

How to effectively use visualisation while doing Exploratory Data Analysis

Kaspar Märtens

18 October 2018

Today:

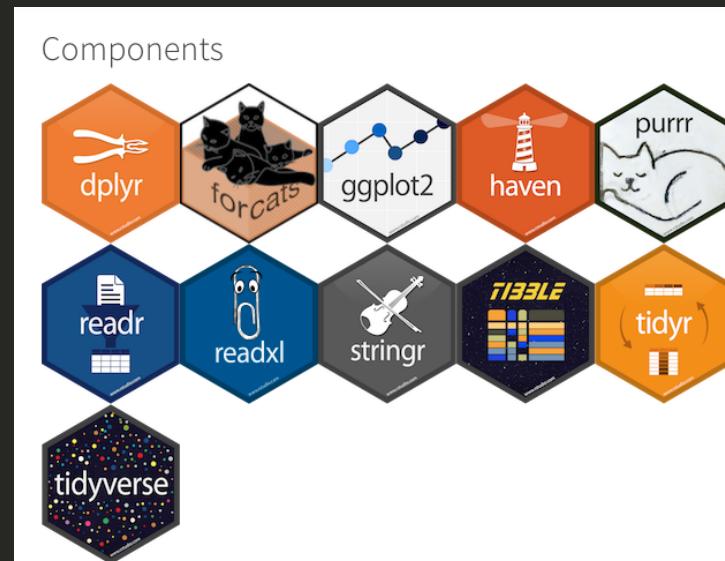
1. Why I think R is really great
2. Why visualise data at all?
3. Good practices and design choices involved when making graphs
4. Focus on the use of visualisations as part of data exploration -- ggplot2 is especially useful for this

Why I think R is great

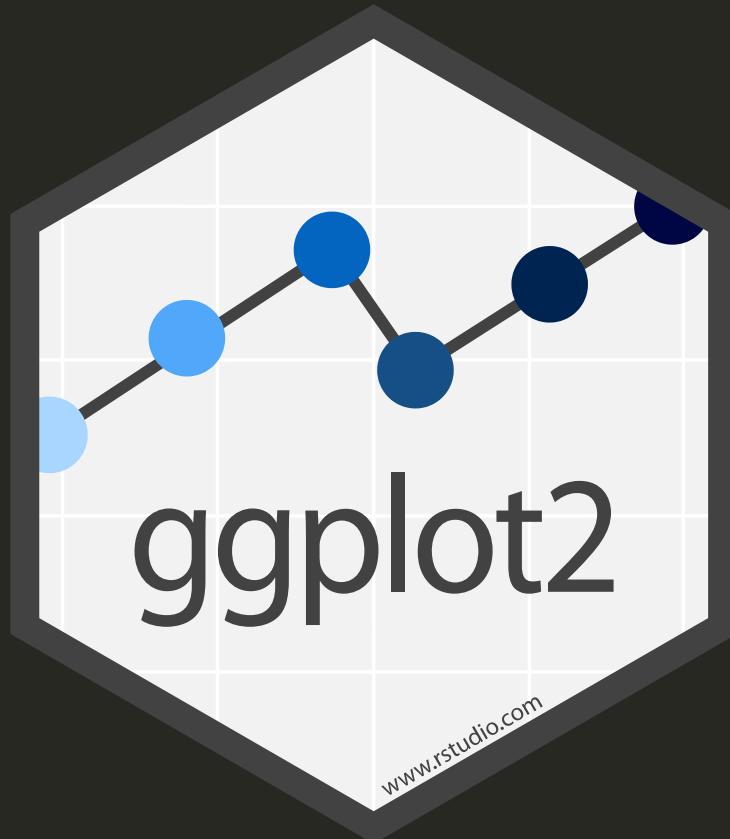
Tidyverse

A popular collection of R packages for data import, manipulation, exploration and visualization.

- Unified and consistent design and API (unlike base R)
- Intuitive underlying philosophy
- Easy-to-learn
- Easy to get help: large community on stackoverflow, twitter #rstats etc



ggplot2



R Markdown

~/Documents/r4ds/r4ds/rmarkdown - master - RStudio

diamond-sizes.Rmd x Environment History Git

Files Plots Packages Help Viewer

rmardown

```
1 ---  
2 title: "Diamond sizes"  
3 date: 2016-08-25  
4 output: html_document  
5 ---  
6  
7 `r setup, include = FALSE}  
8 library(ggplot2)  
9 library(dplyr)  
10  
11 smaller <- diamonds %>%  
12   filter(carat <= 2.5)  
13  
14  
15 We have data about `r nrow(diamonds)` diamonds. Only  
16 `r nrow(diamonds) - nrow(smaller)` are larger than  
17 2.5 carats. The distribution of the remainder is shown  
18 below:  
19  
20 `r, echo = FALSE}  
21 smaller %>%  
22   ggplot(aes(carat)) +  
23     geom_freqpoly(binwidth = 0.01)  
24  
25
```

8:17 Chunk 1: setup R Markdown

Console R Markdown

~/Documents/r4ds/r4ds/rmarkdown ↵

Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |

Diamond sizes
2016-08-25

We have data about 53940 diamonds. Only 126 are larger than 2.5 carats. The distribution of the remainder is shown below:

The figure is a histogram of diamond carat weights. The x-axis is labeled 'carat' and ranges from 0 to 2.5. The y-axis is labeled 'count' and ranges from 0 to 2000. The distribution is highly right-skewed, with a very sharp peak at approximately 0.2 carats (count ~2500) and several smaller peaks at integer carat values (e.g., 1.0, 1.5, 2.0).

R Markdown

The screenshot shows the RStudio interface with the following components:

- Left Panel (Code Editor):** The file `diamond-sizes.Rmd` is open. The code includes R Markdown syntax like `<!--` and `---` for YAML front matter, and ````{r}`` blocks for R code. A note explains that the data set contains 53,940 diamonds, with only 126 being larger than 2.5 carats.- Console:** Displays the R startup message and information about the R environment (version 3.4.0).
- Right Panel (Viewer):** The title "Diamond sizes" is displayed above a date "2016-08-25". Below this, a note states there are 53,940 diamonds and only 126 are larger than 2.5 carats. A histogram titled "Distribution of diamond carat sizes" is shown, plotting "count" against "carat". The x-axis ranges from 0 to 2.5, and the y-axis ranges from 0 to 2,000. The distribution is highly right-skewed, with major peaks at approximately 0.25, 0.75, 1.25, and 1.75 carats.

Note that these slides have been written in RMarkdown, using the xaringan package
<https://github.com/yihui/xaringan>

Shiny

Turn your static figures into an interactive app with little effort using Shiny
<https://shiny.rstudio.com/gallery/>

Shiny from R Studio

Get Started **Gallery** Articles Reference Deploy Help Contribute

Interactive visualizations

Shiny is designed for fully interactive visualization, using JavaScript libraries like d3, Leaflet, and Google Charts.

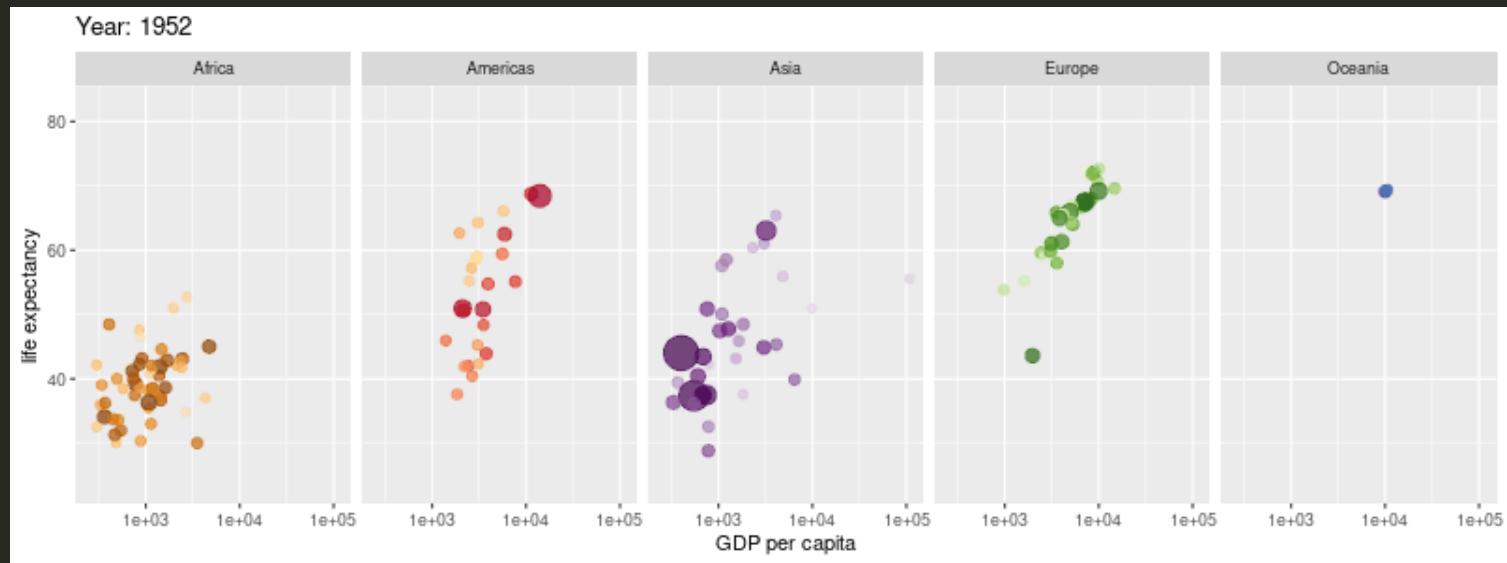
SuperZip example Bus dashboard Movie explorer Google Charts

Start simple

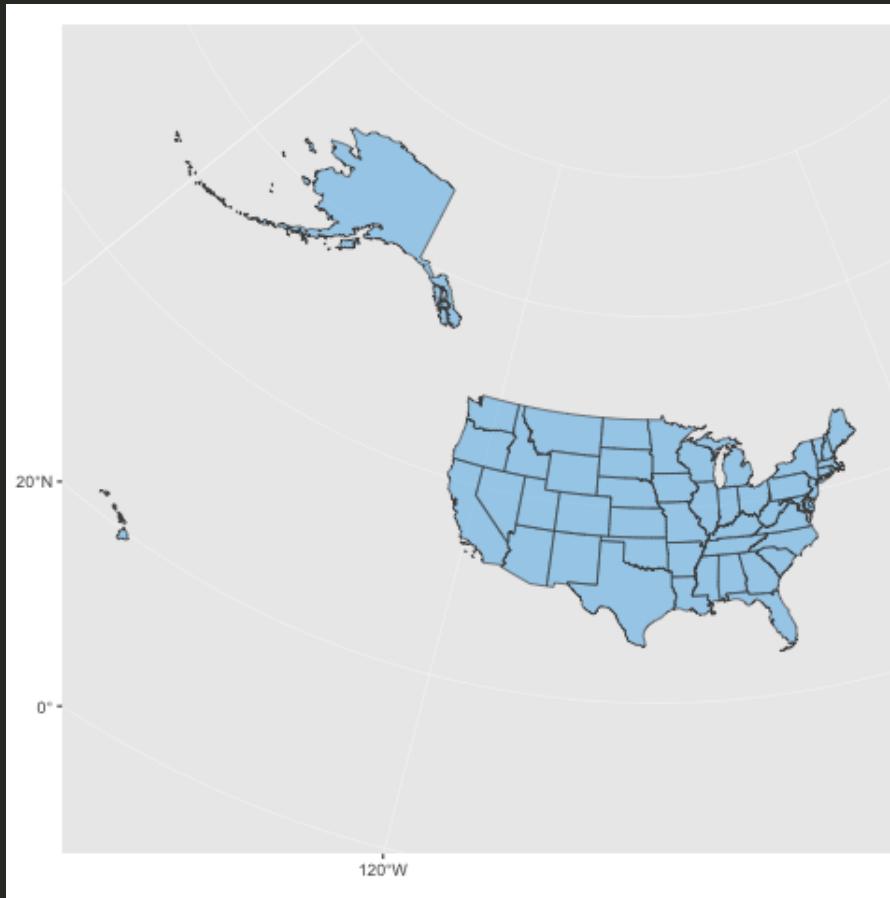
If you're new to Shiny, these simple but complete applications are designed for you to learn from.

Kmeans example Telephones by region Faithful Word cloud

gganimate



gganimate



Source: <https://github.com/thomasp85/gganimate/wiki/Moving-Hawaii-and-Alaska>

Are visualisations useful?

Consider the following data ($N=20$, $p=2$)

x	y
1.851	0.757
0.891	1.791
0.598	1.909
0.546	1.924
-0.195	1.991
-1.388	1.440
-1.458	1.369
-1.493	1.331
-2.000	0.029
-1.794	-0.884

	x	y
11	-1.377	-1.450
12	-1.063	-1.694
13	-0.771	-1.845
14	-0.404	-1.959
15	0.249	-1.984
16	1.593	-1.209
17	1.606	-1.192
18	1.676	-1.091
19	1.880	-0.681
20	1.997	-0.102

```
mean(x)
```

```
## [1] 0.04724152
```

```
mean(y)
```

```
## [1] -0.07760144
```

```
cor(x, y)
```

```
## [1] -0.0304444
```

```
mean(x)
```

```
## [1] 0.04724152
```

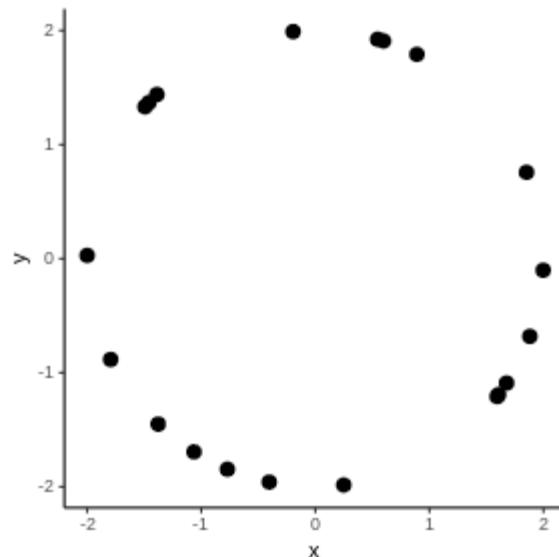
```
mean(y)
```

```
## [1] -0.07760144
```

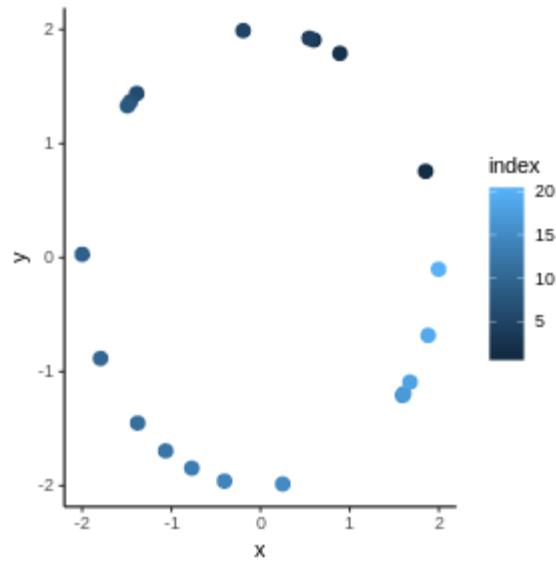
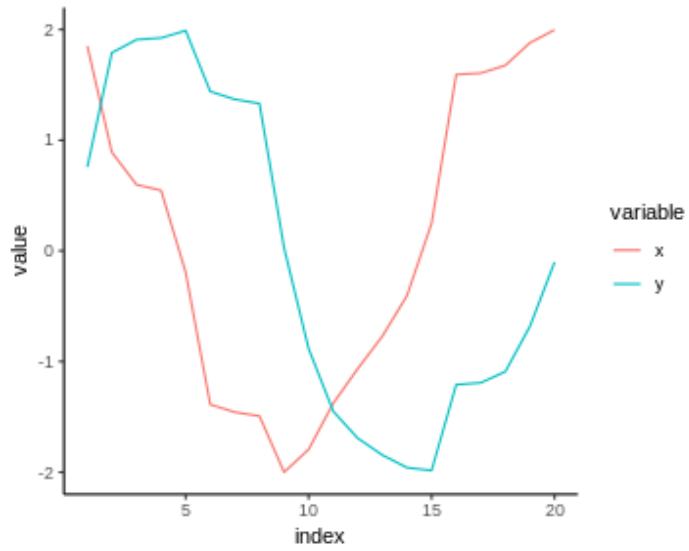
```
cor(x, y)
```

```
## [1] -0.03044444
```

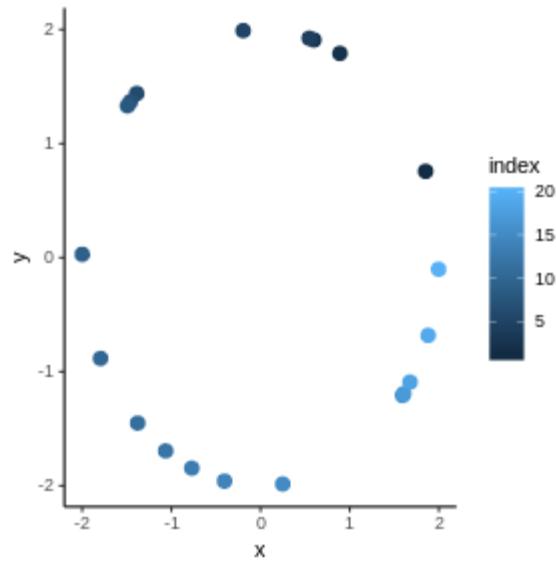
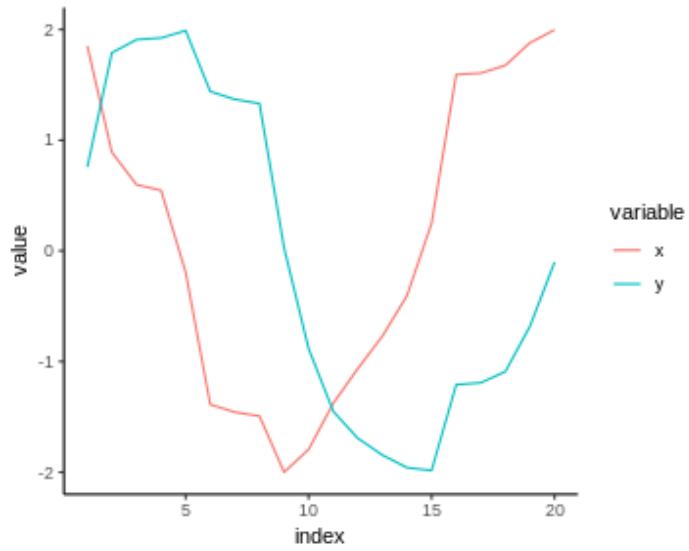
Scatterplot immediately reveals the structure



Some visualisations are more useful than others



Some visualisations are more useful than others



Inspired by the talk "How Humans See Data" by John Rauser

How can we visualise the distribution of a continuous variable?

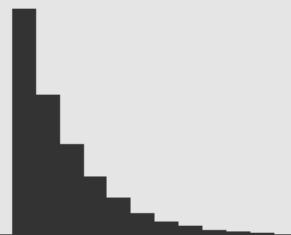
histogramm 1



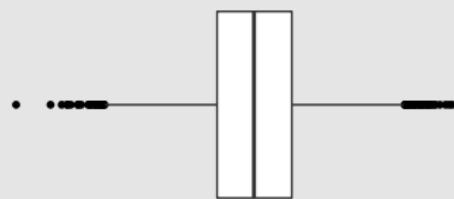
histogramm 2



histogramm 3



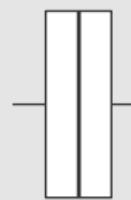
A



B



C



histogramm 1



histogramm 2



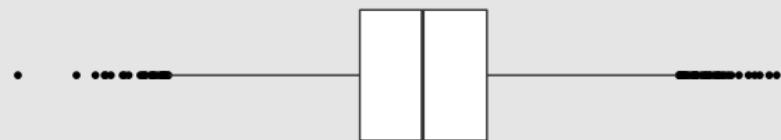
histogramm 3



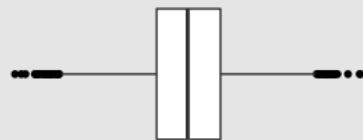
histogramm 4



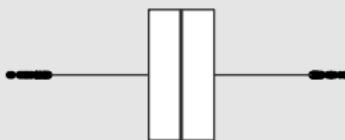
A



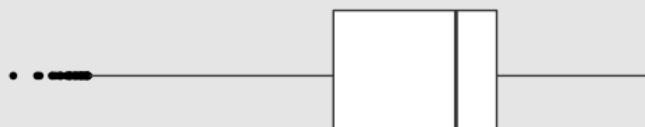
B



C

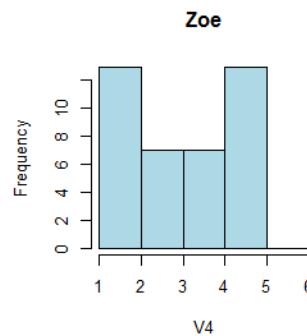
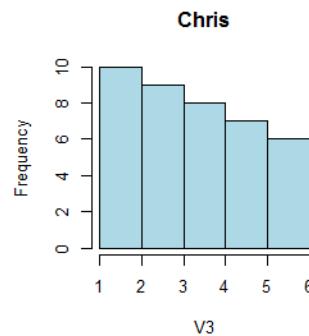
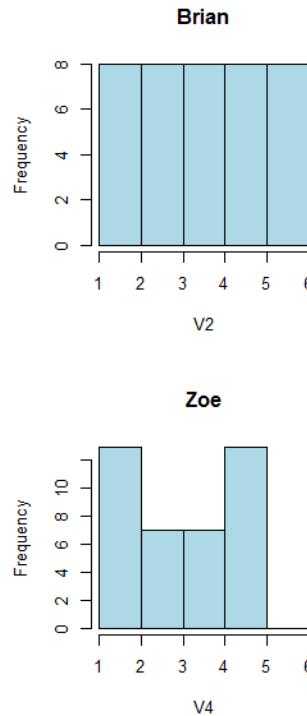
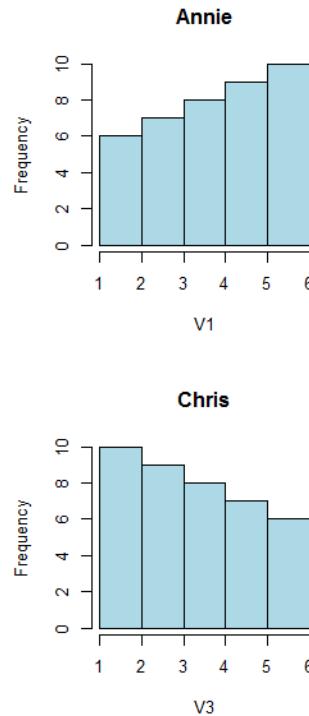


D

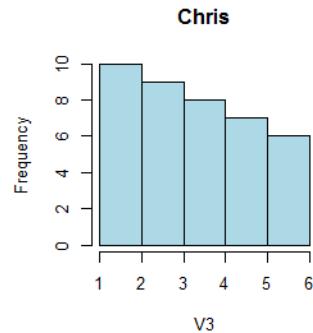
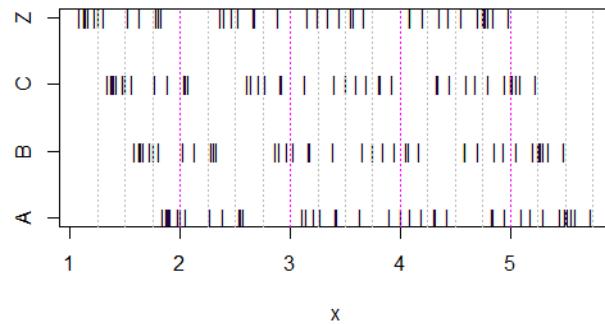
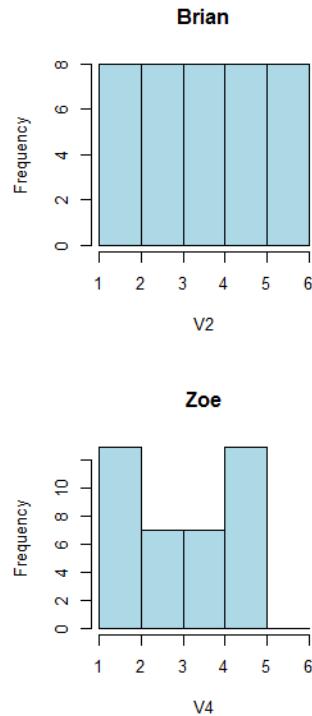
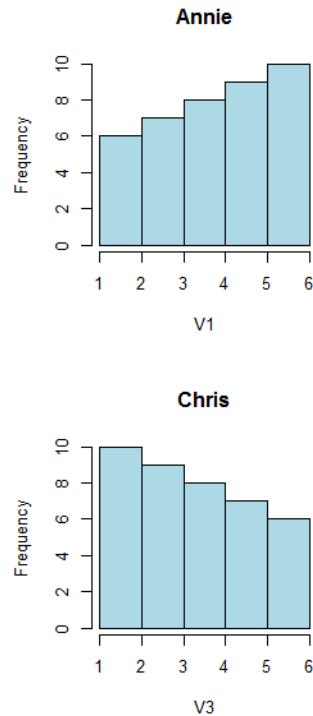


Can a histogram be misleading?

Can a histogram be misleading?



Can a histogram be misleading?



Source: <https://stats.stackexchange.com/a/51753>

How can we visualise the distribution of a continuous variable, compared across two (or more) groups?

Try to think of as many different approaches as you can. What are the pros and cons of each?

What if there are more than two or three variables?

That is, how to visualise high-dimensional data?

What if there are more than two or three variables?

That is, how to visualise high-dimensional data?

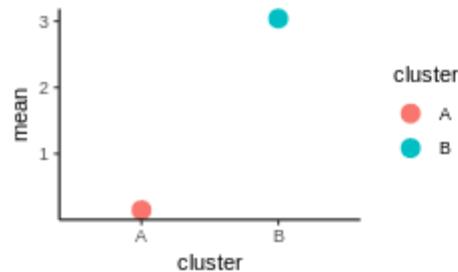
Options include:

- visualise a subset of variables,
 - randomly selected
 - selection based on summary statistics
- compute summary statistics and visualise those instead
- apply a dimensionality reduction method such as PCA

Do summary statistics capture everything?

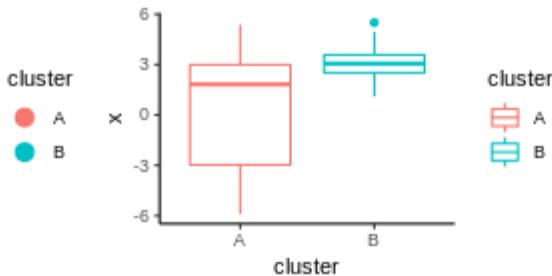
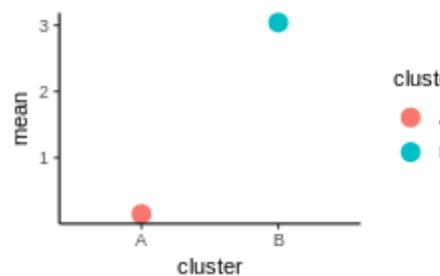
Do summary statistics capture everything?

Do groups A and B have a significantly different measurements x?



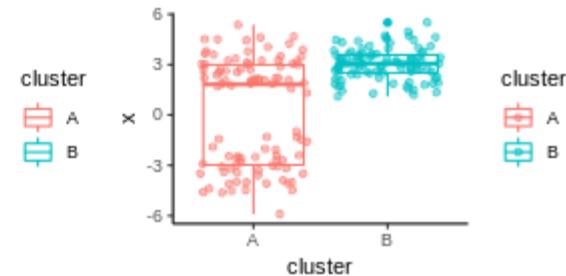
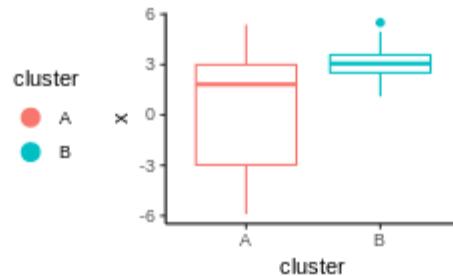
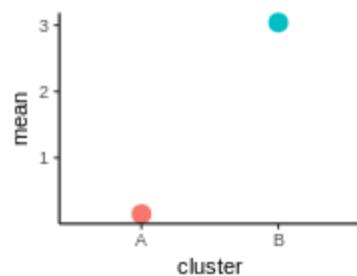
Do summary statistics capture everything?

Do groups A and B have a significantly different measurements x?



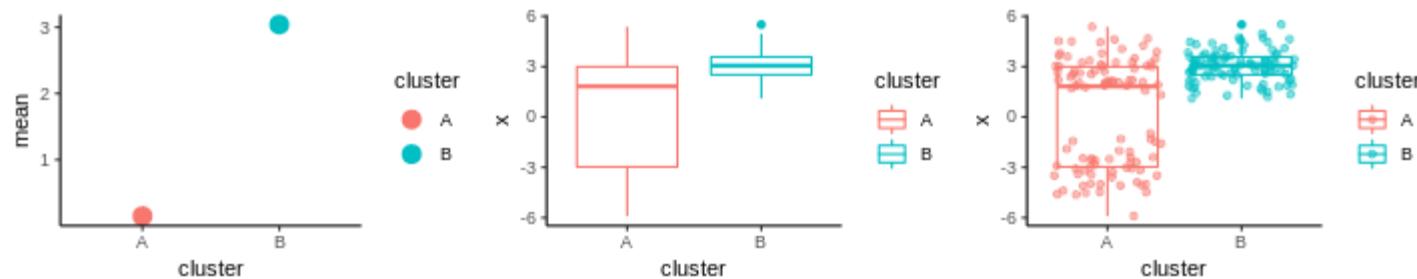
Do summary statistics capture everything?

Do groups A and B have a significantly different measurements x?



Do summary statistics capture everything?

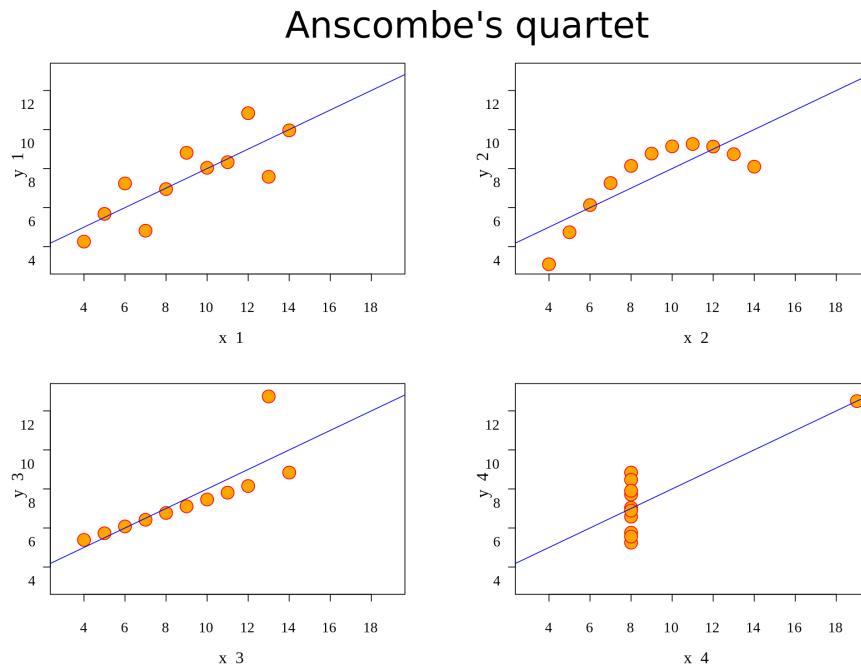
Do groups A and B have a significantly different measurements x?



single summary statistic < multiple summary statistics < data

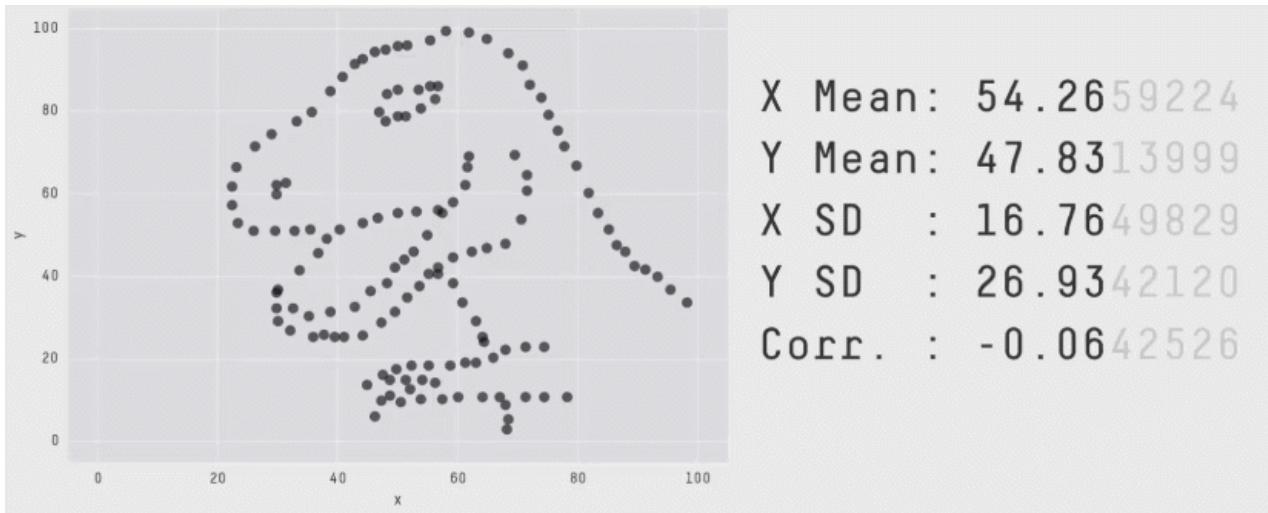
Do summary statistics capture everything?

Is there a significant association between two continuous variables x and y?



Source: https://en.wikipedia.org/wiki/Anscombe%27s_quartet

Do summary statistics capture everything?

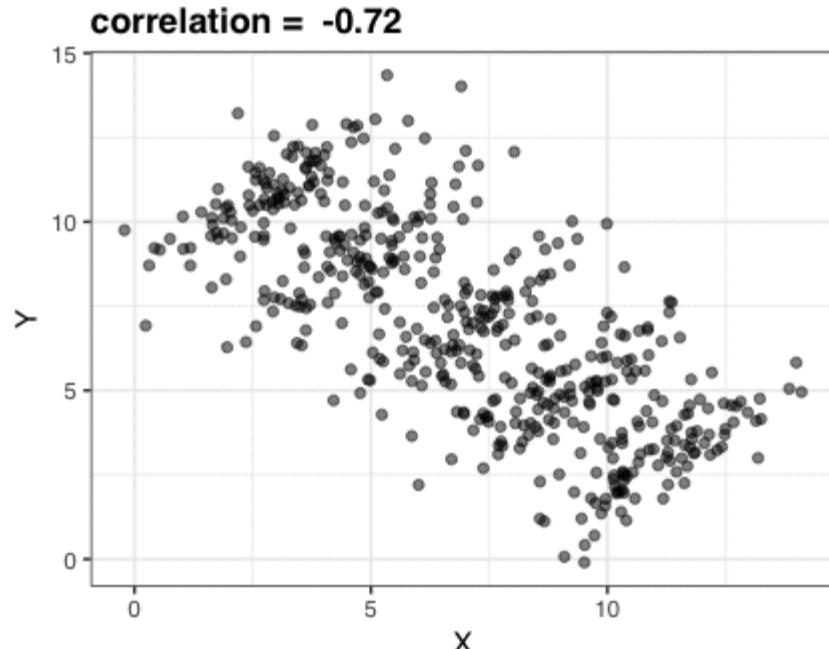


Source: <https://www.autodeskresearch.com/publications/samestats>

Simpson's paradox

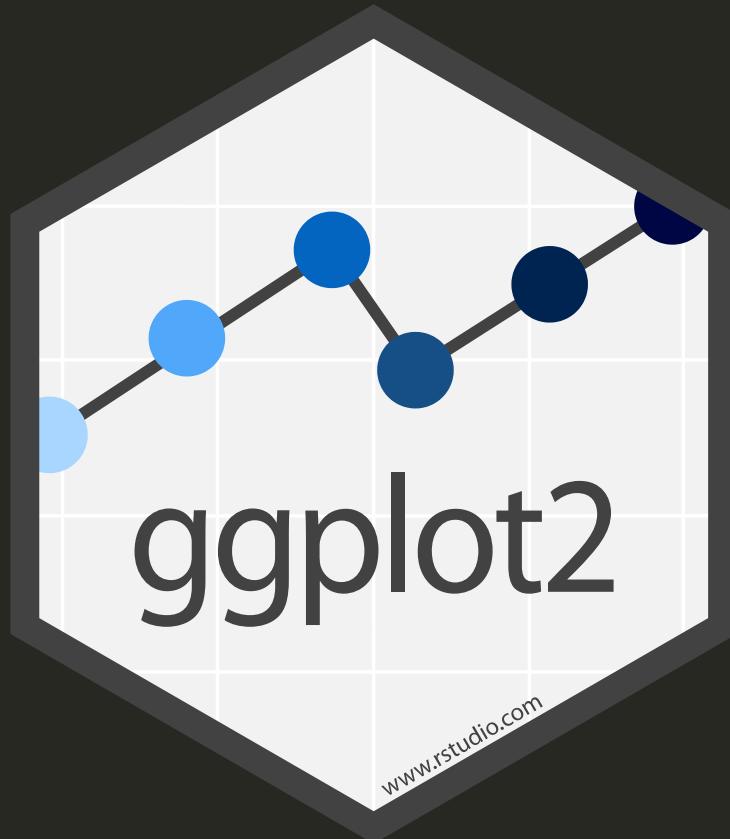
Can we trust the patterns that we see?

Subgroup structure (or other variables) may help to explain the patterns



Source: <https://simplystatistics.org/2017/08/08/code-for-my-educational-gifs/>

ggplot2



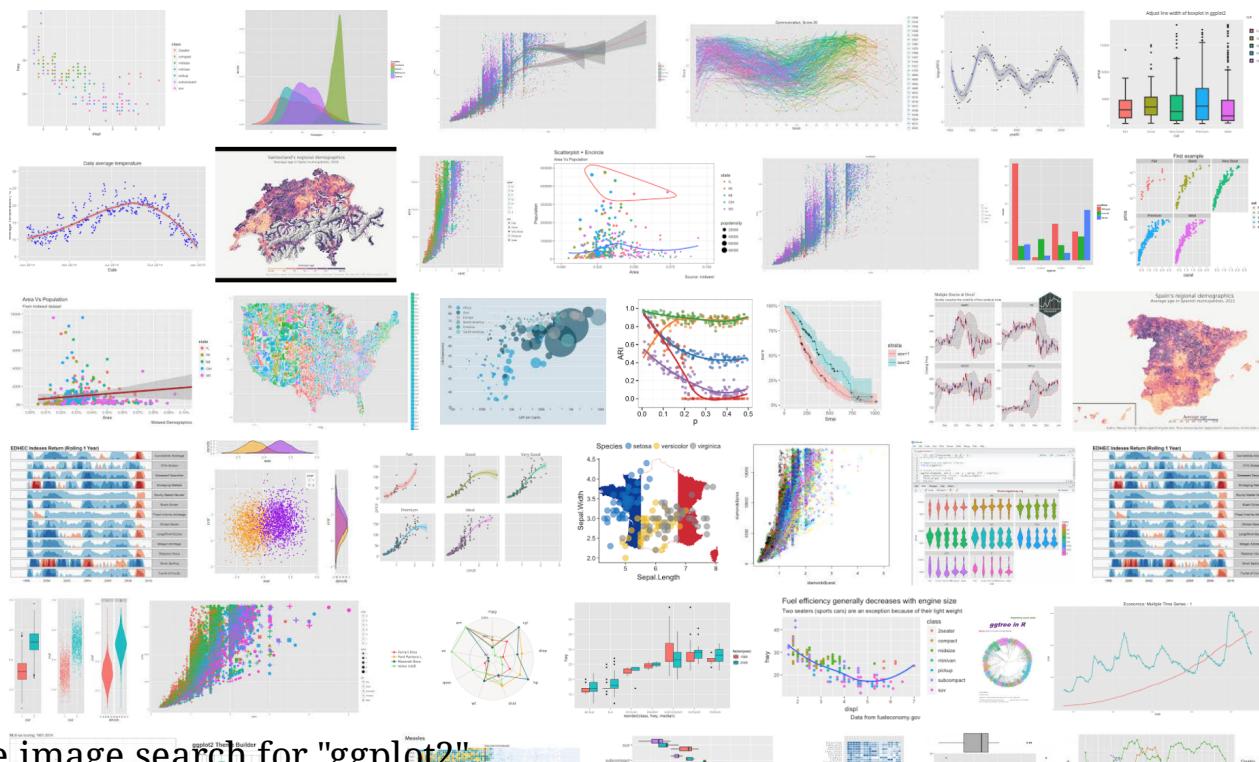
Hadley Wickham (<http://hadley.nz/>):

- author of many widely used R packages
 - including ggplot2
 - introduced the concept of tidy data
 - key contributor to the tidyverse
- author of many excellent books, including
 - R for Data Science <http://r4ds.had.co.nz/>
 - Advanced R <http://adv-r.had.co.nz/>
 - etc



ggplot2

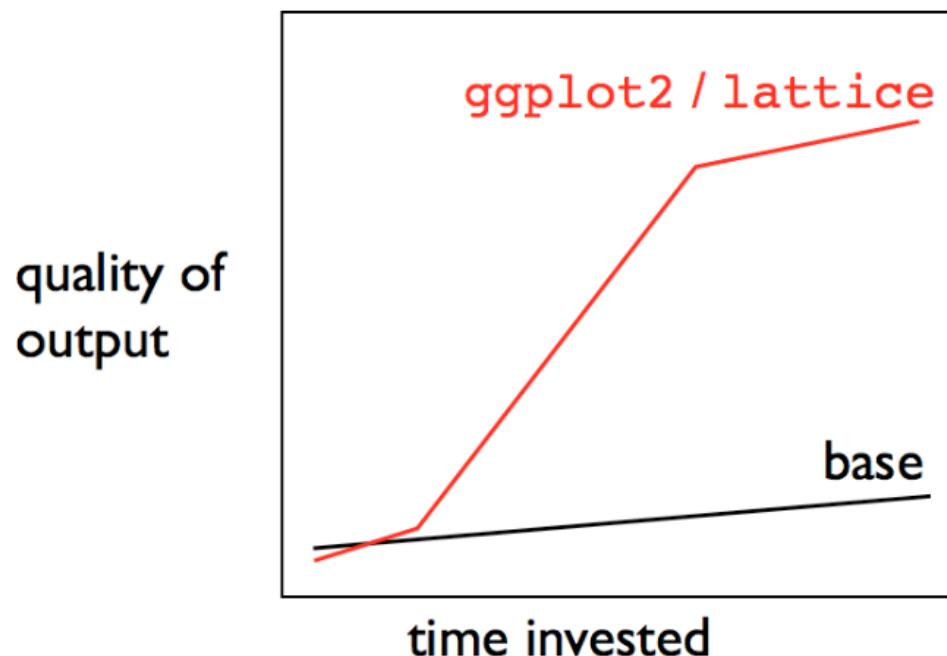
- Quickly iterate over a variety of plots (only making minimal changes to the code).
- Create publication-quality plots with minimal tweaking.



google image search for "ggplot2"

ggplot2 learning curve

week one

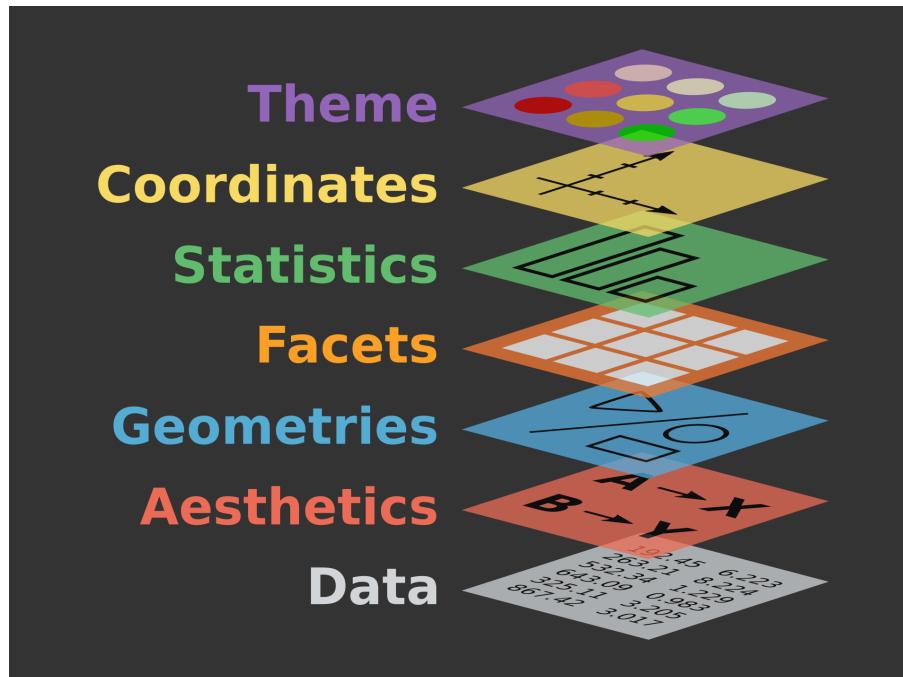


Source: <https://github.com/jennybc/ggplot2-tutorial>

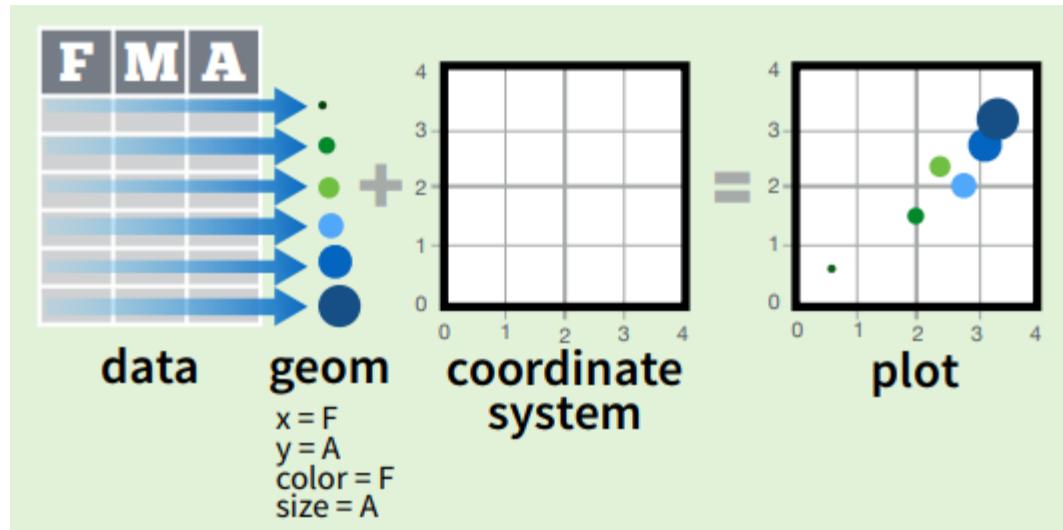
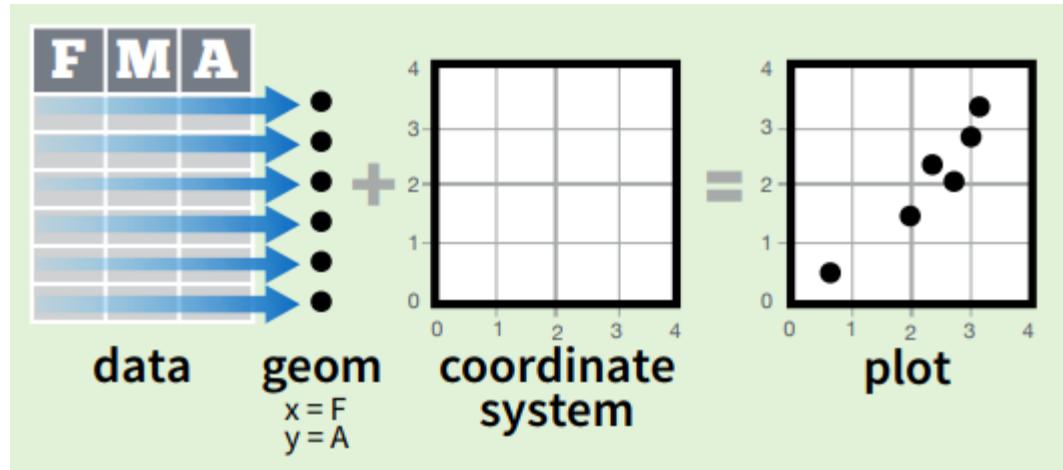
* figure is totally fabricated but, I claim, still true

ggplot2

Based on the Grammar of Graphics (book by Leland Wilkinson, 1999/2005) -- idea that every graph can be built from the same components. Results in a very flexible framework for plotting.



ggplot2



Gapminder data

- See the famous TED talk by Hans Rosling
https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen
- R package "gapminder" conveniently provides a subset of that data
<https://github.com/jennybc/gapminder>

Gapminder data

- See the famous TED talk by Hans Rosling
https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen
- R package "gapminder" conveniently provides a subset of that data
<https://github.com/jennybc/gapminder>

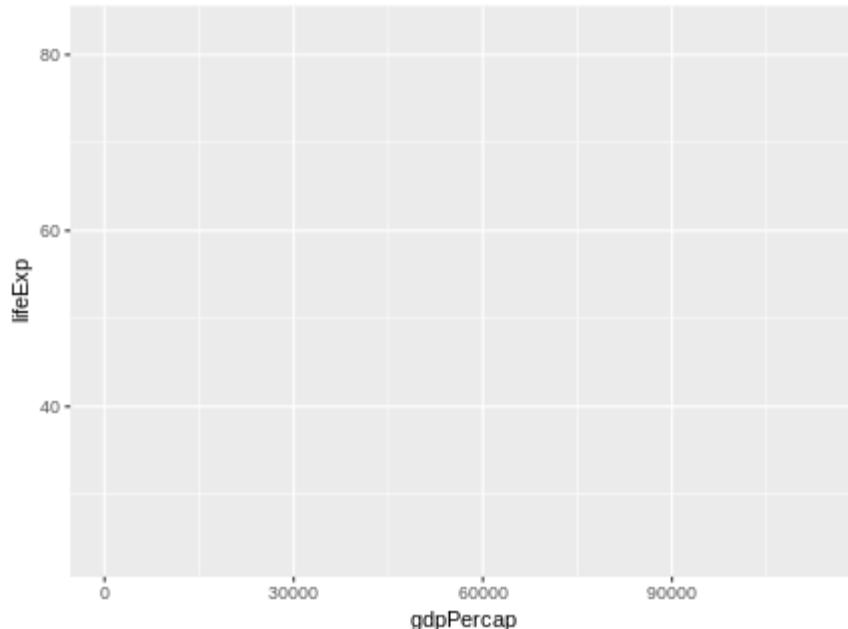
```
library(gapminder)
```

```
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country   continent year lifeExp      pop gdpPercap
##   <fct>     <fct>    <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia     1952     28.8  8425333    779.
## 2 Afghanistan Asia     1957     30.3  9240934    821.
## 3 Afghanistan Asia     1962     32.0  10267083   853.
## 4 Afghanistan Asia     1967     34.0  11537966   836.
## 5 Afghanistan Asia     1972     36.1  13079460   740.
## 6 Afghanistan Asia     1977     38.4  14880372   786.
```

ggplot2 call

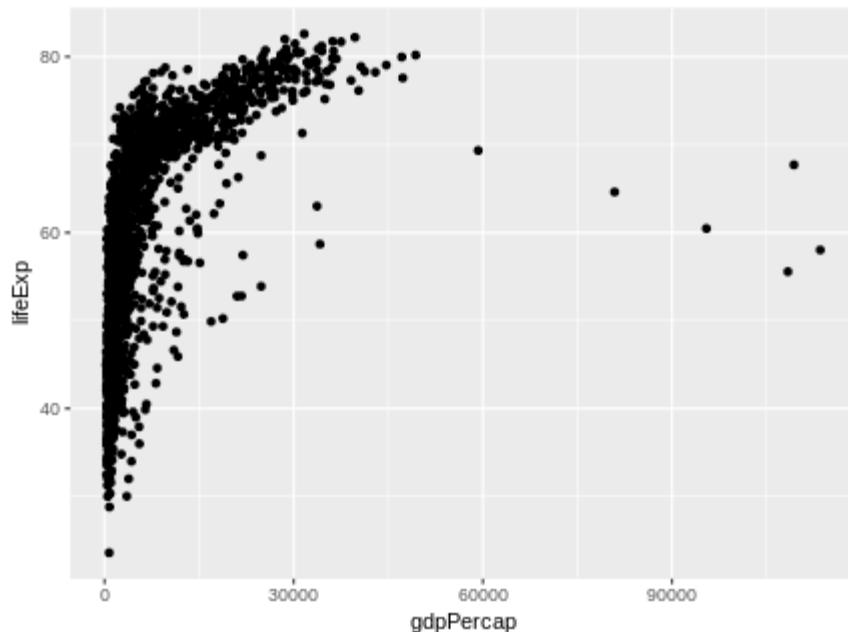
```
library(ggplot2)  
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp))
```



ggplot2 call: scatterplot

```
library(ggplot2)

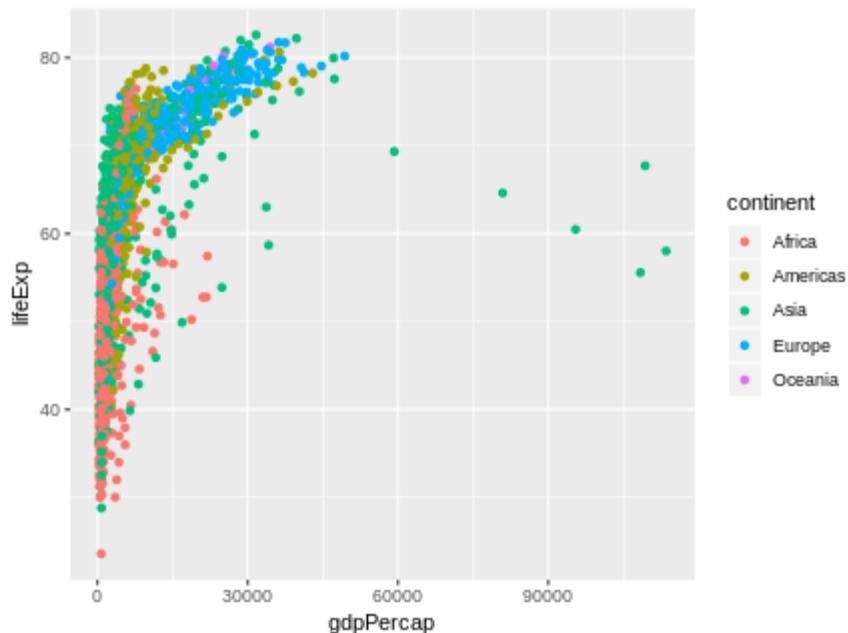
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```



colour by continent

```
library(ggplot2)

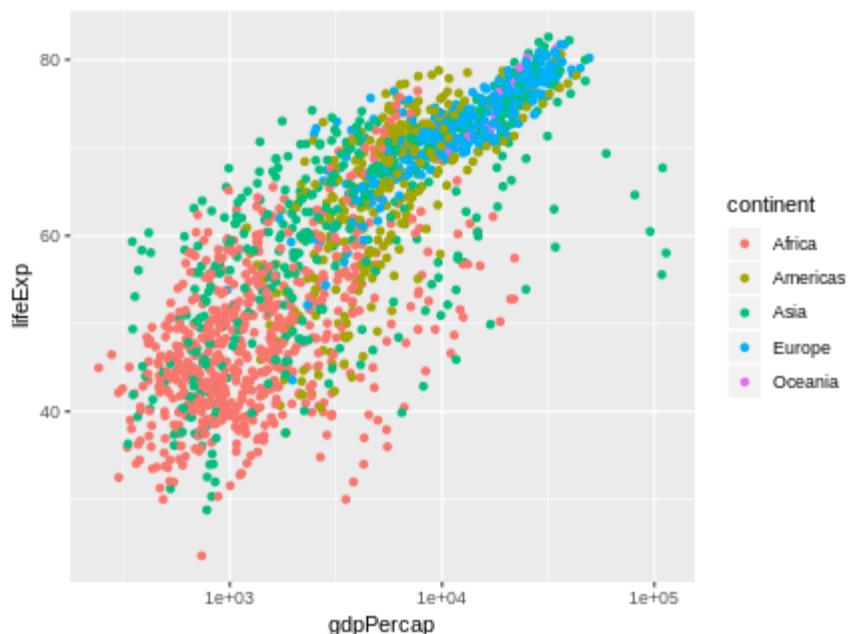
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, col = continent))
  geom_point()
```



change x-axis scale

```
library(ggplot2)

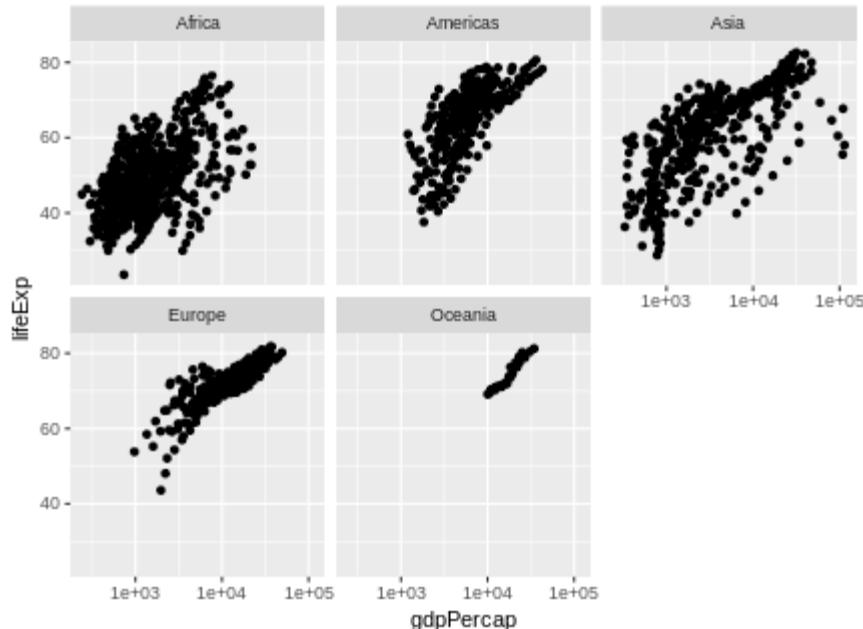
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, col = continent))
  geom_point() +
  scale_x_log10()
```



facet by continent

```
library(ggplot2)

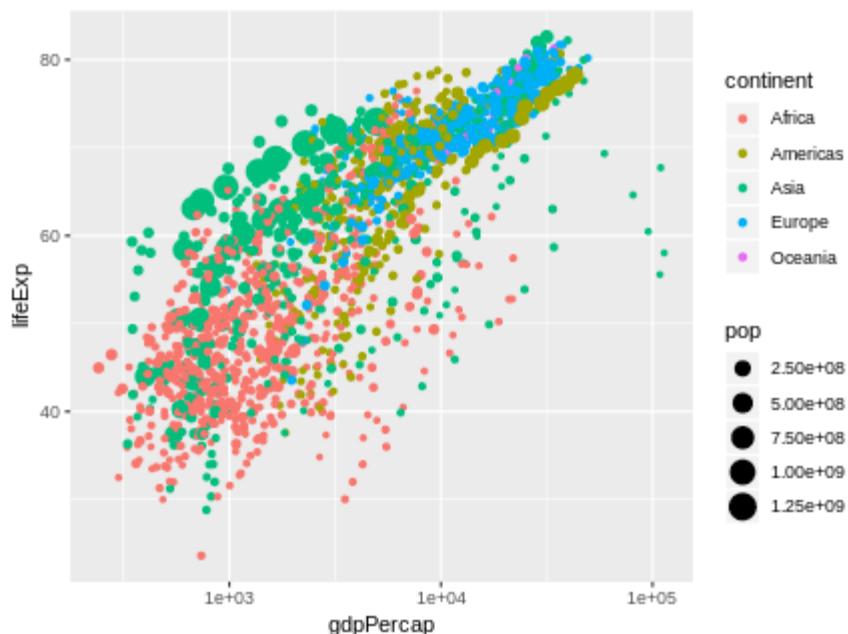
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  facet_wrap(~ continent) +
  scale_x_log10()
```



colour and size

```
library(ggplot2)

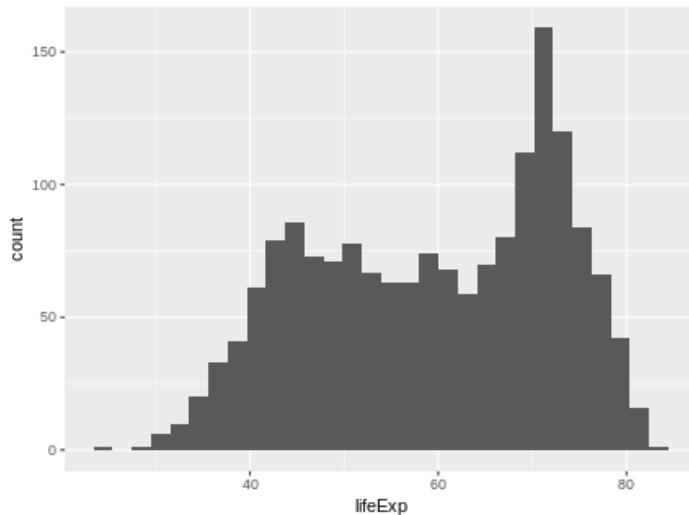
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, col = continent,
  geom_point() +
  scale_x_log10()
```



Let's explore life expectancy

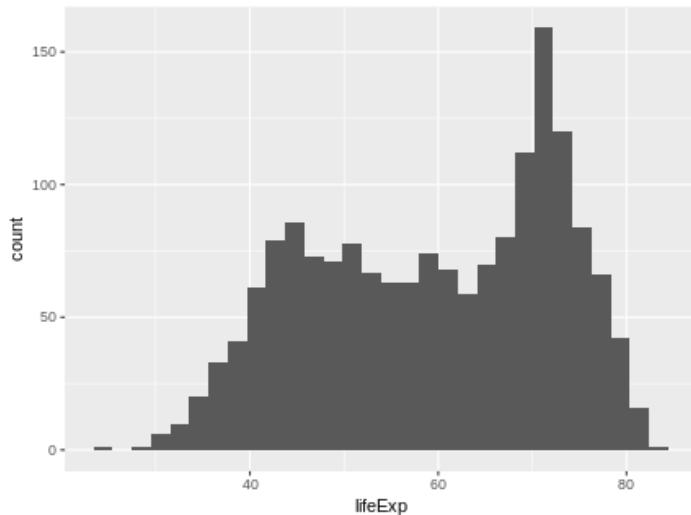
Let's explore life expectancy

```
ggplot(gapminder, aes(x = lifeExp))  
  geom_histogram()
```

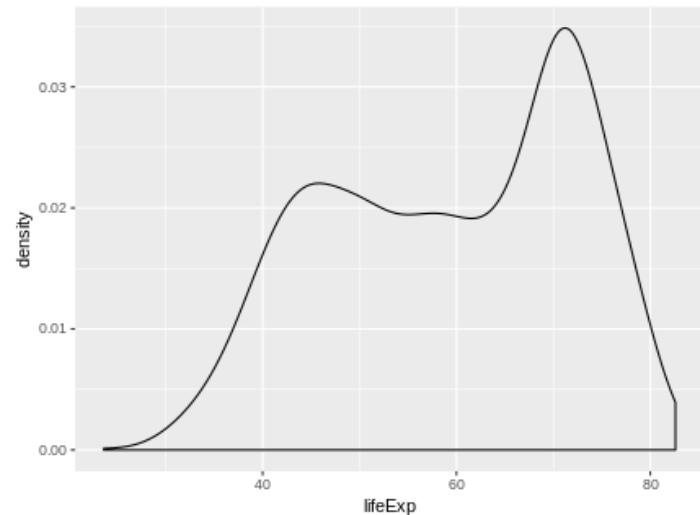


Let's explore life expectancy

```
ggplot(gapminder, aes(x = lifeExp))  
  geom_histogram()
```

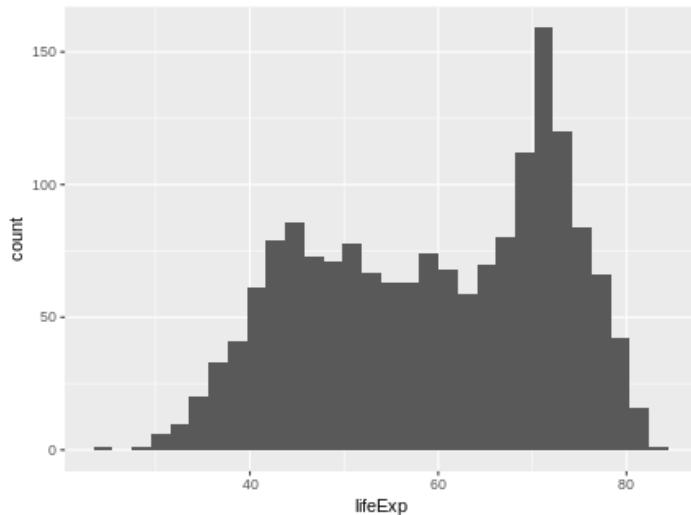


```
ggplot(gapminder, aes(x = lifeExp))  
  geom_density()
```

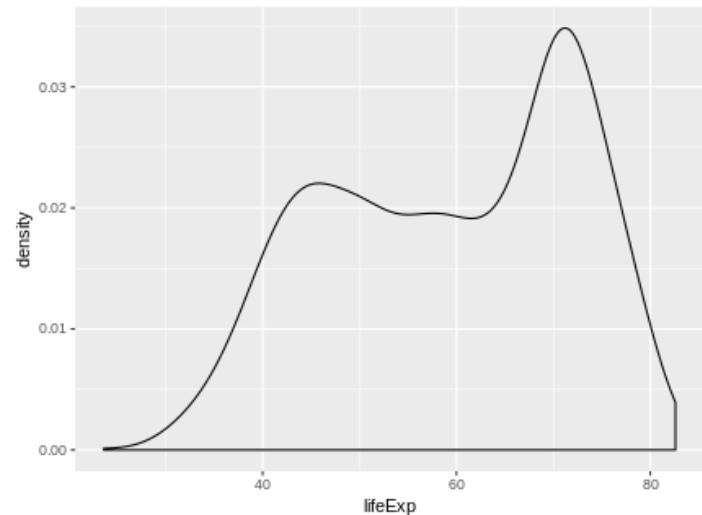


Let's explore life expectancy

```
ggplot(gapminder, aes(x = lifeExp))  
  geom_histogram()
```



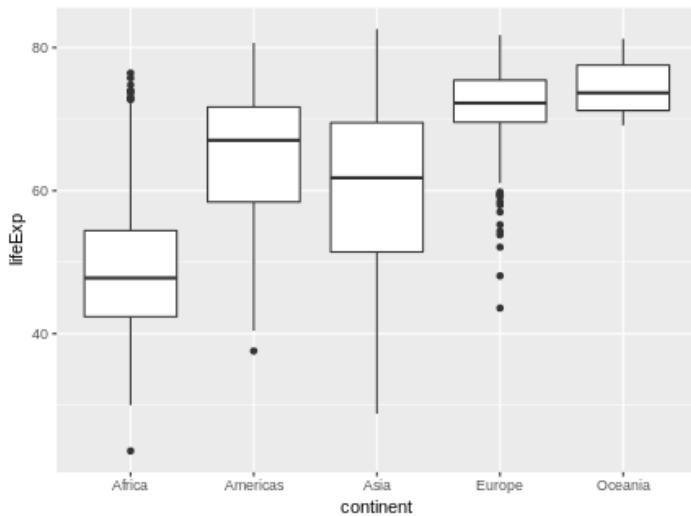
```
ggplot(gapminder, aes(x = lifeExp))  
  geom_density()
```



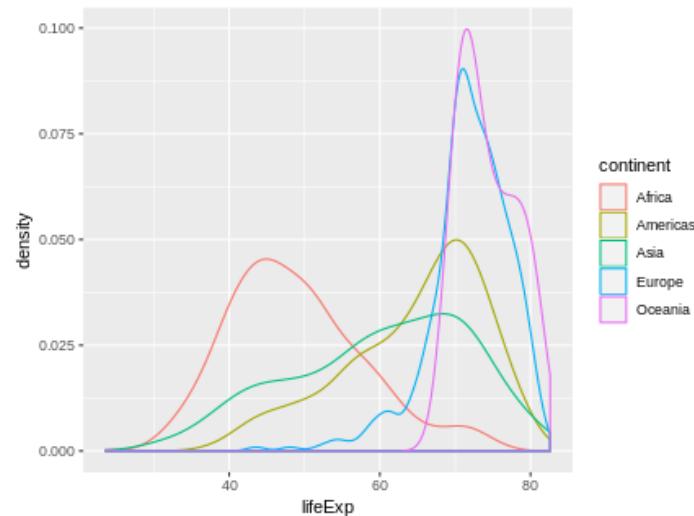
How does this compare across continents? Across years?

Let's explore life expectancy (across continents)

```
ggplot(gapminder, aes(x = continent, y = lifeExp))  
  geom_boxplot()
```

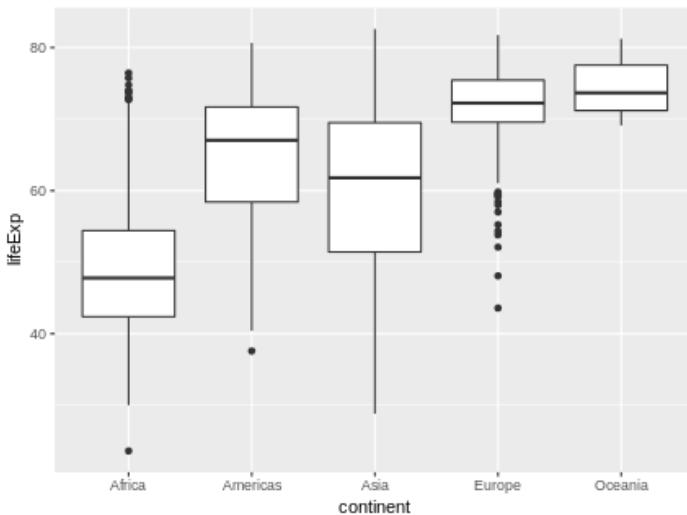


```
ggplot(gapminder, aes(x = lifeExp))  
  geom_density()
```

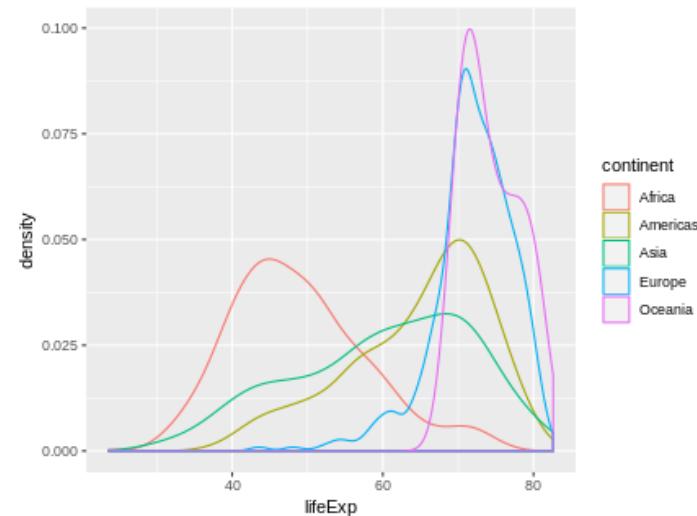


Let's explore life expectancy (across continents)

```
ggplot(gapminder, aes(x = continent, y = lifeExp))  
  geom_boxplot()
```



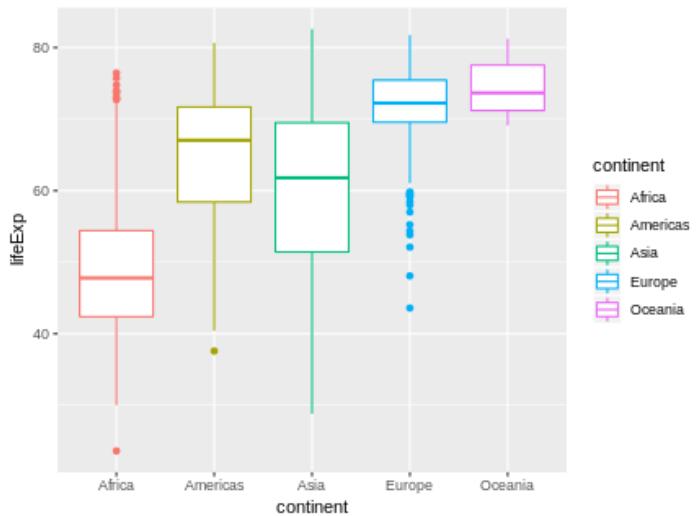
```
ggplot(gapminder, aes(x = lifeExp))  
  geom_density()
```



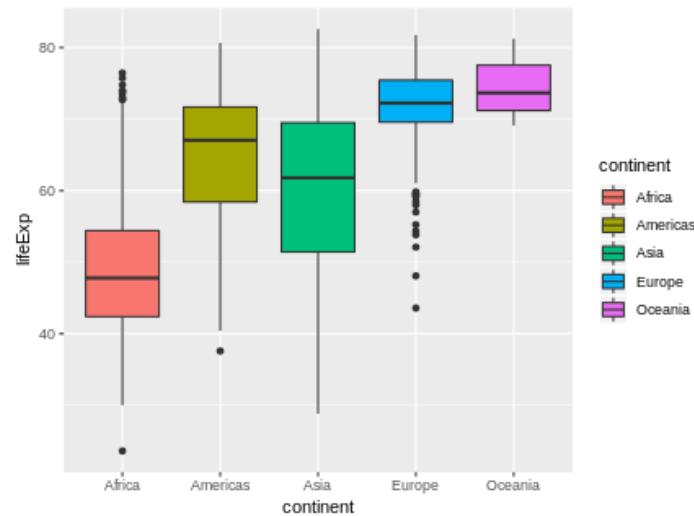
Can you think of other ways? Try them out!

colour vs fill

```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +  
  geom_boxplot()
```



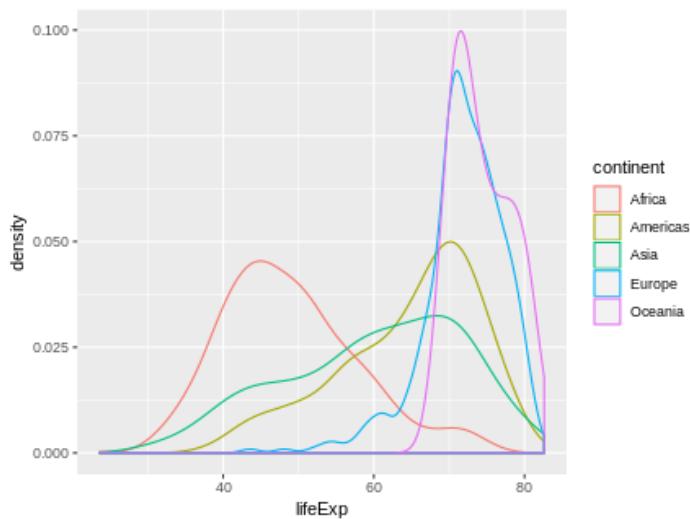
```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +  
  geom_boxplot(fill = "white")
```



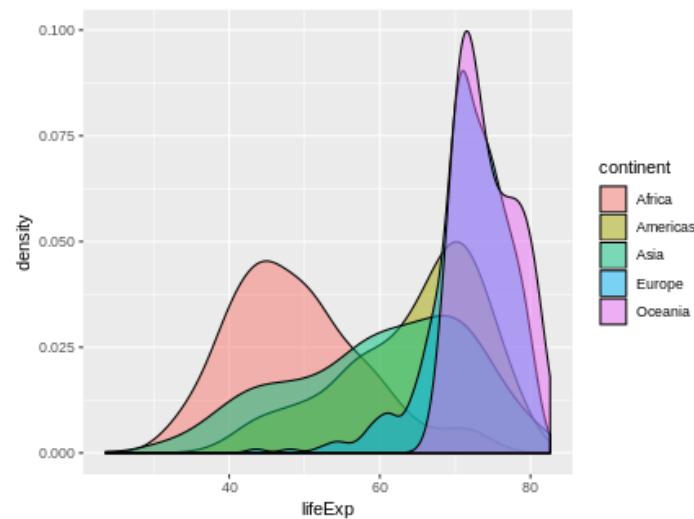
colour vs fill

Transparency with alpha

```
ggplot(gapminder, aes(lifeExp, ..density..)) +  
  geom_density()
```



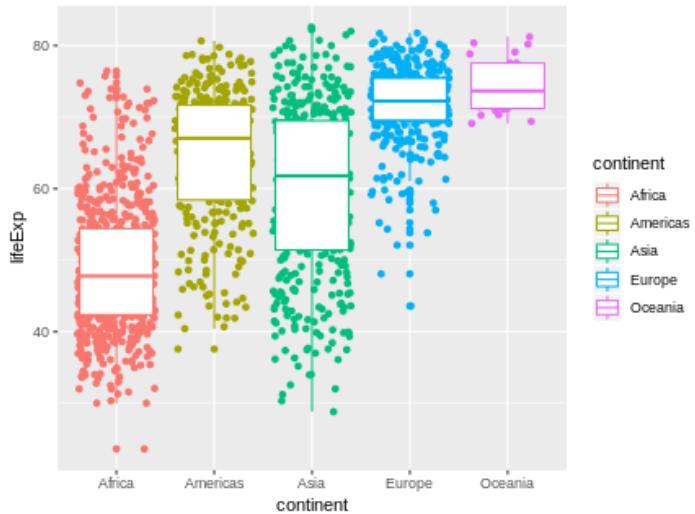
```
ggplot(gapminder, aes(lifeExp,  
                      ..density..)) +  
  geom_density(alpha = 0.5)
```



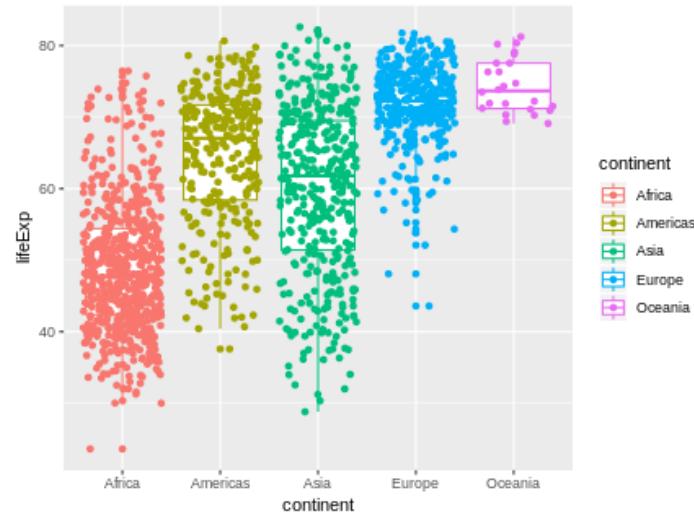
Multiple layers can be combined

Note the order of plotting

```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +  
  geom_jitter() +  
  geom_boxplot()
```



```
ggplot(gapminder, aes(x = continent, y = lifeExp)) +  
  geom_boxplot() +  
  geom_jitter()
```



aes() for different layers

Every layer can have its own aesthetics.

Here, we have only one layer `geom_point()`, so all of the following are equivalent

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, col = continent))  
  geom_point()
```

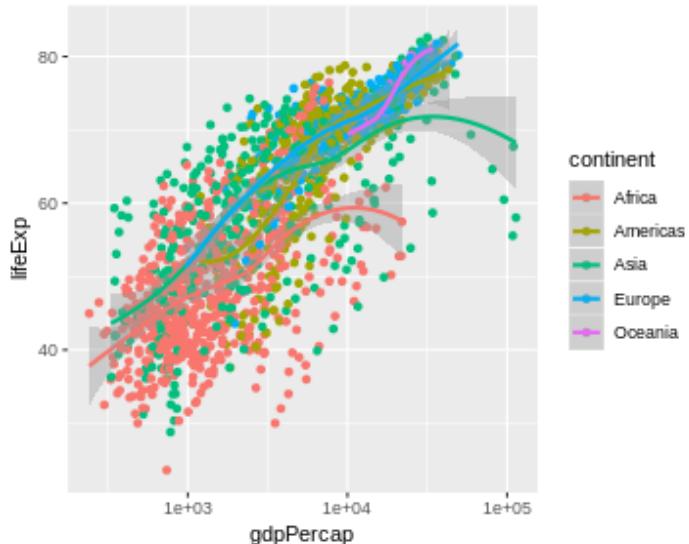
```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(aes(col = continent))
```

```
ggplot(gapminder) +  
  geom_point(aes(x = gdpPercap, y = lifeExp, col = continent))
```

aes() for different layers

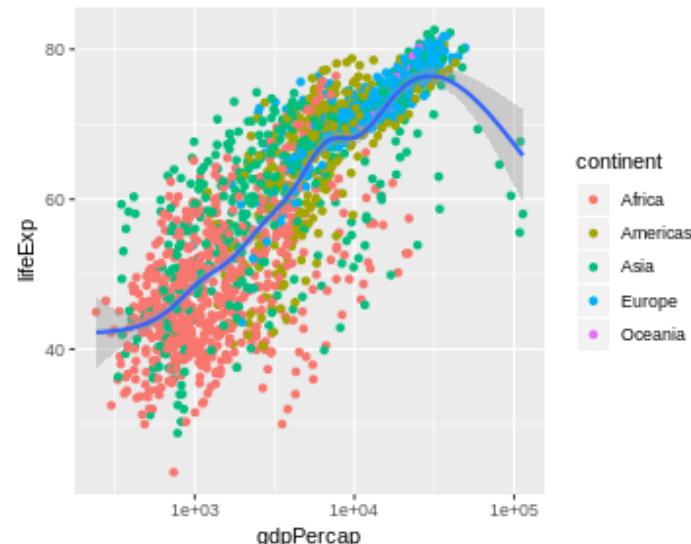
Here `aes(col = continent)` is shared for both layers.

```
ggplot(gapminder, aes(gdpPercap  
geom_point() +  
geom_smooth() +  
scale_x_log10()
```



Here `aes(col = continent)` is specified for `geom_point()` only.

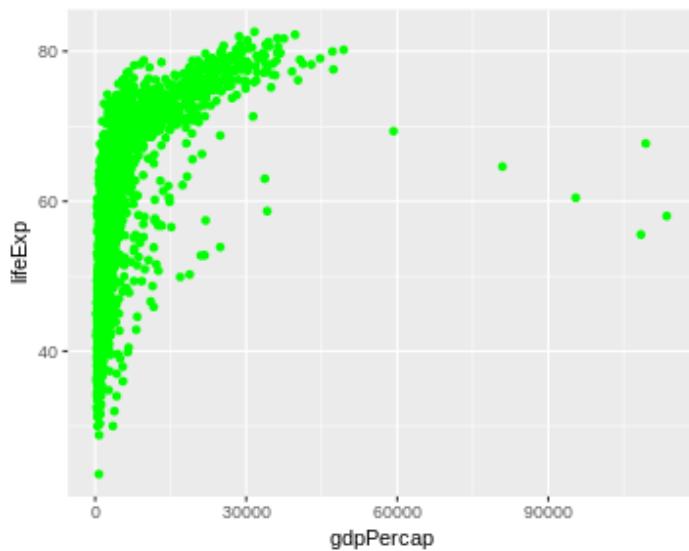
```
ggplot(gapminder, aes(gdpPercap  
geom_point(aes(col = continent)) +  
geom_smooth() +  
scale_x_log10()
```



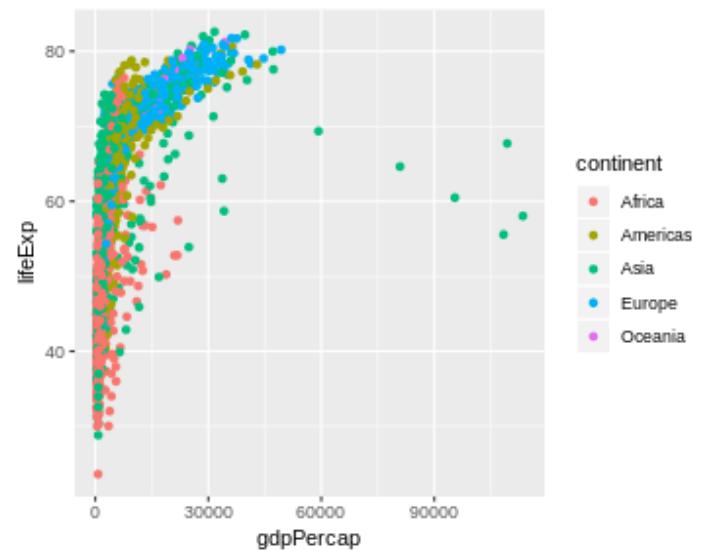
aes(): specifying colour manually

col = "green" versus aes(col = continent)

```
ggplot(gapminder, aes(gdpPercap  
geom_point(col = "green")
```



```
ggplot(gapminder, aes(gdpPercap  
geom_point(aes(col = continent
```

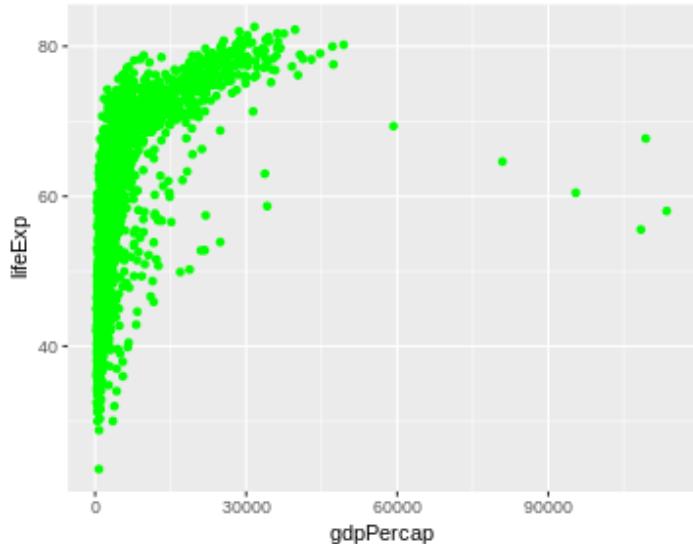


continent

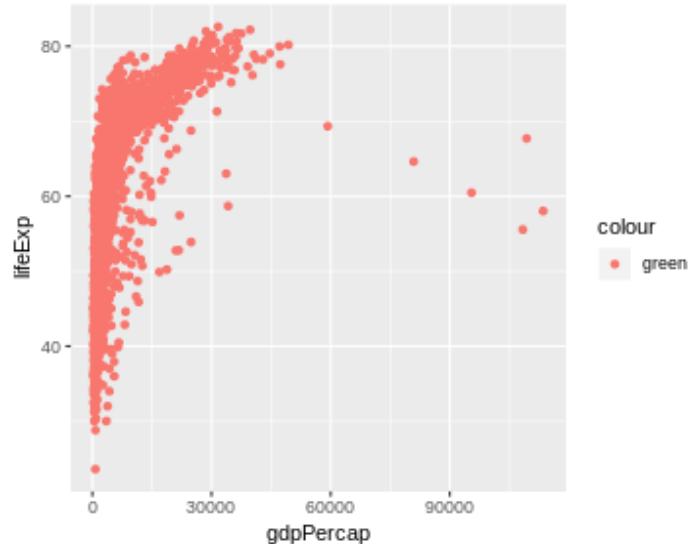
- Africa
- Americas
- Asia
- Europe
- Oceania

aes(): specifying colour manually

```
ggplot(gapminder, aes(gdpPercap  
geom_point(col = "green"))
```

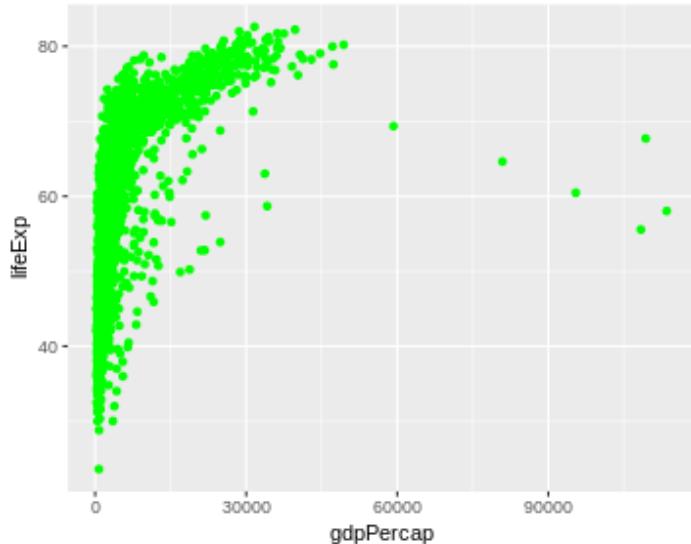


```
ggplot(gapminder, aes(gdpPercap  
geom_point(aes(col = "green"))
```

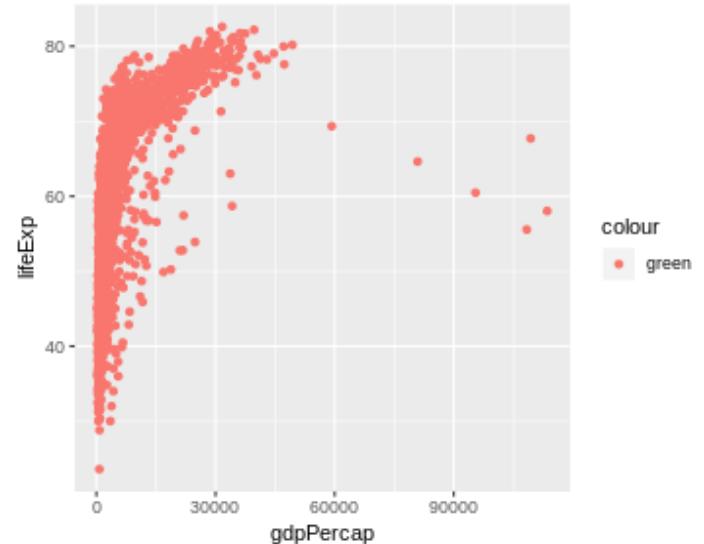


aes(): specifying colour manually

```
ggplot(gapminder, aes(gdpPercap  
geom_point(col = "green"))
```



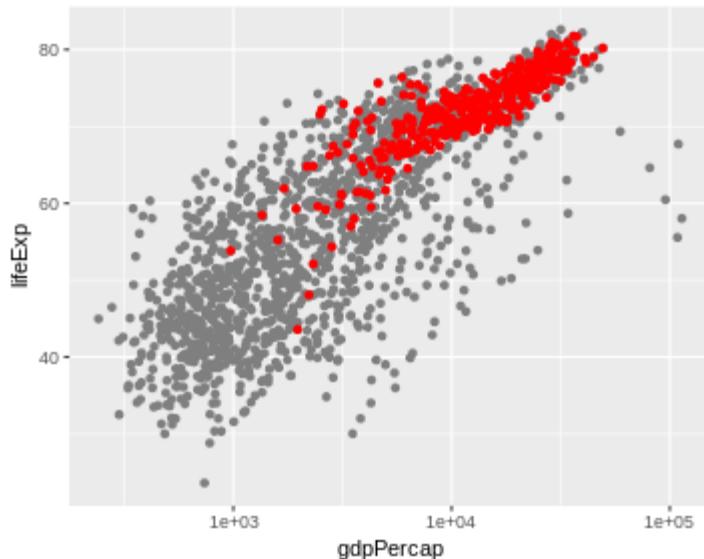
```
ggplot(gapminder, aes(gdpPercap  
geom_point(aes(col = "green"))
```



Good resource for choosing colours: <http://colorbrewer2.org>

Similarly: every layer can be based on different data

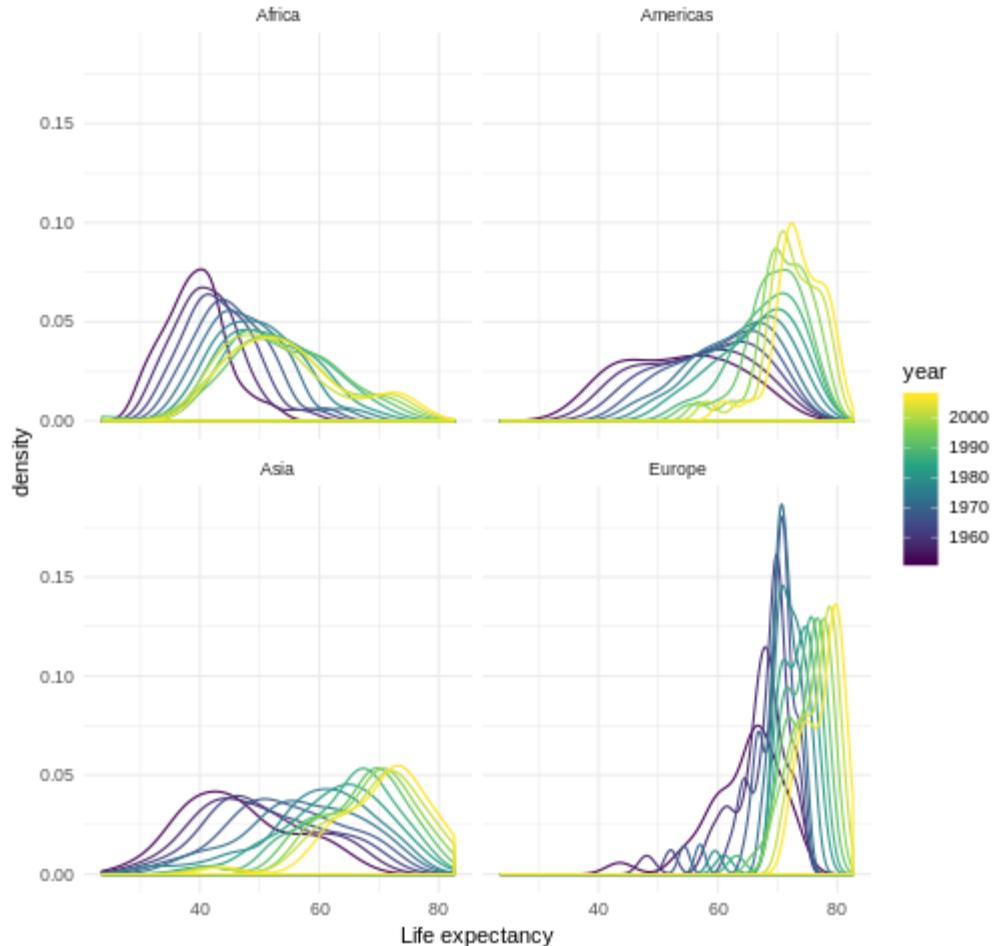
```
df_europe <- subset(gapminder, continent == "Europe")  
  
ggplot() +  
  geom_point(aes(gdpPercap, lifeExp), data = gapminder, col = "grey")  
  geom_point(aes(gdpPercap, lifeExp), data = df_europe, col = "red")  
  scale_x_log10()
```



Tweaking your plot appearance

- Combining layers
- Add transparency using `alpha = 0.5`
- Changing the default theme:
 - `+ theme_bw()`
 - `+ theme_classic()`
 - etc, see also the `ggthemes` package
- Check out the `ggplot2` cheat sheet by RStudio
- Axis limits
 - `+ coord_cartesian(ylim = c(-3, 5))`
- Title, subtitle, caption
 - `+ labs(title = "", subtitle = "", x = "", y = "")`

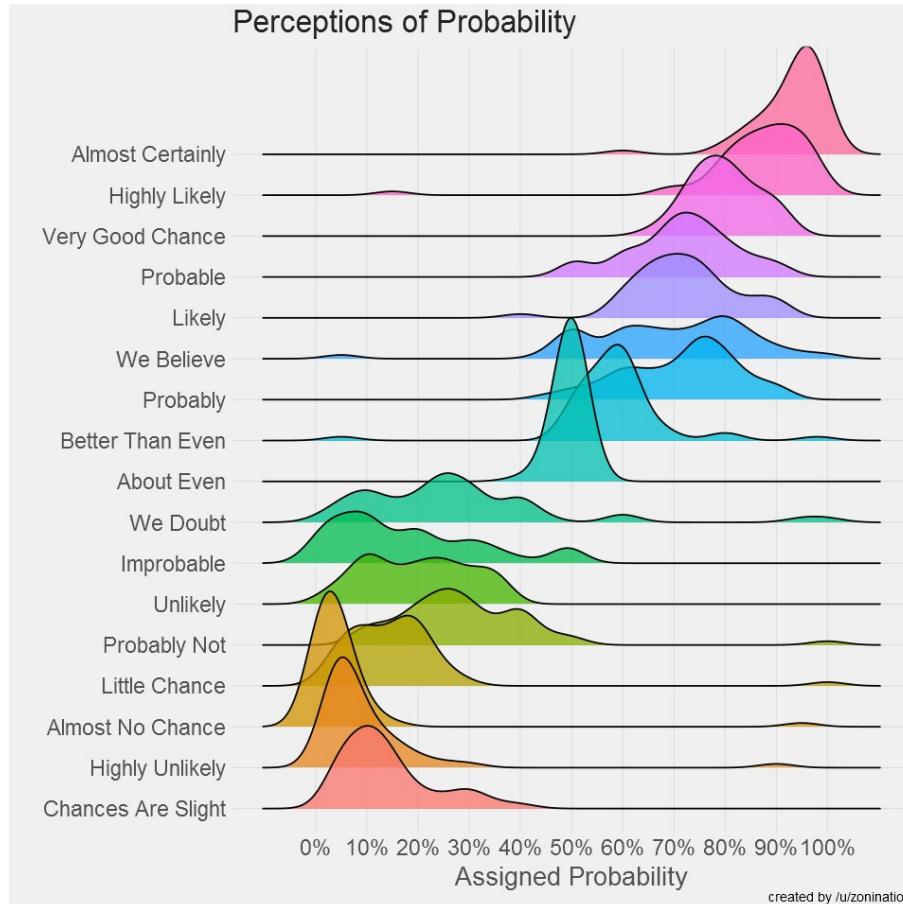
My title



ggplot2 extensions

<http://www.ggplot2-exts.org/>

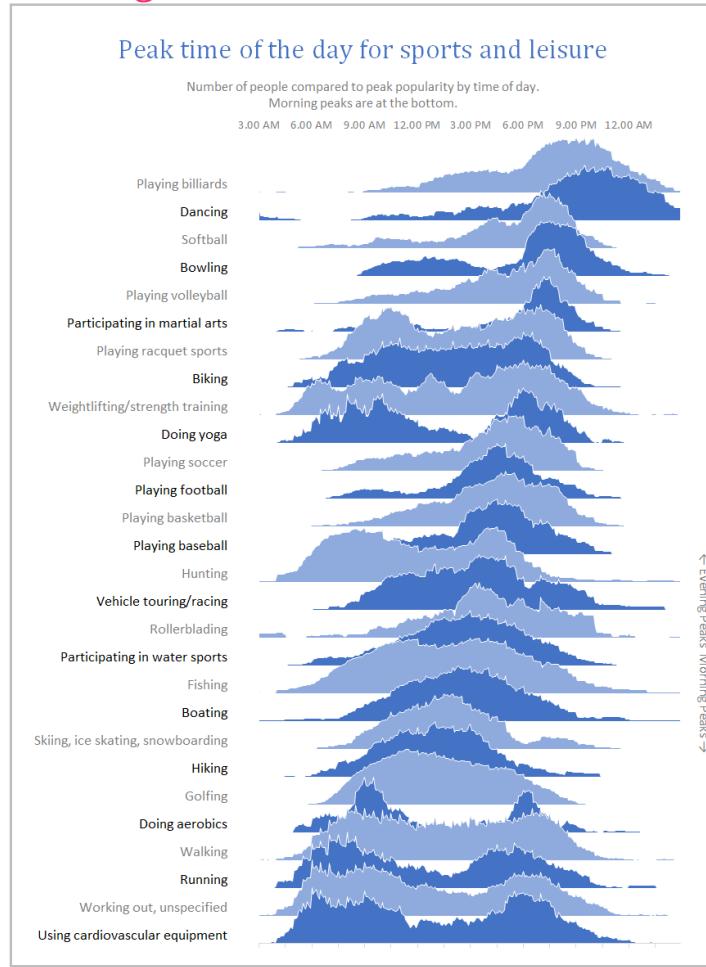
ggridges



ggplot2 extensions

<http://www.ggplot2-exts.org/>

ggridges

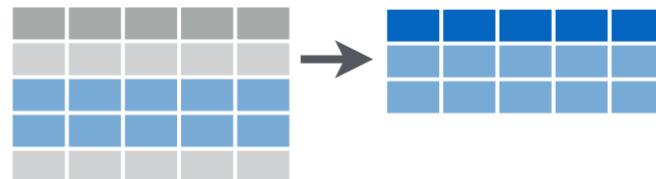


Very brief introduction:

Data transformation using dplyr

Subset rows with filter()

Subset Observations (Rows)



Example:

```
filter(gapminder, continent == "Europe")
```

```
filter(gapminder, continent == "Europe", year > 2000)
```

Subset columns with select()

Subset Variables (Columns)



Example:

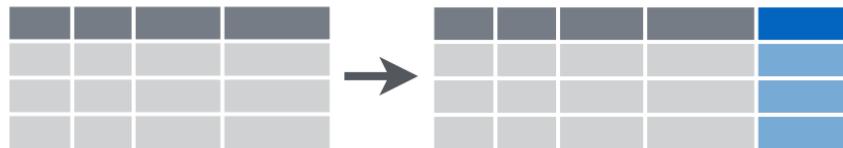
```
select(gapminder, year, continent, lifeExp)
```

Or exclude certain columns

```
select(gapminder, -gdpPercap)
```

Add a new column with mutate()

Make New Variables



Example:

```
mutate(gapminder, total_gdp = pop * gdpPercap)
```

group_by() and summarise()



Example:

```
grouped_df <- group_by(gapminder, continent)
summarise(grouped_df, count = n(), mean_lifeExp = mean(lifeExp))
```

```
## # A tibble: 5 x 3
##   continent count mean_lifeExp
##   <fct>     <int>      <dbl>
## 1 Africa      624       48.9
## 2 Americas    300       64.7
## 3 Asia        396       60.1
## 4 Europe      360       71.9
## 5 Oceania     24        74.3
```

Pipes %>%

Introducing `%>%`, to help you write code in a way that easier to read and understand.

`g(f(x))` can be rewritten using the pipe `x %>% f() %>% g()`

Pipes %>%

Introducing %>%, to help you write code in a way that easier to read and understand.

g(f(x)) can be rewritten using the pipe x %>% f() %>% g()

E.g. the following

```
filtered_df <- filter(gapminder, continent != "Oceania")
grouped_df <- group_by(filtered_df, continent)
summarise(grouped_df, count = n(), mean_lifeExp = mean(lifeExp))
```

can be rewritten

```
gapminder %>%
  filter(continent != "Oceania") %>%
  group_by(continent) %>%
  summarise(count = n(), mean_lifeExp = mean(lifeExp))
```

Pipes %>%

Introducing %>%, to help you write code in a way that easier to read and understand.

g(f(x)) can be rewritten using the pipe x %>% f() %>% g()

E.g. the following

```
filtered_df <- filter(gapminder, continent != "Oceania")
grouped_df <- group_by(filtered_df, continent)
summarise(grouped_df, count = n(), mean_lifeExp = mean(lifeExp))
```

can be rewritten

```
gapminder %>%
  filter(continent != "Oceania") %>%
  group_by(continent) %>%
  summarise(count = n(), mean_lifeExp = mean(lifeExp))
```

- No need for storing intermediate variables.
- Easier to read and write.

Great Resources:

- R for Data Science <http://r4ds.had.co.nz/>
- Fundamentals of Data Visualization <https://serialmentor.com/dataviz>
- TED talk by Hans Rosling
https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen
- Talk by John Rauser on How Humans See Data
<https://www.youtube.com/watch?v=fSgEeI2Xpdc>

 @kasparmartens

 kaspar.martens@gmail.com