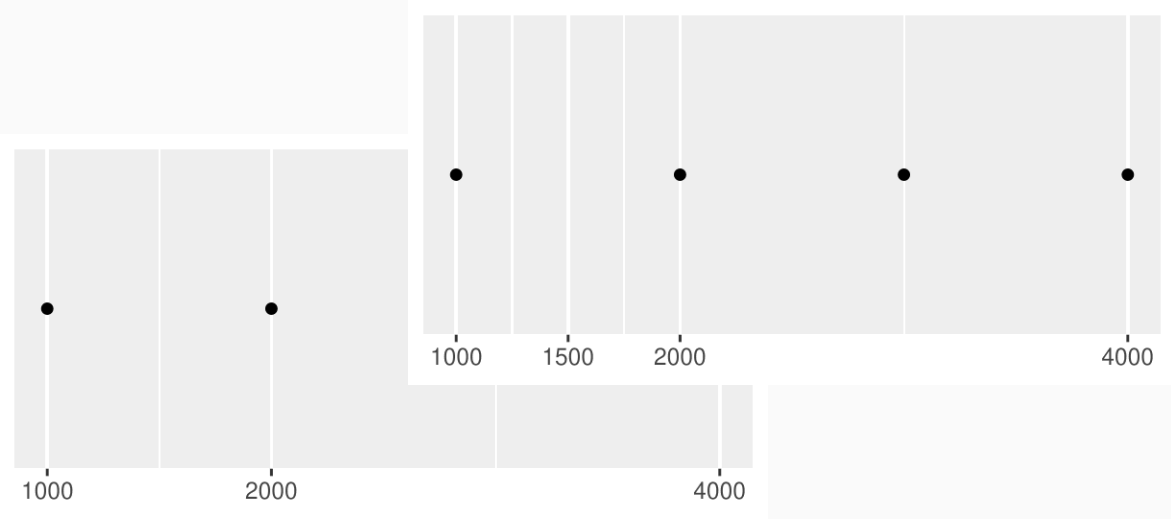
Universiteit Utrecht

Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/individual-geoms
https://rstudio.github.io/cheatsheets/data-visualization.pdf

# Scales: mappings bijsturen

Schalen hoe data gemapt wordt op kleur, positie, grootte, vorm, lijnbreedte, -type…

```
base + scale_x_continuous(breaks = c(1000, 2000, 4000))
base + scale_x_continuous(breaks = c(1000, 1500, 2000, 4000))
```




**Scales** map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

**n <- d + geom_bar(aes(fill = fl))**

**n + scale_fill_manual(**
**values** = c("skyblue", "royalblue", "blue", "navy"),
**limits** = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
**name** = "fuel", labels = c("D", "E", "P", "R"))

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

range of values to include in | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

## GENERAL PURPOSE SCALES

Use with most aesthetics

**scale_*_continuous()** - Map cont' values to visual ones.
**scale_*_discrete()** - Map discrete values to visual ones.
**scale_*_binned()** - Map continuous values to discrete bins.
**scale_*_identity()** - Use data values as visual ones.
**scale_*_manual(**values = c()**)** - Map discrete values to manually chosen visual ones.
**scale_*_date(**date_labels = "%m/%d"),
date_breaks = "2 weeks"**)** - Treat data values as dates.
**scale_*_datetime()** - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.

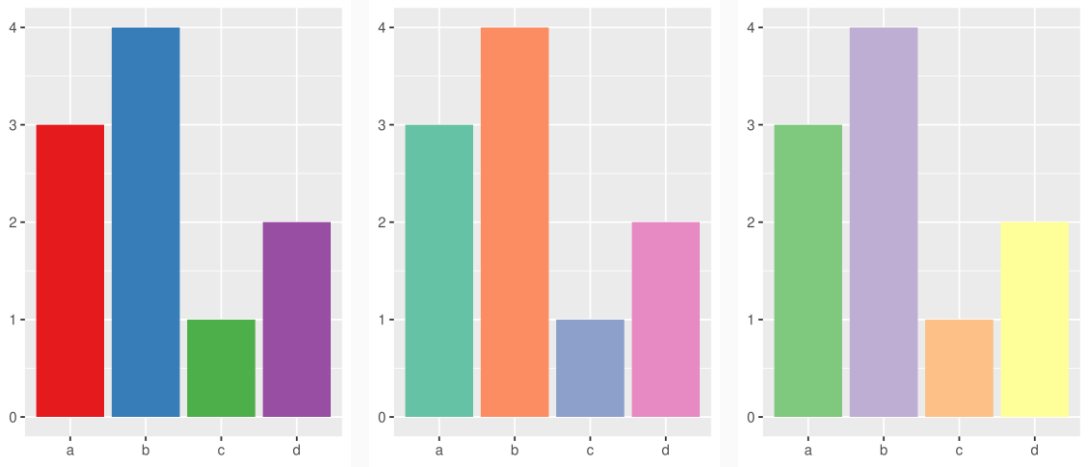


Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/scales-guides
https://ggplot2-book.org/scales

# Scales: mappings bijsturen

Schalen hoe data gemapt wordt op kleur, positie, grootte, vorm, lijnbreedte, -type…

```
bars + scale_fill_brewer(palette = "Set1")
bars + scale_fill_brewer(palette = "Set2")
bars + scale_fill_brewer(palette = "Accent")
```




**Scales** map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

**n <- d + geom_bar(aes(fill = fl))**

**n + scale_fill_manual(**
**values** = c("skyblue", "royalblue", "blue", "navy"),
**limits** = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
**name** = "fuel", labels = c("D", "E", "P", "R"))

**scale_** — **aesthetic to adjust** — **prepackaged scale to use** — **scale-specific arguments**

**range of values to include in** — **title to use in legend/axis** — **labels to use in legend/axis** — **breaks to use in legend/axis**

## GENERAL PURPOSE SCALES

Use with most aesthetics

**scale_*_continuous()** - Map cont' values to visual ones.
**scale_*_discrete()** - Map discrete values to visual ones.
**scale_*_binned()** - Map continuous values to discrete bins.
**scale_*_identity()** - Use data values as visual ones.
**scale_*_manual(**values = c()**)** - Map discrete values to manually chosen visual ones.
**scale_*_date(**date_labels = "%m/%d"), date_breaks = "2 weeks"**)** - Treat data values as dates.
**scale_*_datetime()** - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.

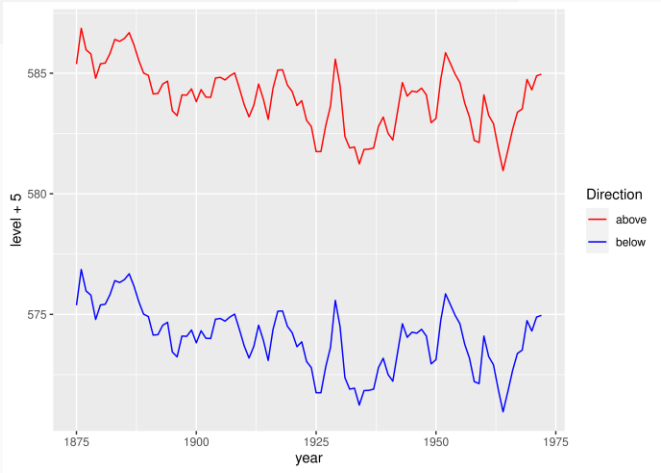


Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/scales-guides
https://ggplot2-book.org/scales

# Scales: mappings bijsturen

Schalen hoe data gemapt wordt op kleur, positie, grootte, vorm, lijnbreedte, -type…

```
ggplot(huron, aes(year)) +
  geom_line(aes(y = level + 5, colour = "above")) +
  geom_line(aes(y = level - 5, colour = "below")) +
  scale_colour_manual("Direction",
    values = c("above" = "red", "below" = "blue")
  )
```




**Scales** map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

**n <– d + geom_bar(aes(fill = fl))**

**n + scale_fill_manual(**
**values** = c("skyblue", "royalblue", "blue", "navy"),
**limits** = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
**name** = "fuel", labels = c("D", "E", "P", "R"))

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

range of values to include in | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

## GENERAL PURPOSE SCALES

Use with most aesthetics

**scale_*_continuous()** - Map cont' values to visual ones.
**scale_*_discrete()** - Map discrete values to visual ones.
**scale_*_binned()** - Map continuous values to discrete bins.
**scale_*_identity()** - Use data values as visual ones.
**scale_*_manual(**values = c()**)** - Map discrete values to manually chosen visual ones.
**scale_*_date(**date_labels = "%m/%d"),
date_breaks = "2 weeks"**)** - Treat data values as dates.
**scale_*_datetime()** - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.



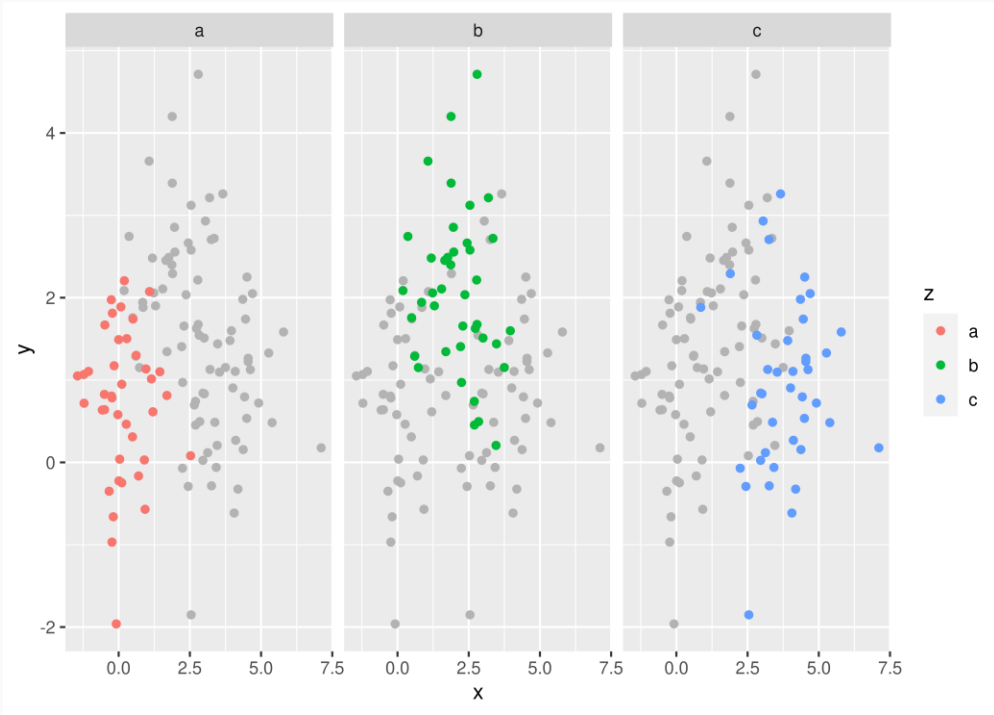
Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/scales-guides
https://ggplot2-book.org/scales

# Faceting: small multiples maken

Gebruik `facet_wrap()` om een
“opgerold lint” van kleine plotjes te maken

```
ggplot(df, aes(x, y)) +
  geom_point(data = df2, colour = "grey70") +
  geom_point(aes(colour = z)) +
  facet_wrap(~z)
```


```
facet_wrap(
  facets,
  nrow = NULL,
  ncol = NULL,
  scales = "fixed",
  shrink = TRUE,
  labeller = "label_value",
  as.table = TRUE,
  switch = deprecated(),
  drop = TRUE,
  dir = "h",
  strip.position = "top",
  axes = "margins",
  axis.labels = "all"
)
```

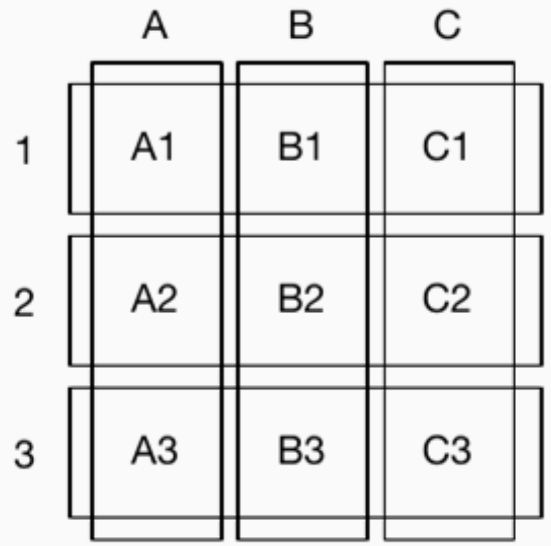




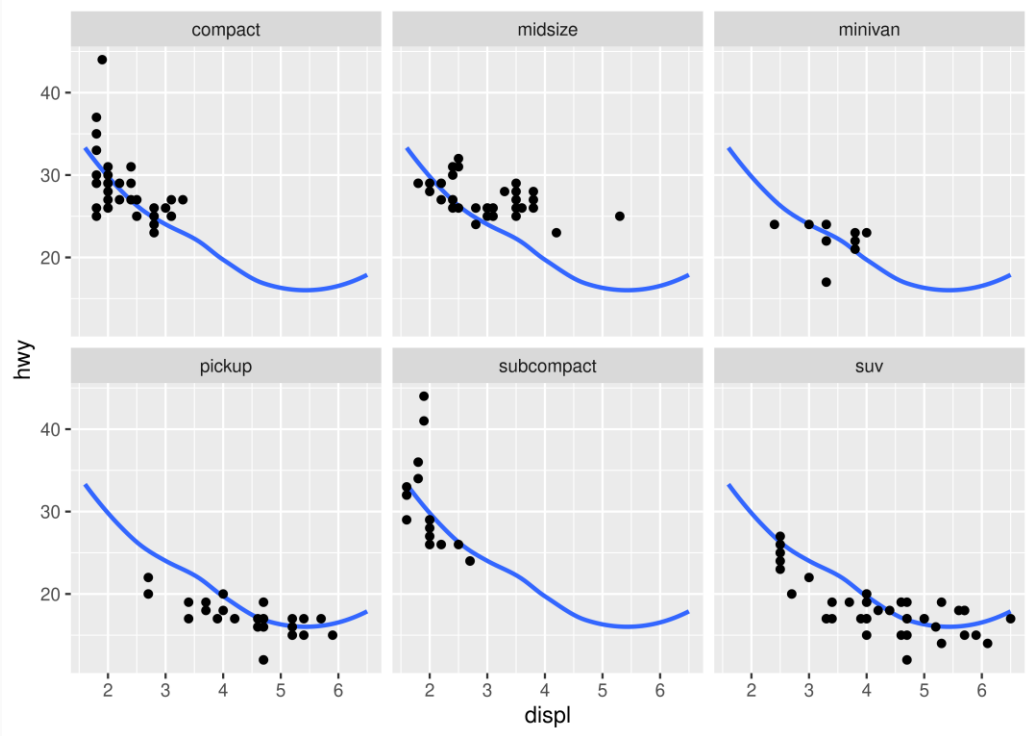

Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/facet

# Faceting: small multiples maken

Gebruik `facet_grid()` om een rooster van kleine plotjes te maken




```
facet_grid(
  rows = NULL,
  cols = NULL,
  scales = "fixed",
  space = "fixed",
  shrink = TRUE,
  labeller = "label_value",
  as.table = TRUE,
  switch = NULL,
  drop = TRUE,
  margins = FALSE,
  axes = "margins",
  axis.labels = "all",
  facets = deprecated()
)
```





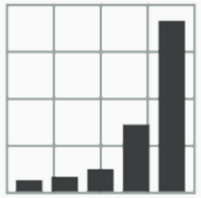

Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/facet

# Themes: het algemene thema aanpassen

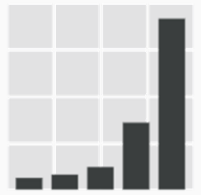
Verschillende thema's passen de algemene lay-out van de vis aan

Meer thema's met de library `ggthemes`

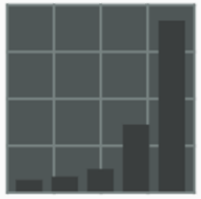
Kies een thema dat dicht ligt bij wat je wilt bereiken en verfijn



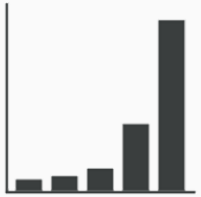
**r + theme_bw()**
White background with grid lines.



**r + theme_gray()**
Grey background (default theme).

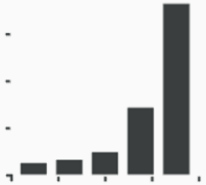


**r + theme_dark()**
Dark for contrast.



**r + theme_classic()**

**r + theme_light()**



**r + theme_linedraw()**

**r + theme_minimal()**
Minimal theme.

**r + theme_void()**
Empty theme.



Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/themes

# Themes: het thema verfijnen

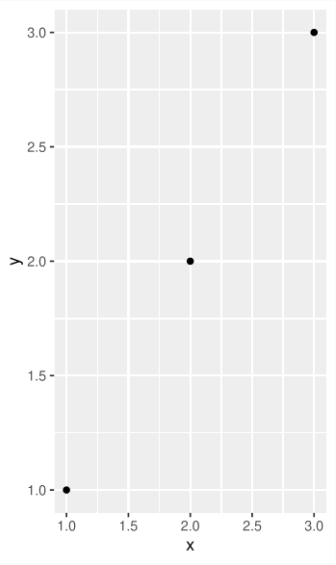
Gebruik `theme()`

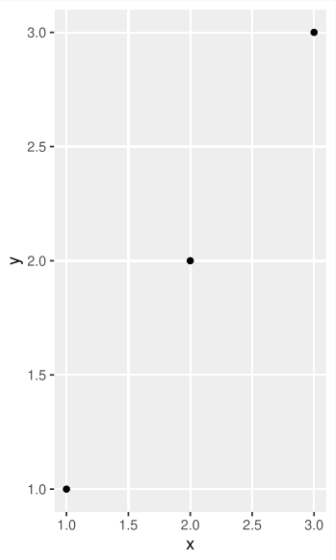
Je kunt erg veel parameters aanpassen,
bv. roosterlijnen, assen en ticks, legende…

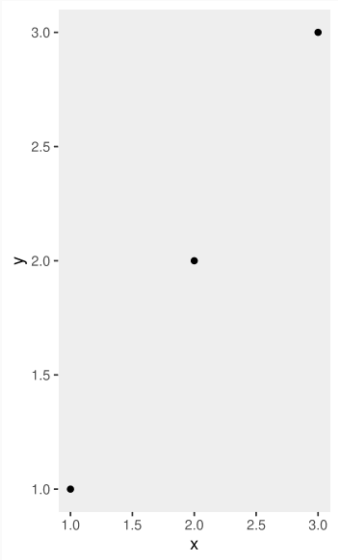
Gebruik een gepast “element”:

- `element_text()`
- `element_line()`
- `element_rect()`
- `element_blank()`

```
base
last_plot() + theme(panel.grid.minor = element_blank())
last_plot() + theme(panel.grid.major = element_blank())
```






Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/themes

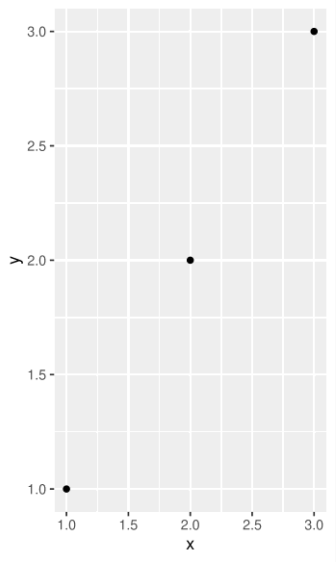
# Themes: het thema verfijnen

Gebruik `theme()`

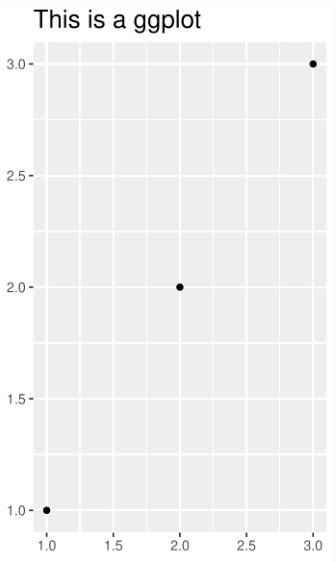
Je kunt erg veel parameters aanpassen,
bv. roosterlijnen, assen en ticks, legende…

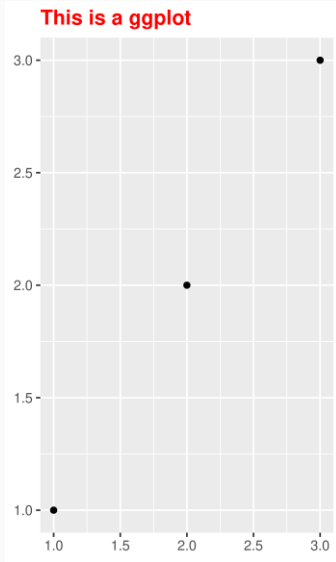
Gebruik een gepast “element”:

- `element_text()`
- `element_line()`
- `element_rect()`
- `element_blank()`




Base




```
base_t <- base + labs(title = "This is a ggplot") + xlab(NULL) + ylab(NULL)
base_t + theme(plot.title = element_text(size = 16))
base_t + theme(plot.title = element_text(face = "bold", colour = "red"))
```



Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org/themes

# Basics: plots opslaan

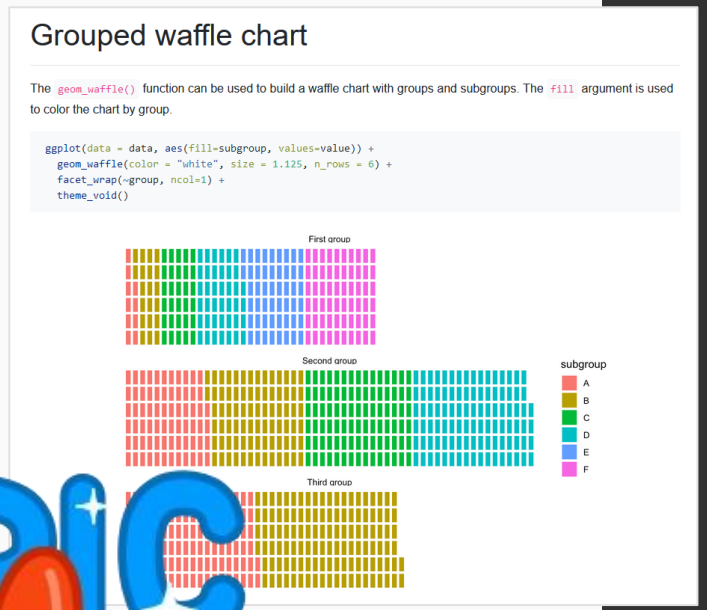
Gebruik `ggsave()`

Als je niet expliciet een plot meegeeft als parameter, dan wordt de laatste plot gebruikt

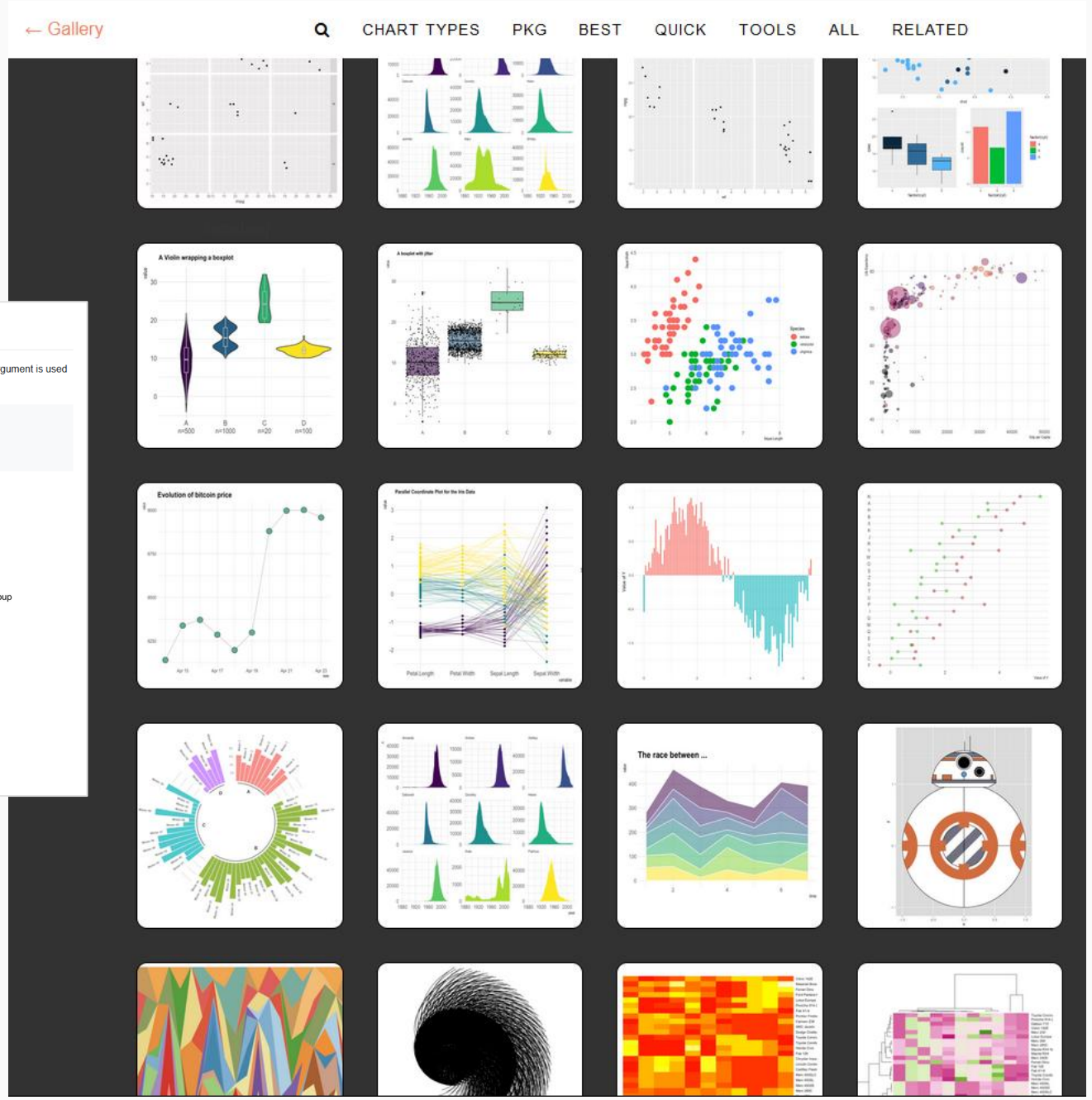
```
ggplot(mpg, aes(displ, cty)) + geom_point()
ggsave("output.pdf")
```

```
ggsave(
  filename,
  plot = last_plot(),
  device = NULL,
  path = NULL,
  scale = 1,
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  limitsize = TRUE,
  bg = NULL,
  create.dir = FALSE,
  ...
)
```

# Extra voorbeelden







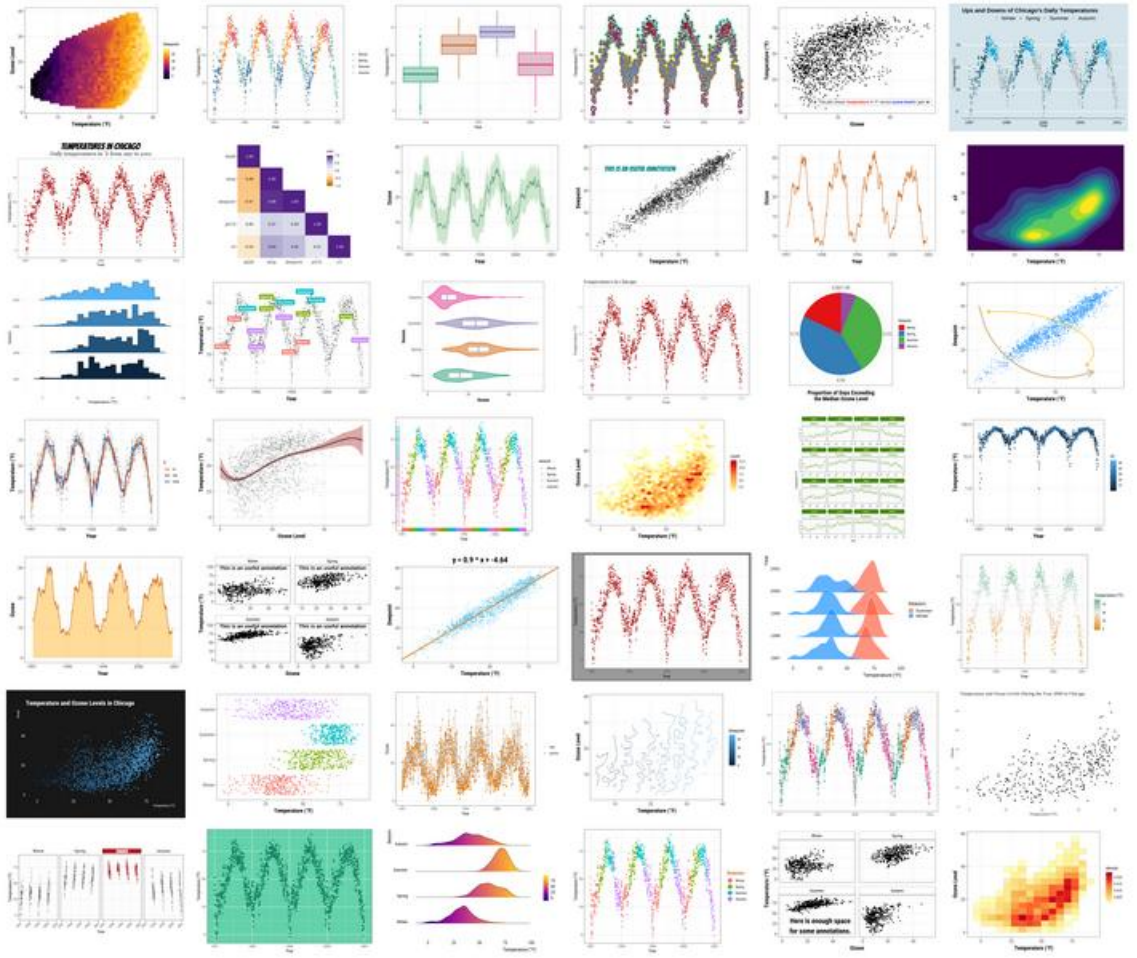

https://r-graph-gallery.com

# Extra tutorial


## Table of Content



`{ggforce}` packages. How to deal with custom fonts and colors. A collection of R packages tailored to create interactive charts. And several other chart types including pie charts because everyone looooves pie charts!)



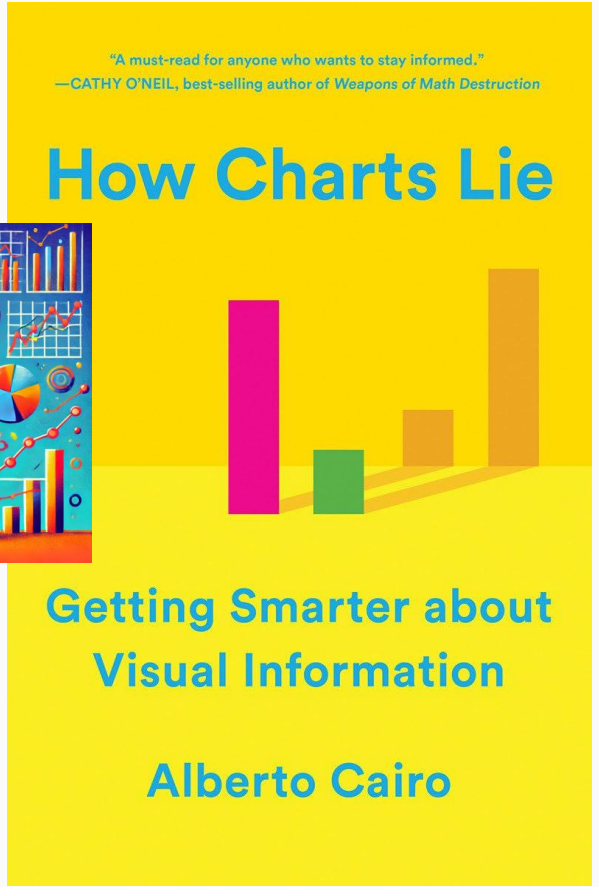
Some exemplary plots included in this tutorial.

https://www.cedricscherer.com/2019/08/05/a-ggplot2-tutorial-for-beautiful-plotting-in-r

Hoorcollege 3

# Introductie tot ggplot2
# Misleiden met visualisaties

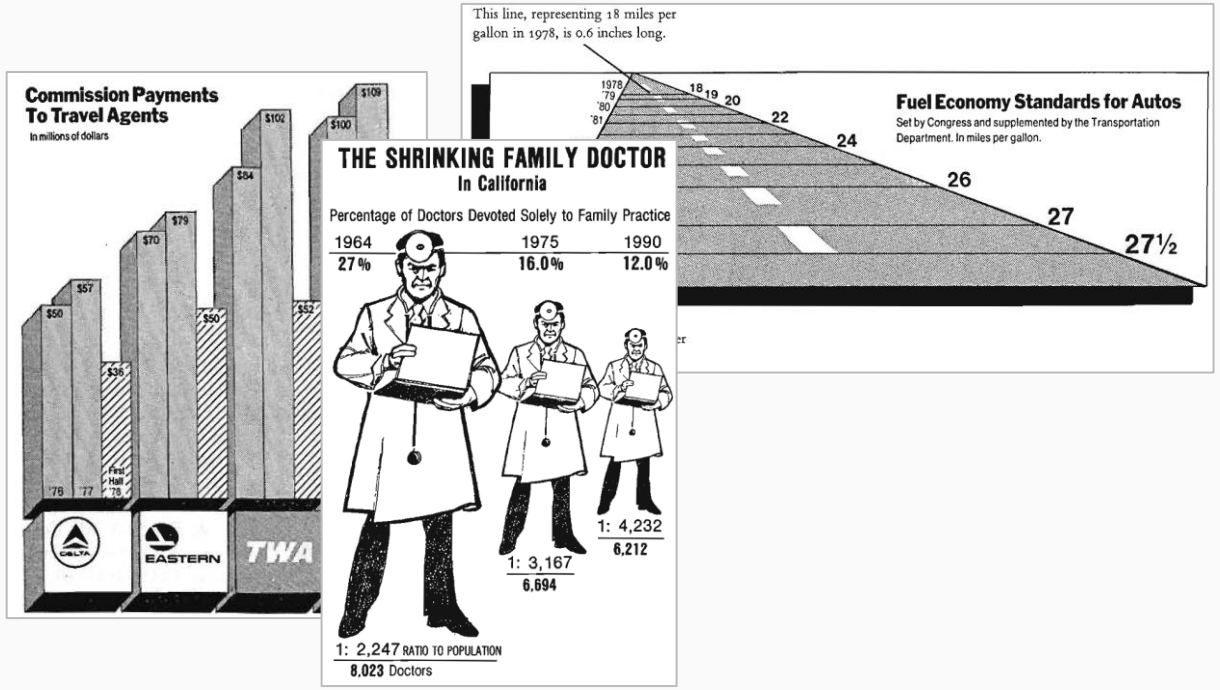
1. Introductie tot ggplot2
2. Misleiden met visualisaties
   a) Slecht ontwerp
   b) Dubieuze data
   c) Te weinig data
   d) Verborgen of onduidelijke onzekerheid
   e) Suggestieve patronen

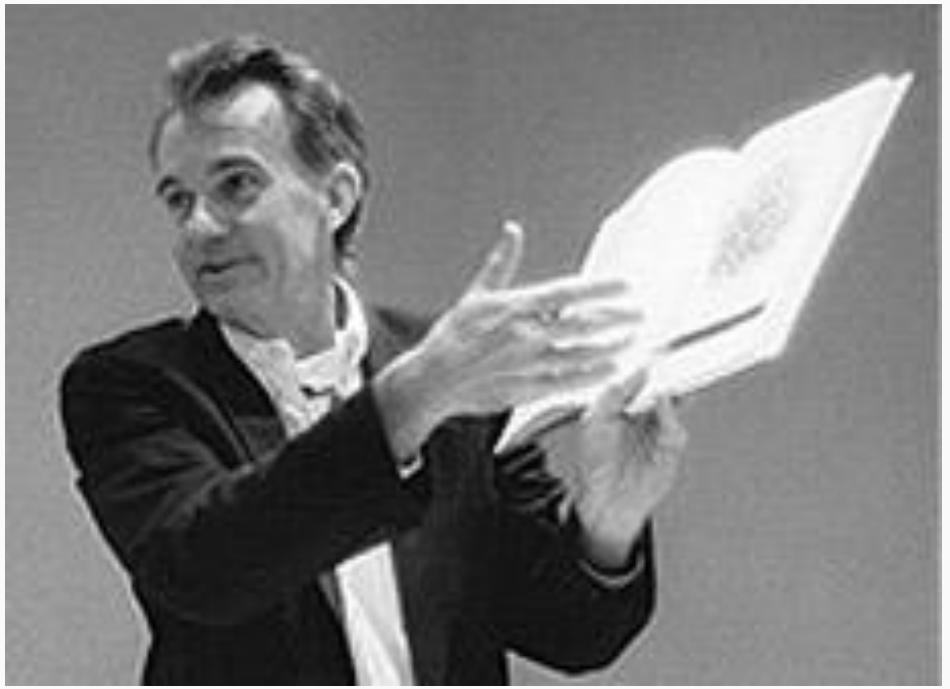



# Visualisatieprincipes van Tufte

**Grafische integriteit** = vertel de waarheid met vis

**Designesthetiek** = maak vis duidelijk en precies

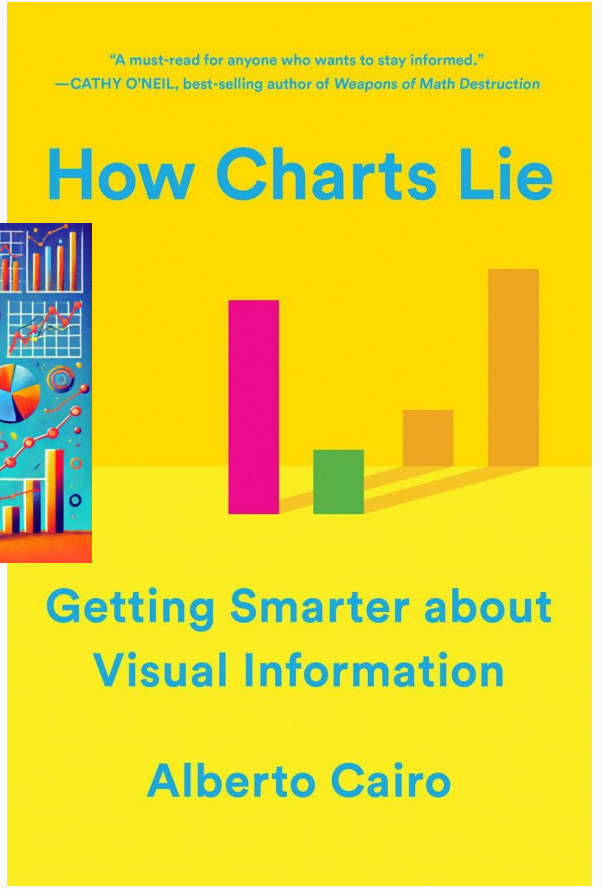


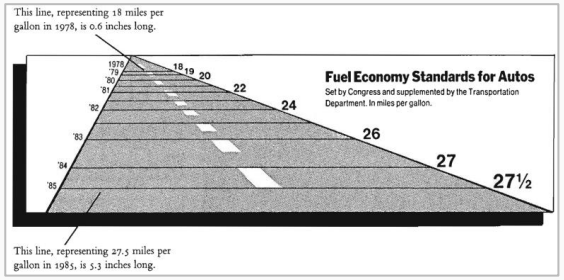
Hoorcollege 3

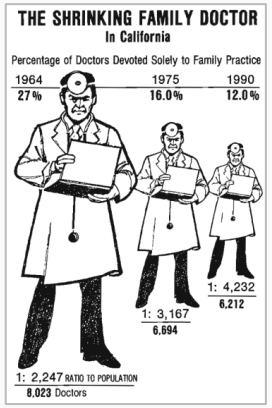
# Introductie tot ggplot2
# Misleiden met visualisaties

1. Introductie tot ggplot2
2. Misleiden met visualisaties
   a) Slecht ontwerp
   b) Dubieuze data
   c) Te weinig data
   d) Verborgen of onduidelijke onzekerheid
   e) Suggestieve patronen

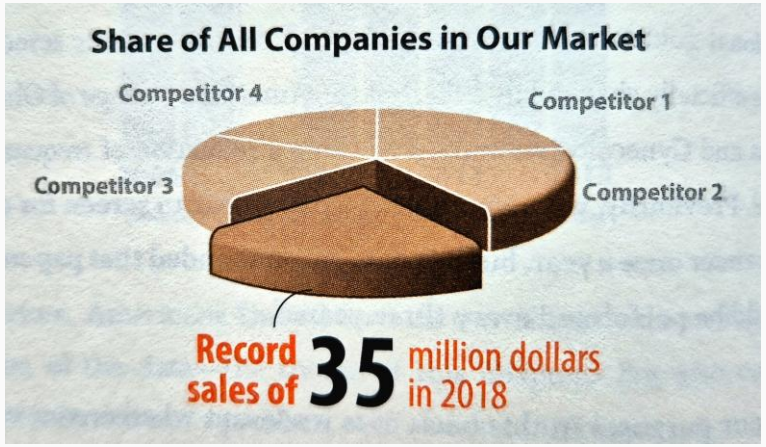



# Slecht ontwerp

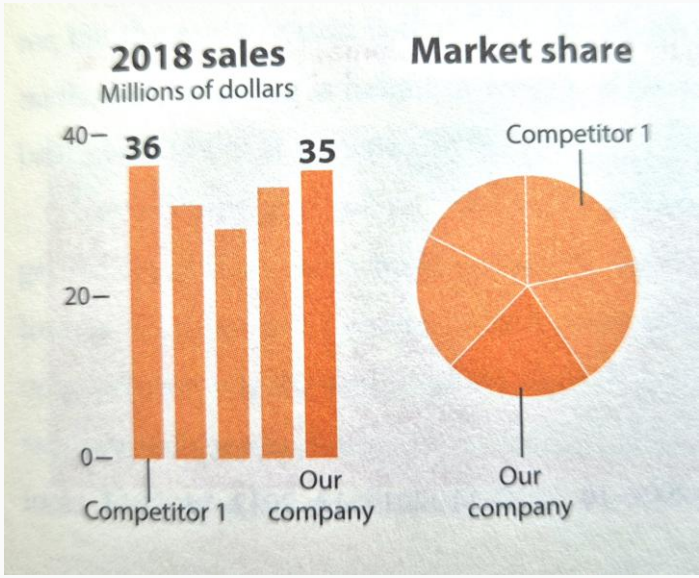



Visuele distortie door 3D-perspectief overdrijft verhoudingen of veranderingen

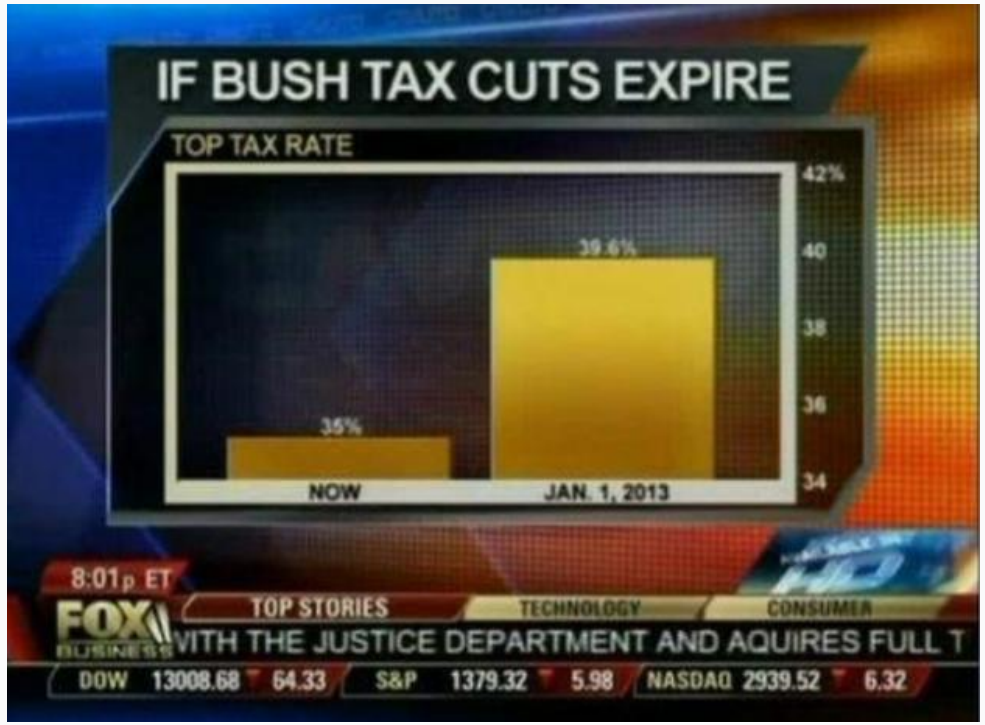

Beter: 2D zonder distortie

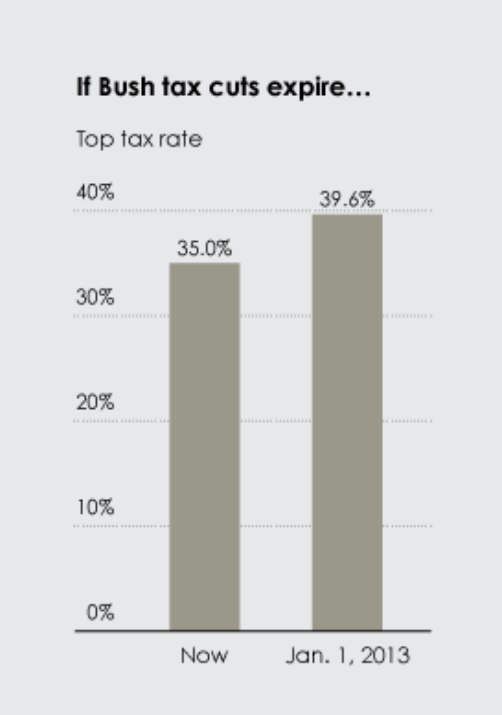





Alberto Cairo. *How charts lie. Getting smarter about visual information.* Norton, 2019. p. 58

# Slecht ontwerp



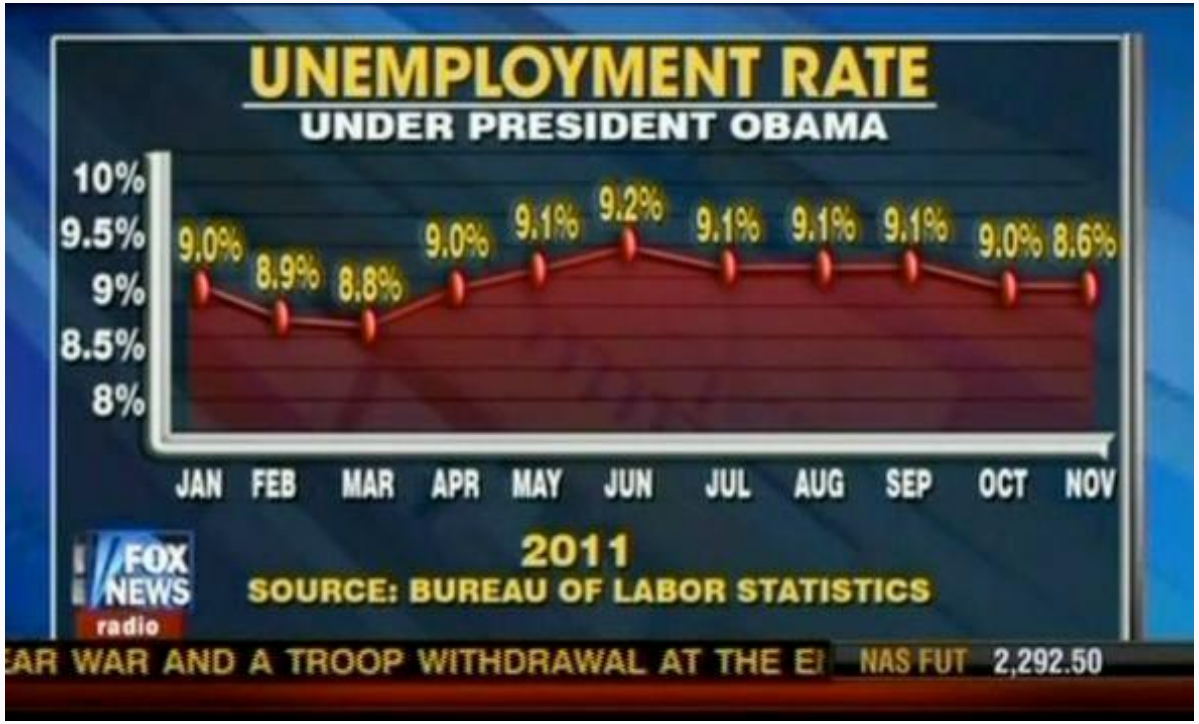



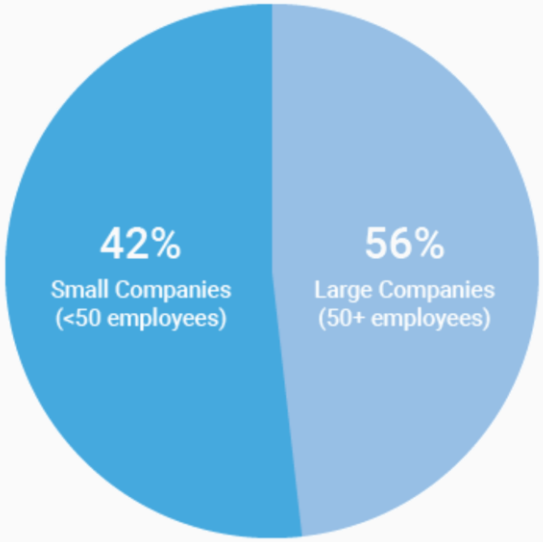

Geef assen betekenisvolle startpunten, bv. 0

Schalen en proporties manipuleren door niet te coderen volgens de data

https://flowingdata.com/2012/08/06/fox-news-continues-charting-excellence

# Slecht ontwerp





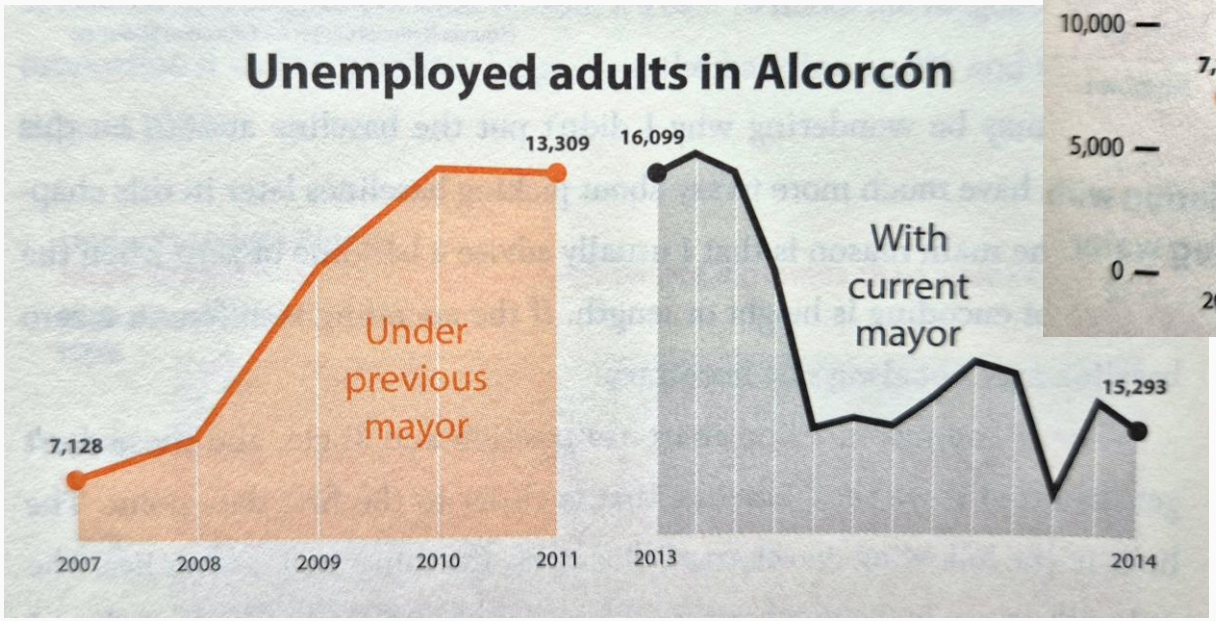
Schalen en proporties manipuleren door niet te coderen volgens de data

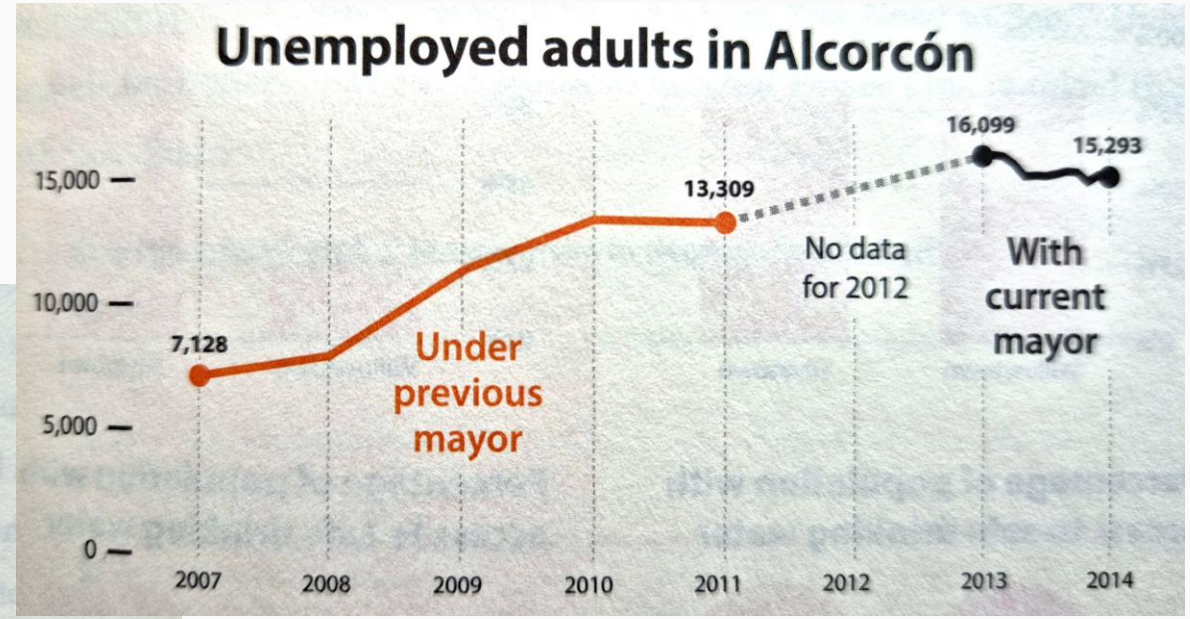
Universiteit Utrecht

http://flowingdata.com/category/statistics/mistaken-data
https://viz.wtf/image/185180504504

Source: Joris Kiers

# Slecht ontwerp



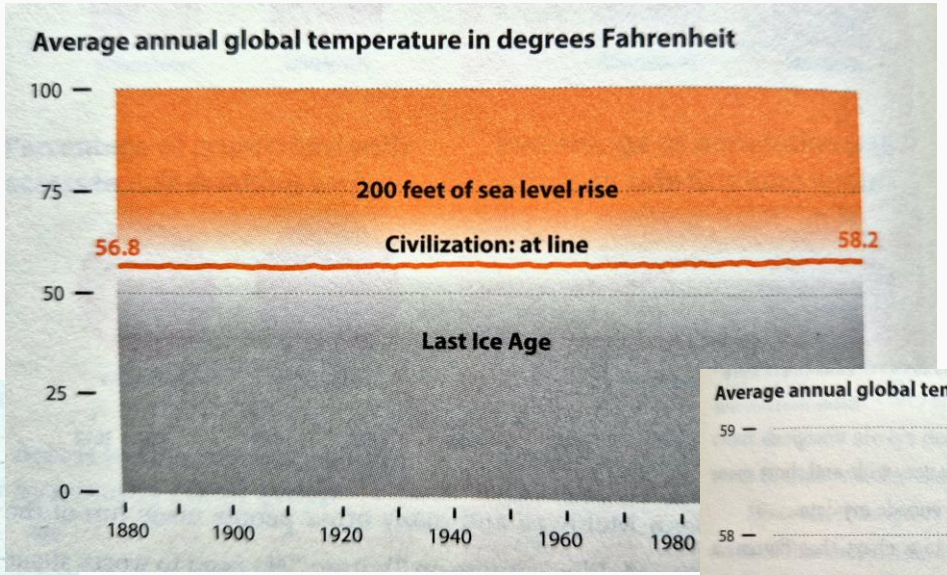

Beter: identiek assenstelsel

Foutieve boodschap suggereren

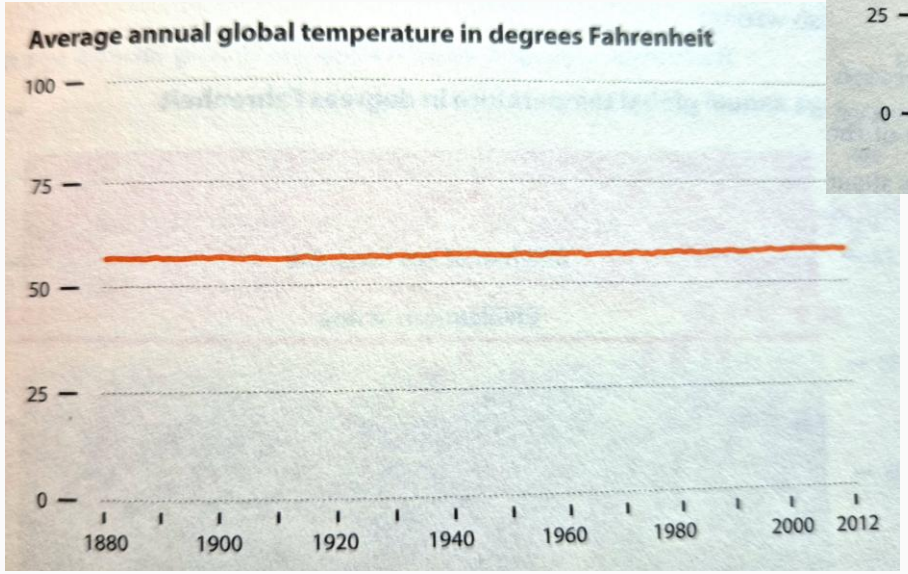


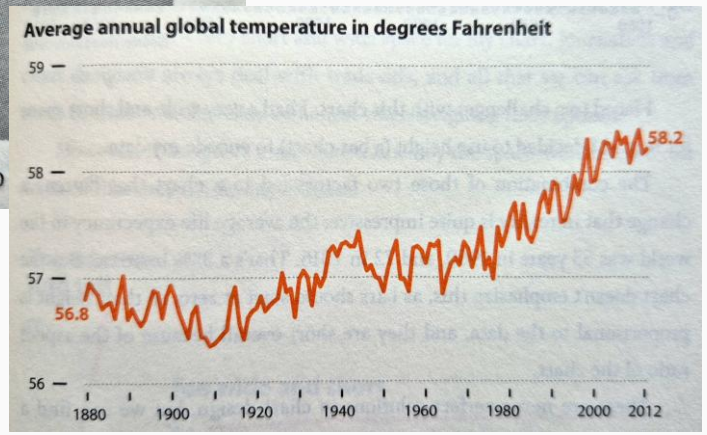
Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 65-67

# Slecht ontwerp




Beter: context



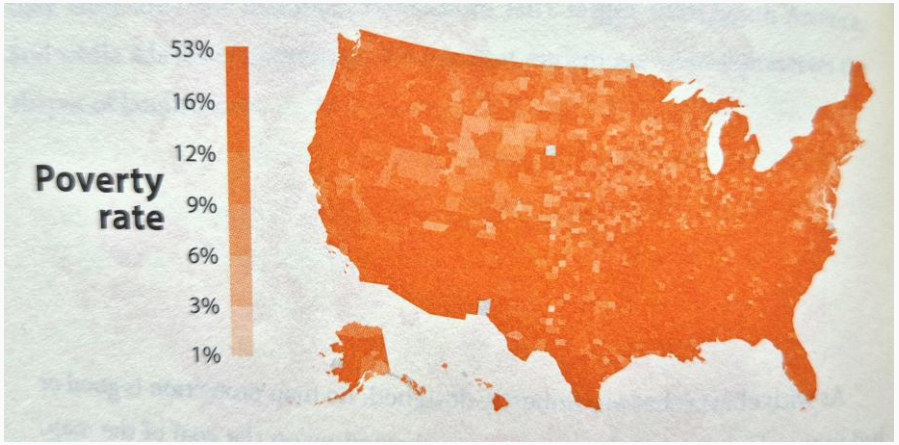




Foutieve boodschap suggereren



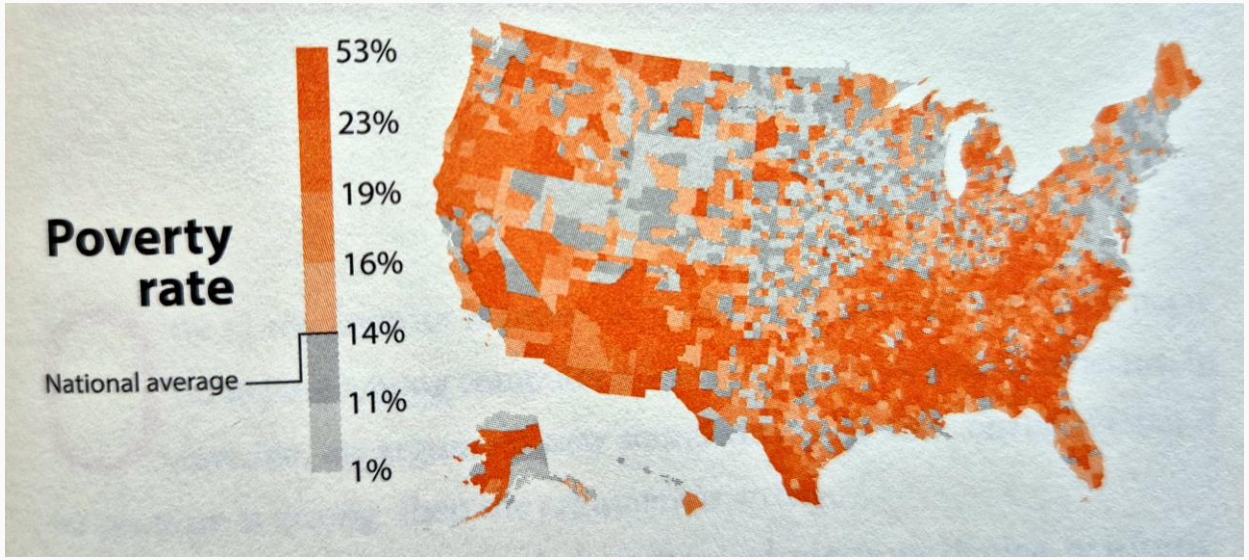
Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 65-67

# Slecht ontwerp


Kleurschalen manipuleren
naargelang de gewenste boodschap

Beter: verdeel datapunten gelijk over de bins



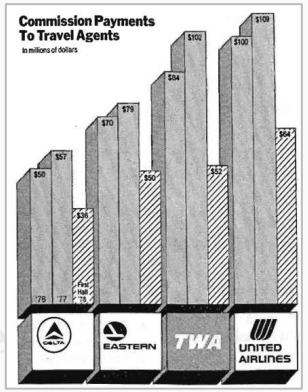


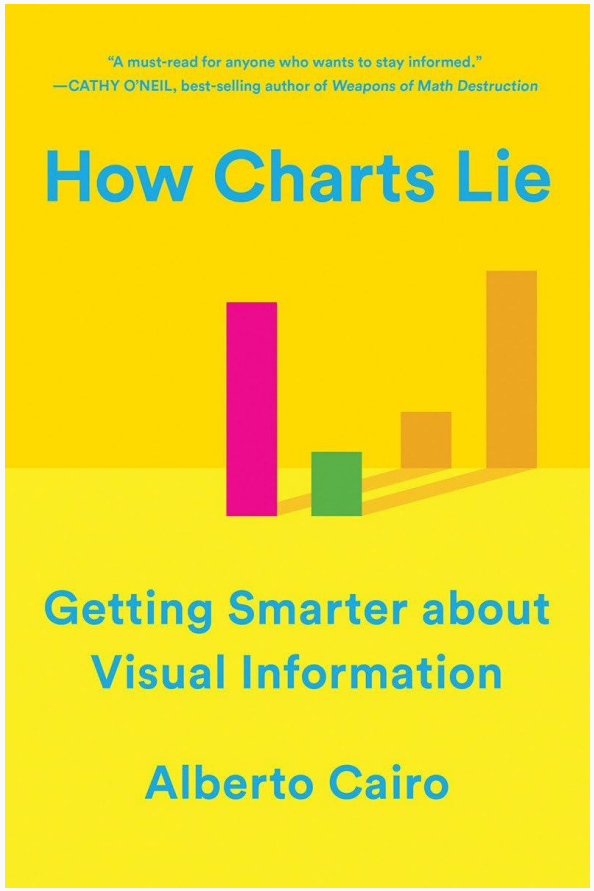

Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 78-79

Hoorcollege 3

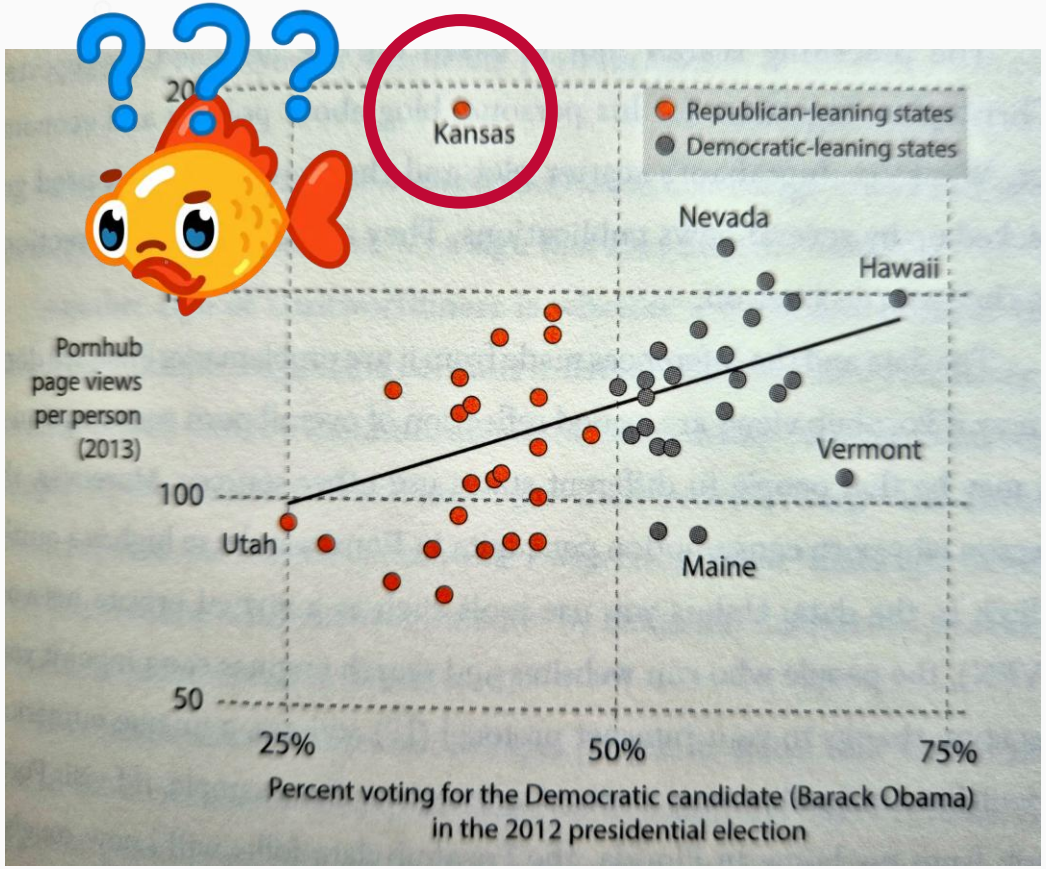
# Introductie tot ggplot2
# Misleiden met visualisaties

1. Introductie tot ggplot2
2. Misleiden met visualisaties
   a) Slecht ontwerp
   b) Dubieuze data
   c) Te weinig data
   d) Verborgen of onduidelijke onzekerhe
   e) Suggestieve patronen

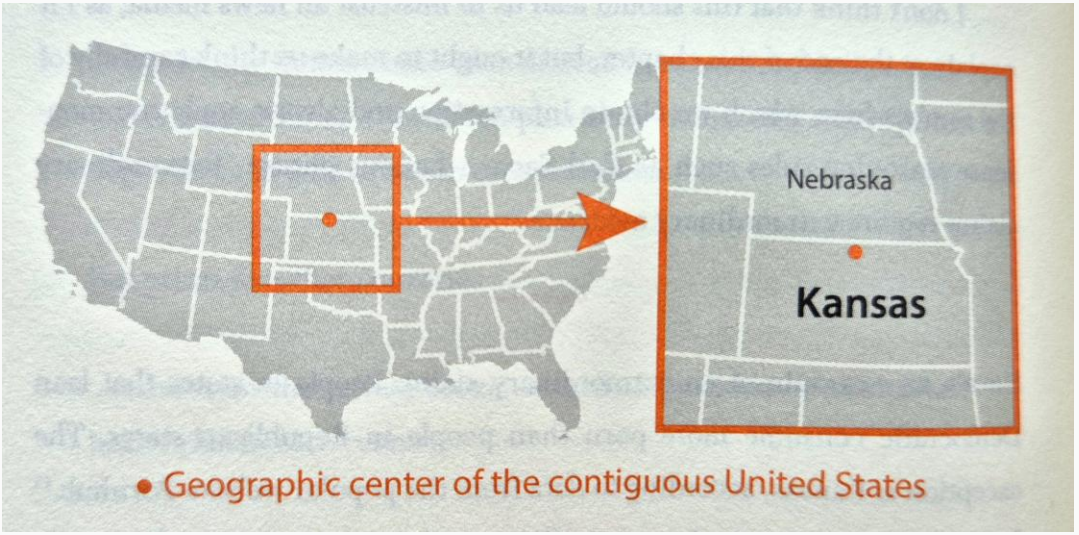


# Dubieuze data

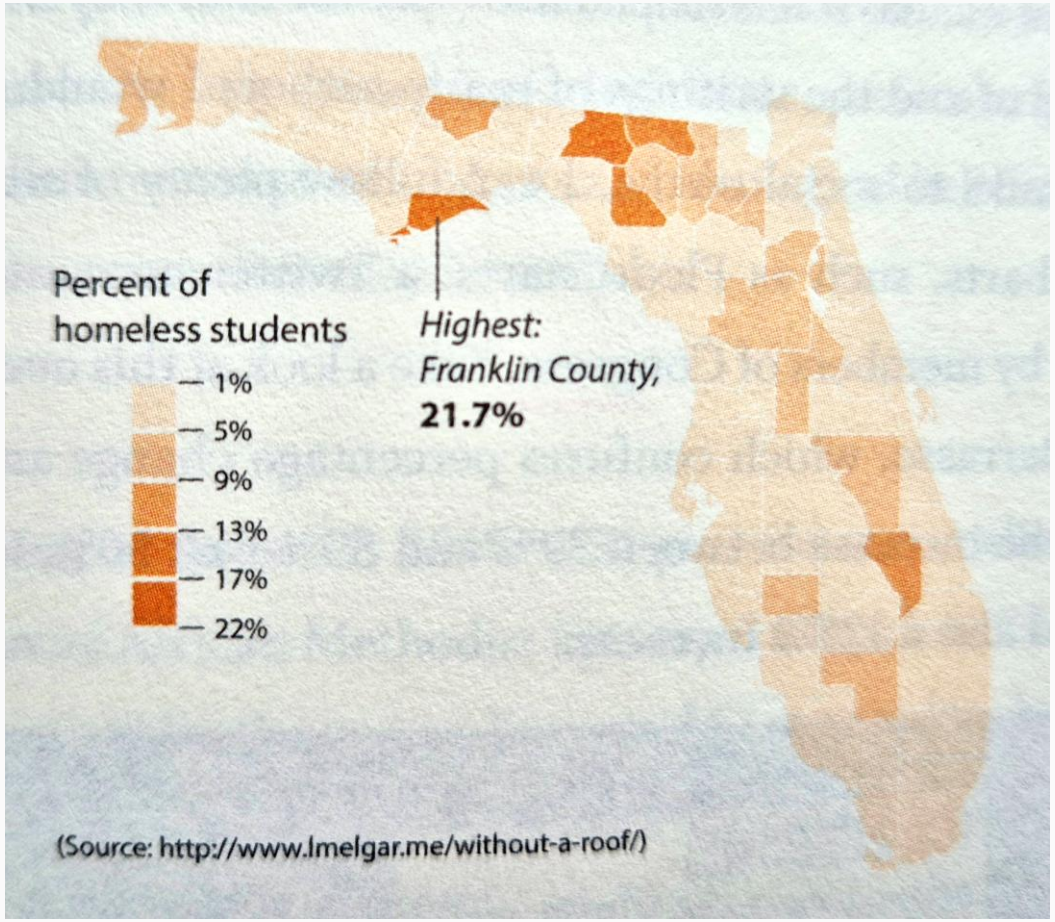



Aannames bij datacollectie
en -interpretatie zijn soms fout




Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 99

# Dubieuze data




Betekenis van gevisualiseerde informatie is niet meteen duidelijk

vb. wat is “homeless”?



Alberto Cairo. *How charts lie. Getting smarter about visual information.* Norton, 2019. p. 95

# Dubieuze data

Female employees

| Employee | Salary ($) | Employee | Salary ($) |
|---|---|---|---|
| Manager | 150,000 | Regular worker | 45,000 |
| Manager | 130,000 | Regular worker | 42,000 |
| Manager | 115,000 | Regular worker | 40,000 |
| Supervisor | 76,000 | Regular worker | 38,000 |
| Supervisor | 74,500 | Regular worker | 36,000 |
| Supervisor | 72,000 | Regular worker | 35,250 |
| Regular worker | 70,000 | Intern | 15,000 |
| | | Intern | 15,000 |

Woman makes more than man in the same position

Male employees

| Employee | Salary ($) | Employee | |
|---|---|---|---|
| Manager | 162,000 | Regular worke | |
| Manager | 138,500 | Regular worke | |
| Manager | 125,000 | Regular worke | |
| Supervisor | 80,000 | Regular worke | |
| Supervisor | 76,000 | Regular worker | 35,500 |
| Supervisor | 73,000 | Regular worker | 35,000 |
| Regular worker | 68,500 | Intern | 14,000 |
| | | Intern | 14,000 |

Man makes more than woman in the same position

3/5 van de vrouwen verdient meer dan mannen

Boodschap is misleidend omdat
slechts 1 perspectief wordt uitgelicht



Alberto Cairo. *How charts lie. Getting smarter about visual information.* Norton, 2019. p. 94

# Dubieuze data

**Vuistregels**:

- Wantrouw vis zonder (betrouwbare) bron
- Varieer de mediakanalen die je gebruikt
- Veronderstel dat men je niet bewust wil misleiden
- Schakel emoties en meningen even uit als je een nieuwe vis ziet

"One reason charts lie is that we are prone to lying to ourselves"

Alberto Cairo, 2019

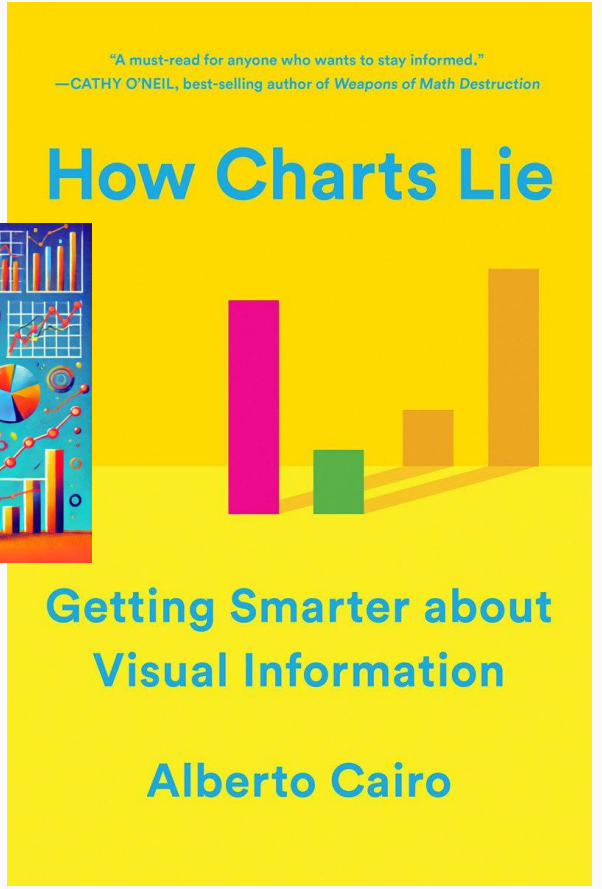


Alberto Cairo. *How charts lie. Getting smarter about visual information.* Norton, 2019. p. 104-106

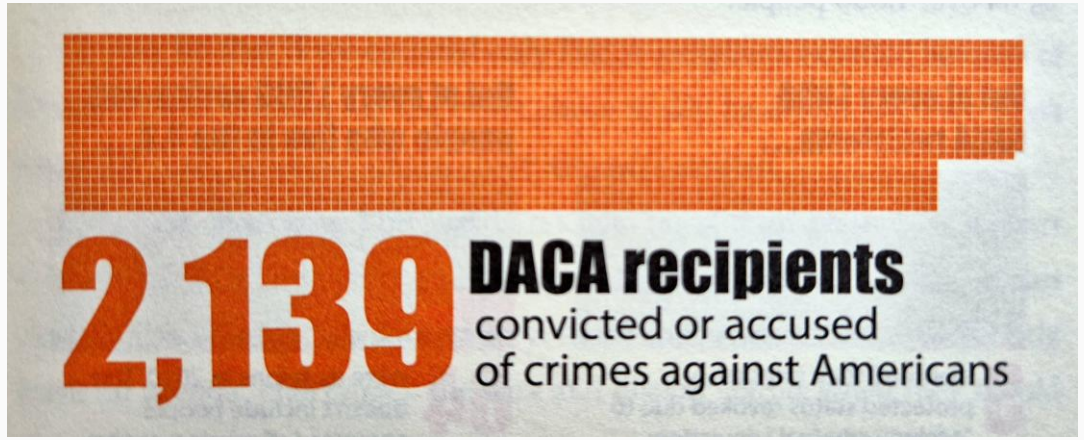
Hoorcollege 3

# Introductie tot ggplot2
# Misleiden met visualisaties

1. Introductie tot ggplot2
2. Misleiden met visualisaties
   a) Slecht ontwerp
   b) Dubieuze data
   c) Te weinig data
   d) Verborgen of onduidelijke onzekerheid
   e) Suggestieve patronen

# Te weinig data

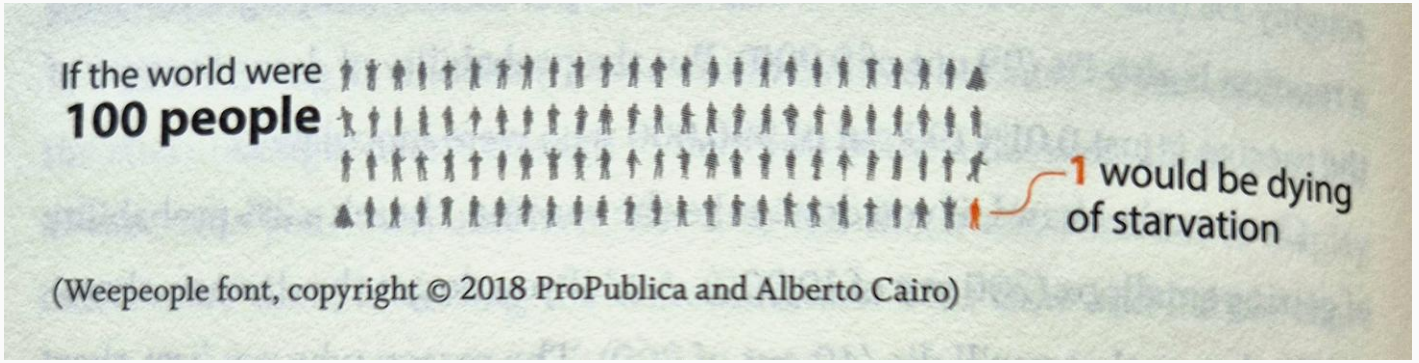



Proporties zijn belangrijk: absolute aantallen dramatiseren soms




Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 111-112

# Te weinig data


In sommige contexten zijn absolute aantallen wel gepast

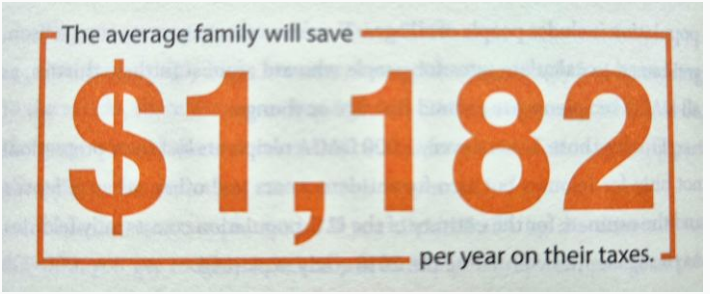
Vergeet de menselijke kant van data niet

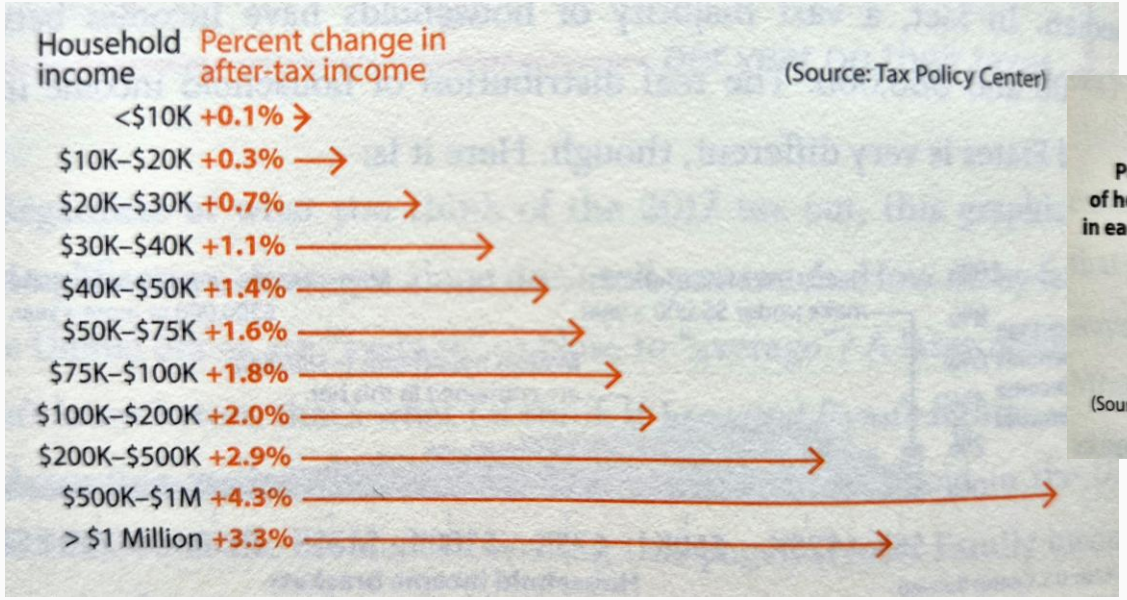


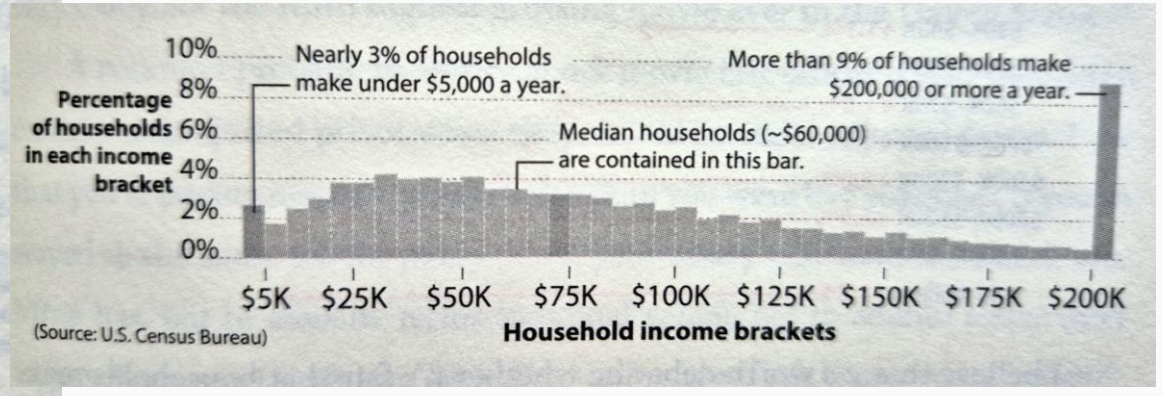
Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 111

# Te weinig data

Inkomensverdeling wordt verzwegen




Beter: toon het hele plaatje

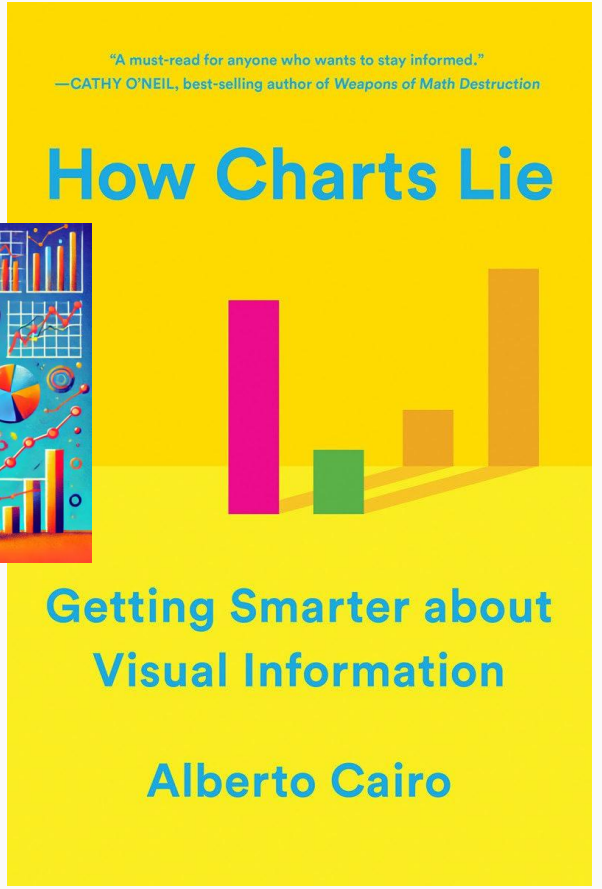


Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 114-116

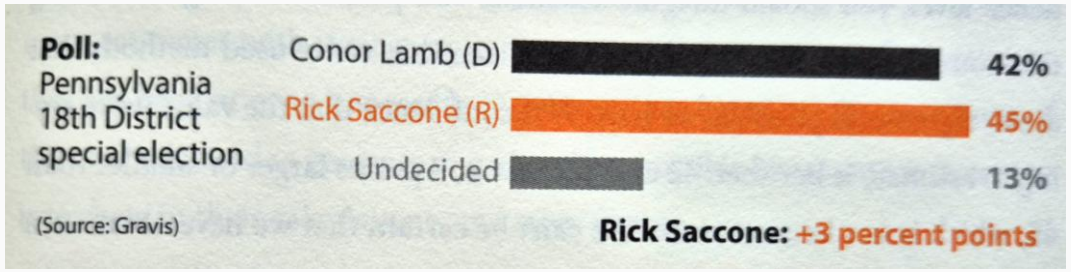
Hoorcollege 3

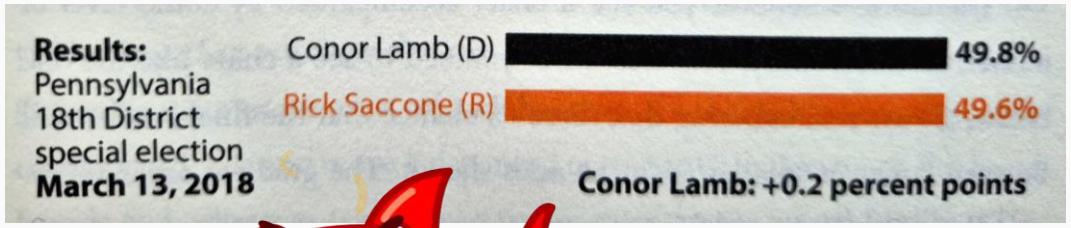
# Introductie tot ggplot2
# Misleiden met visualisaties

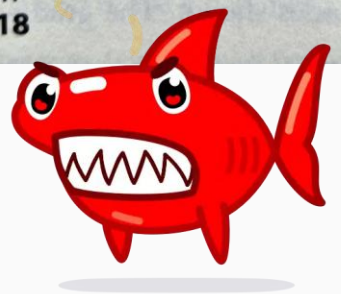
1. Introductie tot ggplot2
2. Misleiden met visualisaties
   a) Slecht ontwerp
   b) Dubieuze data
   c) Te weinig data
   d) Verborgen of onduidelijke onzekerheid
   e) Suggestieve patronen

# Verborgen of onduidelijke onzekerheid

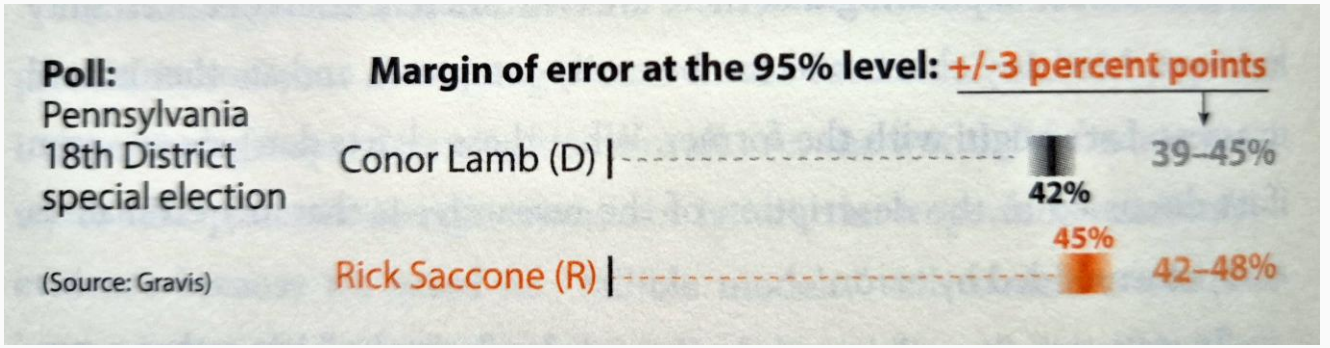


Ontbrekende onzekerheid schept een vals gevoel van zekerheid

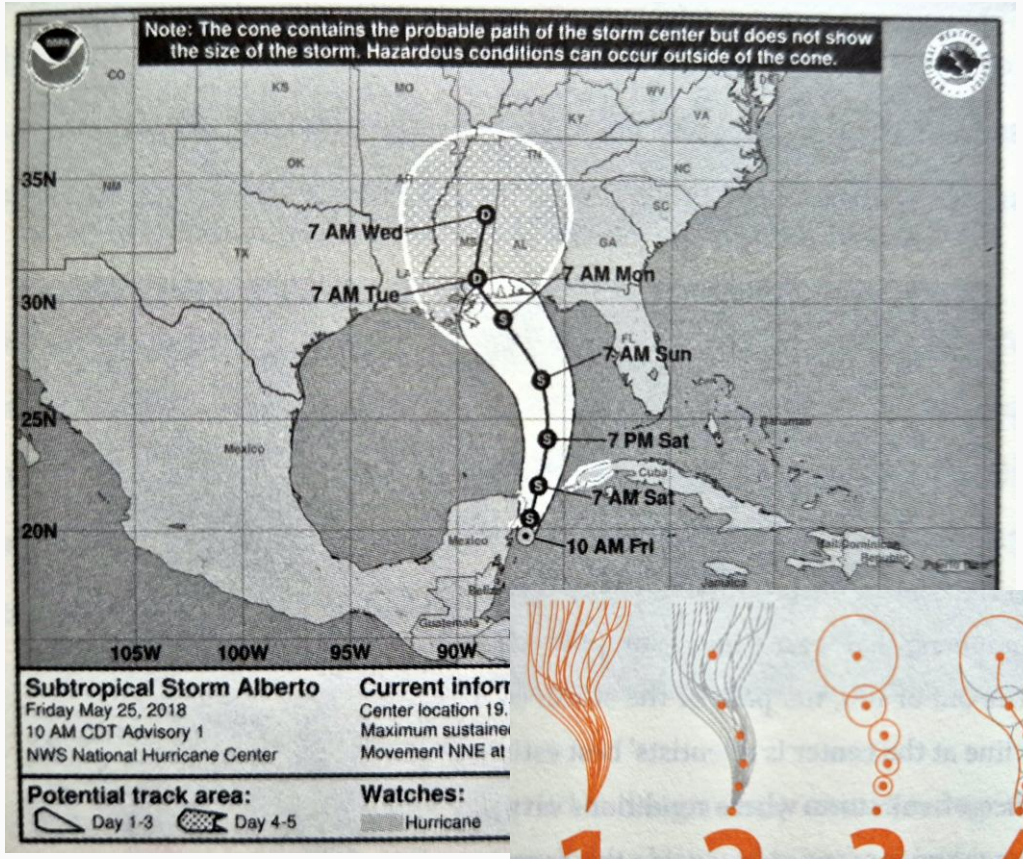
Beter: toon onzekerheid






Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 141-142

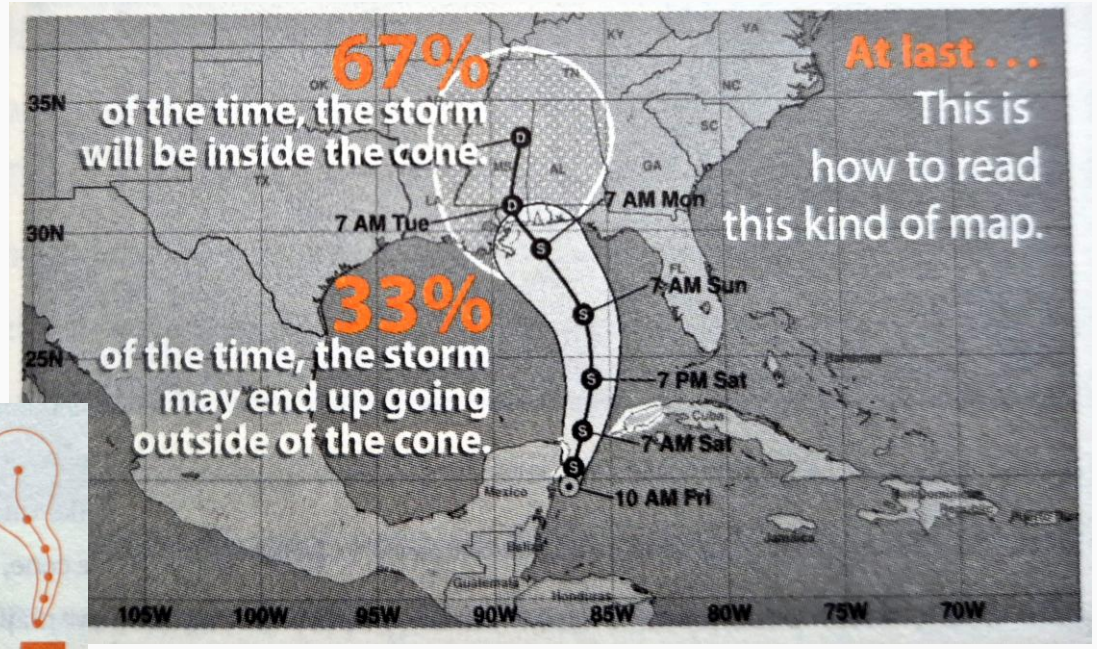
# Verborgen of onduidelijke onzekerheid







Onzekerheid kan verwarrend zijn zonder afdoende duiding

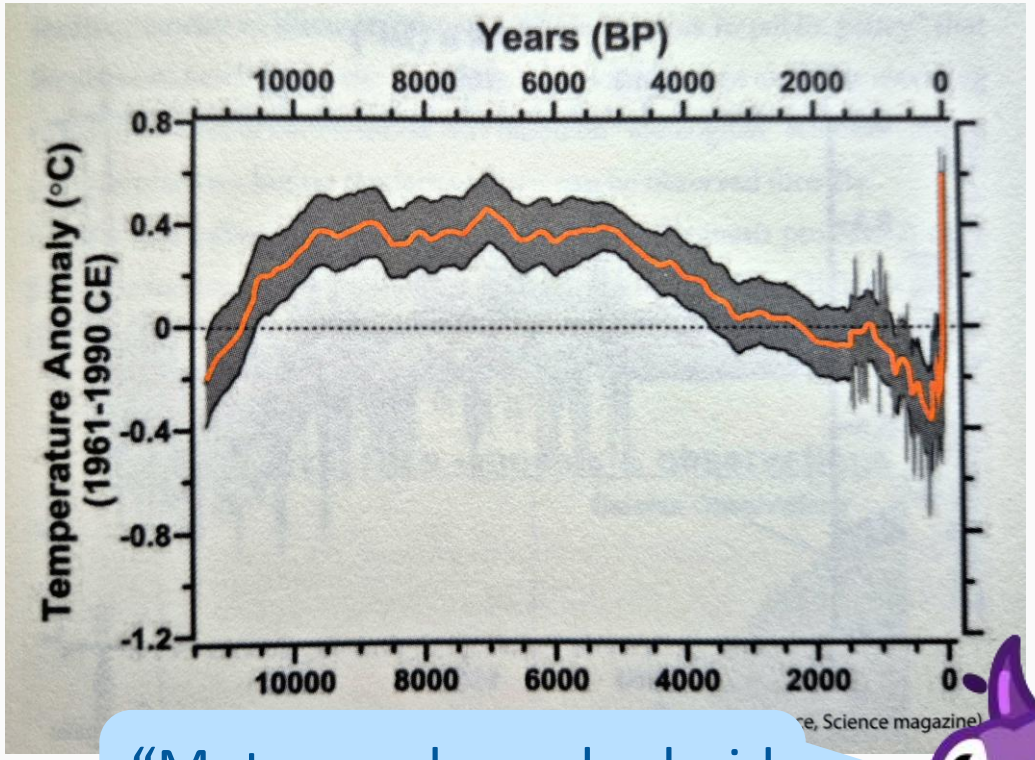



Beter: leg statistische data uit

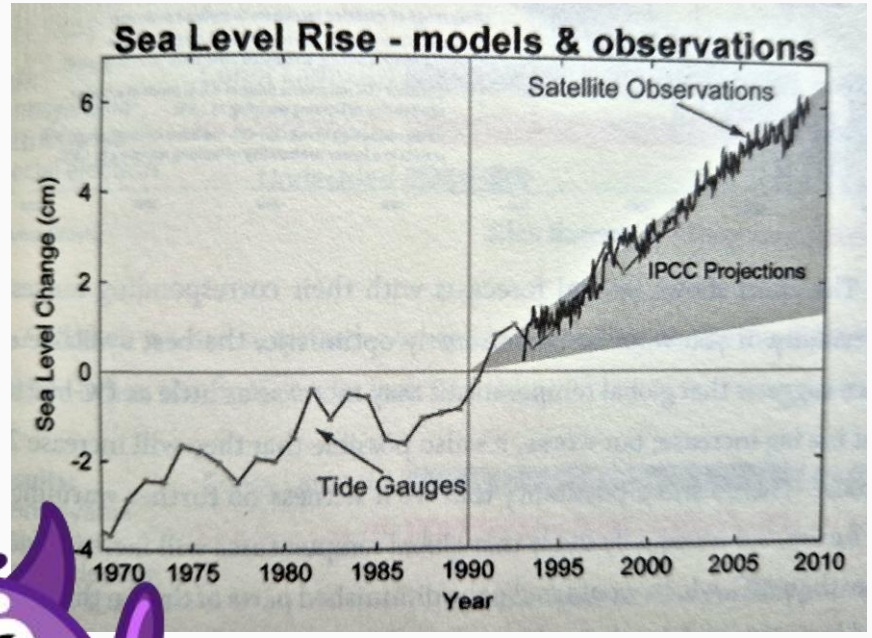


Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 145-150

# Verborgen of onduidelijke onzekerheid




Onzeker ≠ onjuist



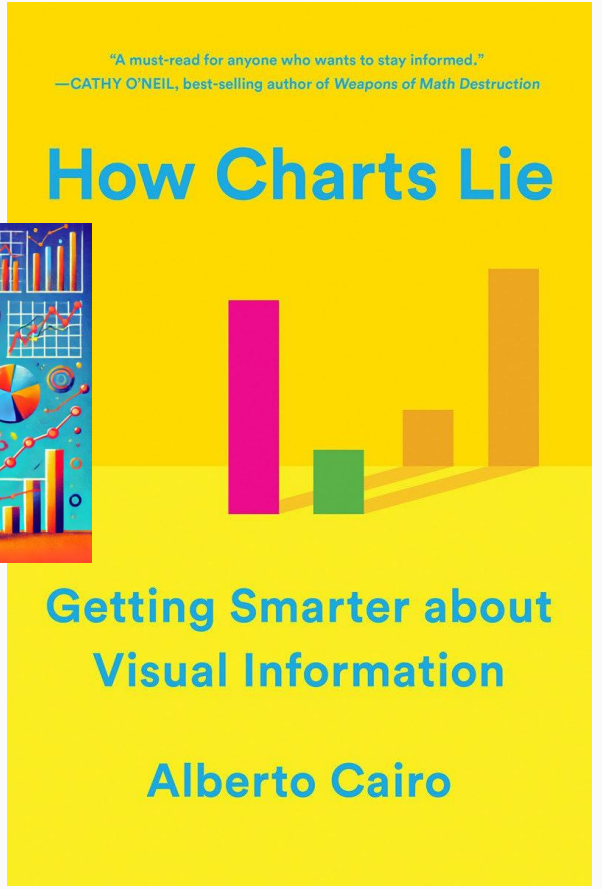

“Met zoveel onzekerheid weet je toch niks?”

Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 145-150

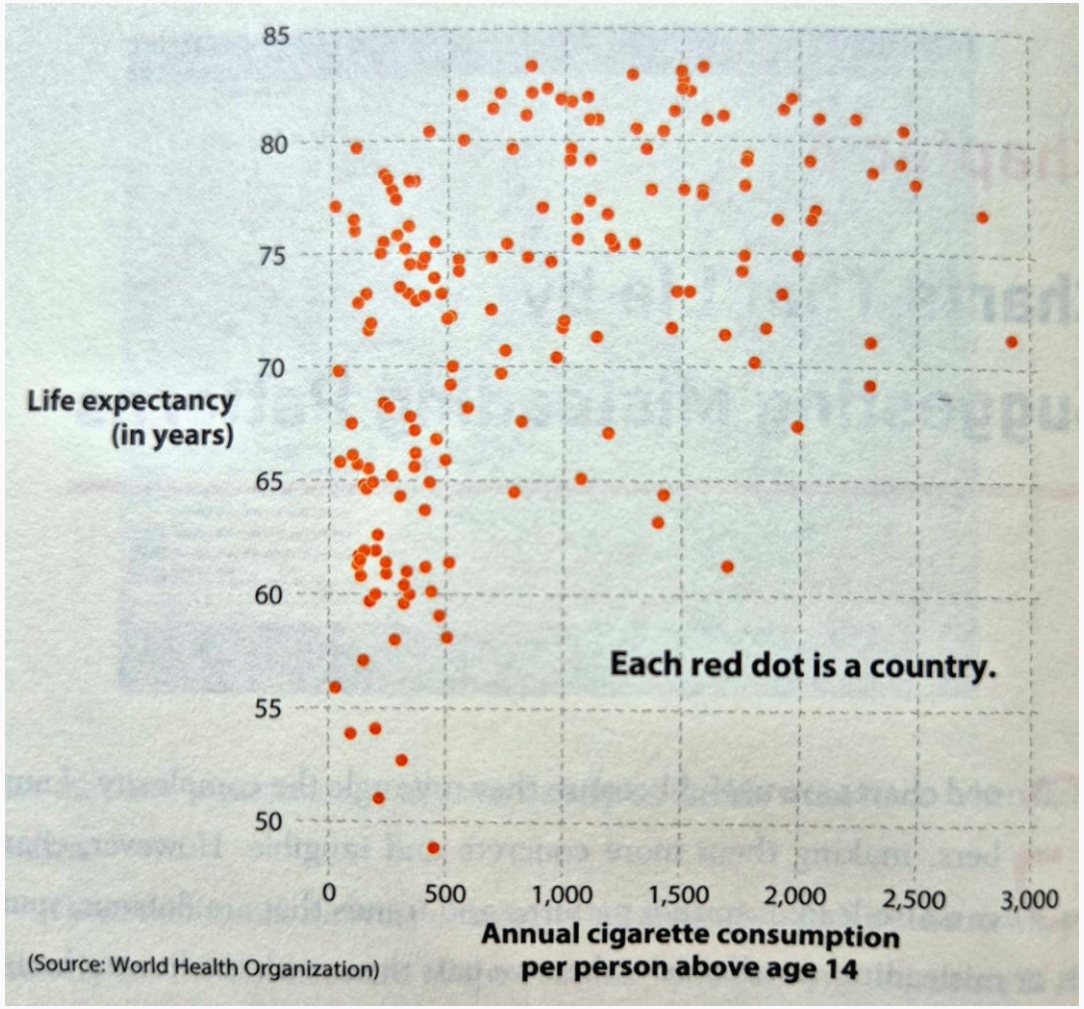
Hoorcollege 3

# Introductie tot ggplot2
# Misleiden met visualisaties

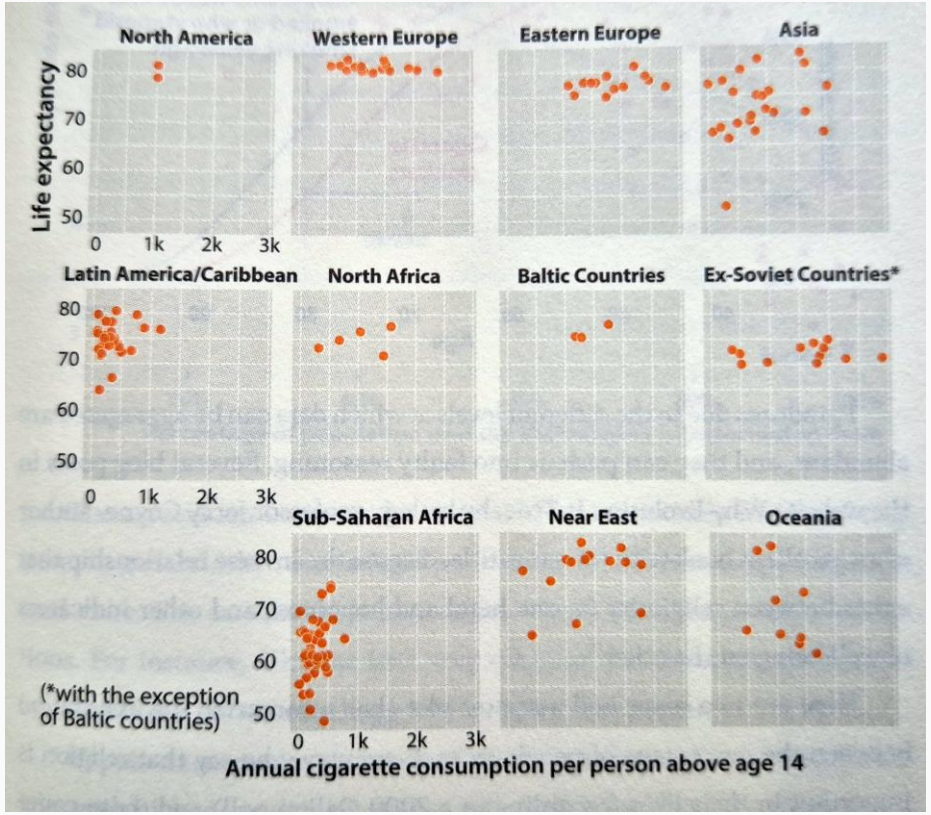
1. Introductie tot ggplot2
2. Misleiden met visualisaties
   a) Slecht ontwerp
   b) Dubieuze data
   c) Te weinig data
   d) Verborgen of onduidelijke onzekerheid
   e) Suggestieve patronen

# Suggestieve patronen



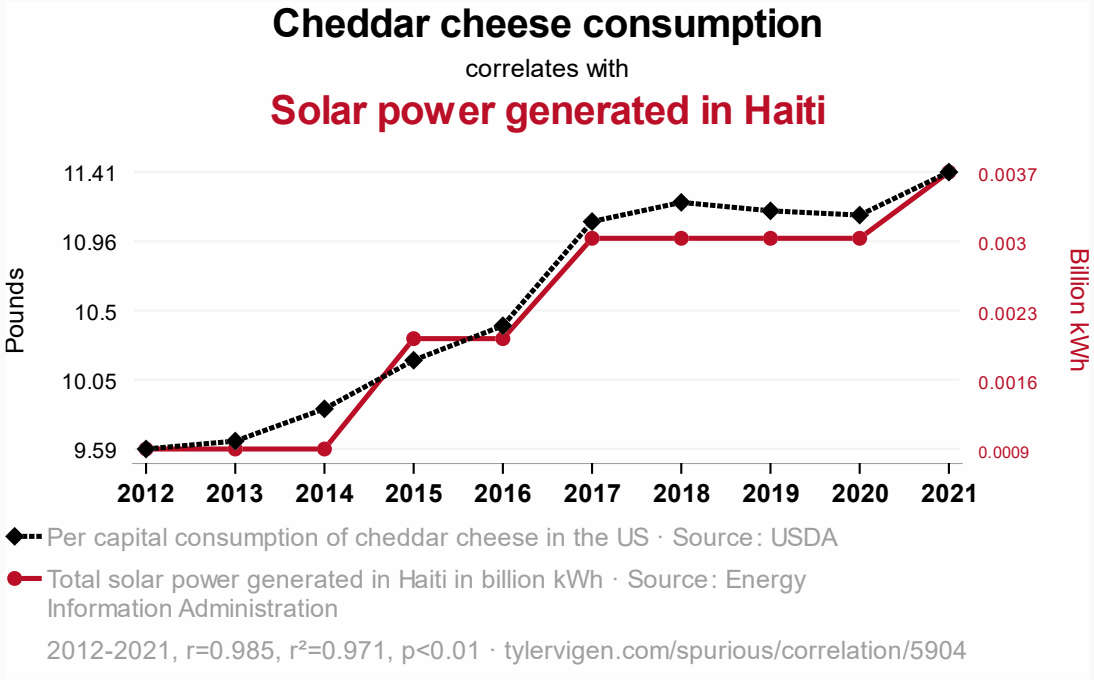

Aggregeren van data toont soms misleidende patronen

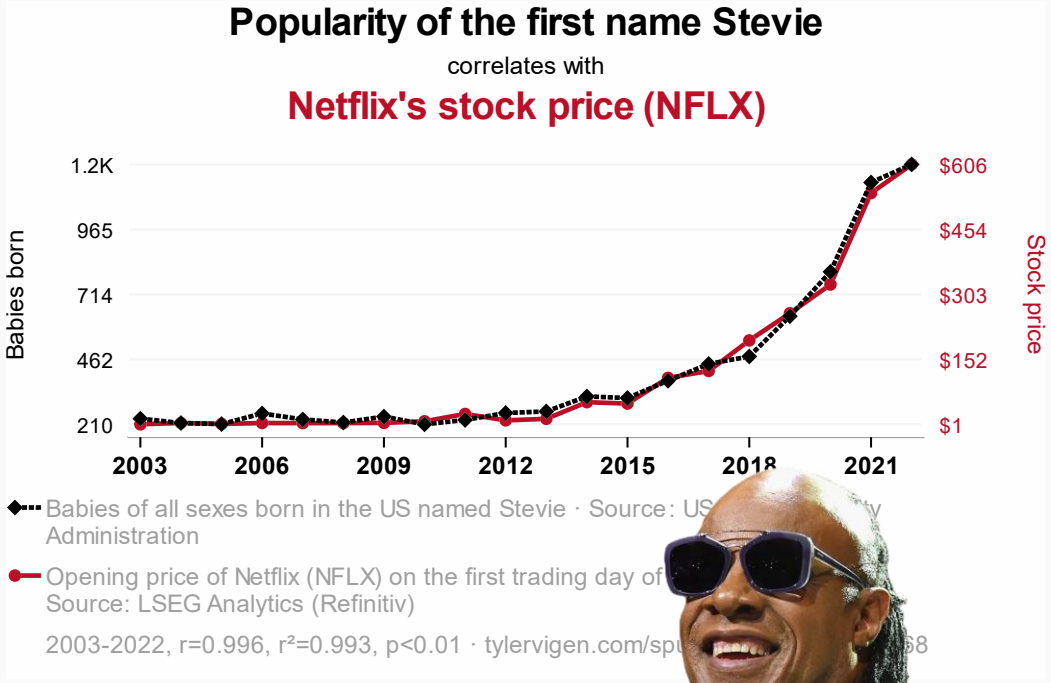





Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 154-157

# Suggestieve patronen



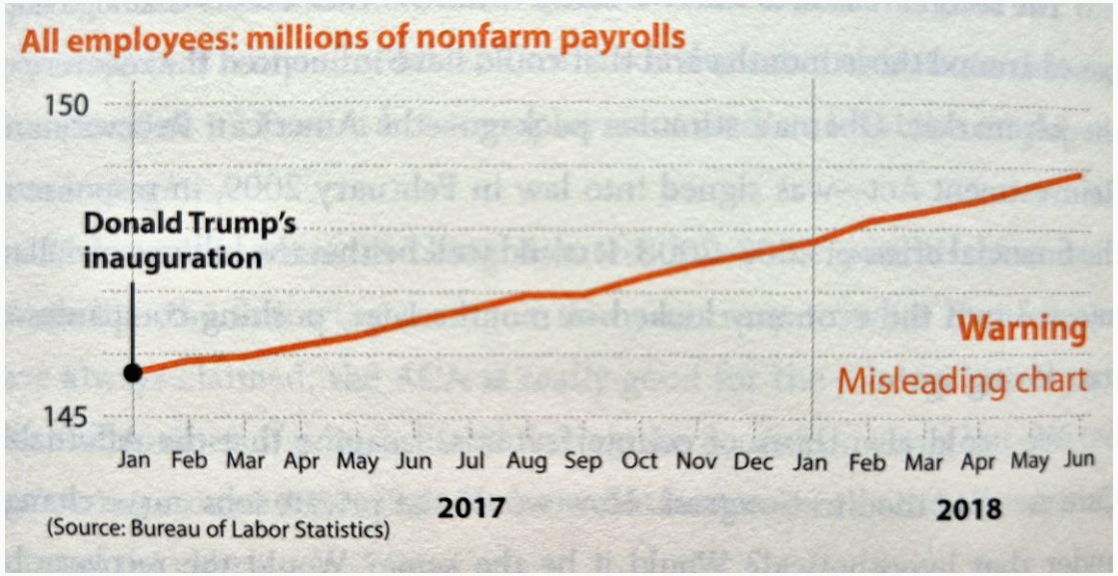



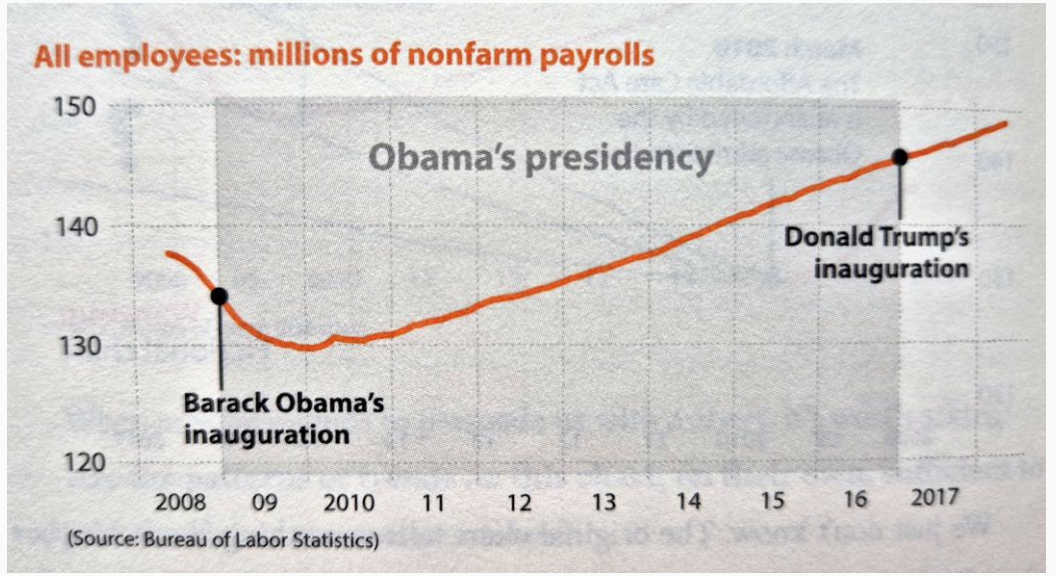

Correlatie ≠ causatie

Het is een Wonder!

https://www.tylervigen.com/spurious-correlations

# Suggestieve patronen



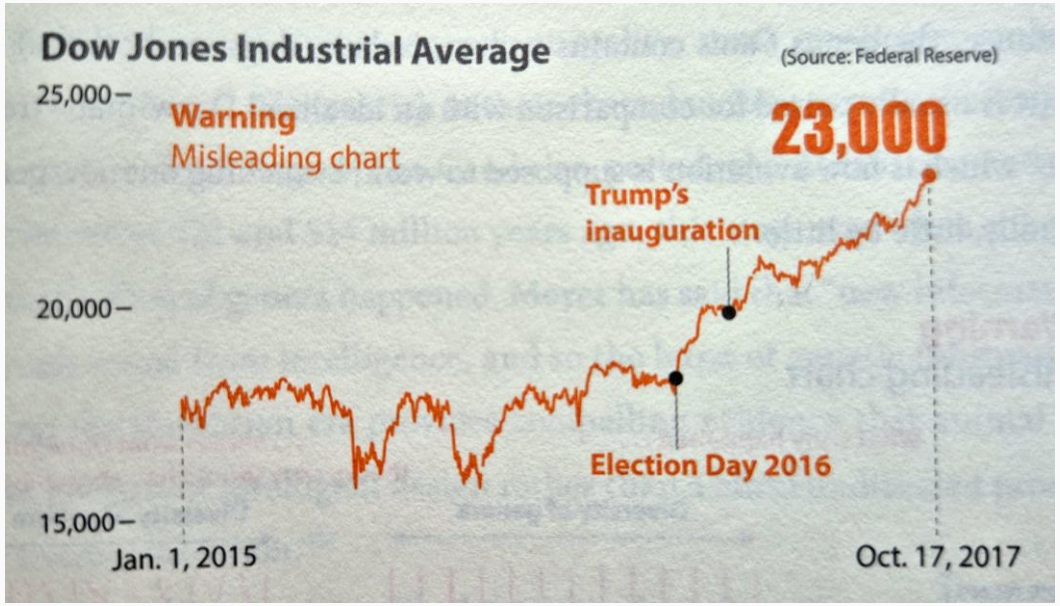

Trend koppelen aan 1 factor die misschien niet causaal is

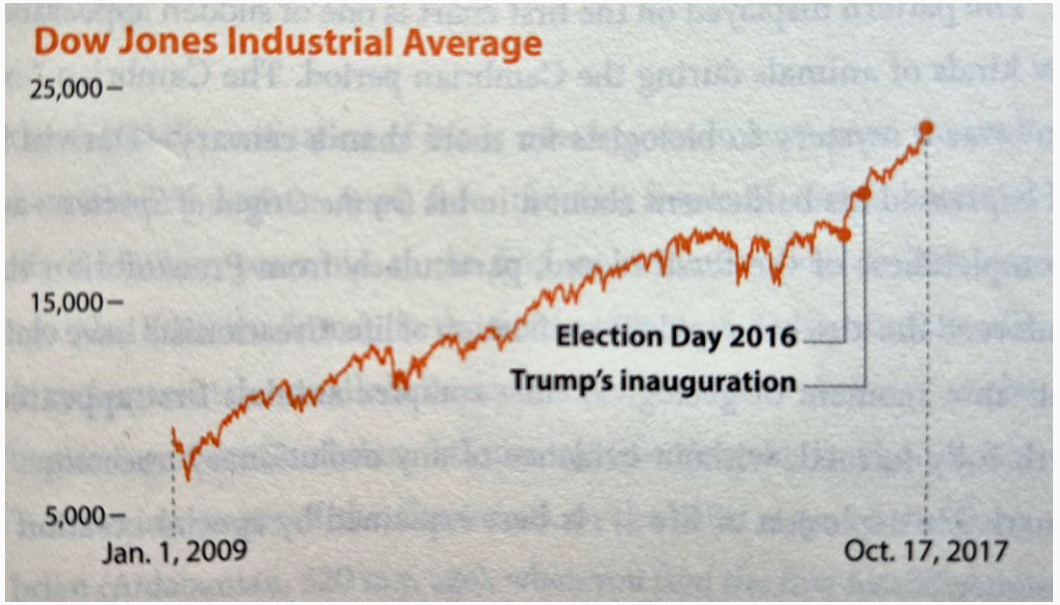
Beter: toon de hele relevante trend



Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 166-169

# Suggestieve patronen



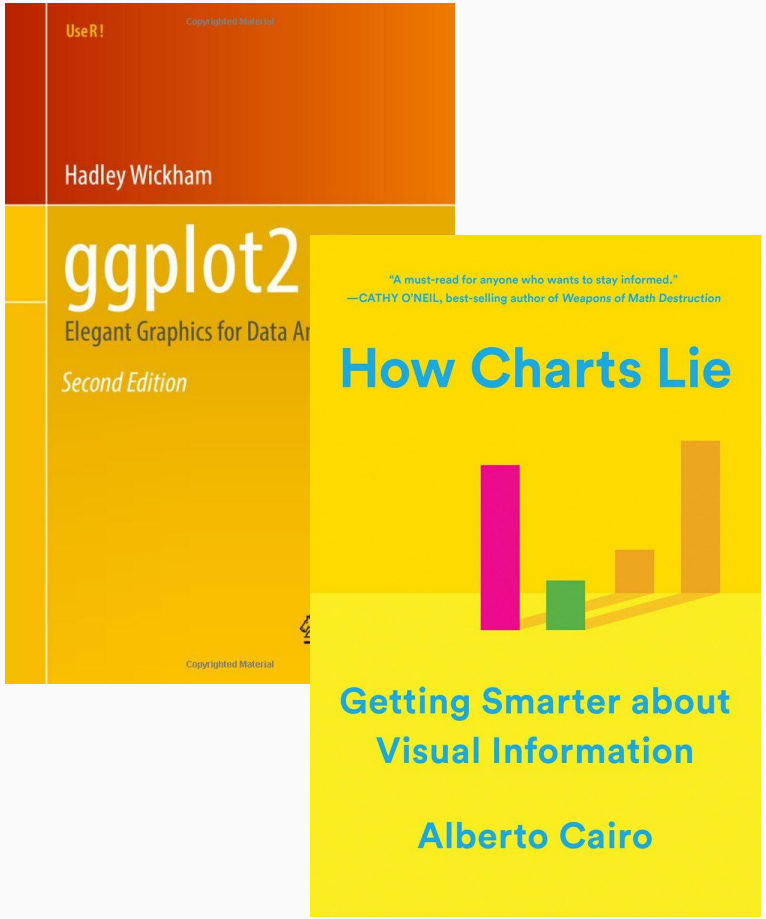

Trend koppelen aan 1 factor die misschien niet causaal is

Beter: toon de hele relevante trend



Alberto Cairo. *How charts lie. Getting smarter about visual information*. Norton, 2019. p. 166-169

# Referenties en credits




**Boek:** Hadley Wickham. *Ggplot2, Elegant graphics for data analysis. Second edition.* Springer, 2015. https://ggplot2-book.org

**Cheatsheet**: https://rstudio.github.io/cheatsheets/html/data-visualization.html

**Voorbeelden**: https://r-graph-gallery.com

**Tutorial**: https://www.cedricscherer.com/2019/08/05/a-ggplot2-tutorial-for-beautiful-plotting-in-r

**Boek:** Alberto Cairo. *How charts lie. Getting smarter about visual information.* Norton, 2019.

# Hoorcollege 3

# Introductie tot ggplot2
# Misleiden met visualisaties


Informatie-uitwisseling: informatievisualisatie

**Jeroen Ooge**

Universiteit Utrecht