Fundamentals of R

Principles and Practices of Data Visualization

Henrique Sposito and Livio Silva-Muller

INSTITUT DE HAUTES
ÉTUDES INTERNATIONALES
ET DU DÉVELOPPEMENT

GRADUATE INSTITUTE
OF INTERNATIONAL AND
DEVELOPMENT STUDIES

What did we learn last week?

Conceptually:

- Tidy datasets
- Wrangling data
- Functional Programming

Practically:

```
# %>% the pipe operator
separate()
unite()
pivot_long()
pivot_wide()
select()
filter()
mutate()
group_by()
summarize()
count()
left_join() #family of functions to join datasets
```

Lecture:

- Principles of good visualization.
- Grammar of Graphs (ggplot2).
- Basic plots.

Practical:

- Bar, line, scatter, and box plots.
- How to size, color, shape and label things.
- How to facet plots.

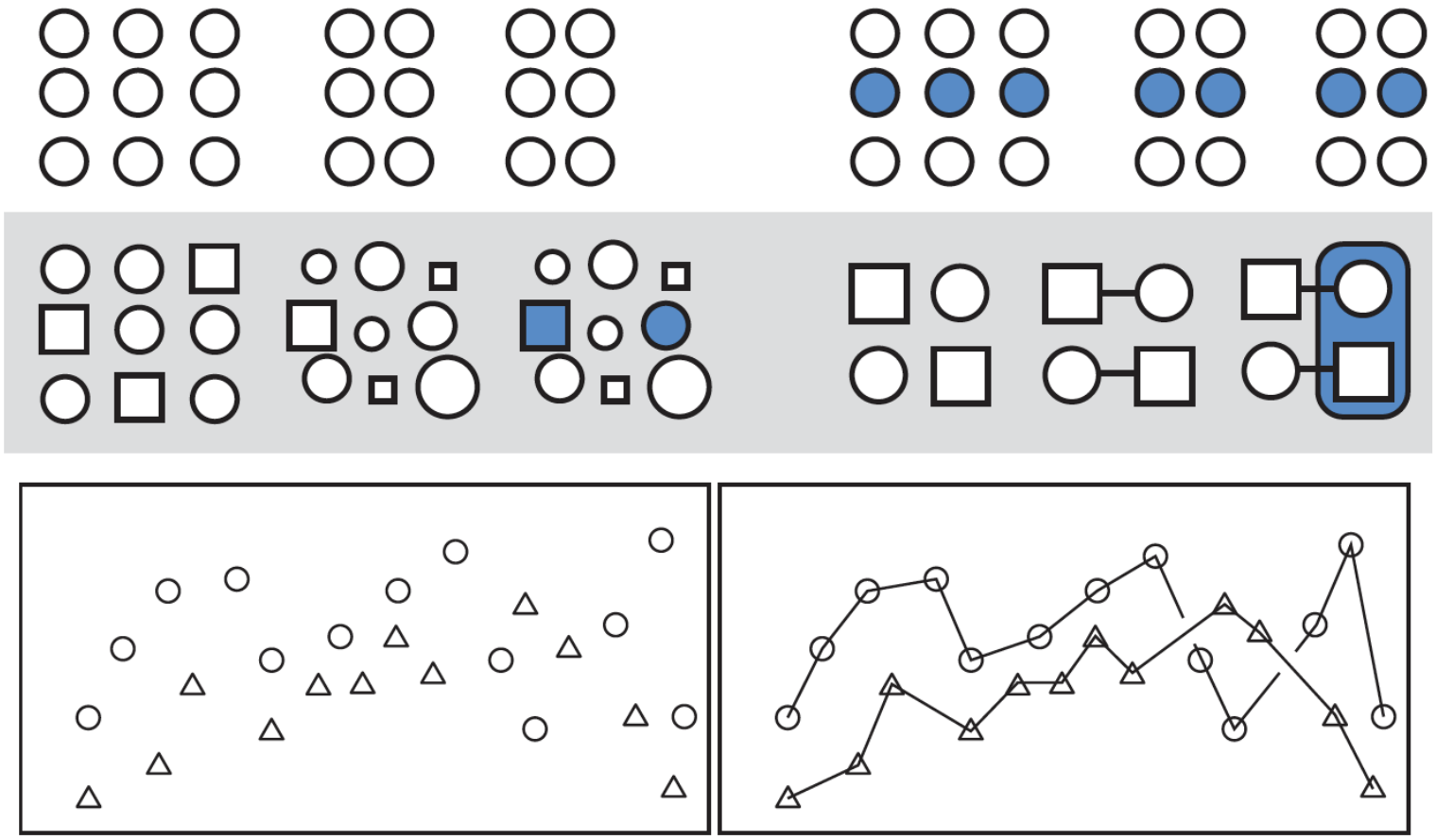
What makes good visualization?

- *Gestalt Principles*: sizes, colors, and shapes can structure a plot.
- *Tufte's Principle*: too much visual information makes comprehension more difficult.

In other words:

- Leverage size, color, and shapes to...
- ...decrease unnecessary information,
- **So one plot, tells one story clearly**.

Gestalt principles (1/6)



Gestalt principles (2/6)

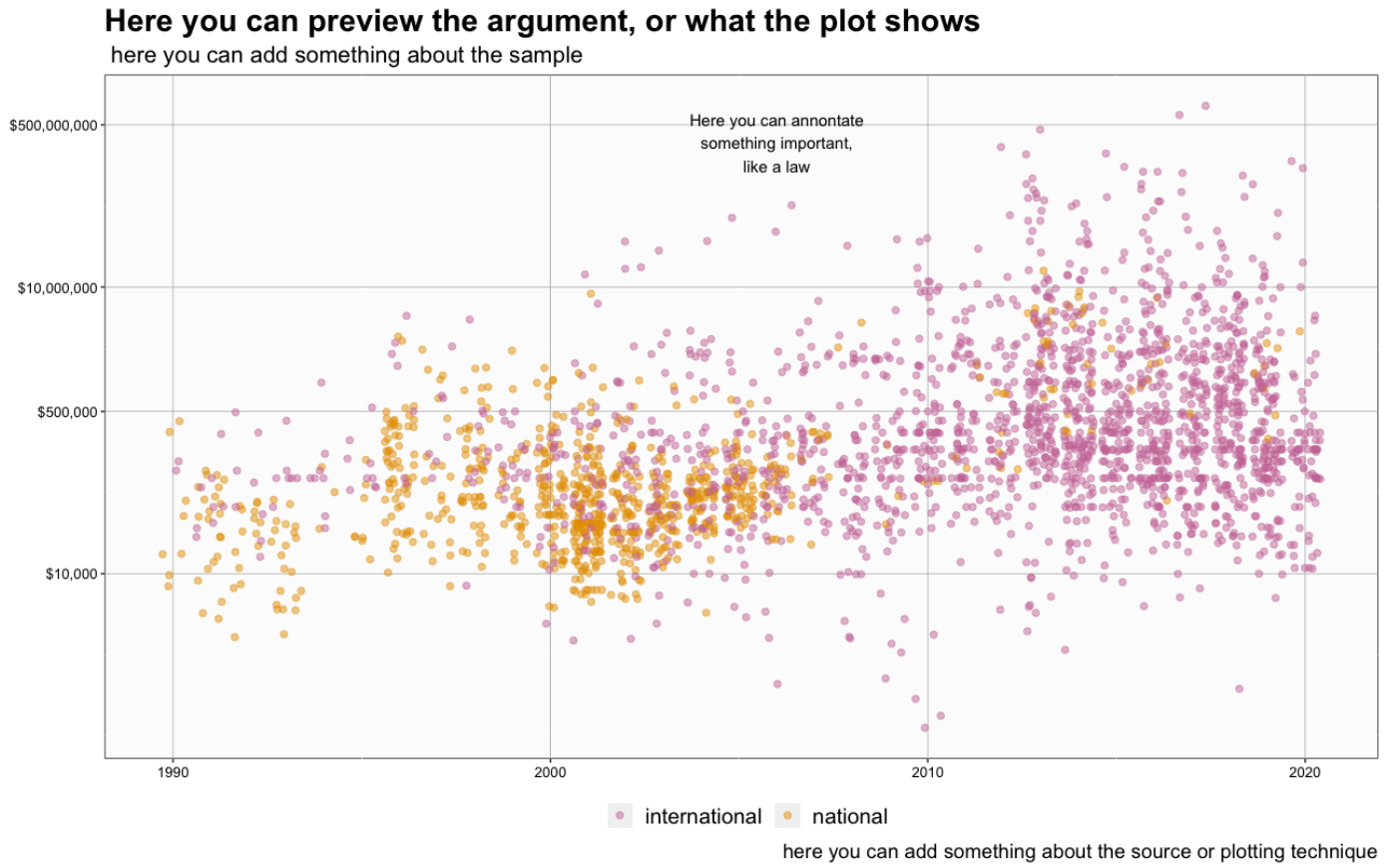
- We look for structure... and we infer relationships from structure!
  - **Proximity**: things that are spatially near to one another seem related.
  - **Similarity**: things that look alike seem to be related.
  - **Connections**: things that are connected seem to be related.

Image from and text adapted from: Healy, 2018

Gestalt principles: sizing things (3/6)

Gestalt principles: sizing things (4/6)

- Size, more than order, will define what is noticed first.
- Size things around and in your plot by their hierarchy of importance.



Gestalt principles: shapping and colouring things (5/6)

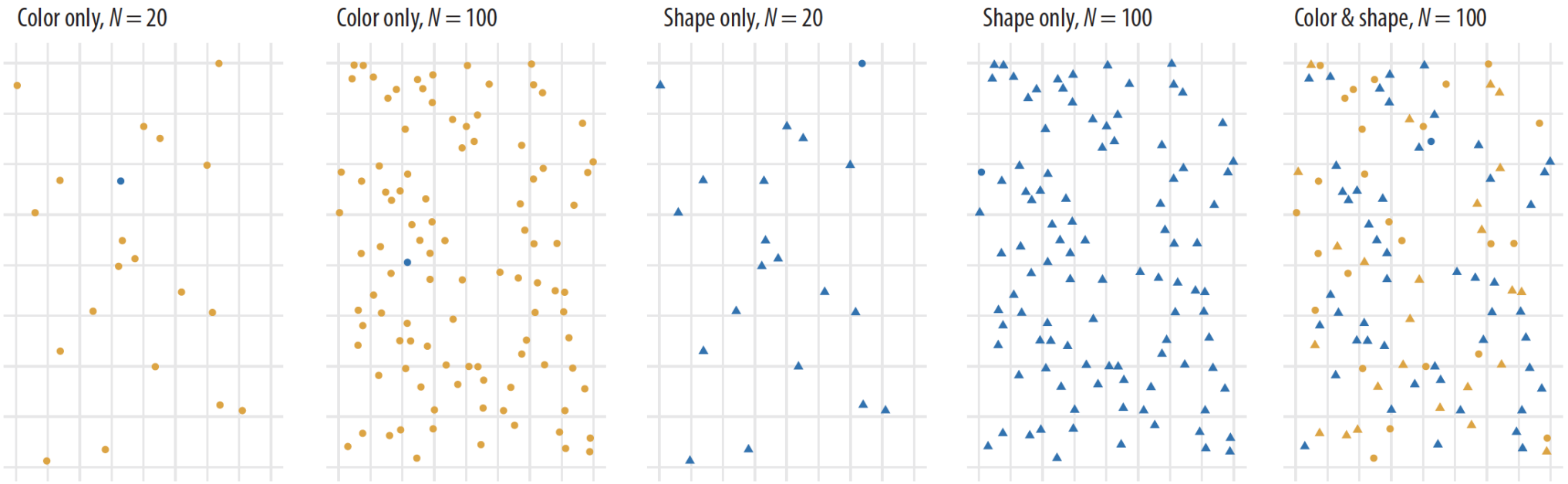


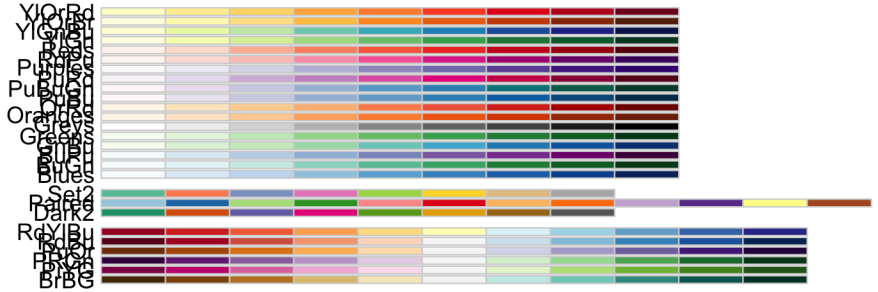
Figure 1.18: Searching for the blue circle becomes progressively harder.

Image from: Kieran Healy

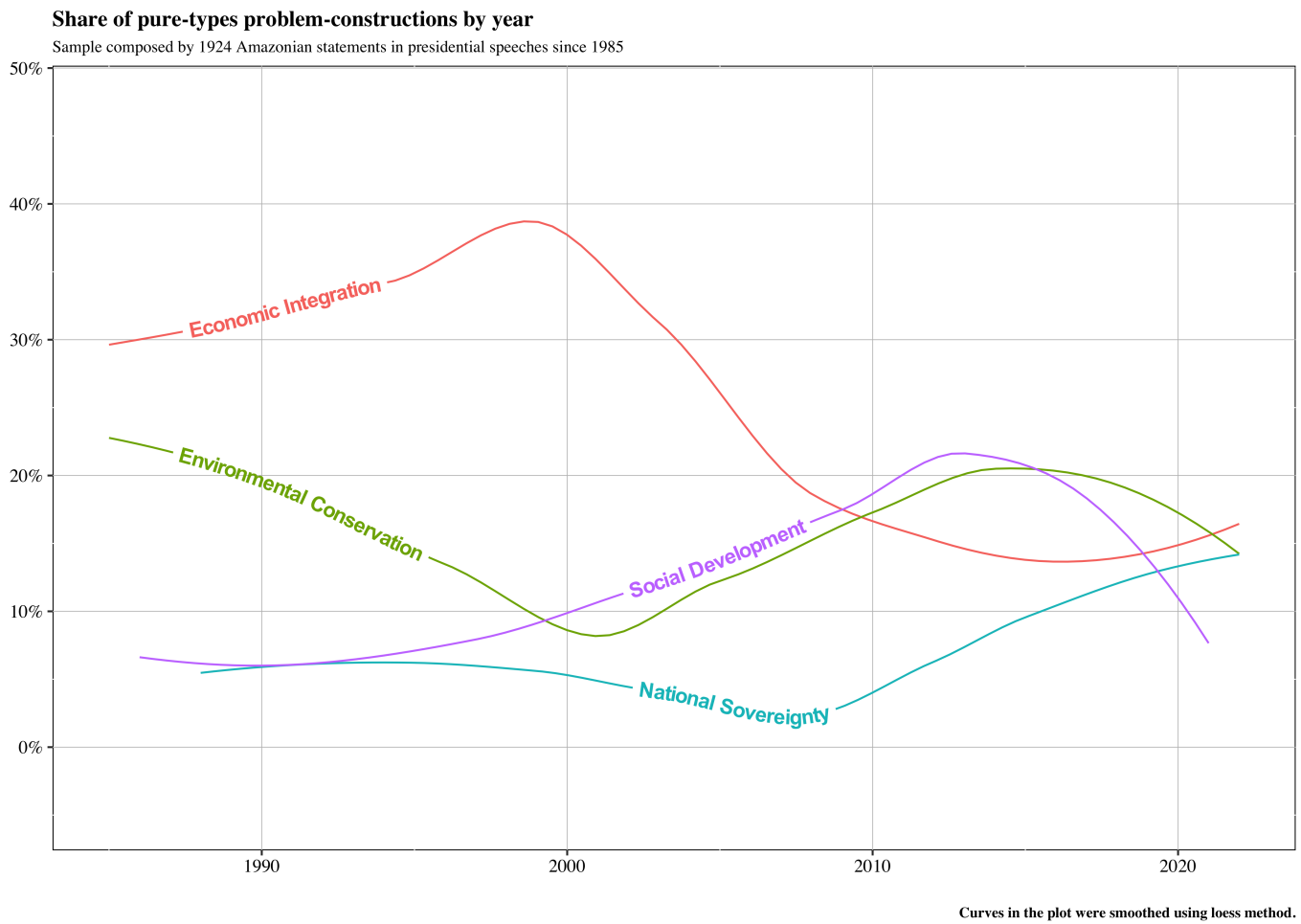
Gestalt principles: shapping and colouring things (6/6)

- Shape or color are good when you want to show contrast.
- Both together are often too much.
  - If you have color, shape, and two axis you are plotting 4 variables!
- Choose colors based on
  - the relationship: continuous or discrete?
  - intution: red for high temperatures, blue for low.
- *Remember: colorblindness is more frequent than you think!*

```
library(RColorBrewer)
display.brewer.all(colorblindFriendly = TRUE)
```

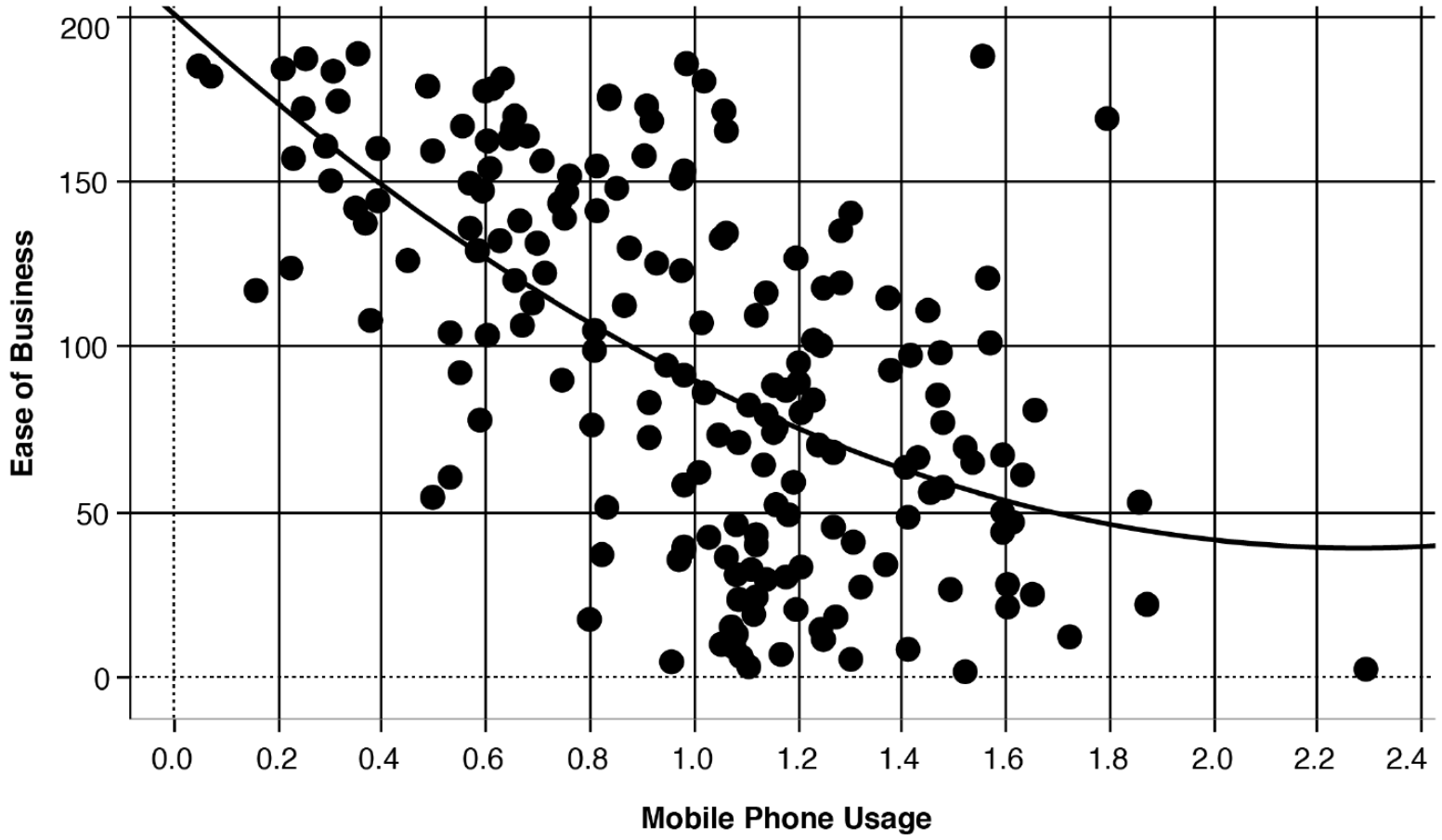


In line legend with `geom_textlabel()`

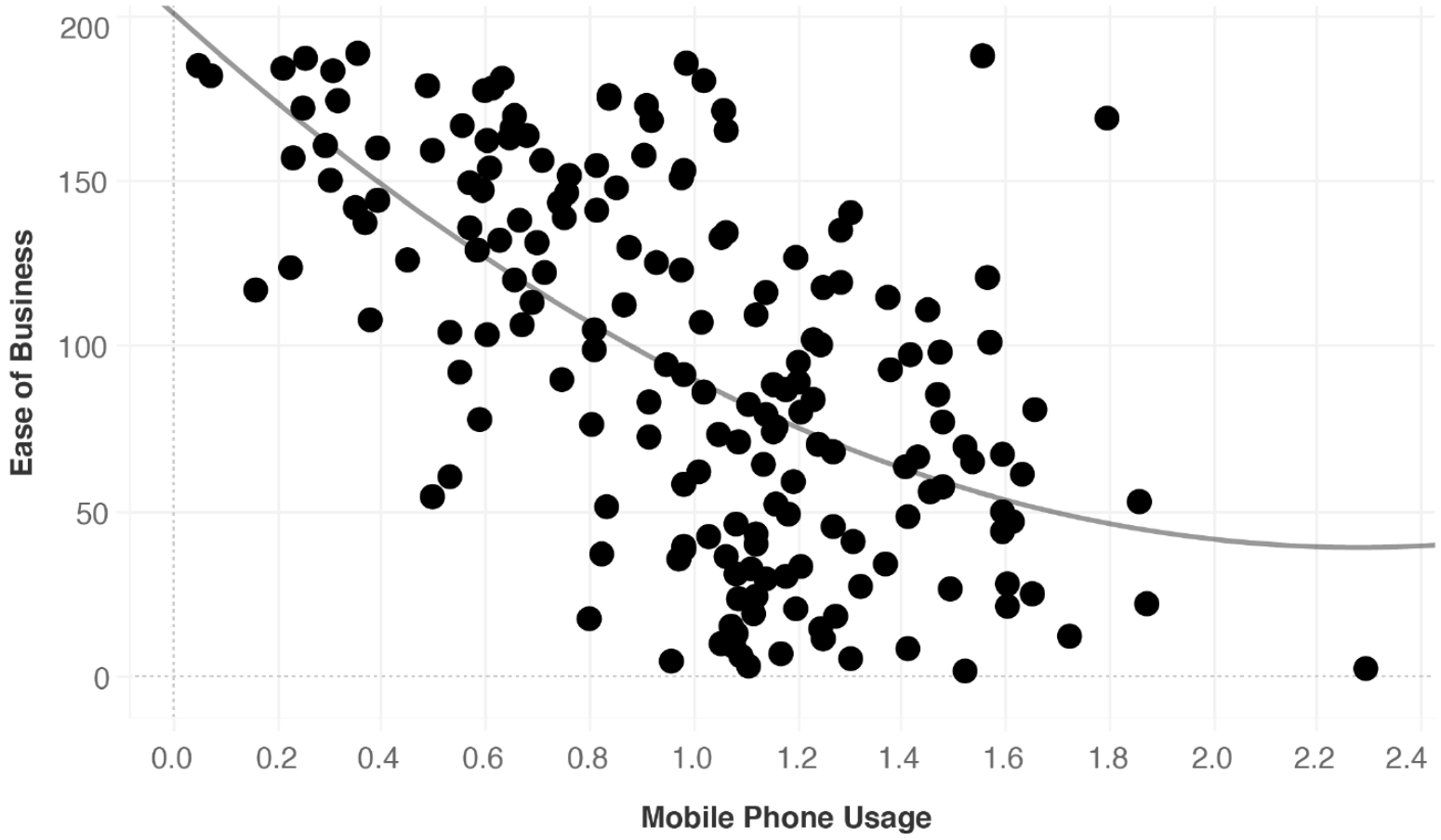


Check out: GeomTextPath

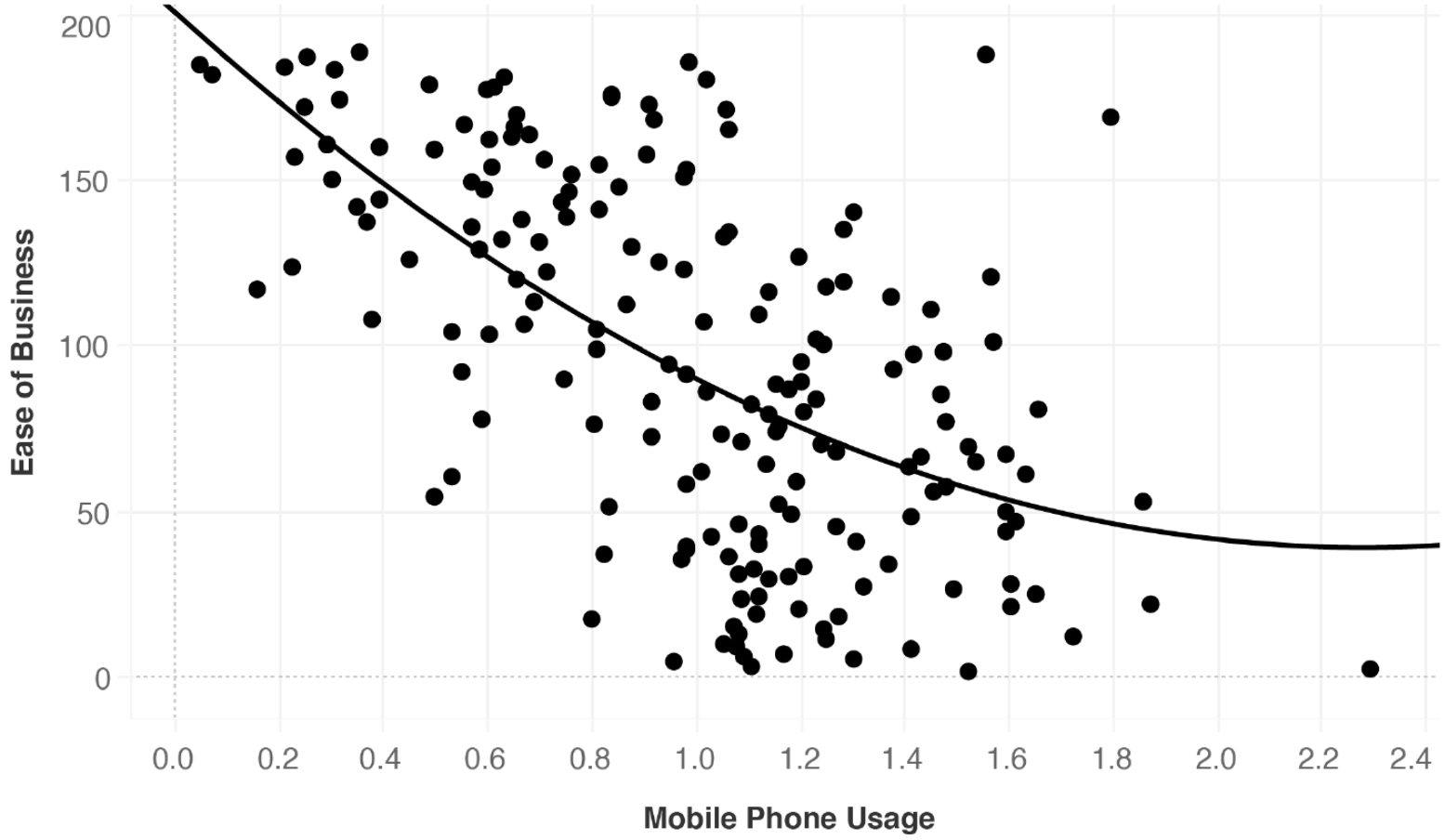
Tufte's Law: Reduce data-to-ink ratio



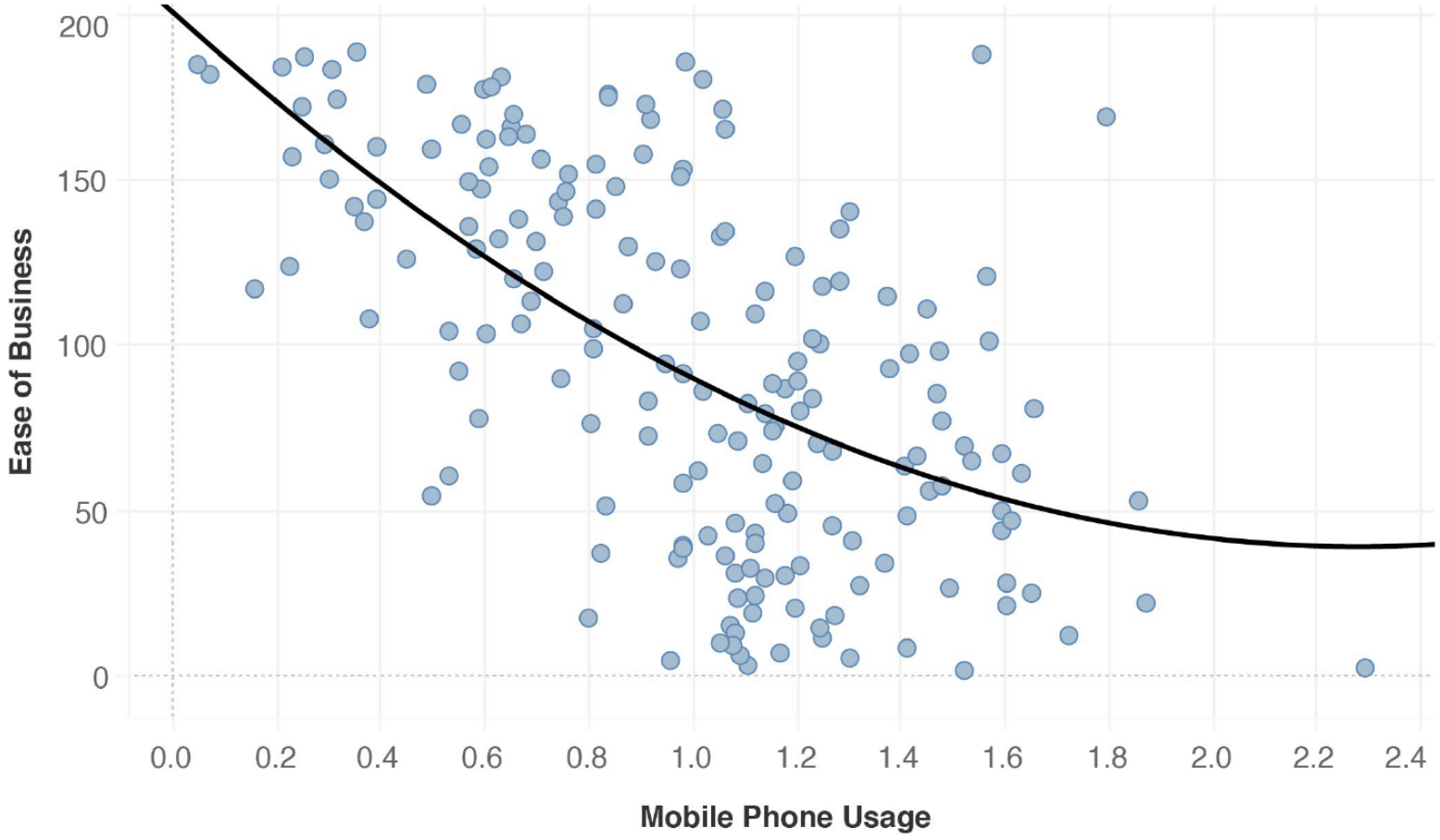
Tufte's Law: Reduce data-to-ink ratio



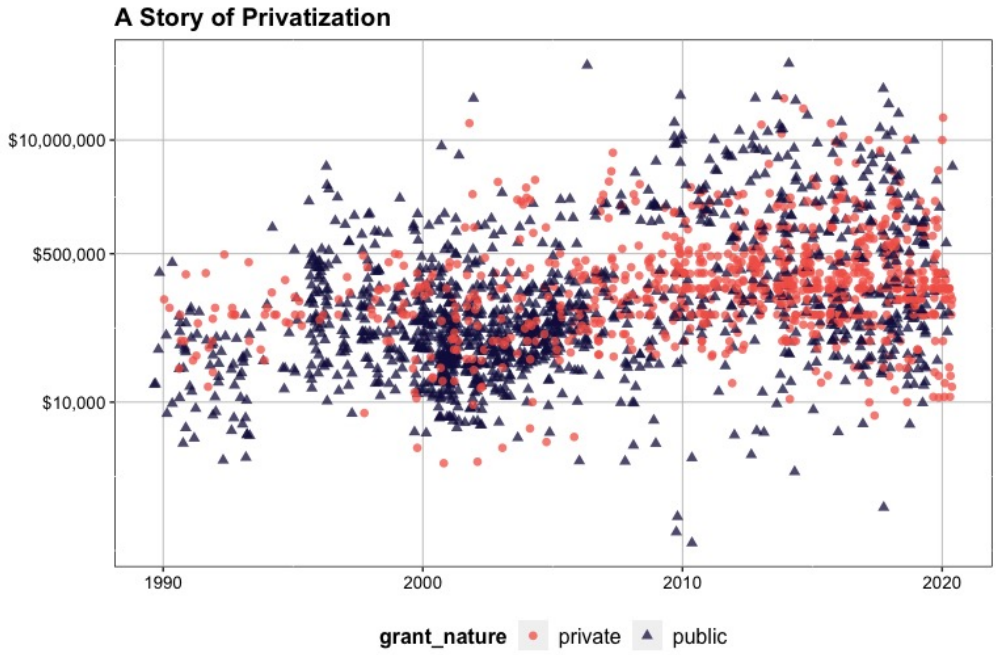
Tufte's Law: Reduce data-to-ink ratio

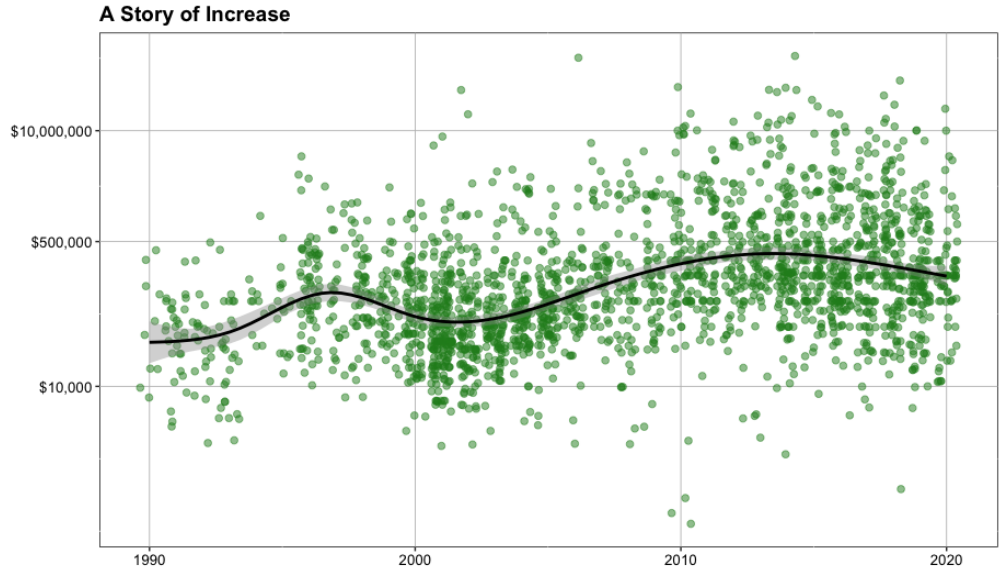


Tufte's Law: Reduce data-to-ink ratio



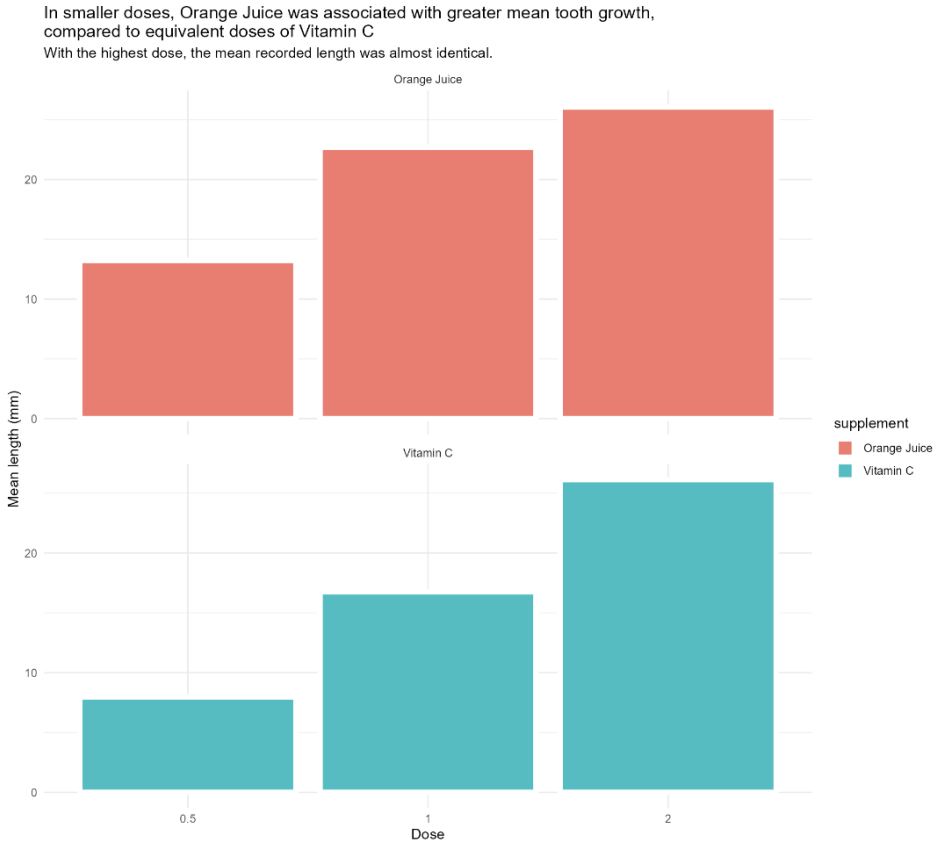
One plot, one clear story!

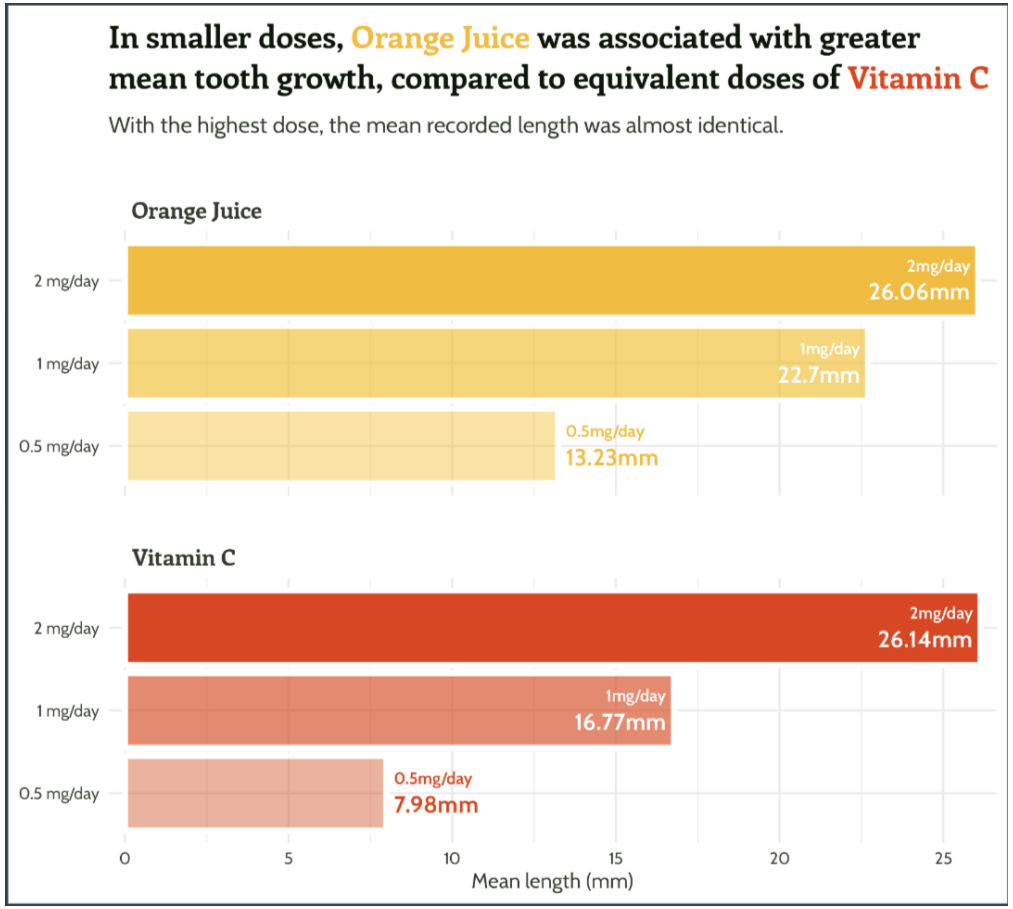




- Overlap shape and color to the same variable to maximize results!

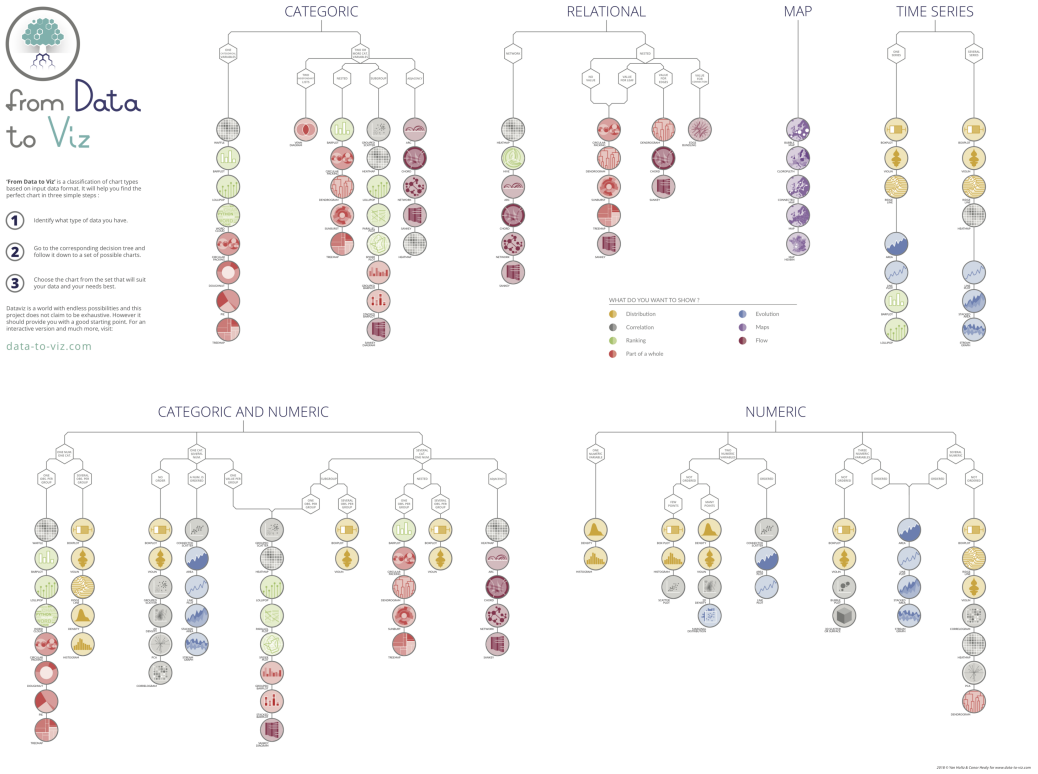
You can always bring it to the next level!





Check: Cara Thompson

Okay, but how do I know which plot is the best plot?



From: Data-to-Viz

How do I plot?

We will use a package called the ´{ggplot2}´

> ggplot2 is a system for declaratively creating graphics, based on **The Grammar of Graphics**. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

- Map variables to aesthetics: linking data do things you see.
- Graphical primitives: type of graph you want (line, box, scatter...)

All slides below are highly inspired from Healy, 2018 check out his blog and book!

ggplot2: a general workflow (1/5)

- Tell ggplot2 **what we want to see**: the data and the variables.
  - What are our axis?
  - Do we want colors?
  - Do we want sizes?
- Tell ggplot2 **how you want to see it**: the types of graphs (points, lines, boxes)
  - What type of graph we want?
  - Do we want more graphs on it?
  - Do we want to add some summary statistics?
- Tell ggplot2 **how to improve what is there**: scales, labels, titles, marks and so forth.
  - Do we want titles, sub-titles, captions?
  - Are the scales meaningful?
  - Are legends and marks explanatory?

the data for the next few slides

```r
library(gapminder) #downloading the package with the data
library(dplyr) # loading dplyr
data("gapminder") # loading the data from the package
sample_n(gapminder,15) #showing a random sample
```

```
#> # A tibble: 15 × 6
#>    country               continent  year lifeExp      pop gdpPercap
#>    <fct>                 <fct>     <int>   <dbl>    <int>     <dbl>
#>  1 Togo                  Africa     1987    56.9  3154264     1202.
#>  2 Argentina             Americas   1977    68.5 26983828    10079.
#>  3 Yemen, Rep.           Asia       2007    62.7 22211743     2281.
#>  4 Panama                Americas   1992    72.5  2484997     6619.
#>  5 Benin                 Africa     1992    53.9  4981671     1191.
#>  6 Sao Tome and Principe Africa     2002    64.3   170372     1353.
#>  7 Lesotho               Africa     1987    57.2  1599200      774.
#>  8 Trinidad and Tobago   Americas   1977    68.3  1039009     7900.
#>  9 Germany               Europe     1982    73.8 78335266    22032.
#> 10 Taiwan                Asia       1997    75.2 21628605    20207.
#> 11 Equatorial Guinea     Africa     1952    34.5   216964      376.
#> 12 Botswana              Africa     2007    50.7  1639131    12570.
#> 13 Cambodia              Asia       1962    43.4  6083619      497.
#> 14 Bulgaria              Europe     1962    69.5  8012946     4254.
#> 15 Botswana              Africa     1952    47.6   442308      851.
```

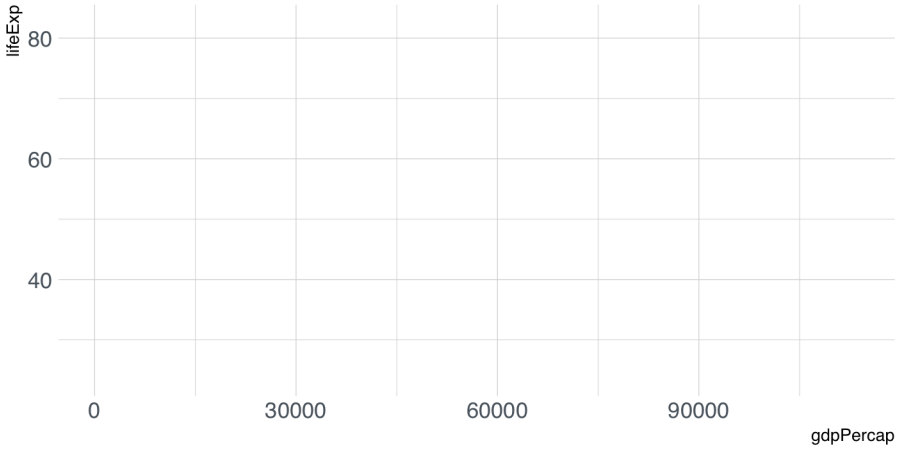
`

ggplot2: what we want to see (2/5)

First, let's tell ggplot what data we have, and how to map variables to aesthetics.

- Our data is gapminder.
- Our x will be GDP per capita, our y life expectancy.

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

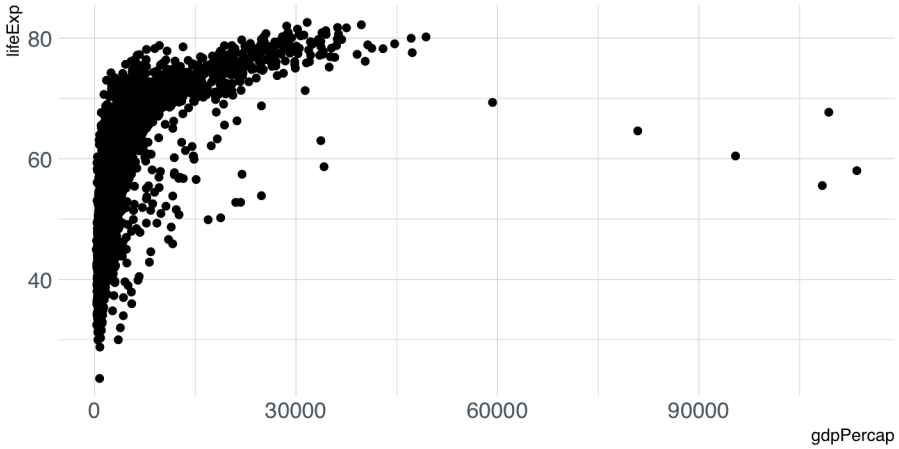


ggplot2: how we want to see (3/5)

Second, let's tell ggplot2 how we want to see the relationships.

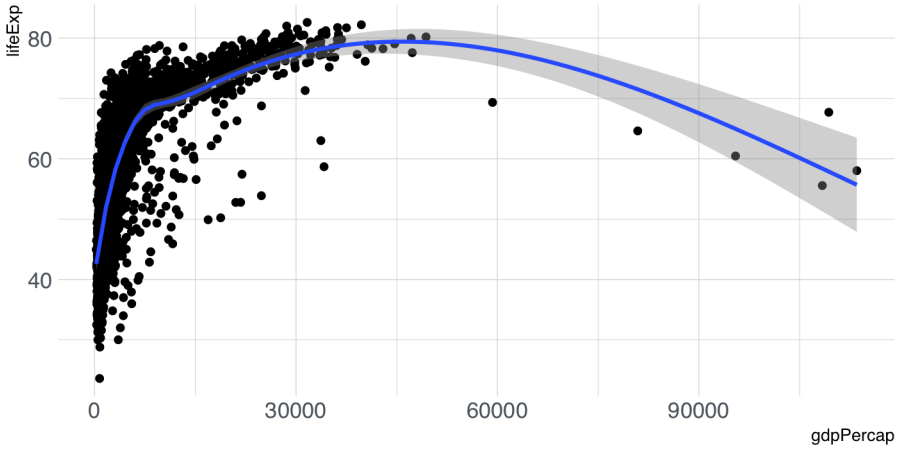
- `geom_point()` inherits the x and y as we specified inside the `ggplot()`, and draws points for observations.
- We can just **add** the geom to the object p we created.
- In ggplot2 we add things, because we are **layering** rather than piping.

```
p ← p + geom_point()
```



ggplot2: how we want to see (4/5)

- `geom_()` is a family of functions that built various types of graphs (bar, lines, boxes)
- There are multiple kinds of geoms, you can check them in the cheat sheet: ´{ggplot2}´
- Following the logic of layering, you can just add a new geom to the object we have.
- In this case, we are adding `geom_smooth()`.



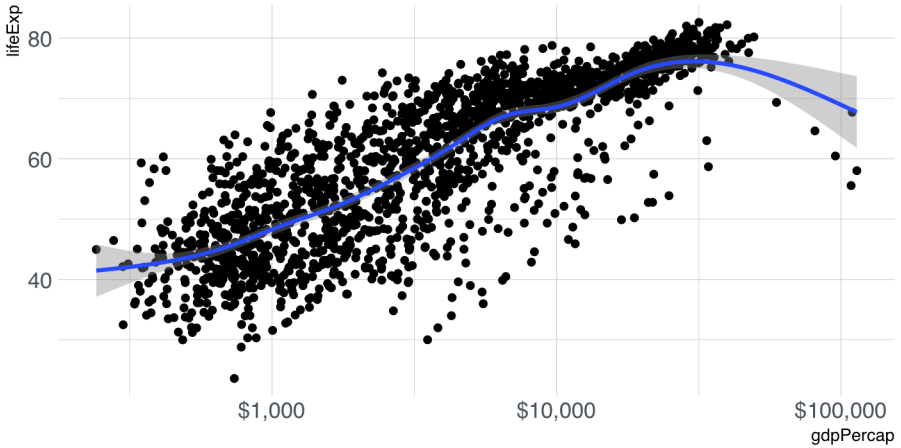

```
p← p + geom_smooth() #smooth plots a trend line.
```

`

ggplot2: how to improve what we see (5/5)

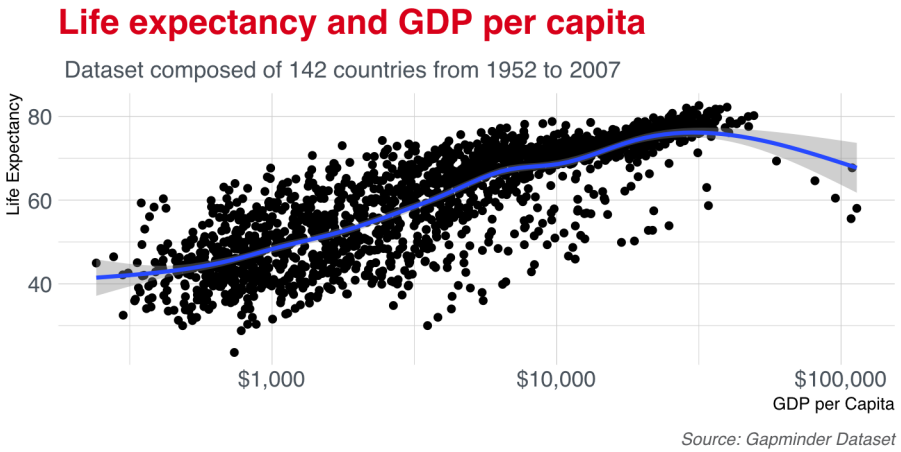
- Third (and finally), we improve what we see.
  - We can scale x in log because the range of gdpPercap is too big: `scale_x_log10()`.
  - Inside the `scale_()` is also a familiy of various functions doing different things.
  - Inside `scale_x_log10()`, we can specify the unit of measurement as dollars.

```
p← p +
  scale_x_log10(labels=scales::dollar)
# scale_x_log10() rescales our x to improve
visualization
```



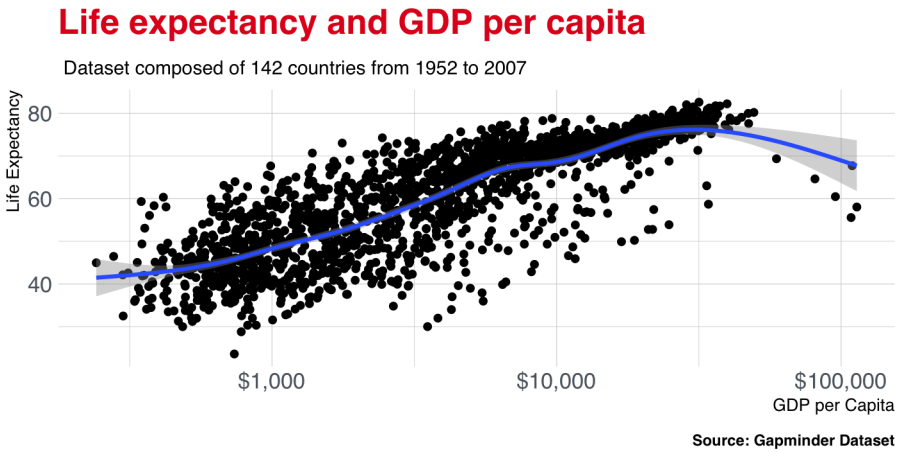
We can add titles, subtitles, axis-titles and caption with `labs()`: read at home

```
p← p +
    labs( y="Life Expectancy", x=" GDP per
Capita",
        title= "Life expectancy and GDP per
capita",
        subtitle=" Dataset composed of 142
countries from 1952 to 2007",
        caption= "Source: Gapminder Dataset")
# labs add axis titles, titles, subtitles, and
captions.
```

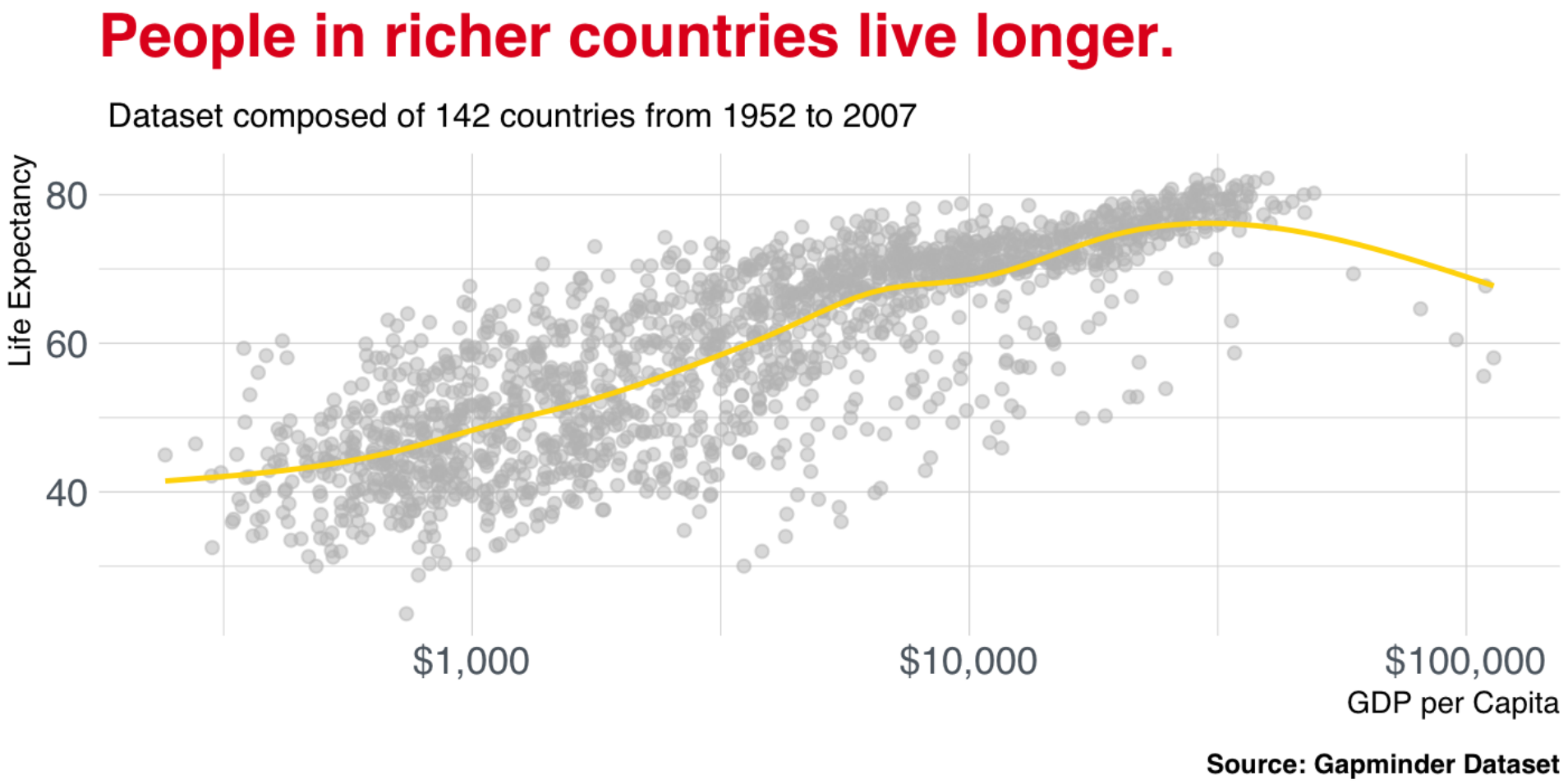


We can change sizes, font, and face with `theme()`: read at home

```
p← p +
      theme(title =
element_text(color="black", size=12,
face="bold"),
        axis.text =
element_text(color="black", size=10),
        plot.subtitle =
element_text(color="black", size=10, face=
"plain"),
        plot.caption =
element_text(color="black", size=8,
face="bold"))
  #theme() resizes, changes face, font and
many others.
```

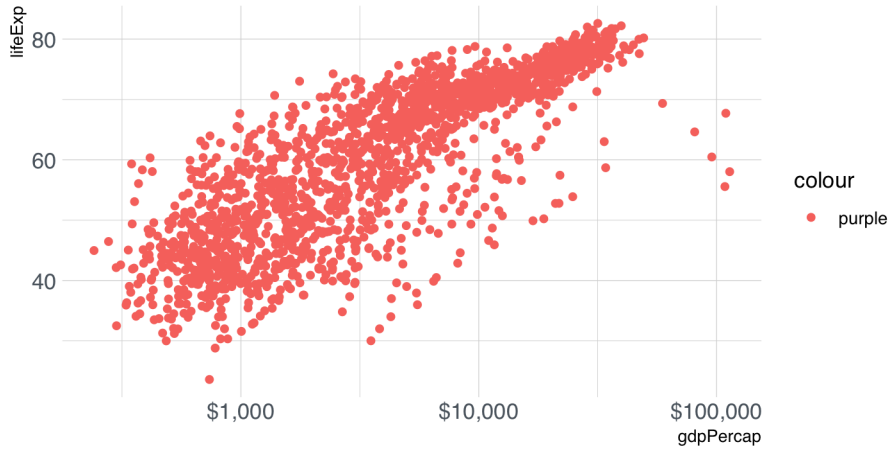


with a few details,we get a nicer plot: read at home



Mapping aesthetics vs setting aesthetics (1/3)

```
purp ← ggplot(data = gapminder, mapping =
aes(x = gdpPercap, y = lifeExp,
color = "purple"))+
  geom_point()+
  scale_x_log10(labels=scales::dollar)
```



-Color is a possible aesthetic in ggplot2; it expects a variable.

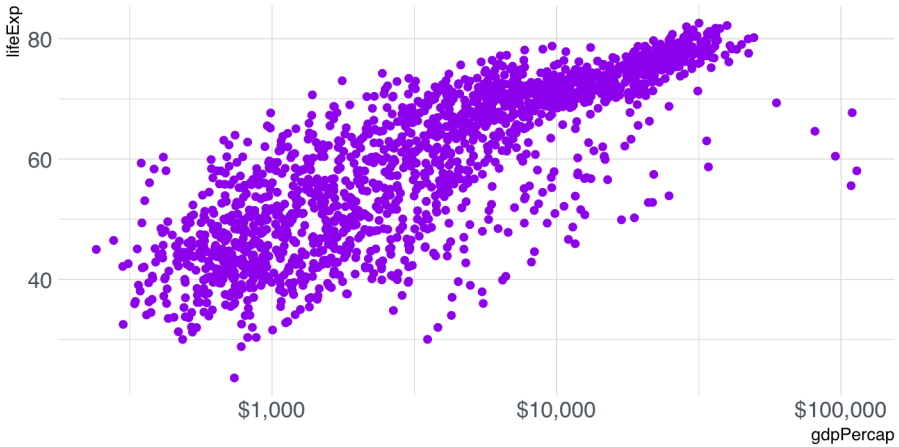
-The code is creating a new variable with value "purple" for observations.

-All observation have the same value ("purple") for color, so they take the same color.

Mapping aesthetics vs setting aesthetics (2/3)

- If we want the points to be purple, we need to **set** them as purple.
- **Setting** is different than mapping to an aesthetic, and it is done *outside* the `ggplot(mapping=aes())`.
- You **set** at the `geom()` level.

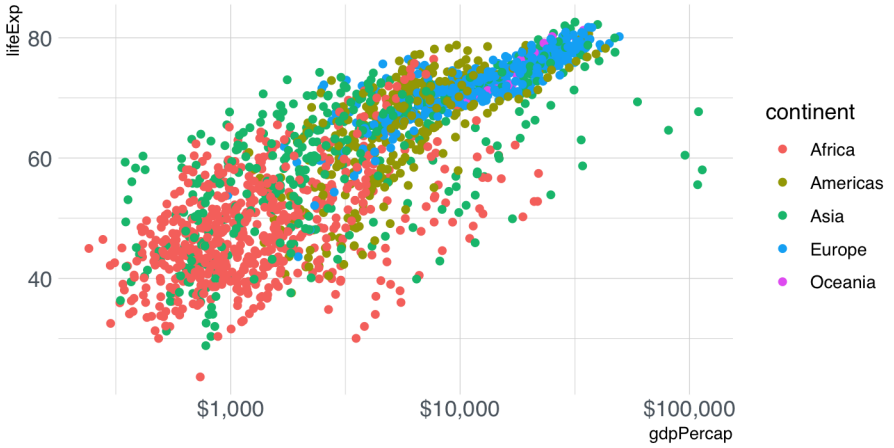
```
purp2←ggplot(data = gapminder, mapping =
aes(x = gdpPercap, y = lifeExp))+
geom_point(color="purple")+
scale_x_log10(labels=scales::dollar)
```



Mapping aesthetics vs setting aesthetics (3/3)

- If we want the points to be colored by a variable, we need map the variable to an aesthetics.
- This means: `ggplot(mapping=aes(color=continent))`.
- This reads: map the variable continent to the aesthetic color.

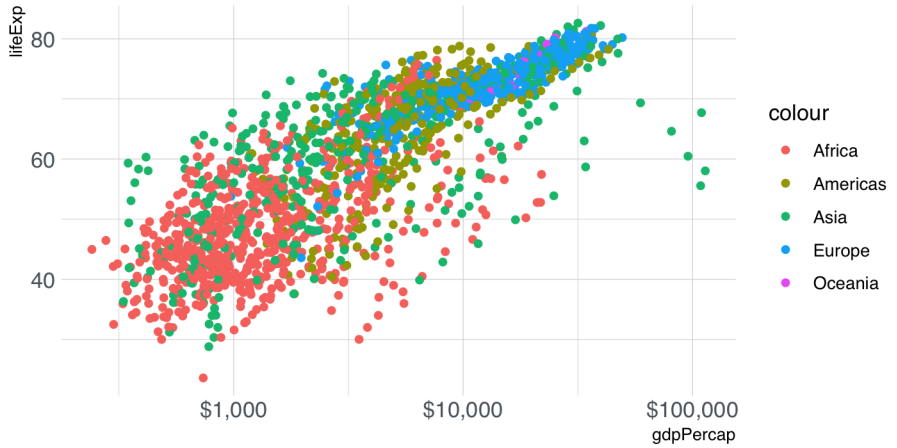
```
purp3←ggplot(data = gapminder, mapping =
aes(x = gdpPercap, y = lifeExp,
color=continent))+
  geom_point()+
  scale_x_log10(labels=scales::dollar)
```



Aesthetics can also be mapped per `geom()`

- You might want different `geom()` to show different things.
- That can be easily achieve that by mapping at the `geom()` level.
- Whatever is mapped at the `geom()` level, will override what is mapped at `ggplot()`.

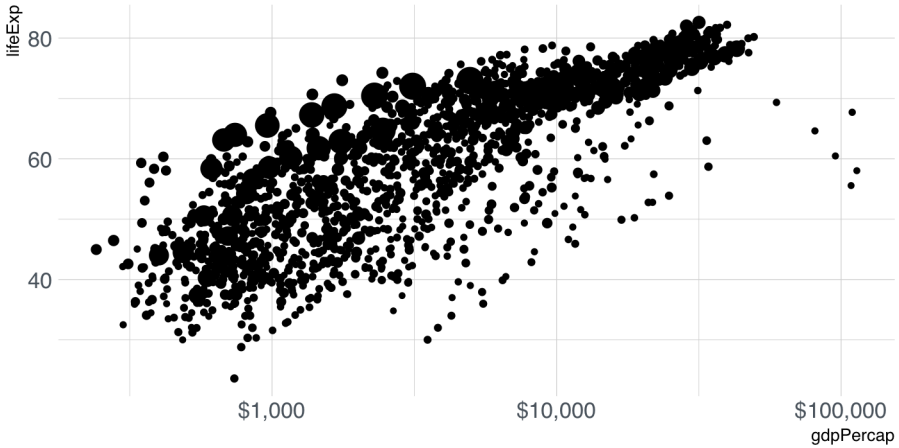
```
purp5←ggplot(data = gapminder, mapping =
aes(x = gdpPercap, y = lifeExp,
color="purple"))+
  geom_point(mapping=aes(color=continent))+
  scale_x_log10(labels=scales::dollar)
```



PS: same logic applies for other aesthetics.

- If we want the points to be sized by a variable, we need map the variable to an aesthetics.
  - These are all aesthetics in ggplot2:
  - size and shape.
  - line-type.
  - color (outside color) and fill (inside color).

```
purp4←ggplot(data = gapminder, mapping =
aes(x = gdpPercap, y = lifeExp, size= pop))+
  geom_point()+
  scale_x_log10(labels=scales::dollar)+
  theme(legend.position="none")
```



quick look at the data again!

```
library(gapminder) #downloading the package with the data
library(dplyr) # loading dplyr
data("gapminder") # loading the data from the package
sample_n(gapminder,15) #showing a random sample
```

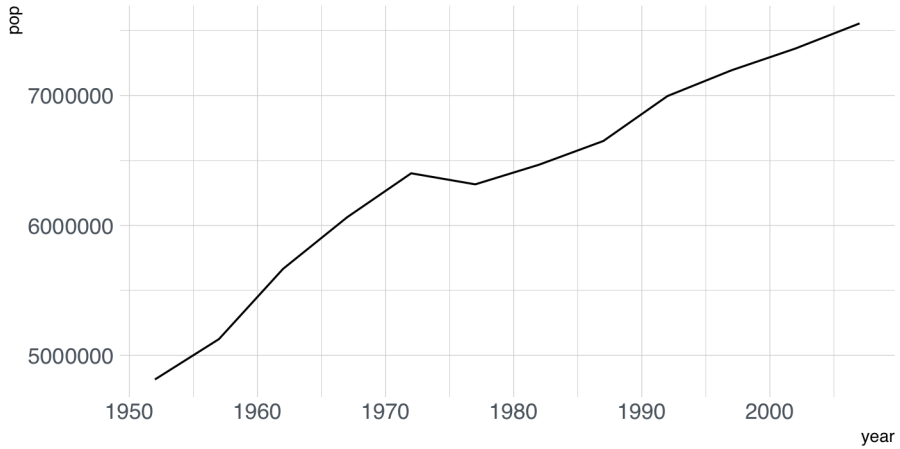
```
#> # A tibble: 15 × 6
#>    country        continent  year lifeExp      pop gdpPercap
#>    <fct>          <fct>     <int>   <dbl>    <int>     <dbl>
#>  1 Libya          Africa     1987    66.2  3799845    11771.
#>  2 El Salvador    Americas   1962    52.3  2747687     3777.
#>  3 Uruguay        Americas   1982    70.8  2953997     6920.
#>  4 Senegal        Africa     1962    41.5  3430243     1655.
#>  5 Botswana       Africa     1992    62.7  1342614     7954.
#>  6 Tanzania       Africa     1987    51.5 23040630      832.
#>  7 Ireland        Europe     1997    76.1  3667233    24522.
#>  8 Cote d'Ivoire  Africa     1972    49.8  6071696     2378.
#>  9 Afghanistan    Asia       1987    40.8 13867957      852.
#> 10 Montenegro     Europe     1962    63.7   474528     4650.
#> 11 Austria        Europe     1992    76.0  7914969    27042.
#> 12 Pakistan       Asia       1972    51.9 69325921     1050.
#> 13 Czech Republic Europe     1997    74.0 10300707    16049.
#> 14 Netherlands    Europe     1952    72.1 10381988     8942.
#> 15 Egypt          Africa     1972    51.1 34807417     2024.
```

`

Last, ggplot2 works with pipes!

- Just remember the difference between piping (%>%) and layering (+)!
  - The '%>%' pipes the output of an operation as the first argument of the next one.
  - The '+', instead, adds layers to a ggplot2 plot (here)

```
plot ← gapminder %>%
          filter(country=="Switzerland")%>%
          ggplot(mapping=aes(x=year, y=pop)) +
          geom_line()
```



Repository: exporting high-resolution images in four quick steps

```r
# 1 Set your working directory:
setwd("~/Desktop")

# 2 Run a 'graphics devices': tiff(); jpeg(); png()...
tiff("figure_money.tiff", units="in", width=10, height=7,res=1800)
# Tag Image Format File (.tiff) → very high-quality, but sizable files.
# Image size increases with resolution (measured in ppi)

# 3 Run your ggplot code OR call the object:
ggplot(data=gapminder, mapping=aes(x=gdpPercap, y=lifeExp)) +
  geom_point(color="grey", alpha=.5, size=3)+
  geom_smooth(se=FALSE, size=1.25, color="gold")+
  scale_x_log10(labels=scales::dollar)+
  labs( y="Life Expectancy", x=" GDP per Capita",
        title= "People in richer countries live longer.",
        subtitle=" Dataset composed of 142 countries from 1952 to 2007",
        caption= "Source: Gapminder Dataset" )+
  theme(title = element_text(color="black", size=10, face="bold"),
        axis.text = element_text(color="black", size=10),
        plot.subtitle = element_text(color="black", size=10, face= "plain"),
        plot.caption = element_text(color="black", size=8, face="bold"))

#  4 Turn off the graphic device, file placed in your working directory:
dev.off()
```

```
#> quartz_off_screen
```