

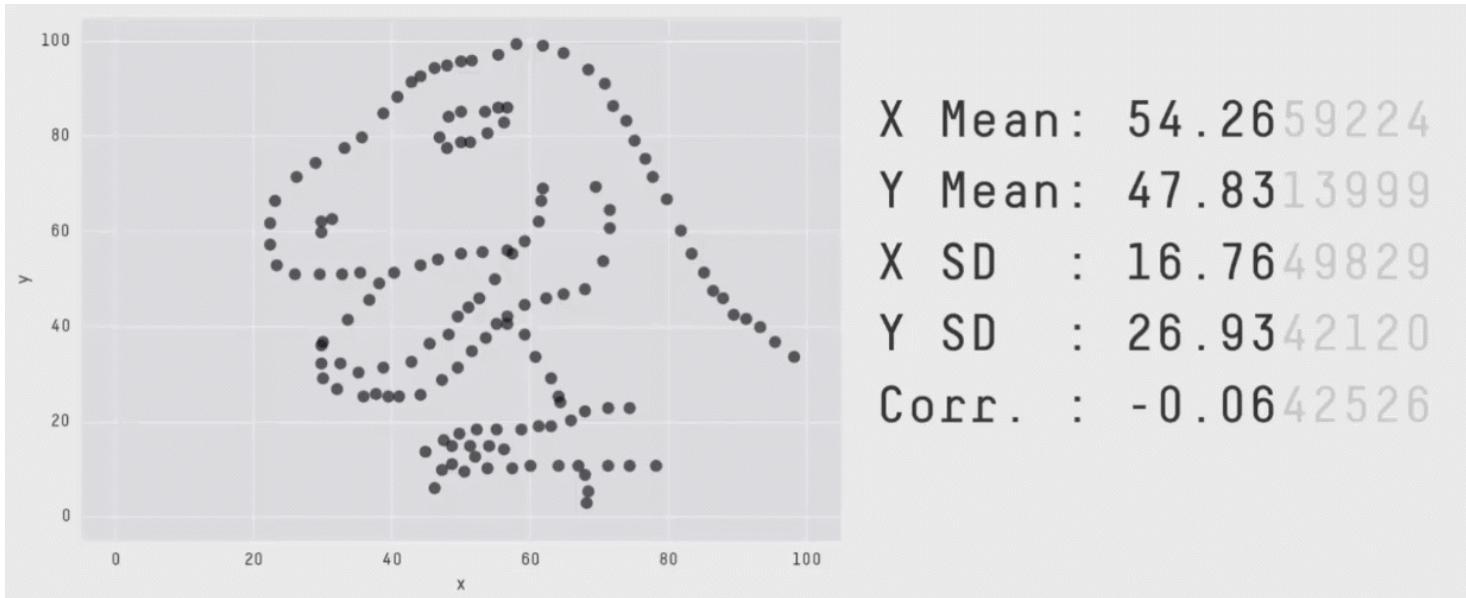
Data visualisation with `ggplot2`

Francisco Rodríguez-Sánchez

<https://frodriguezsanchez.net>

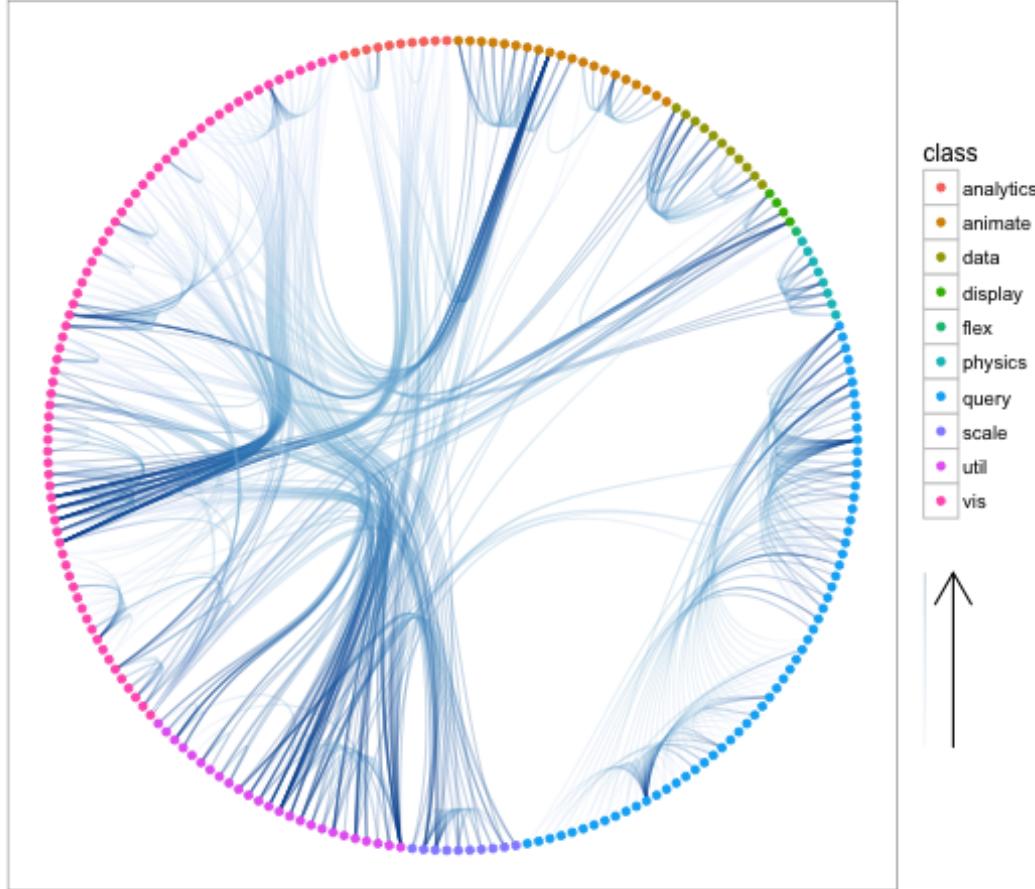
@frod_san

Always plot data!



<https://github.com/stephlocke/datasauRus>

Made with ggplot



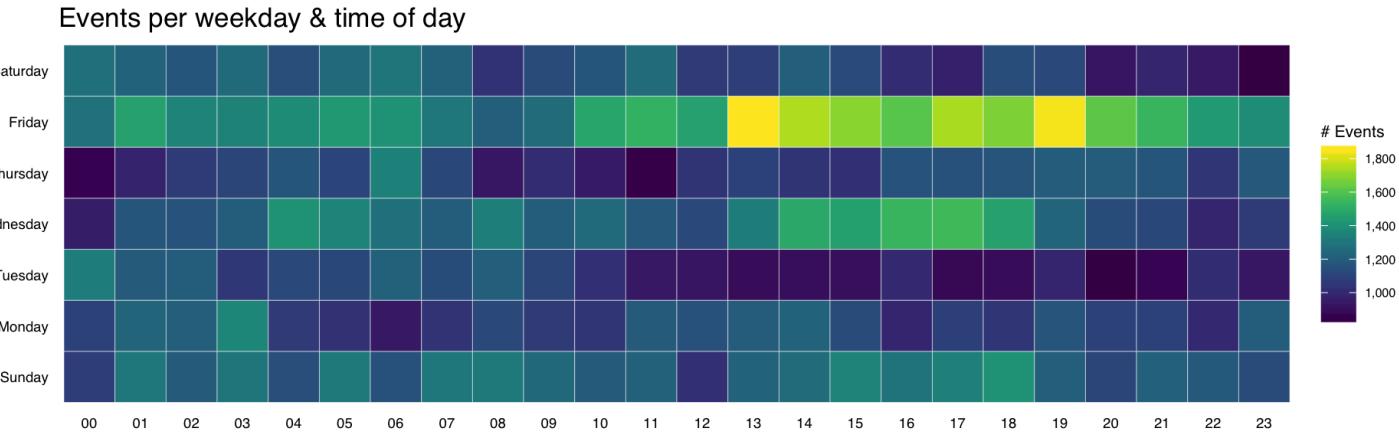
<https://github.com/thomasp85/ggraph>

Made with ggplot



<http://spatial.ly/2012/02/great-maps-ggplot2/>

Made with ggplot

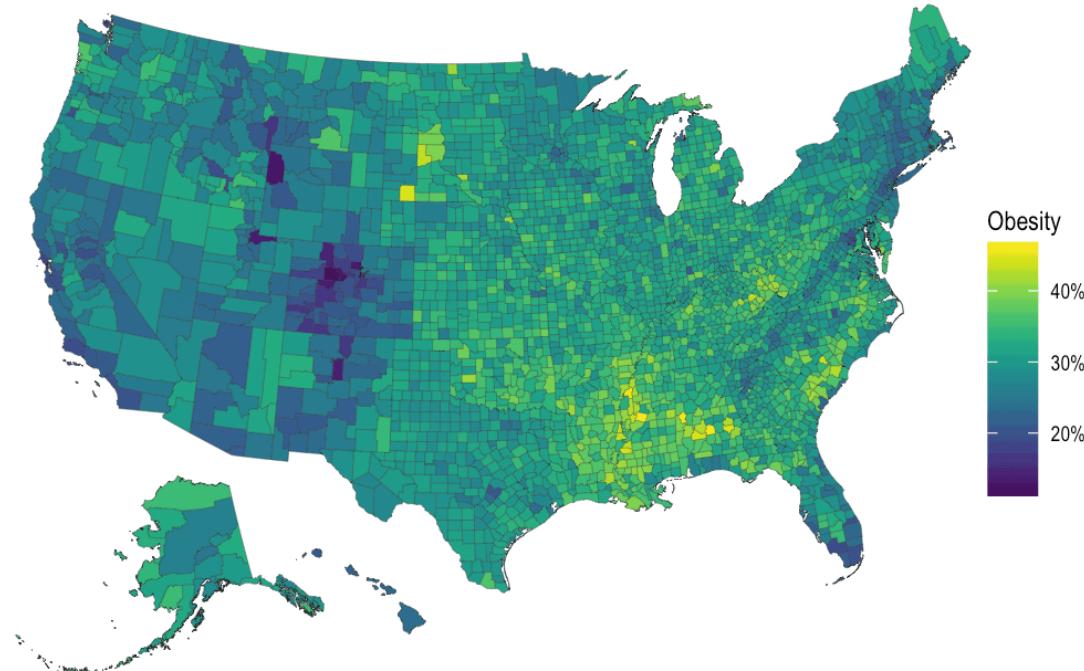


<https://rud.is/b/2016/02/14/making-faceted-heatmaps-with-ggplot2/>

Made with ggplot

U.S. Obesity Rate by County (2012)

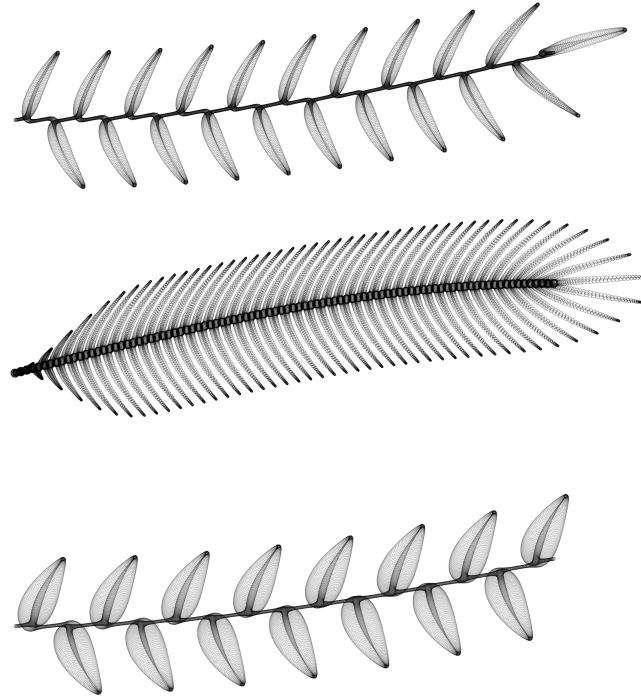
Content source: Centers for Disease Control and Prevention



Data from http://www.cdc.gov/diabetes/atlas/countydata/County_ListofIndicators.html

<https://rud.is/b/2016/03/29/easier-composite-u-s-choropleths-with-albersusa/>

Made with ggplot



<https://github.com/marcusvolz/mathart>

Why ggplot

- Extremely powerful and flexible
- Consistent (grammar of graphics)
- Very powerful user base and active development

Very good documentation and tutorials

- Official `ggplot2` documentation
- `ggplot2` book
- Data visualisation chapter in R for Data Science
- R graphics cookbook and `Cookbook for R`
- Data visualization: a practical introduction (K. Healy)
- Fundamentals of data visualization (C. Wilke)

Cheatsheet

Data Visualization with ggplot2 Cheat Sheet

R Studio

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>) +  
  geom_function(mapping = aes(...),  
    stat = <STAT>,  
    position = <POSITION>  
  ) +  
  geom_aesthetic(mapping = <MAP>,  
    effect = <EFFECT>,  
    scale = <SCALE>,  
    theme = <THEME>)
```

Required

Not required, unless defaults supplied

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

geom(mapping = <MAP>, data = <DATA>, geom = <GEOM>)
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

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.

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))
c <- geom_blank()
Useful for expanding limits
- b <- geom_curve(bendyend = list(lat = 1, xend -> x, yend -> y, alpha, angle, color, curvature, linetype, size))
- c <- geom_polygon(aes(group = group))
x, y, alpha, color, group, linetype, size
- d <- geom_rect(aes(minx = long, miny = ymin, maxx = long + 1, maxy = ymax), alpha, color, fill, group, linetype, size)
- e <- geom_ribbon(aes(ymin = yimmunemploy - 900, ymax = yimmunemploy + 900), alpha, color, fill, group, linetype, size)
- f <- geom_rect(aes(xmin = x, ymin = y, xmax = x + 1, ymax = y + 1), alpha, color, fill, group, linetype, size)
- g <- geom_point()
x, y, alpha, color, fill, shape, size
- h <- geom_quandle()
x, y, alpha, color, group, linetype, size, stroke
- i <- geom_hex()
x, y, alpha, colour, fill, shape, size
- j <- geom_density2d()
x, y, alpha, color, fill, shape, size
- k <- geom_hex2d()
x, y, alpha, colour, group, linetype, size, weight

Two Variables

- Continuous X, Continuous Y
e <- ggplot(mpg, aes(cty, hwy))
nudge_y <- 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family, fontface, fontstyle, lineweight, size, yintercept
- Continuous Bivariate Distribution
h <- ggplot(diamonds, aes(carat, price))
x, y, alpha, color, fill, linetype, size, weight
- Continuous Function
i <- ggplot(economics, aes(date, unemploy))
j <- geom_area()
x, y, alpha, color, fill, linetype, size
- geom_line()
x, y, alpha, color, group, linetype, size
- geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

Visualizing error

- of <- data.hanei(gp = c("A", "B"), fit = 4.5, se = 1.2)
j <- ggplot(of, aes(gp, fit, ymin = fit - se, ymax = fit + se))
k <- geom_crossbar(latten = 2)
x, ymin, ymax, alpha, color, fill, group, linetype, size
- l <- geom_baplot()
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
- m <- geom_hexplot()
x, y, alpha, color, fill, group, linetype, size, weight
- n <- geom_violin(effect = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight
- o <- geom_dotplot()
x, y, alpha, color, fill
- p <- geom_freqpoly()
x, y, alpha, color, group, linetype, size
- q <- geom_histogram(bandwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
- r <- geom_qq(sample = hwy)
x, y, alpha, color, fill, linetype, size, weight
- s <- geom_boxplot()
x, y, alpha, color, fill, group, linetype, size
- t <- geom_hex()
x, y, alpha, color, fill, group, linetype, size, weight
- u <- geom_hex2d()
x, y, alpha, colour, group, linetype, size, weight

Discrete X, Continuous Y

- v <- ggplot(mpg, aes(class, hwy))
w <- geom_col()
x, y, alpha, color, fill, group, linetype, size
- x <- geom_baplot()
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
- y <- geom_hexplot()
x, y, alpha, color, fill, group, linetype, size, weight
- z <- geom_violin()
x, y, alpha, color, fill, group, linetype, size, weight
- aa <- geom_dotplot()
x, y, alpha, color, fill, group, linetype, size
- bb <- geom_hex()
x, y, alpha, color, fill, group, linetype, size, weight
- cc <- geom_hex2d()
x, y, alpha, colour, group, linetype, size, weight

Discrete X, Discrete Y

- dd <- ggplot(diamonds, aes(cut, color))
ee <- geom_count()
x, y, alpha, color, fill, shape, size, stroke

Maps

- data <- data.frame(murder = USArrests\$Murder,
state = USArrests\$State, nrares = USArrests\$NRArrests)
map <- map_data("us-states")
l <- geom_map(mapping = map, id = state, map = map) +
expand_limits(x = mapping\$x, y = mapping\$y)
- ff <- geom_raster(aes(fill = z), hjust = 0.5,
vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill
- gg <- geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight
- hh <- geom_tile(aes(fill = z))
x, y, alpha, color, fill, linetype, size, width

Learn more at docs.ggplot2.org and www.ggplot2-exists.org • ggplot2 2.1.0 • Updated 11/16

<https://www.rstudio.com/resources/cheatsheets/>

Repos of figures + code

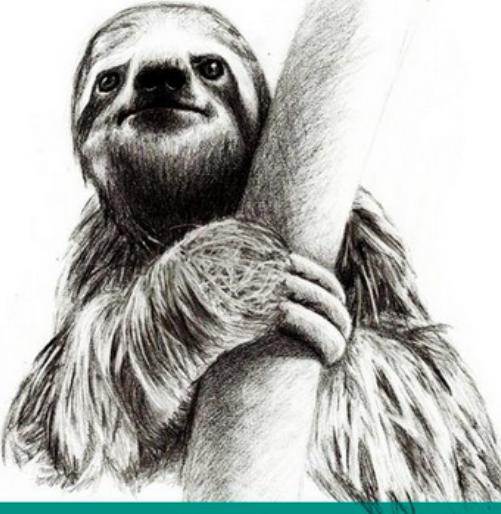
- From Data to Viz
- The R graph gallery
- R graphics cookbook
- Cookbook for R: Graphs
- Graphical data analysis with R
- R graph catalog

Find answers in Stack Overflow, Rstudio Community, R4DS...

DEV The Practical Dev
@ThePracticalDev

The last programming book you'll ever need

Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting
from Stack Overflow

Building a ggplot figure

Our example dataset: paper planes flying experiment

```
library(paperplanes)
data(paperplanes)
head(paperplanes)
```

```
# A tibble: 6 × 8
  id hour person gender age plane      paper distance
  <int> <fct>  <chr>   <fct> <dbl> <chr>      <int>     <dbl>
1 1 [17,18) Roland male     30 Standard80     80      7.8
2 2 [17,18) Astrid female   female    30 Concorde120  120      2.7
3 3 [17,18) Roland male     30 Standard120    120      9.2
4 4 [17,18) Isabella female female   48 Standard120  120       6
5 5 [17,18) Fabienne female  female   17 Standard120  120      7.3
6 6 [17,18) Fabienne female  female   17 Standard120  120      7.8
```

Ensuring 'paper' is factor, not numeric

Using dplyr:

```
paperplanes <- paperplanes %>%  
  mutate(paper = as.factor(paper))
```

R base:

```
paperplanes$paper <- as.factor(paperplanes$paper)
```

Data must be a tidy data frame

country	year	cases	population
Afghanistan	1999	745	1998071
Afghanistan	2000	2666	2059360
Brazil	1999	37737	17206362
Brazil	2000	80488	174504898
China	1999	212258	1272515272
China	2000	213766	128042583

variables

country	year	cases	population
Afghanistan	1999	745	1998071
Afghanistan	2000	2666	2059360
Brazil	1999	37737	17206362
Brazil	2000	80488	174504898
China	1999	212258	1272515272
China	2000	213766	128042583

observations

country	year	cases	population
Afghanistan	99	745	1998071
Afghanistan	00	2666	2059360
Brazil	99	37737	17206362
Brazil	00	80488	174504898
China	99	212258	1272515272
China	00	213766	128042583

values

```
tidy::gather(table4, key = "year", value = "cases", "1999", "2000")
```

country	year	cases	1999	2000
Afghanistan	1999	745	745	2666
Afghanistan	2000	2666		2666
Brazil	1999	37737	37737	80488
Brazil	2000	80488		80488
China	1999	212258	212258	213766
China	2000	213766		213766

table4

Calling ggplot

```
library(ggplot2)  
ggplot(paperplanes)
```

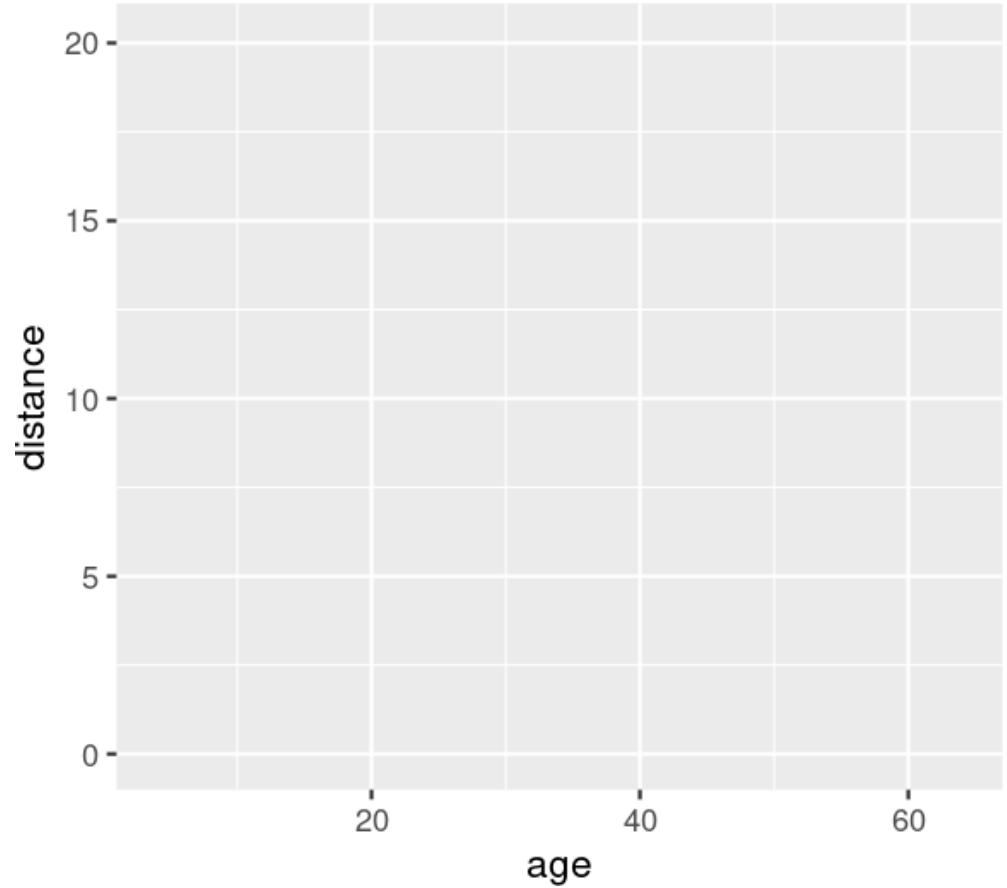
First argument is a tidy data frame

```
ggplot(paperplanes)
```

What variables as axes?

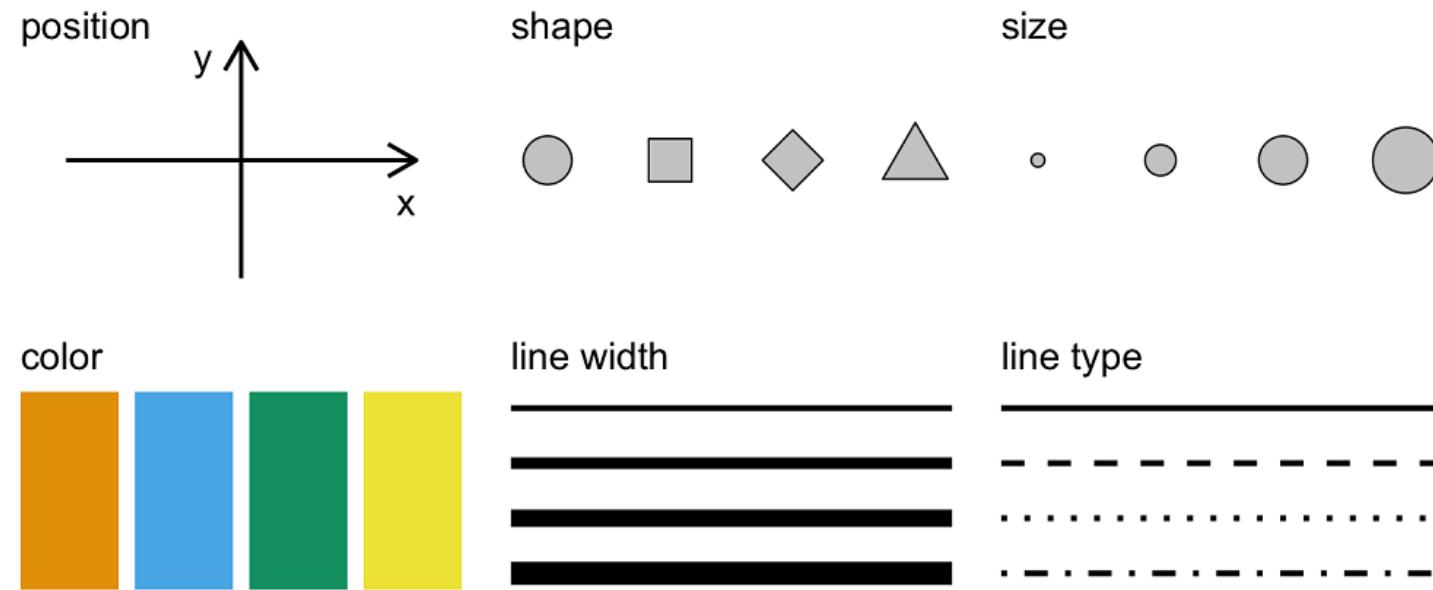
Note syntax: + followed by new line

```
ggplot(paperplanes) +  
  aes(x = age, y = distance)
```



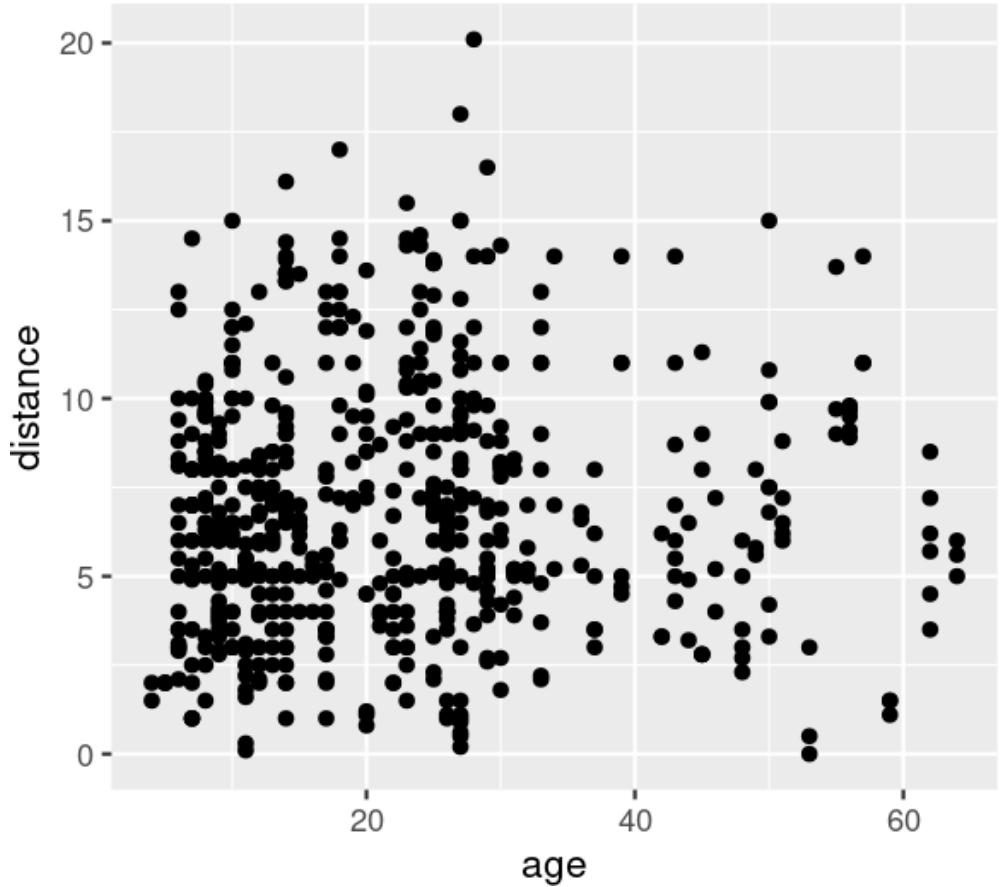
Aesthetics (*aes*) map data variables (*age*, *distance*) to graphic elements (*axes*)

```
ggplot(paperplanes) +  
  aes(x = age, y = distance)
```



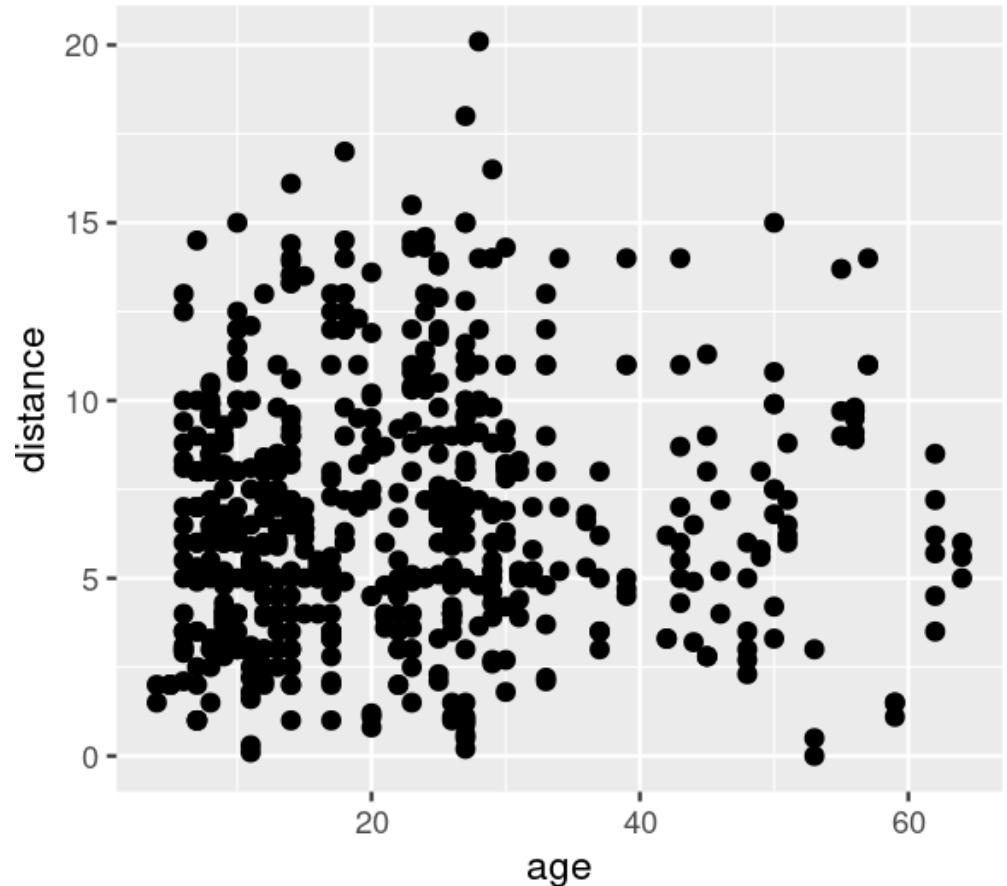
Add layers (geoms)

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point()
```



Change point size and type

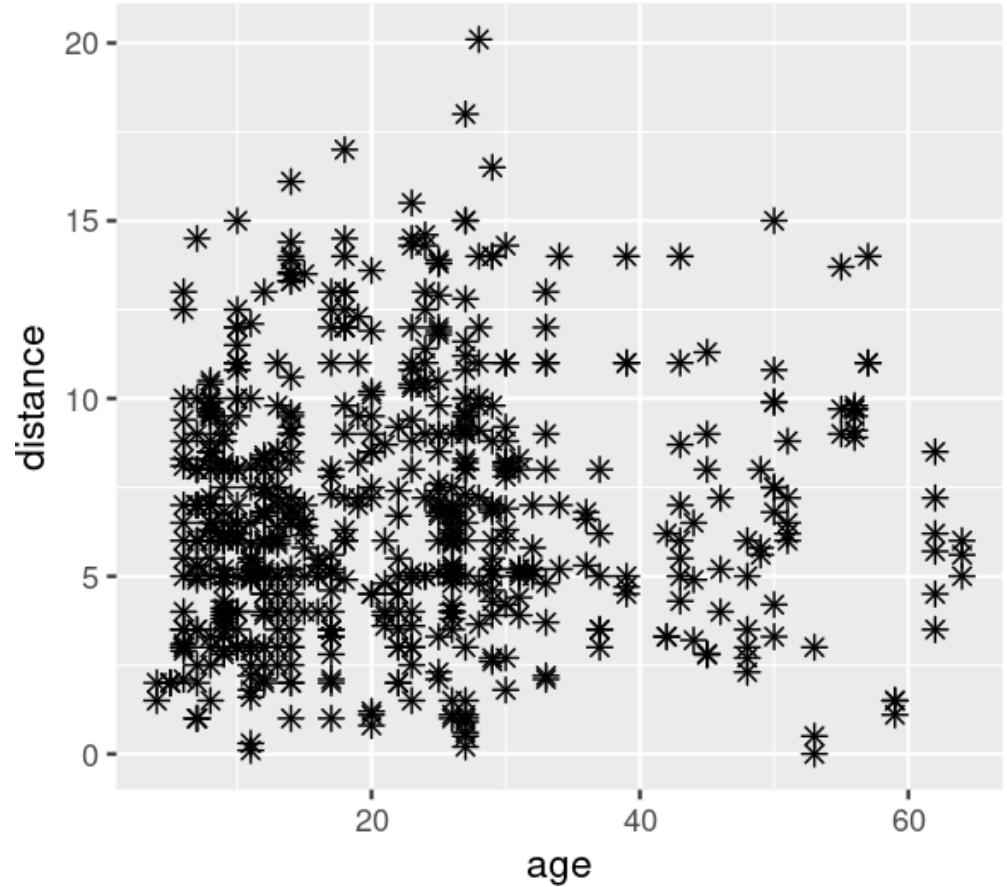
```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(size = 2)
```



Check out `geom_point` help [here](#)

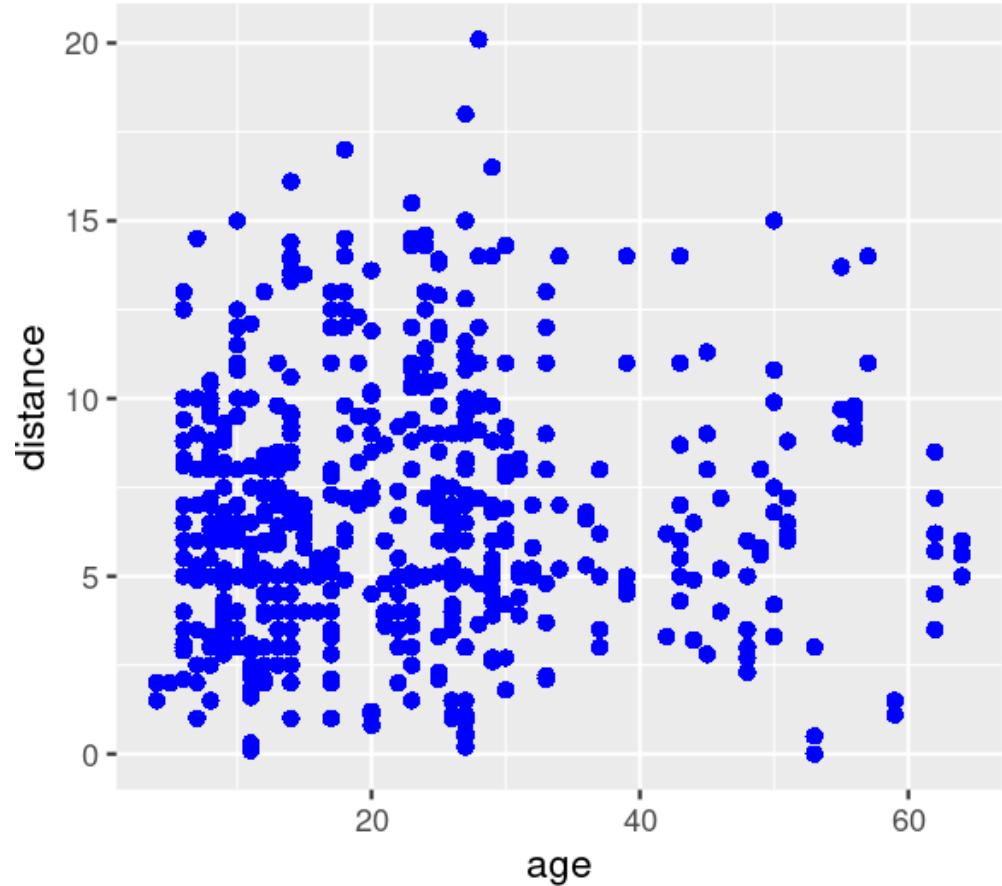
Change point size and type

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(size = 2, shape = 8)
```



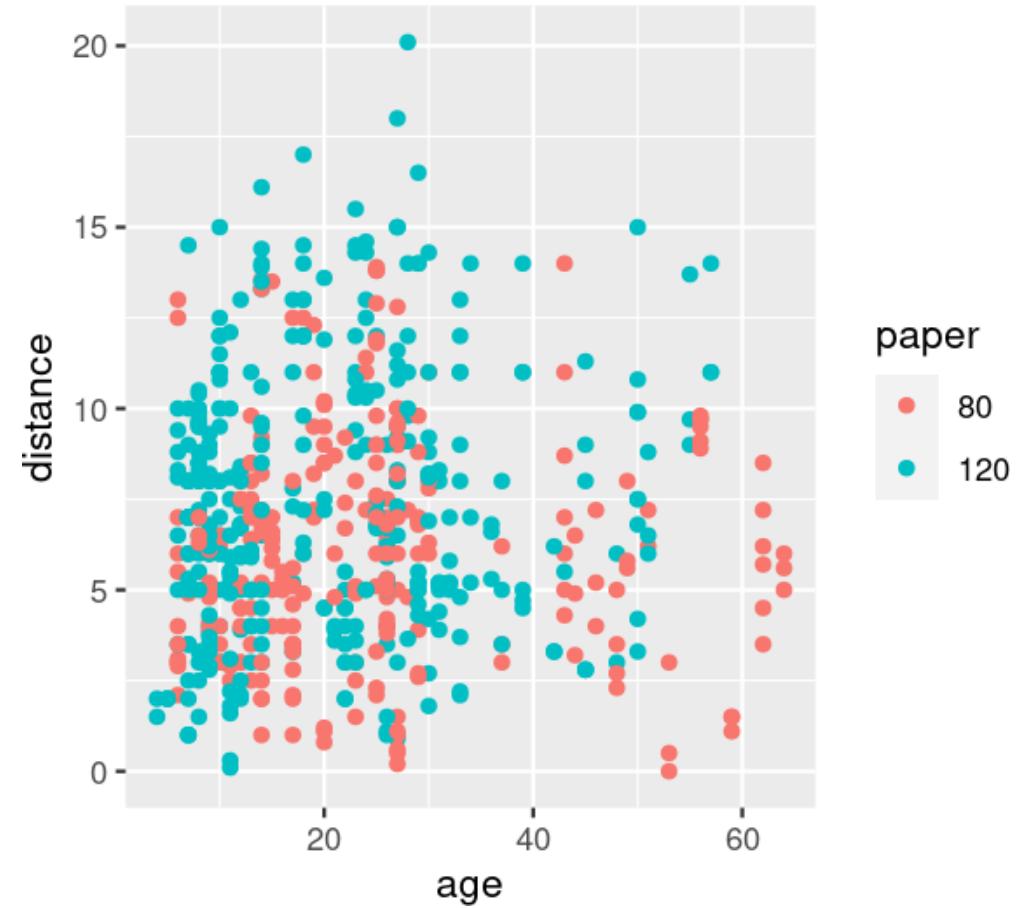
Change point size and type

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(size = 2, shape = 16,  
             colour = "blue")
```



Map geom aesthetics (e.g. colour) to variable

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper))
```



Remember:

'aes' relates some graphical characteristic

(colour, size, shape...)

to a variable in the data

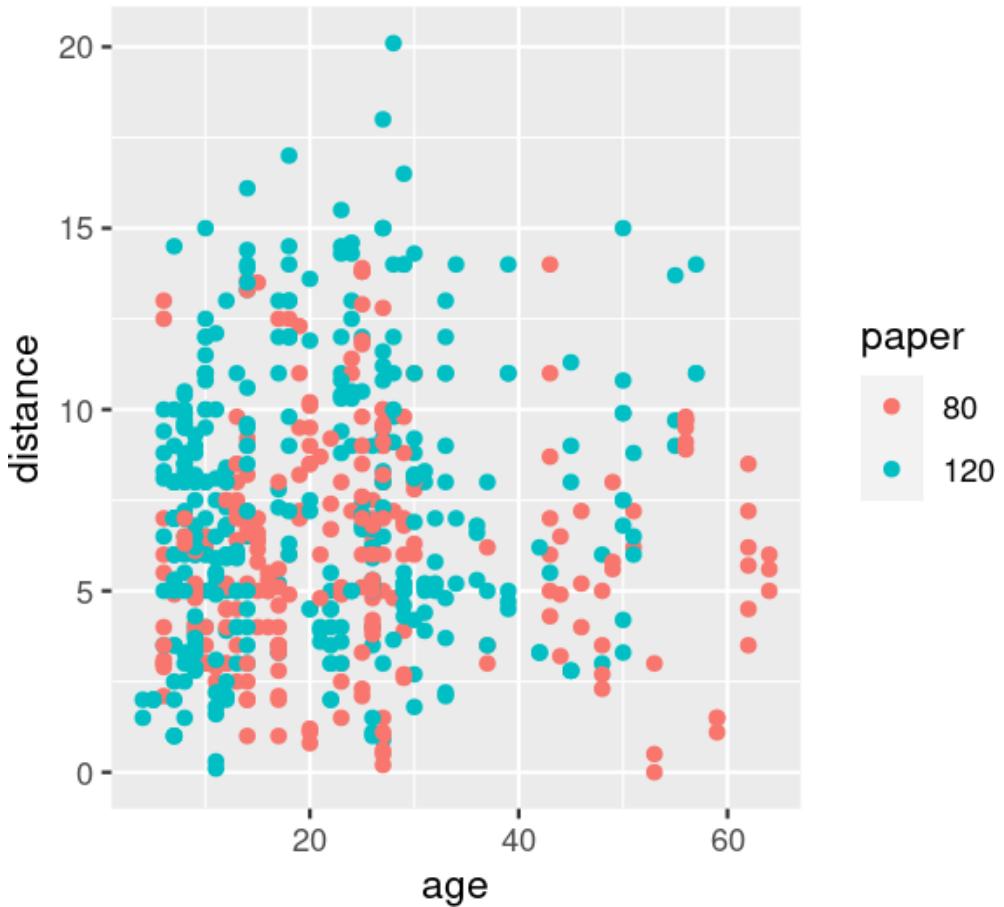
Note difference between

```
geom_point(colour = "blue")
# colour is given a concrete value ('blue')
```

```
geom_point(aes(colour = gender))
# colour maps a *variable* in the data (gender) USING `aes`
```

This works:

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper))
```



This doesn't work:

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(colour = paper)
```

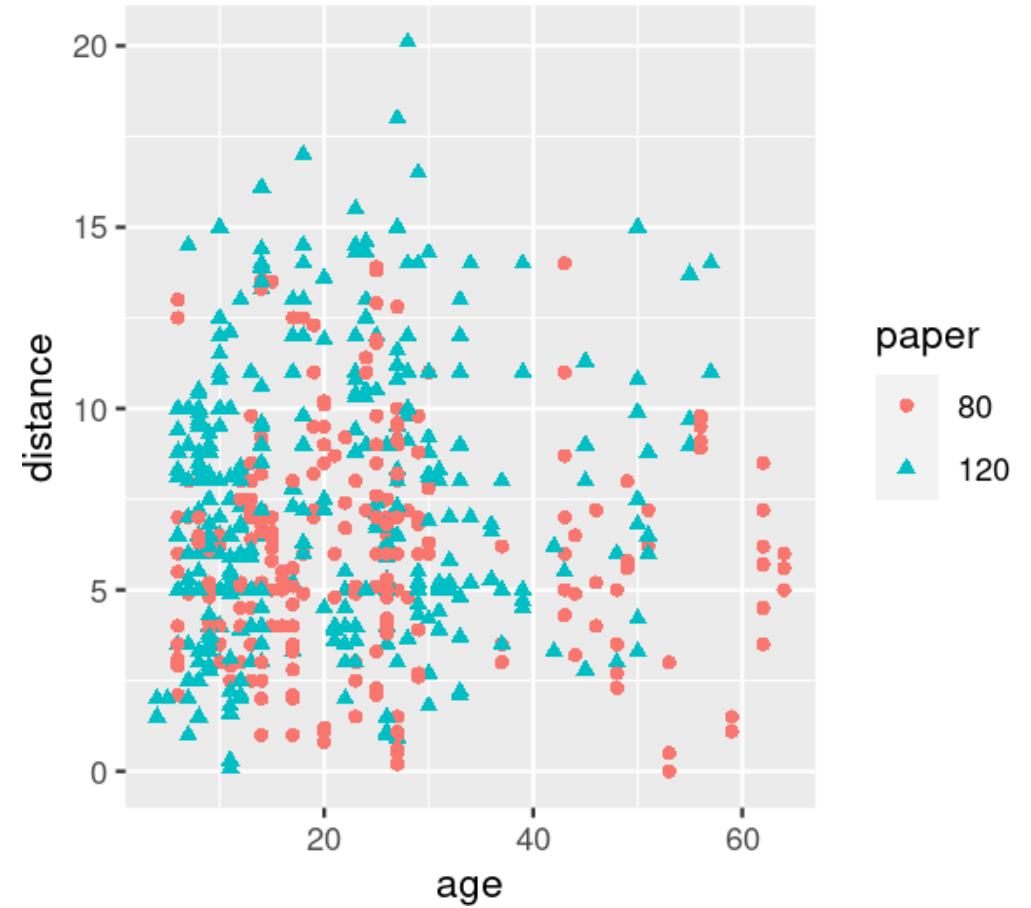
*Error in layer(data = data, mapping = mapping,
stat = stat, geom = GeomPoint, : object 'paper'
not found*

'paper' is a variable in dataframe, hence

must use aes

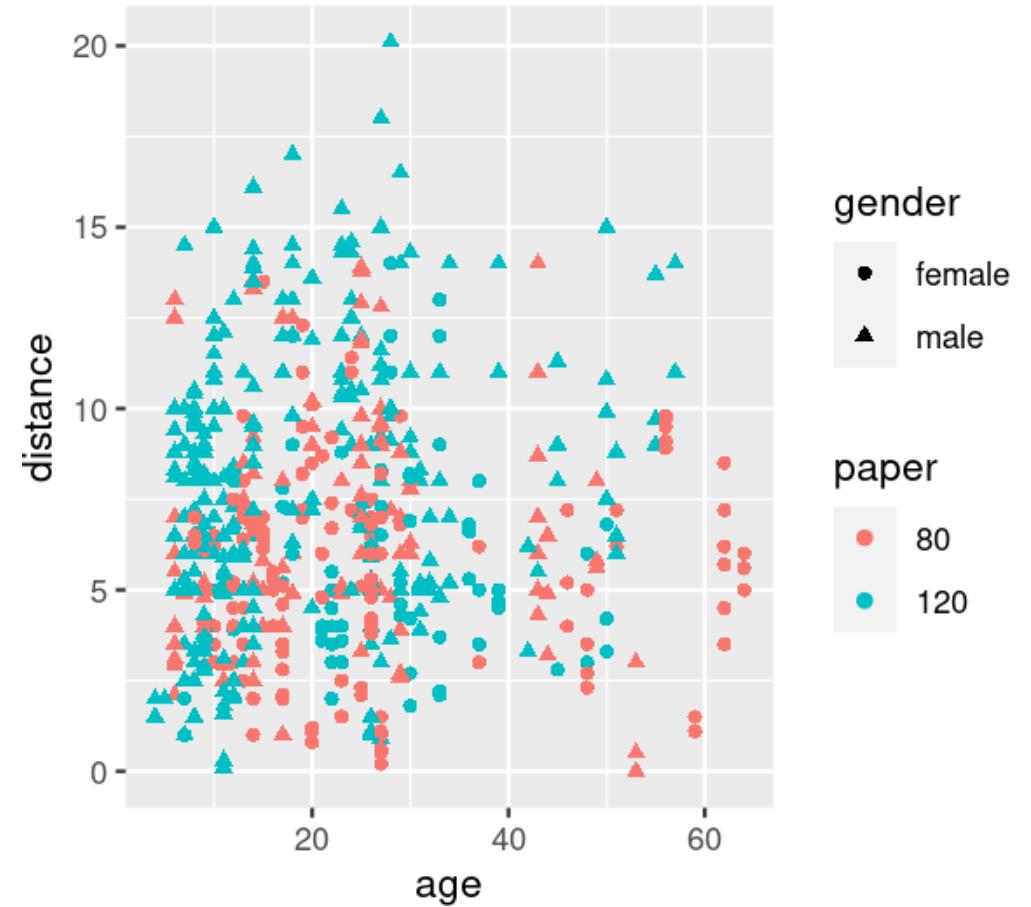
Map geom aesthetics (colour, shape) to variable

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper,  
                 shape = paper))
```



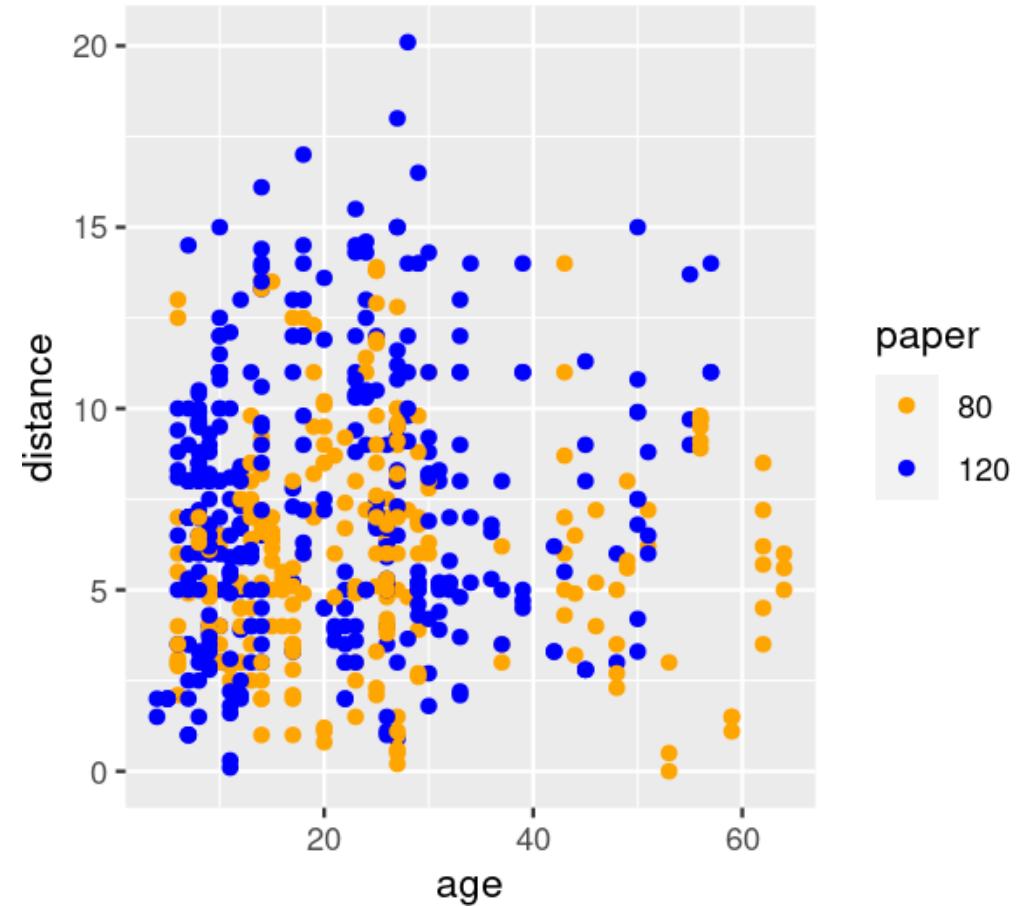
Map geom aesthetics (colour, shape) to variable

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper,  
                 shape = gender))
```



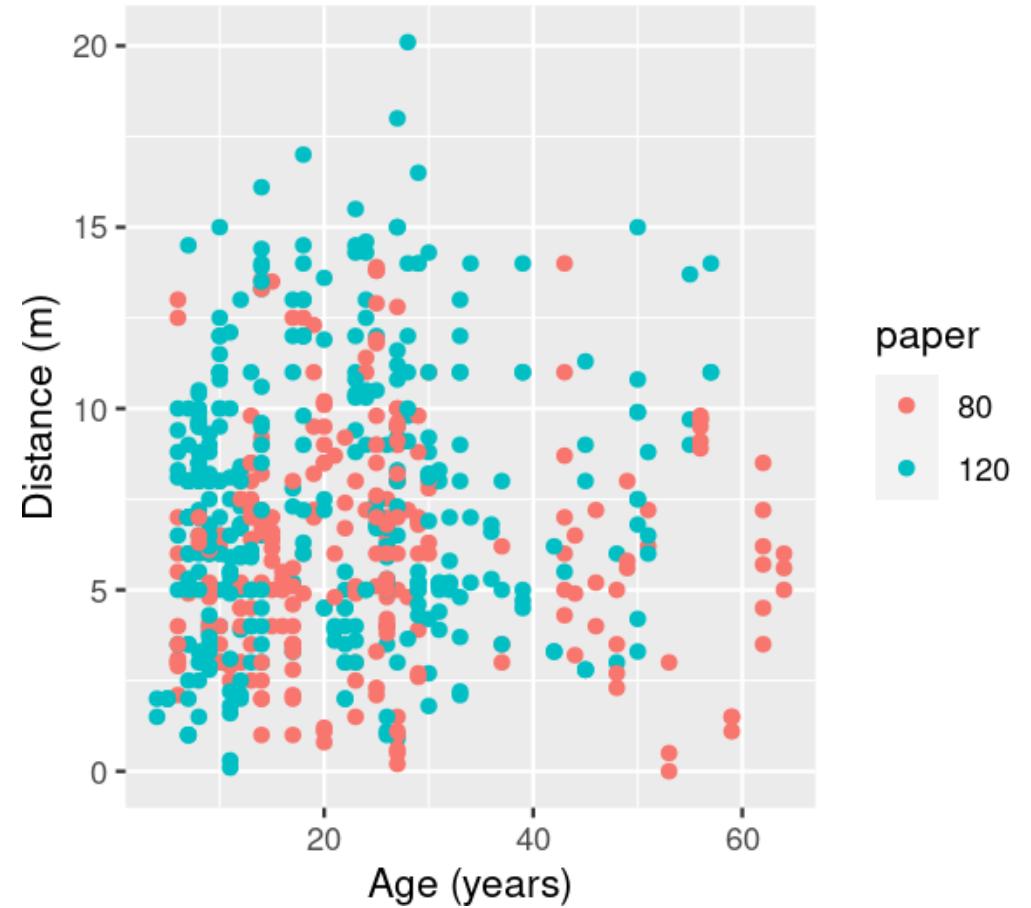
Change colour scale

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper)) +  
  scale_colour_manual(values = c("orange", "blue"))
```



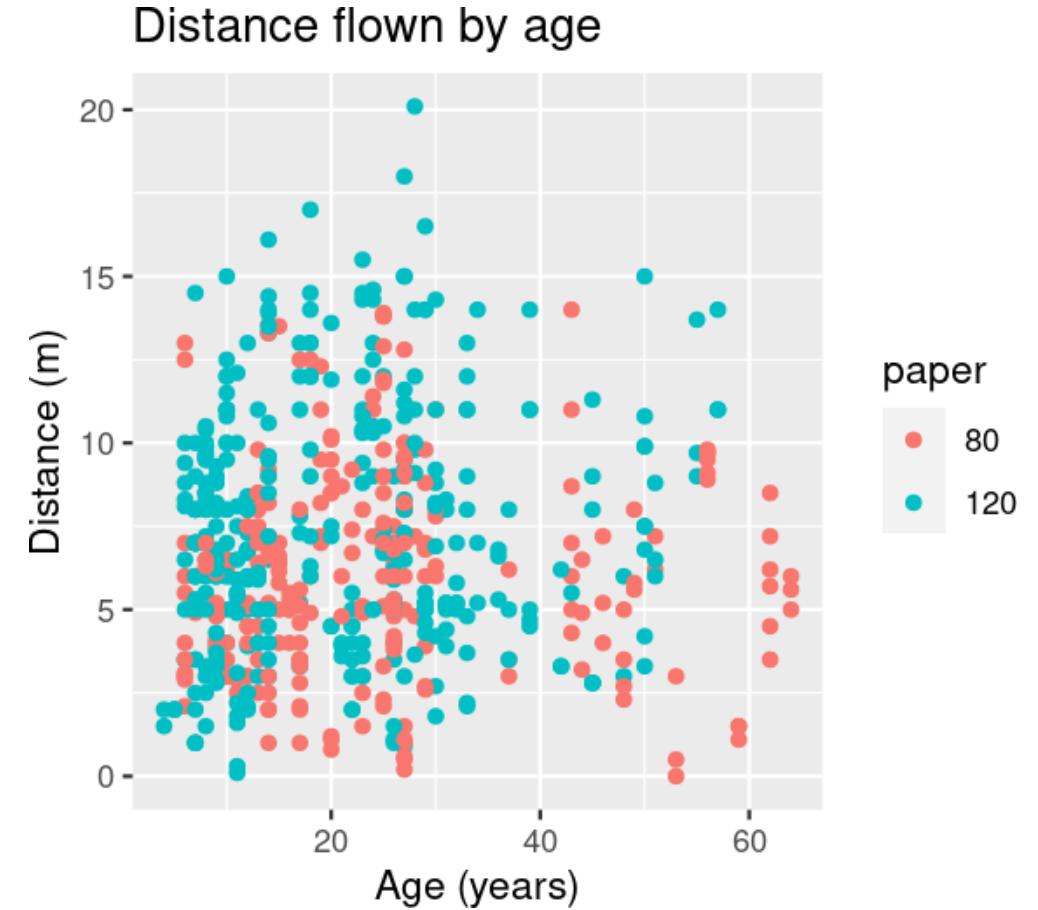
Change axis labels: xlab & ylab

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper)) +  
  labs(x = "Age (years)",  
       y = "Distance (m)")
```



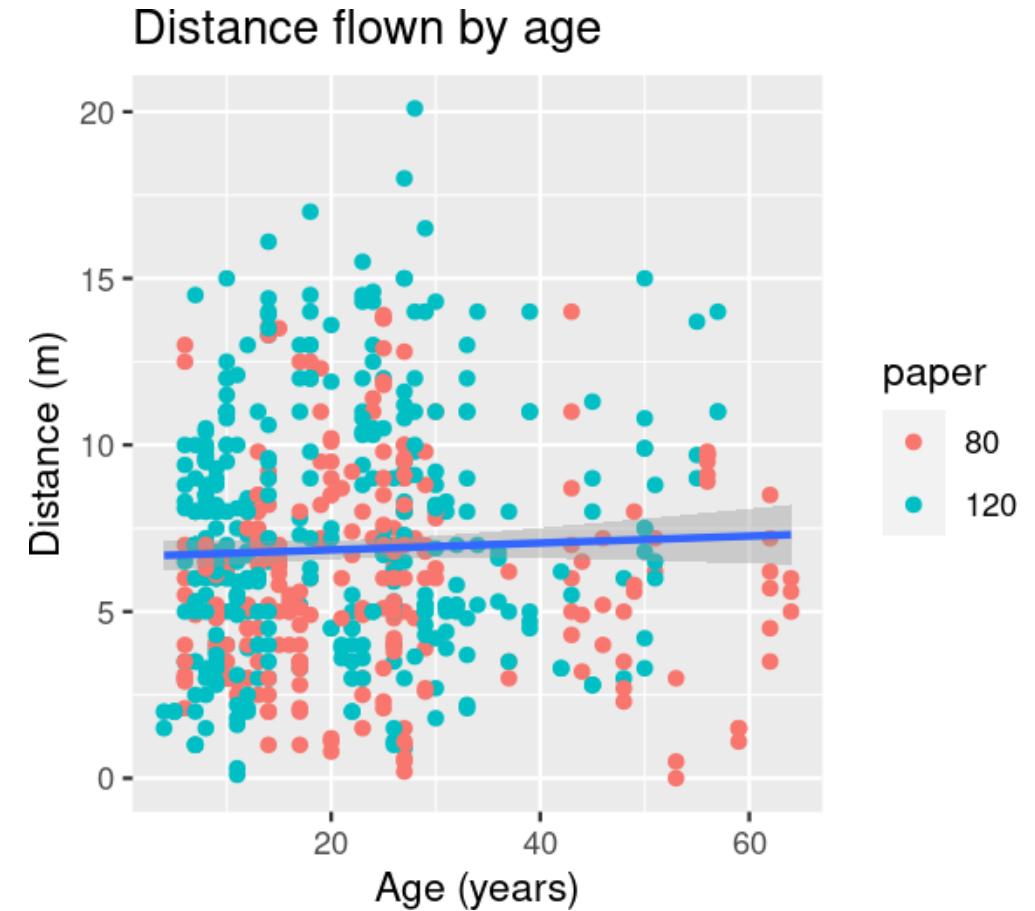
Set title

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper)) +  
  labs(x = "Age (years)",  
       y = "Distance (m)") +  
  labs(title = "Distance flown by age")
```



Adding more layers

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper)) +  
  labs(x = "Age (years)",  
       y = "Distance (m)") +  
  labs(title = "Distance flown by age") +  
  geom_smooth(method = "lm")
```



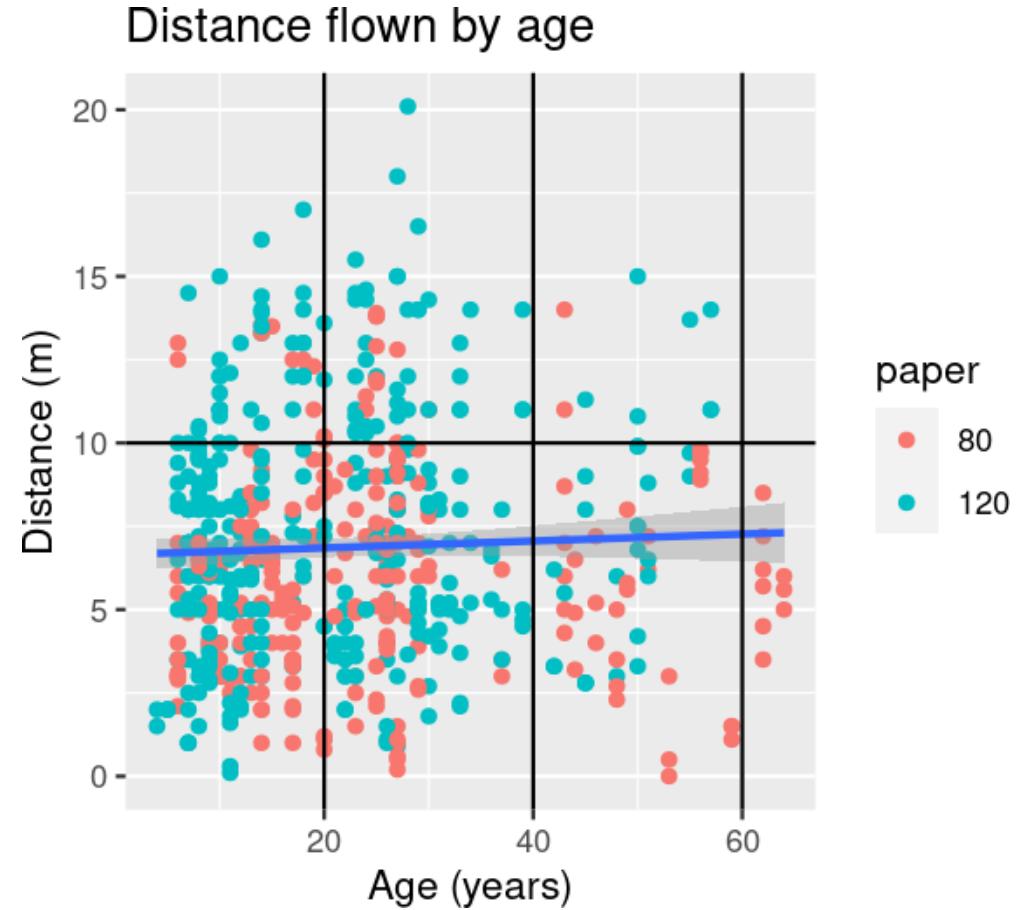
Adding more layers

```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper)) +  
  labs(x = "Age (years)",  
       y = "Distance (m)") +  
  labs(title = "Distance flown by age") +  
  geom_smooth(method = "lm") +  
  geom_vline(xintercept = c(20, 40, 60))
```



Adding more layers

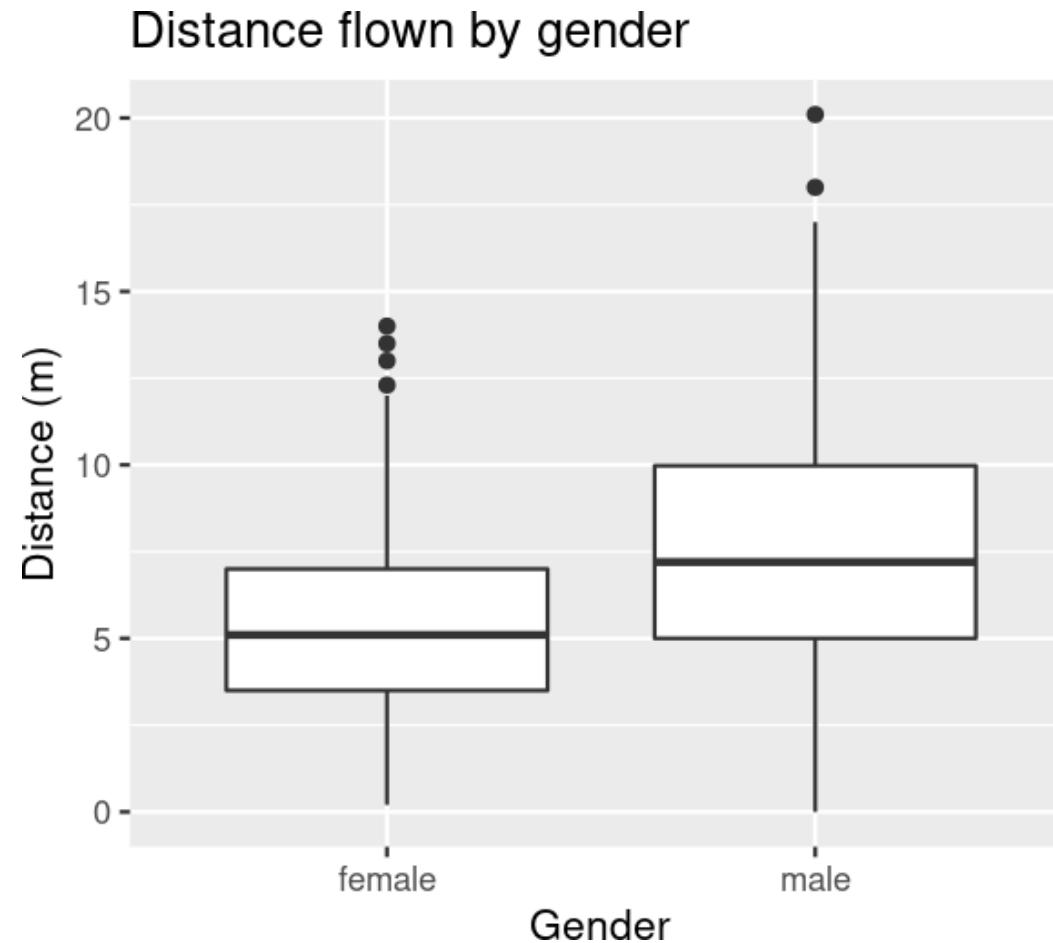
```
ggplot(paperplanes) +  
  aes(x = age, y = distance) +  
  geom_point(aes(colour = paper)) +  
  labs(x = "Age (years)",  
       y = "Distance (m)") +  
  labs(title = "Distance flown by age") +  
  geom_smooth(method = "lm") +  
  geom_vline(xintercept = c(20, 40, 60)) +  
  geom_hline(yintercept = 10)
```



Summary

```
ggplot(paperplanes) +          # Name of (tidy) data frame  
  aes(x = age, y = distance) + # Aesthetics (variables to map in axes)  
  geom_point()                # Geoms: geometric objects
```

Exercise: Make a plot like this one



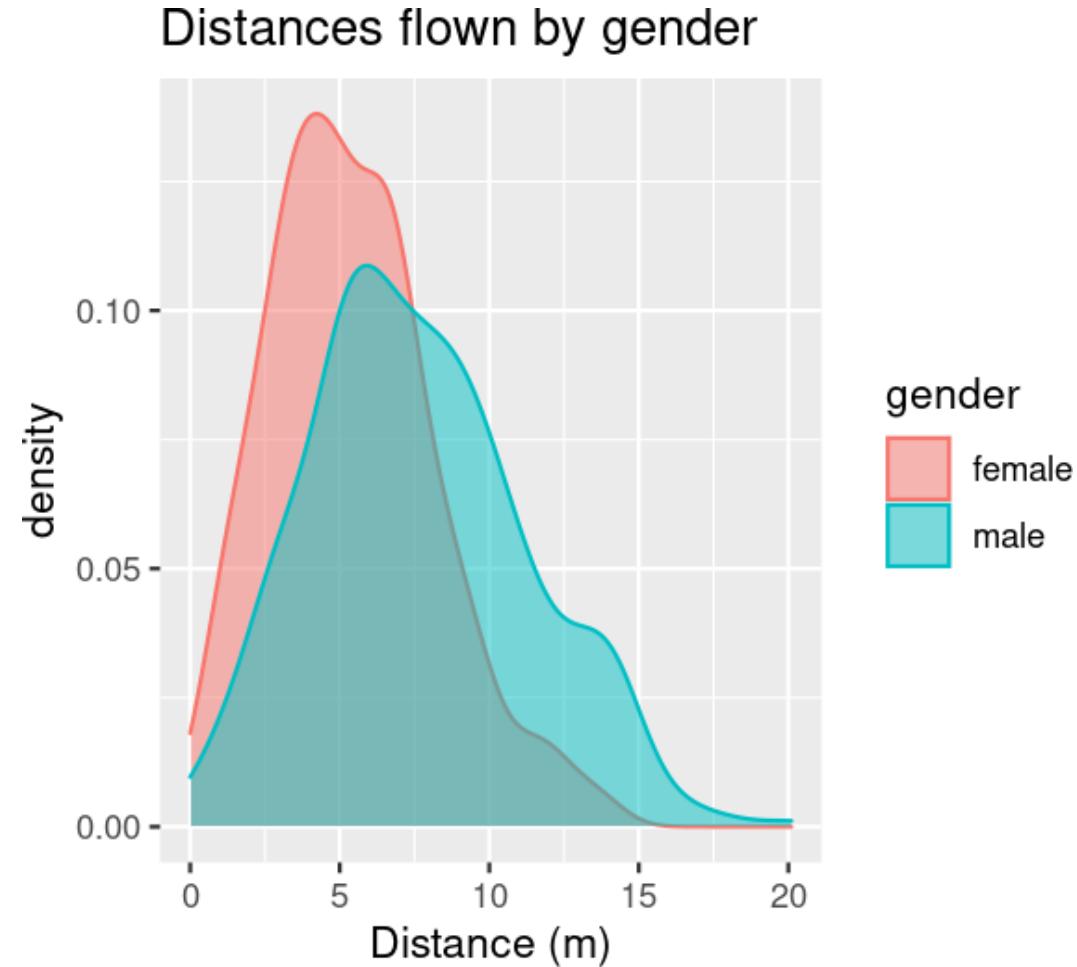
Exercise: Make a plot like this one



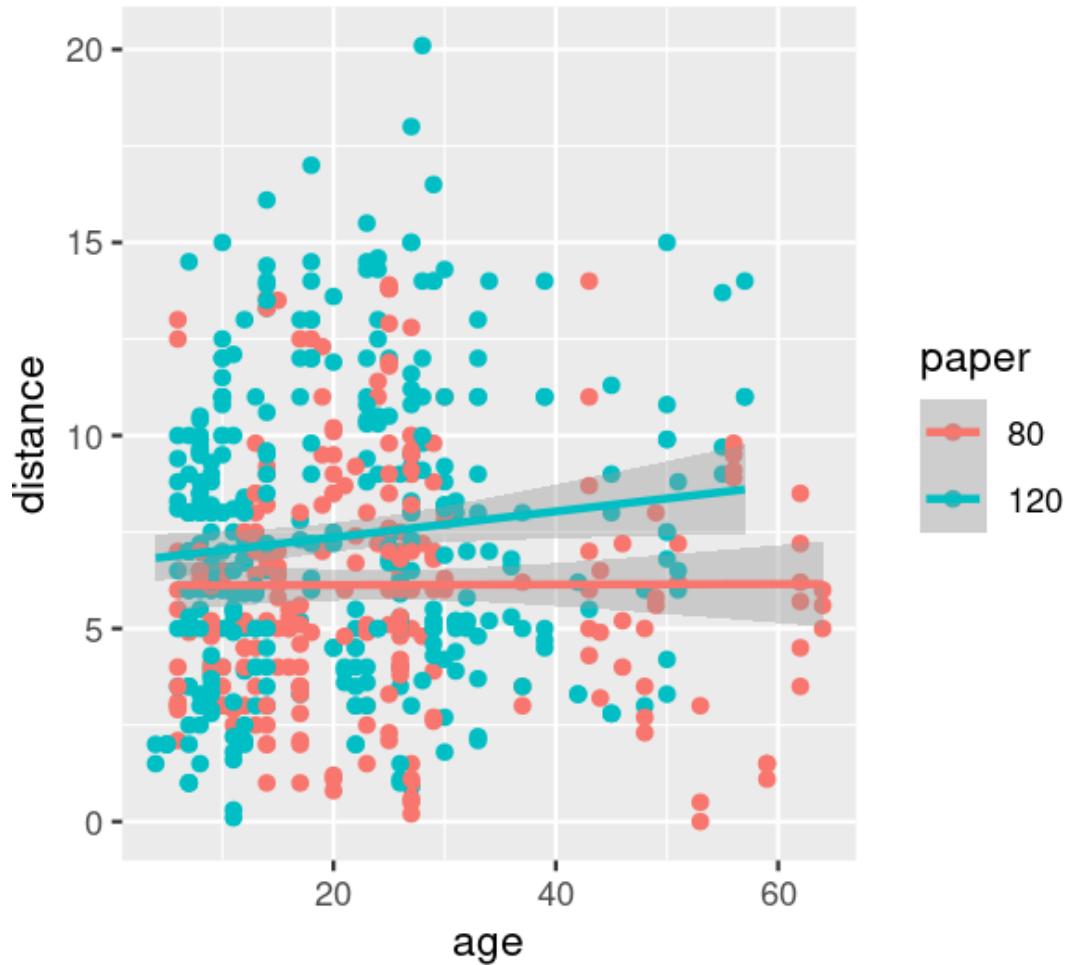
Exercise: Make a plot like this one



Exercise: Make a plot like this one



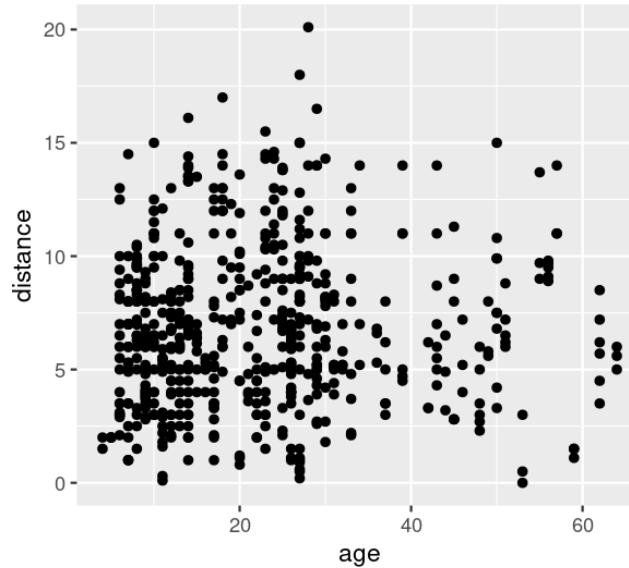
Exercise: Make a plot like this one



`ggplot2` figures can be assigned to R objects

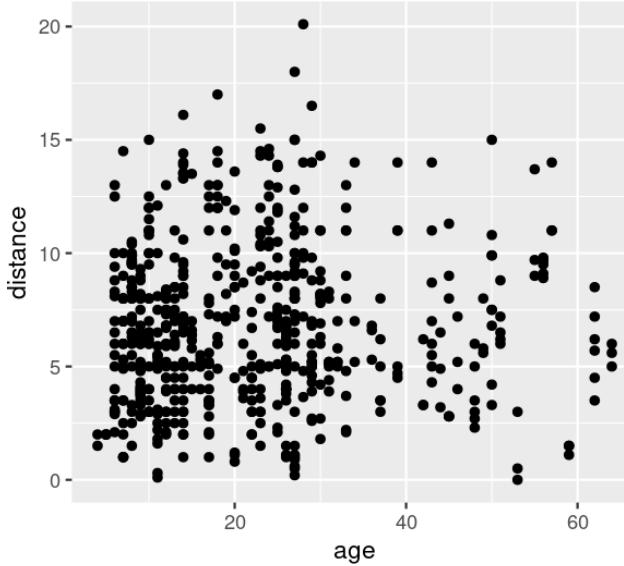
Assigning ggplot objects

```
myplot <- ggplot(paperplanes) +  
  aes(x = age, y = distance)  
myplot + geom_point()
```



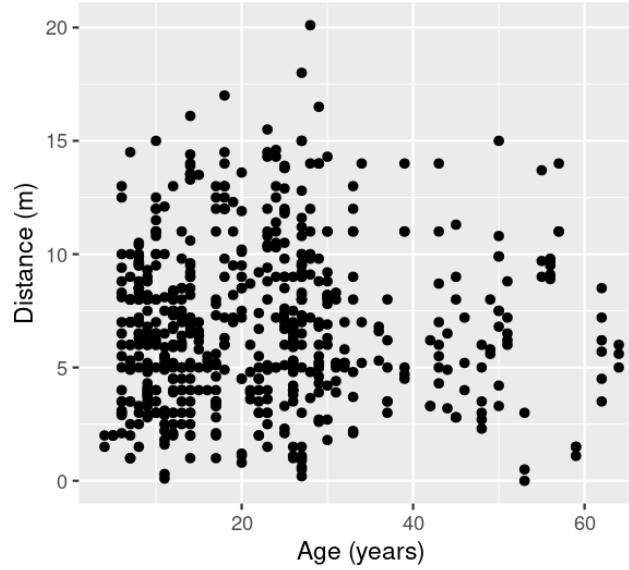
Assigning ggplot objects

```
myplot <- ggplot(paperplanes) +  
  aes(x = age, y = distance)  
myplot <- myplot + geom_point()  
myplot
```



Assigning ggplot objects

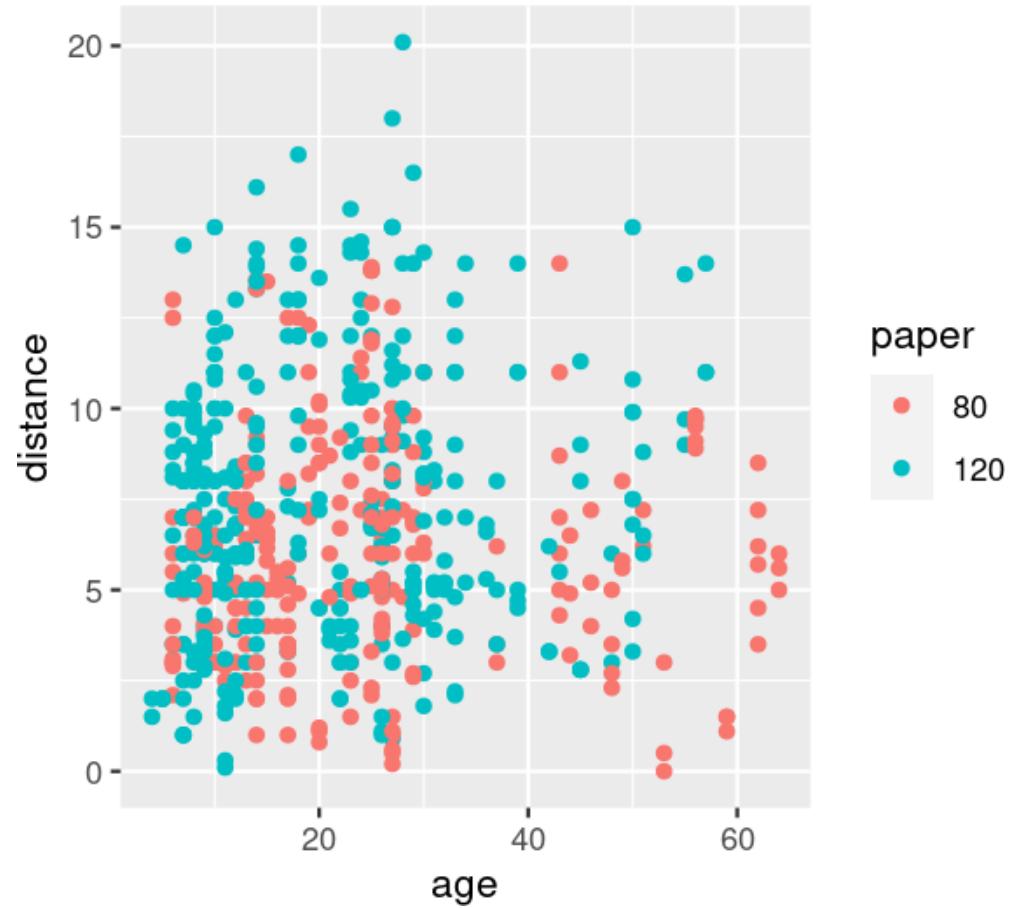
```
baseplot <- ggplot(paperplanes) +  
  aes(x = age, y = distance)  
scatterplot <- baseplot + geom_point()  
labelled <- scatterplot + labs(x = "Age (years)", y = "Distance (m)")  
labelled
```



Themes: changing plot appearance

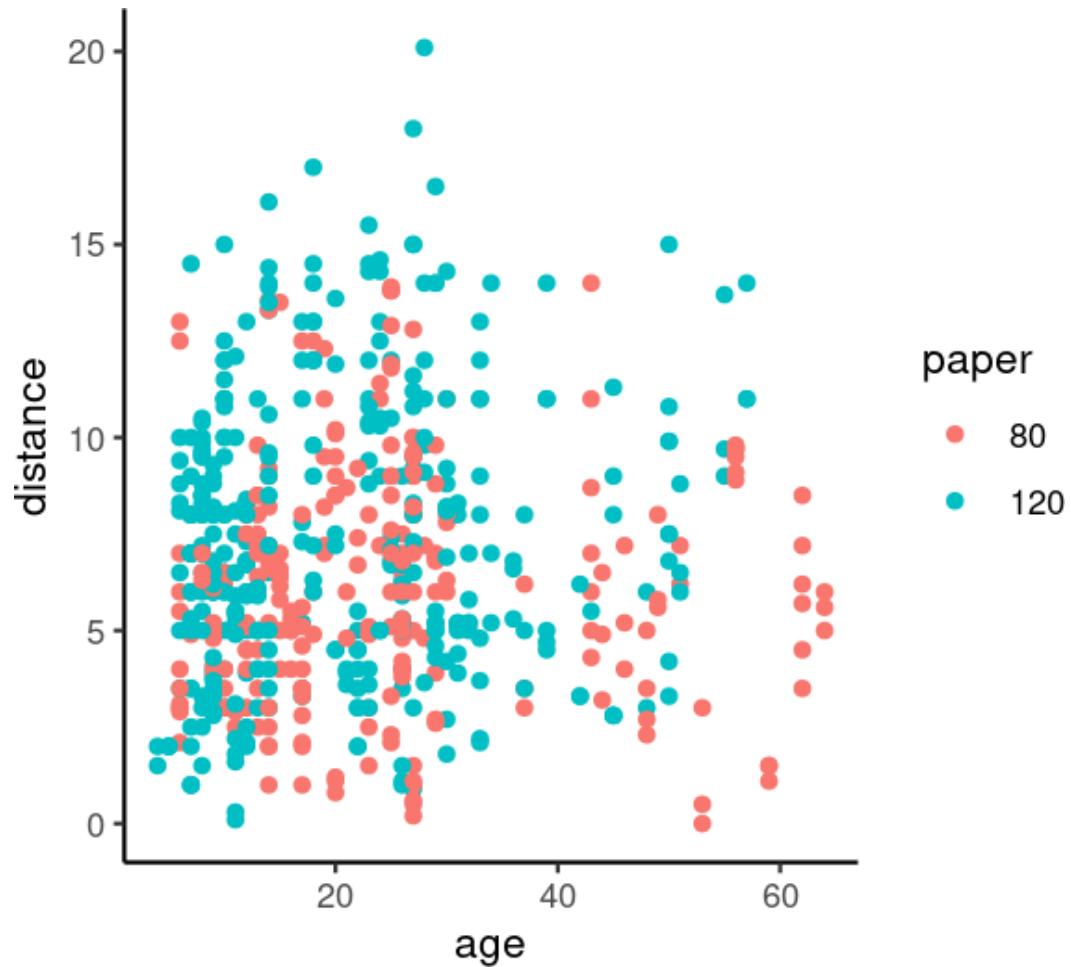
Create 'myplot'

```
myplot <- ggplot(paperplanes) +  
  aes(x = age,  
      y = distance,  
      colour = paper) +  
  geom_point()  
  
myplot
```



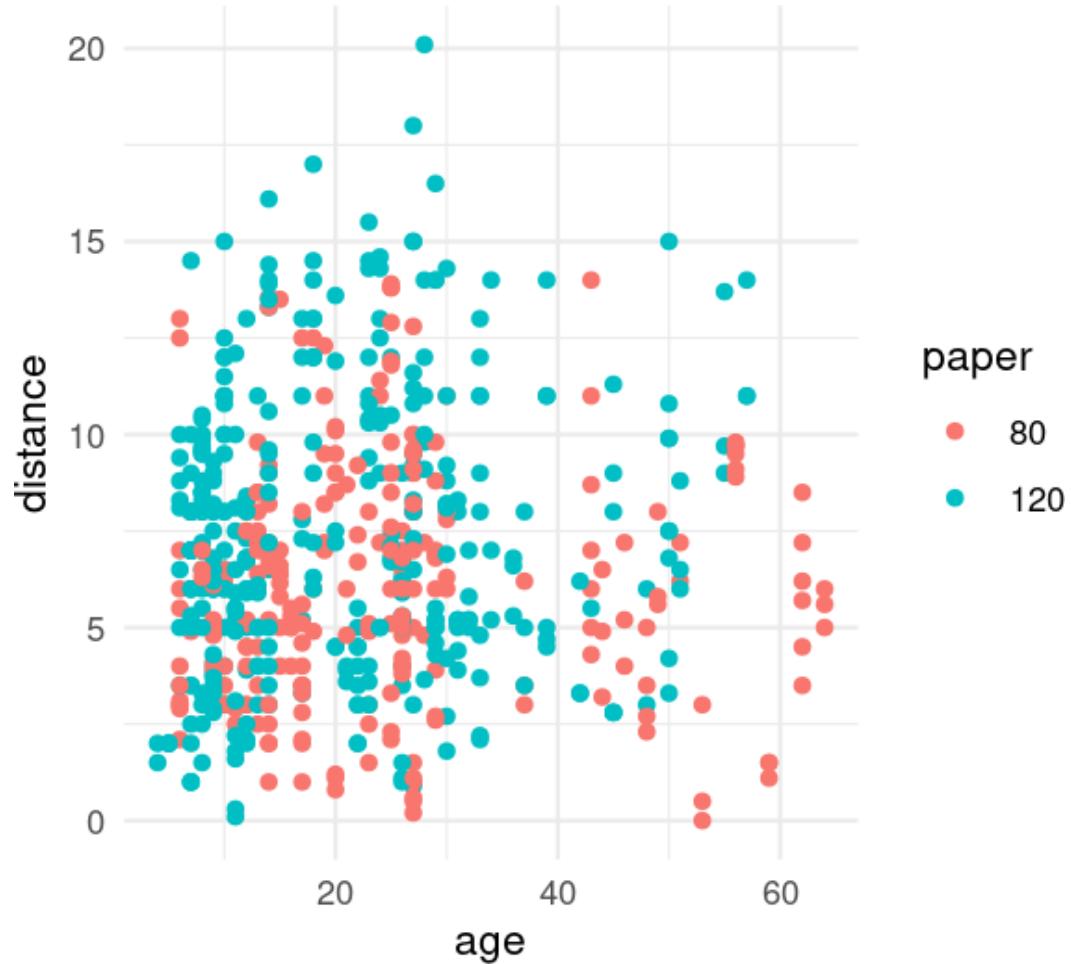
Use theme_classic

```
myplot + theme_classic()
```



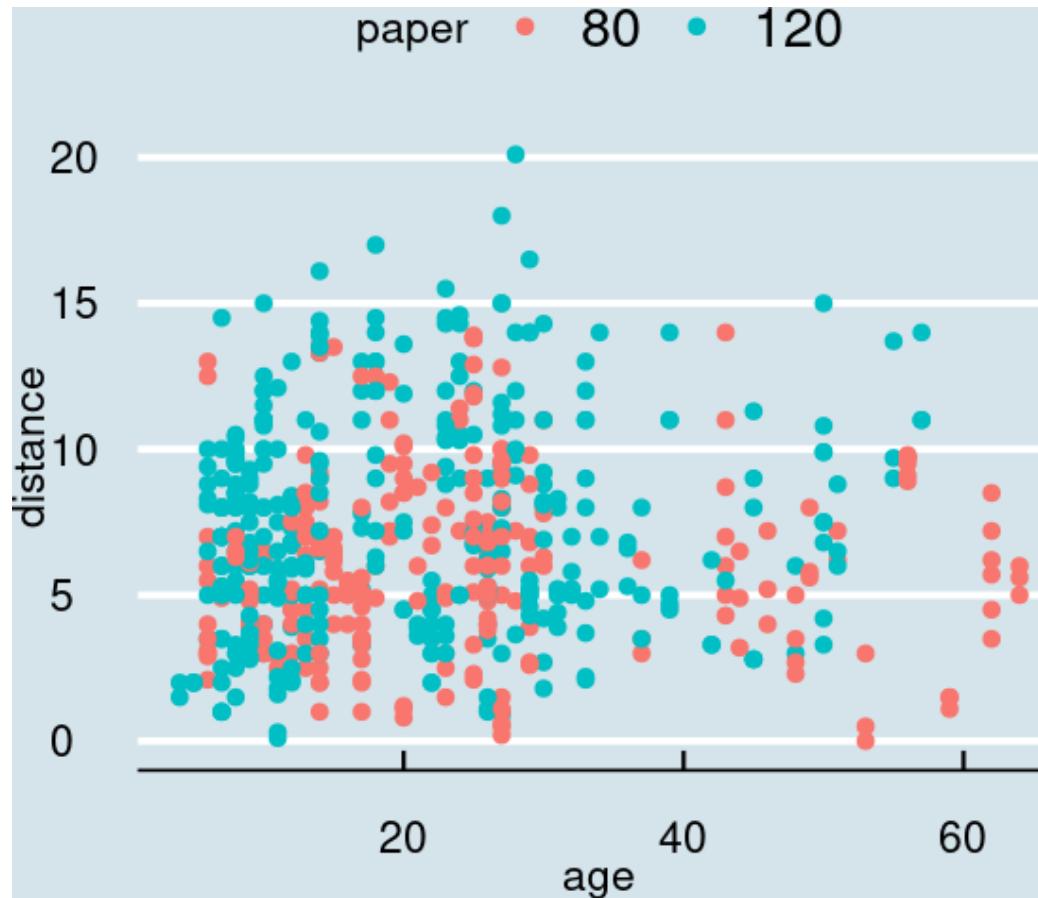
theme_minimal

```
myplot + theme_minimal()
```



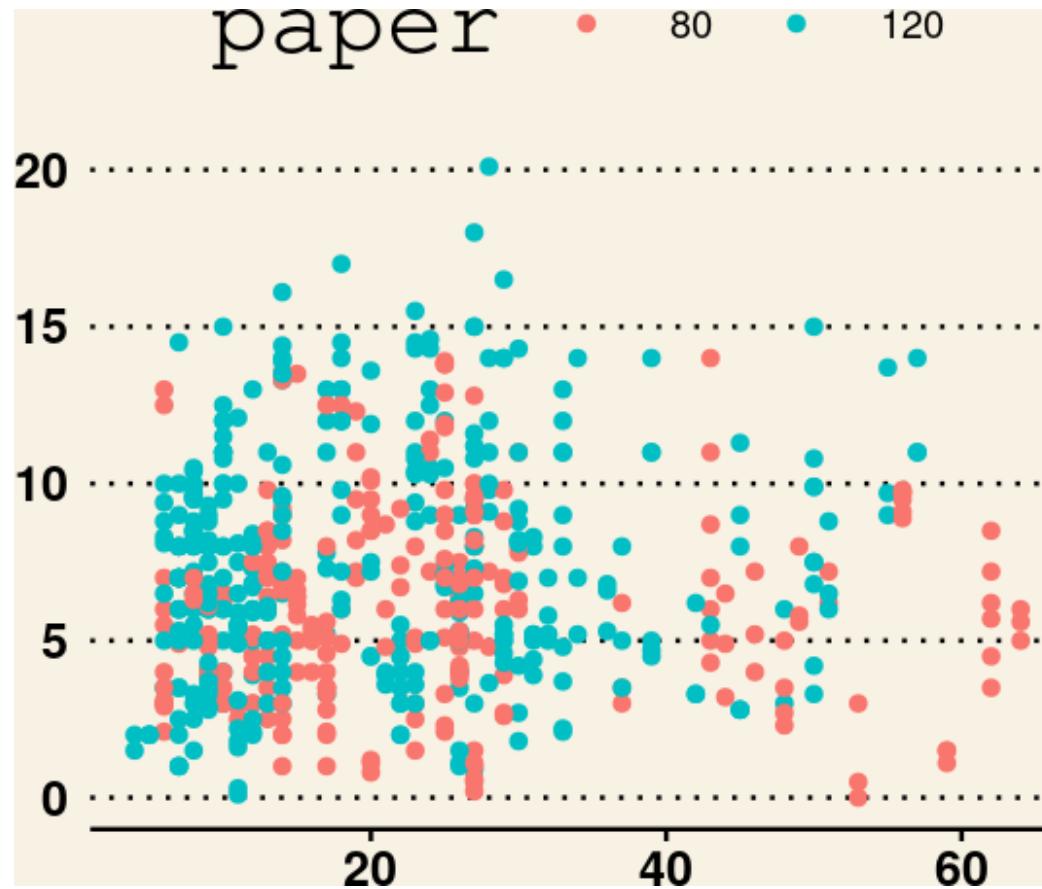
Lots of themes out there

```
library(ggthemes)  
myplot + theme_economist()
```



Lots of themes out there

```
myplot + theme_wsj()
```

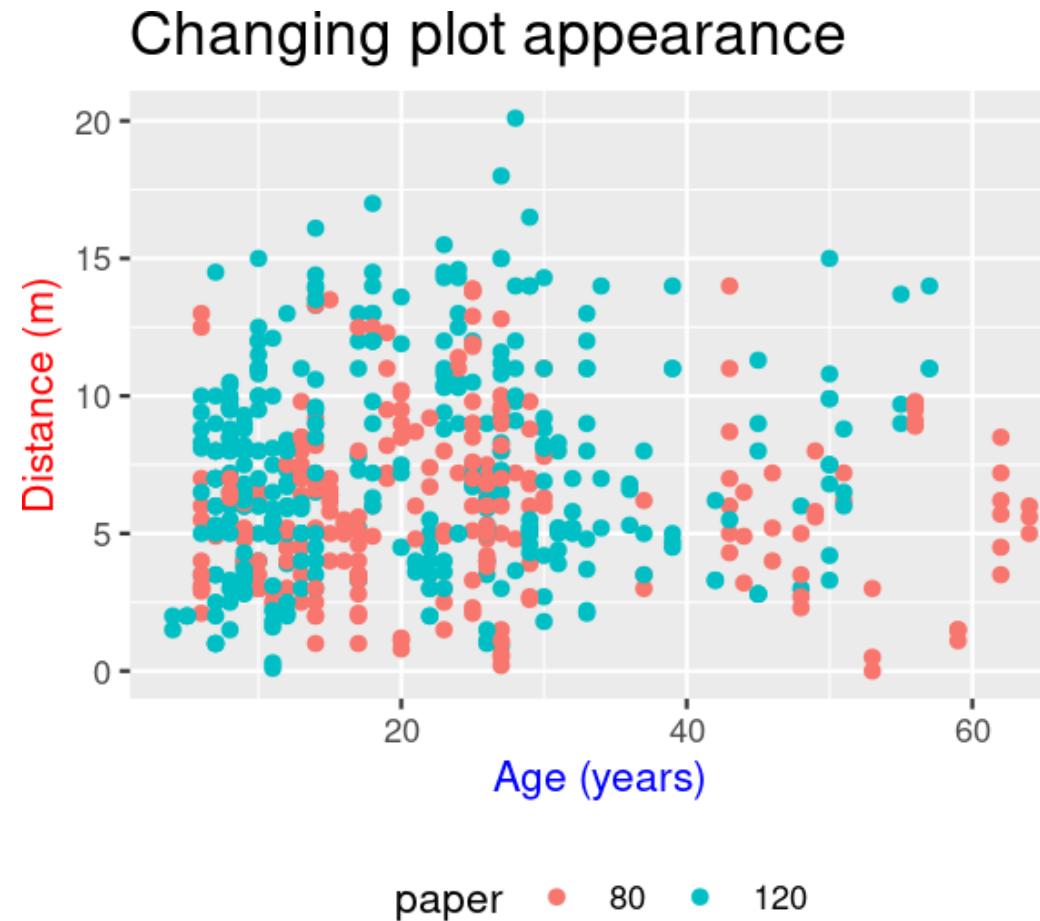


Editing themes

```
?theme
```

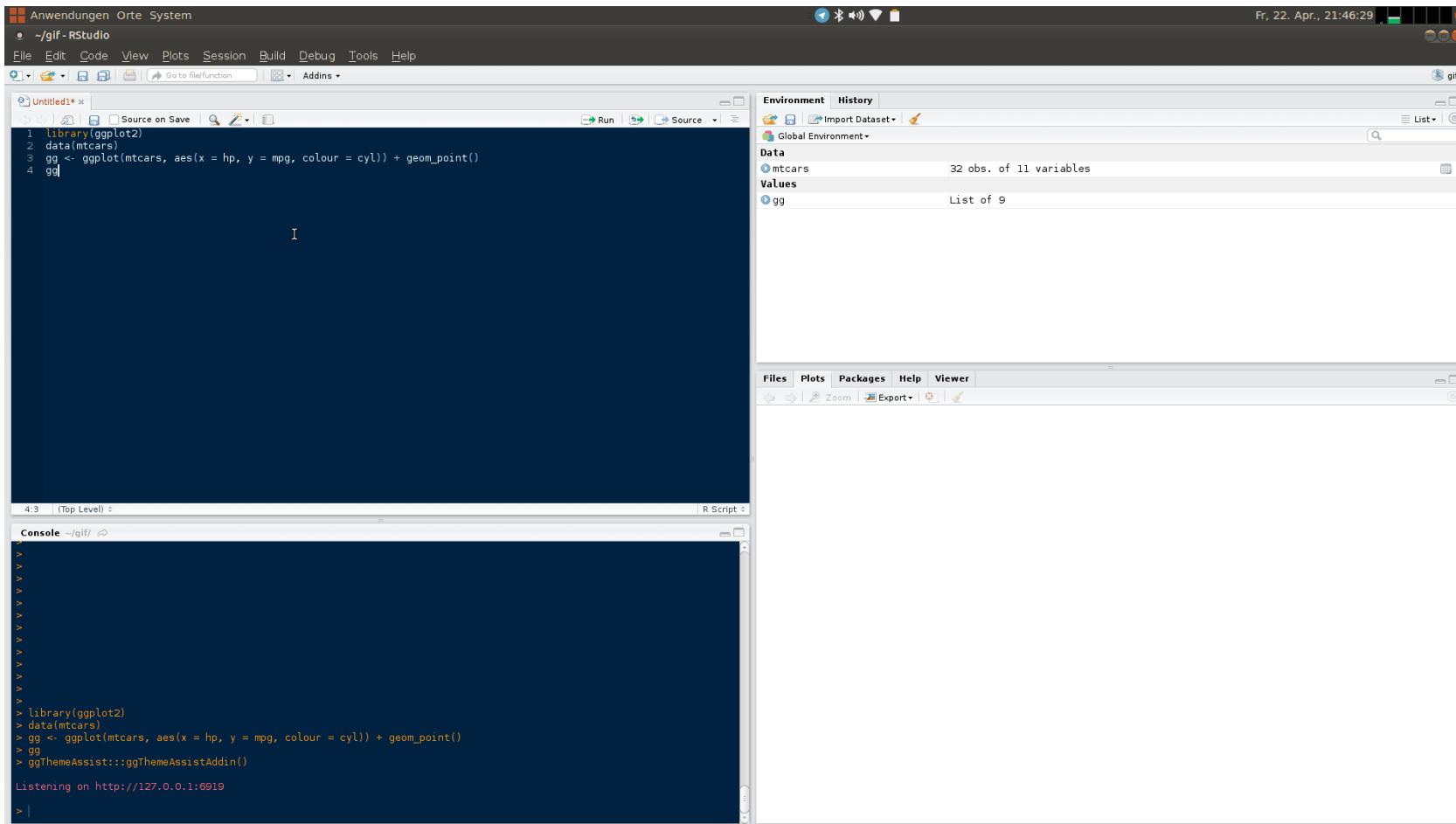
- element_blank
- element_text
- element_line
- element_rect (borders & backgrounds)

Exercise: make a plot like this one



Easily changing appearance with ggthemearrassist (Rstudio addin)

<https://github.com/calligross/ggthemearrassist>

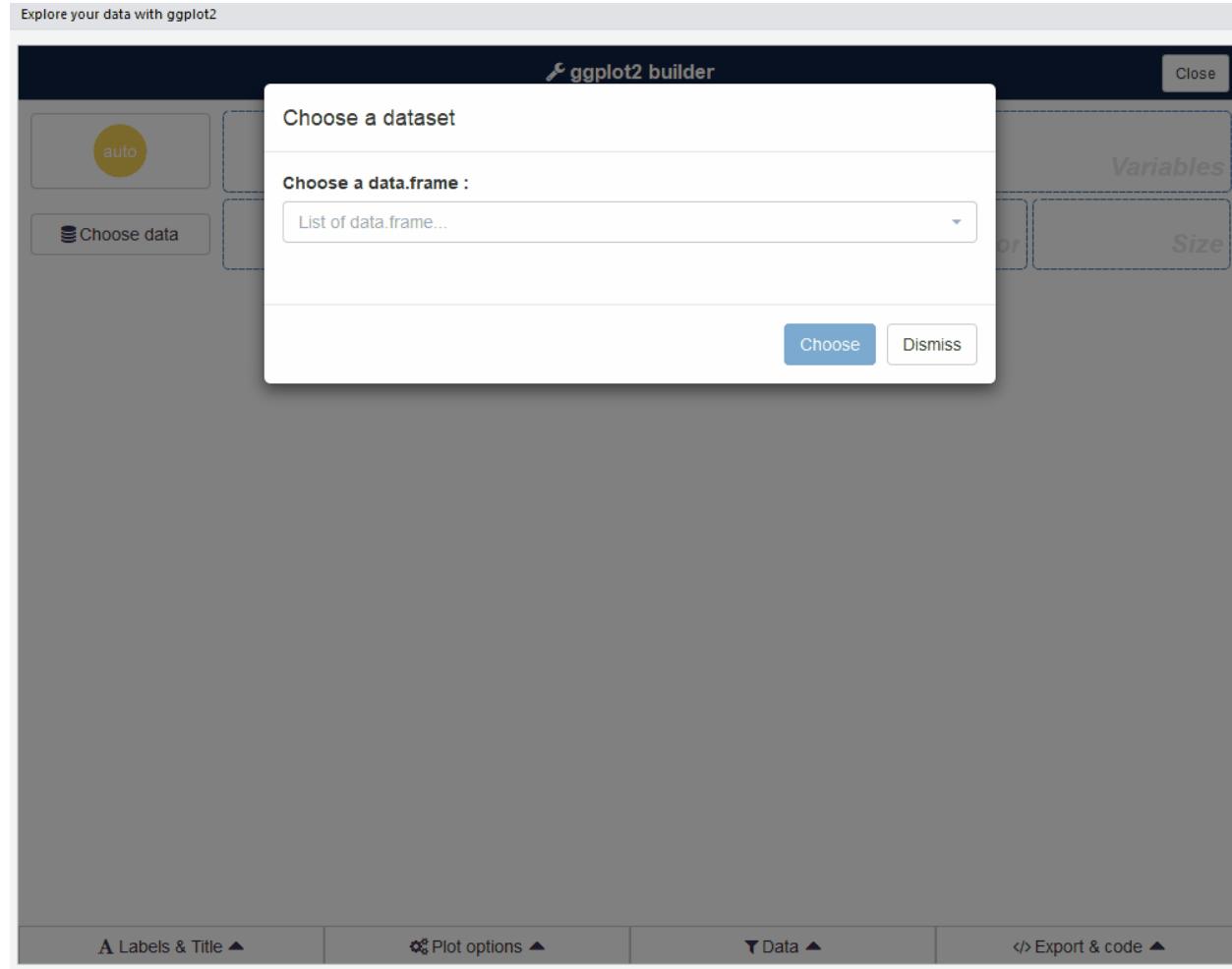


Easily changing appearance with ggedit

<https://github.com/yonicd/ggedit>

esquisse: ggplot2 builder addin

<https://github.com/dreamRs/esquisse>



Think twice before editing plots out of R



Trevor A. Branch
@TrevorABranch

Follow

My rule of thumb: every analysis you do on a dataset will have to be redone 10–15 times before publication. Plan accordingly. #Rstats

Why I think twice before editing plots out of R

Choosing the right visualization software

Think twice before editing plots out of R

Referee #3: "Please increase font size in all figures"

```
myplot +  
  theme(axis.title = element_text(size = 18))
```

Publication-quality plots

```
library(cowplot)  
myplot + theme_cowplot()
```

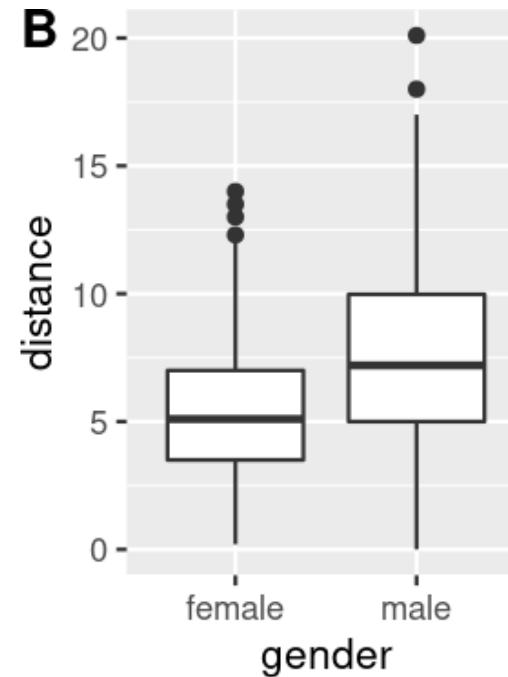
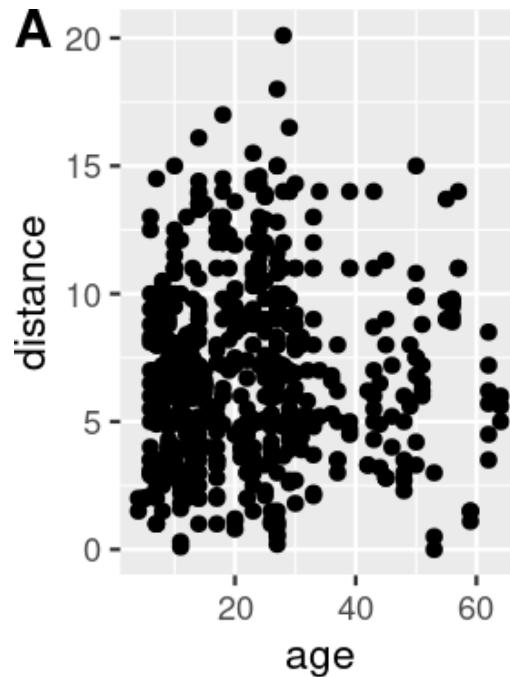
Some publication themes:

<https://gist.github.com/Pakillo/c2c7ea11c528cc2ee20f#themes>

Composite figures

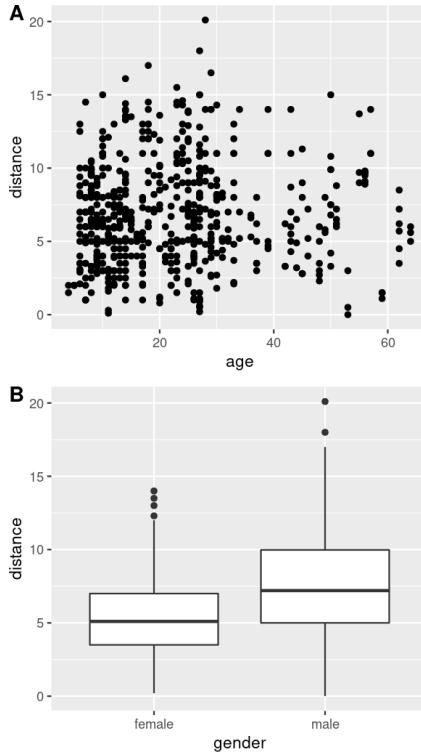
Composite figures: cowplot

```
library(cowplot)
plot1 <- ggplot(paperplanes) + aes(age, distance) + geom_point()
plot2 <- ggplot(paperplanes) + aes(gender, distance) + geom_boxplot()
plot_grid(plot1, plot2, labels = "AUTO")
```

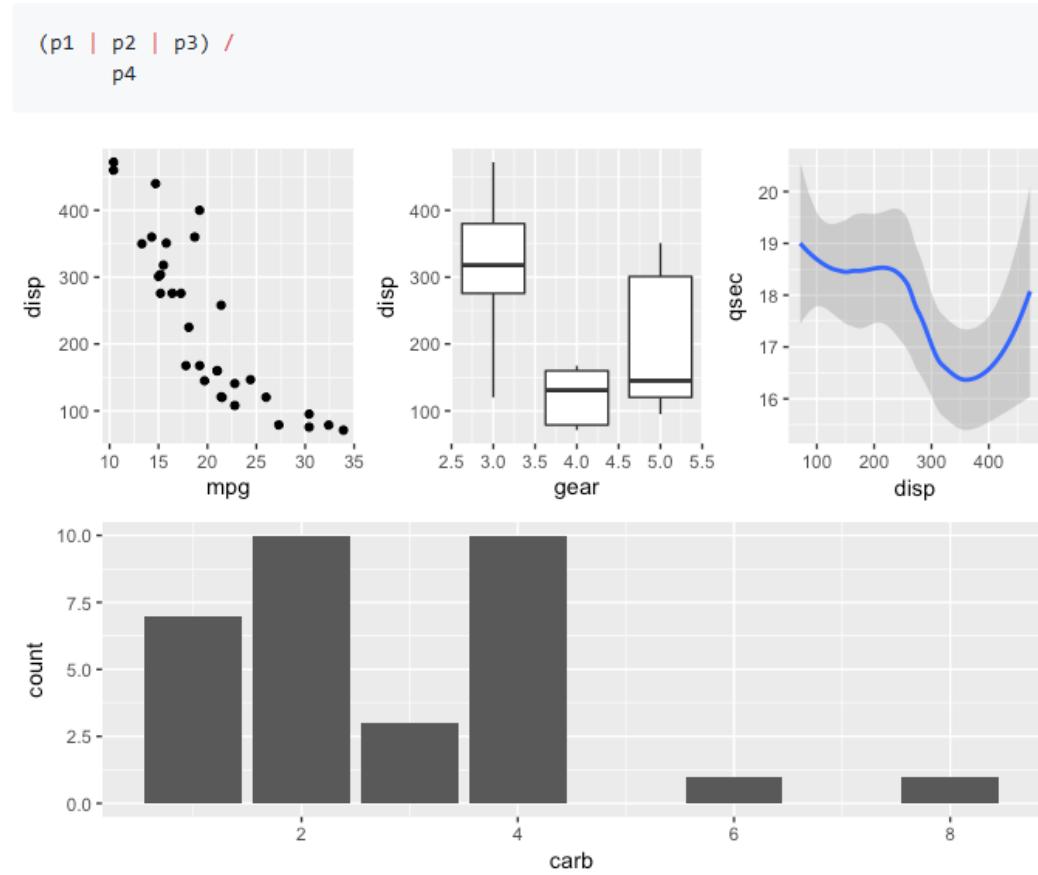


Composite figures

```
plot_grid(plot1, plot2, labels = "AUTO", ncol = 1)
```

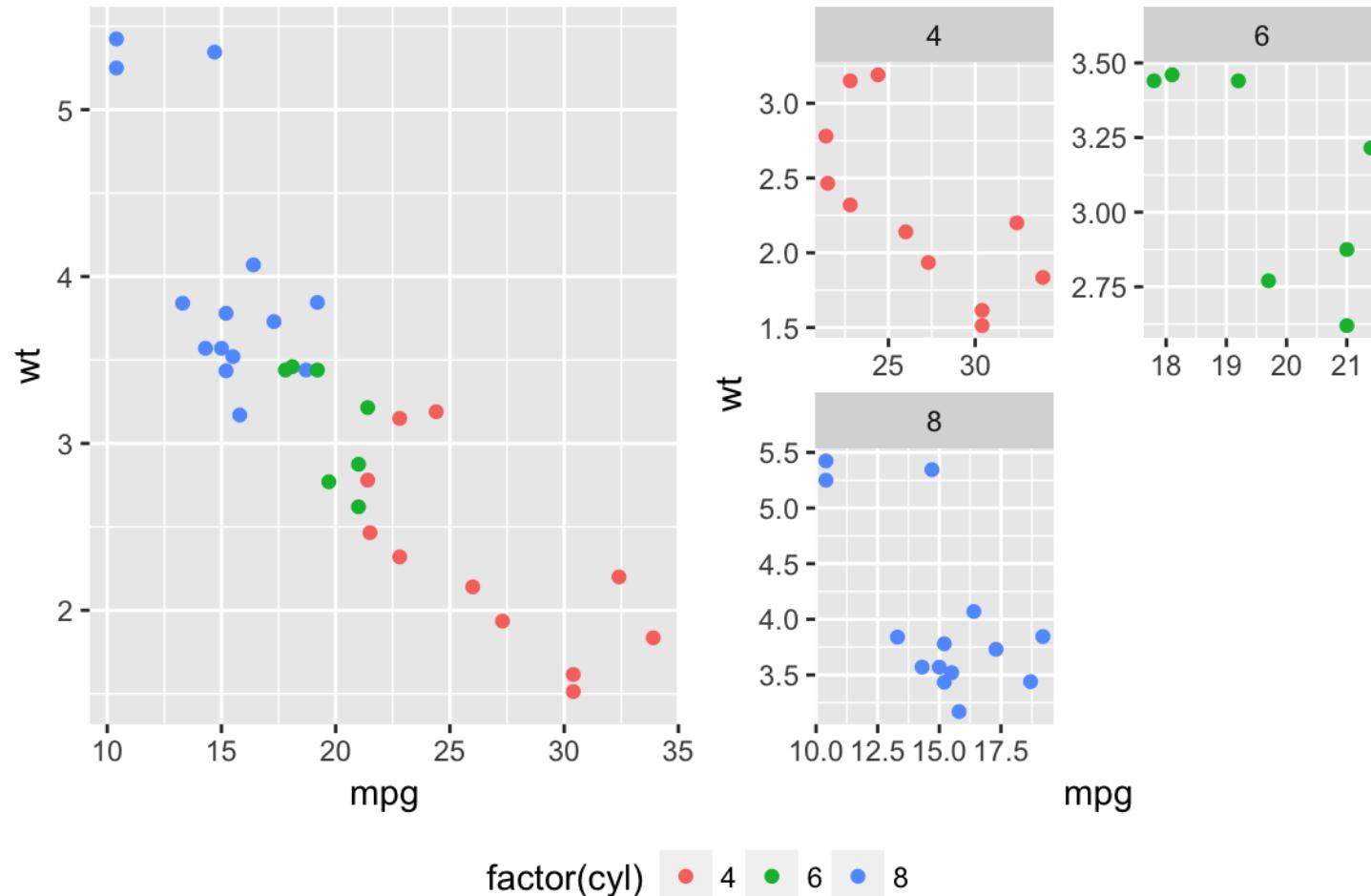


Composite figures: patchwork



<https://github.com/thomasp85/patchwork>

Composite figures: egg



Saving plot

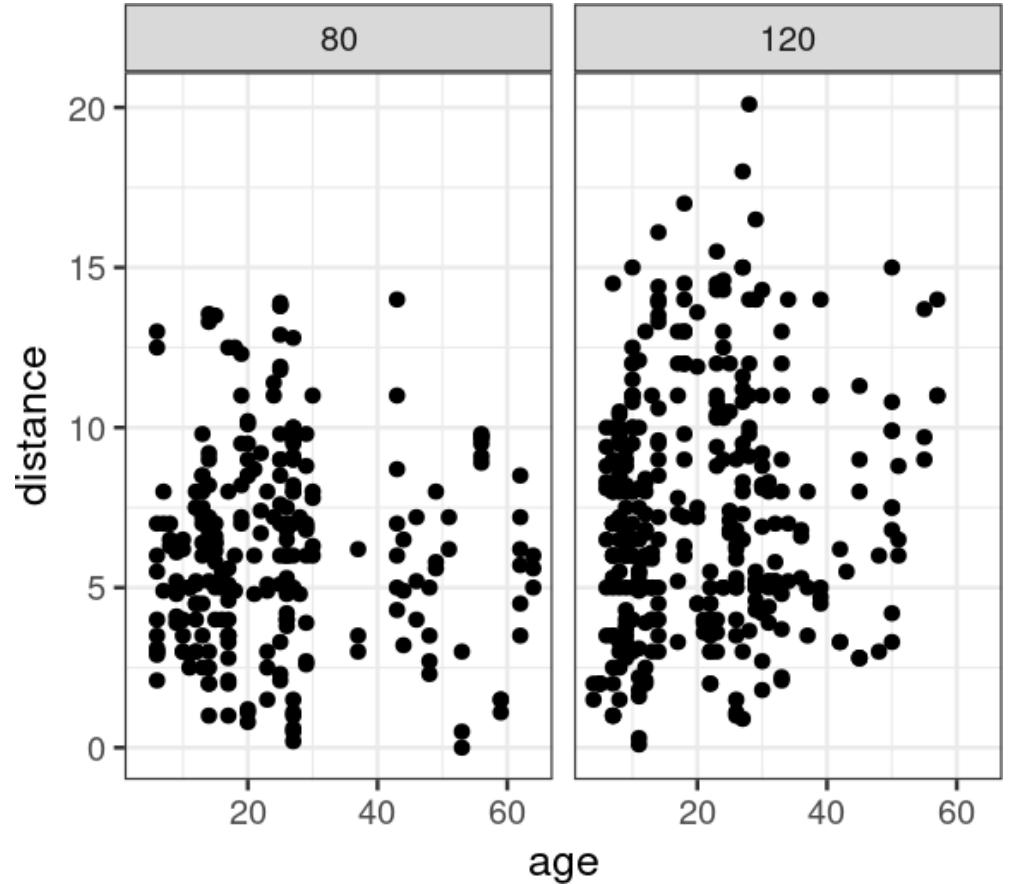
```
ggsave("myplot.pdf")
```

```
save_plot("myplot.pdf")
```

Facetting (small multiples)

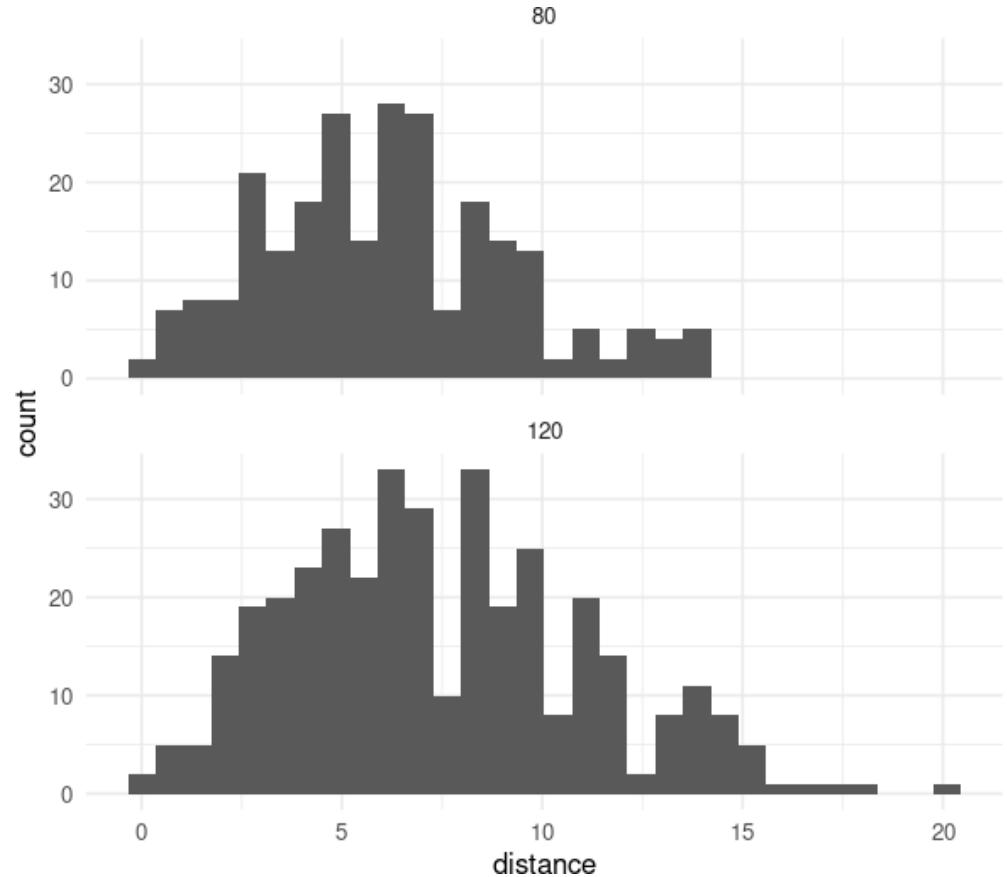
Facetting

```
ggplot(paperplanes) +  
  aes(x = age,  
      y = distance) +  
  geom_point() +  
  theme_bw(base_size = 12) +  
  facet_wrap(~paper)
```



Facetting

```
ggplot(paperplanes) +  
  geom_histogram(aes(distance)) +  
  theme_minimal(base_size = 8) +  
  facet_wrap(~paper, nrow = 2)
```

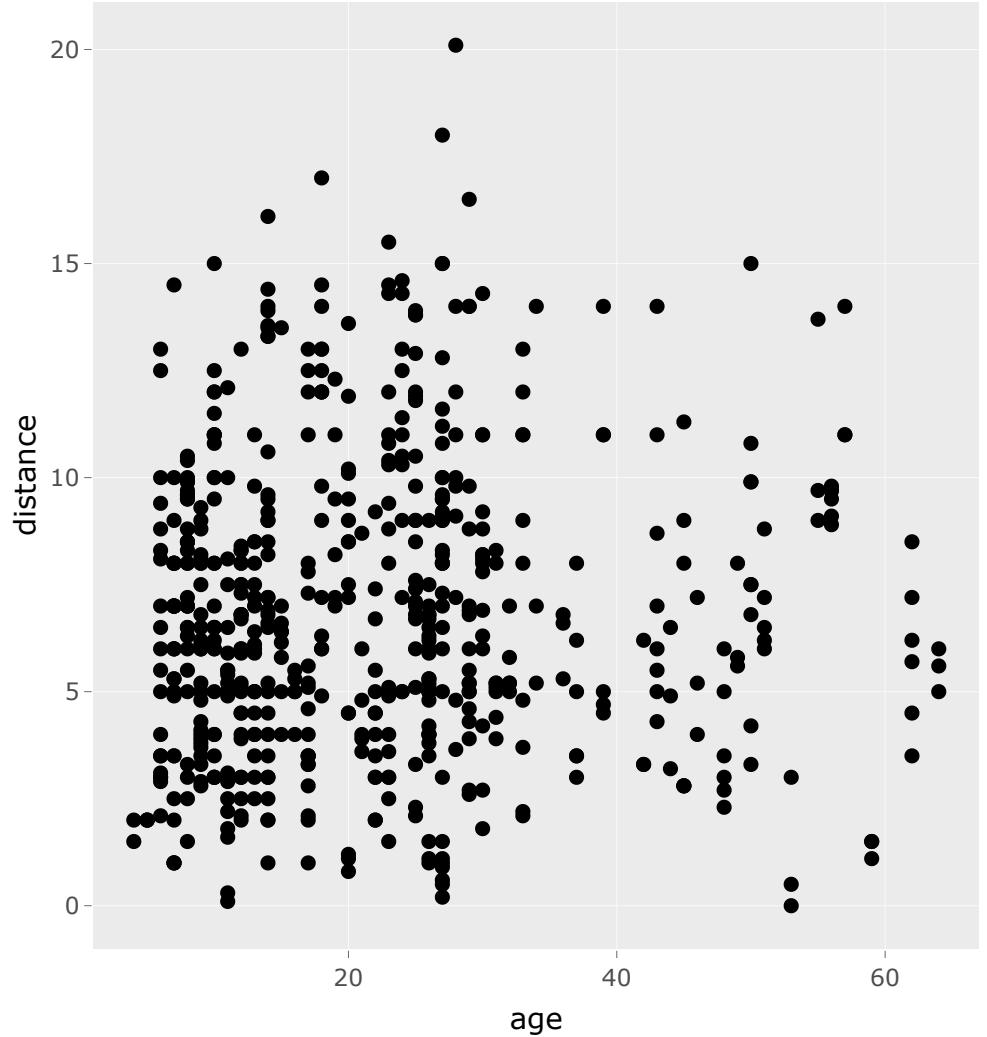


Interactivity: plotly

```
library(plotly)

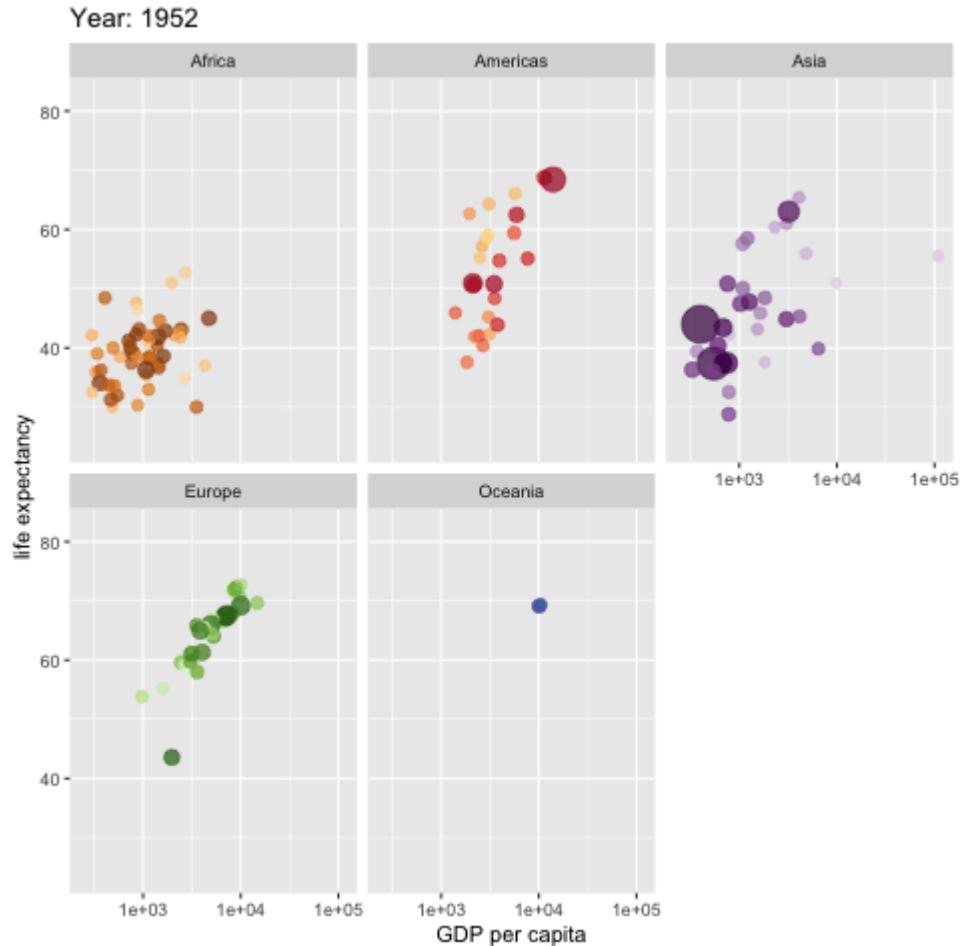
myplot <- ggplot(paperplanes) +
  aes(age, distance) +
  geom_point()

ggplotly(myplot)
```

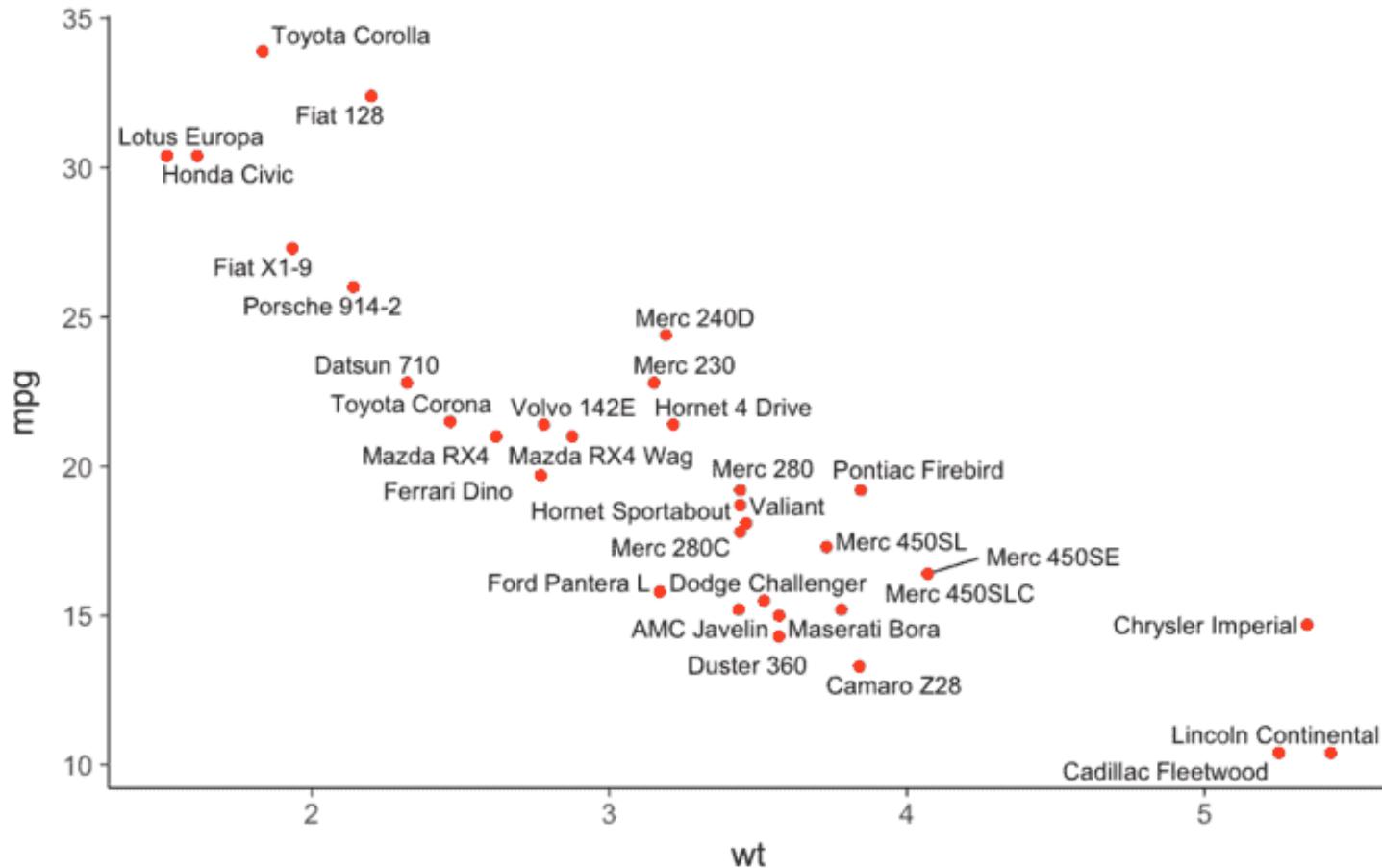


Animated graphs

<https://github.com/thomasp85/gganimate>



Automatic label placement

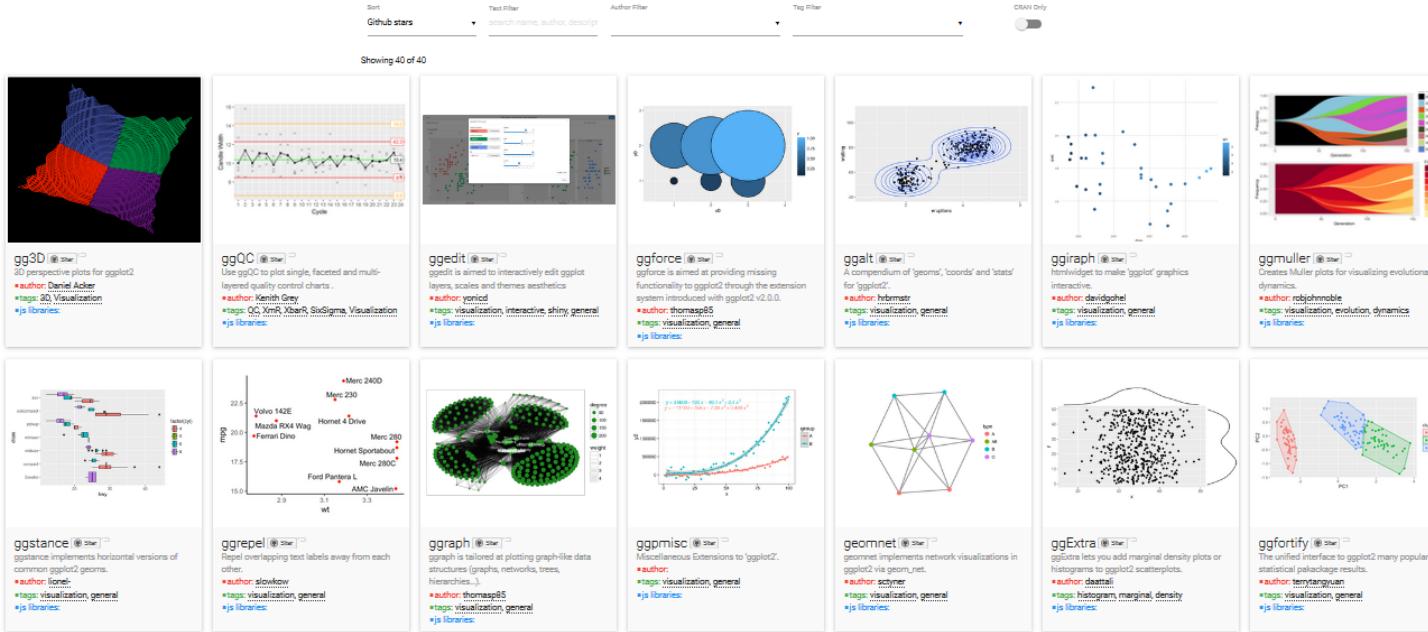


<https://cran.r-project.org/package=ggrepel>

Many extensions!

<https://exts.ggplot2.tidyverse.org/gallery/>

40 registered extensions available to explore

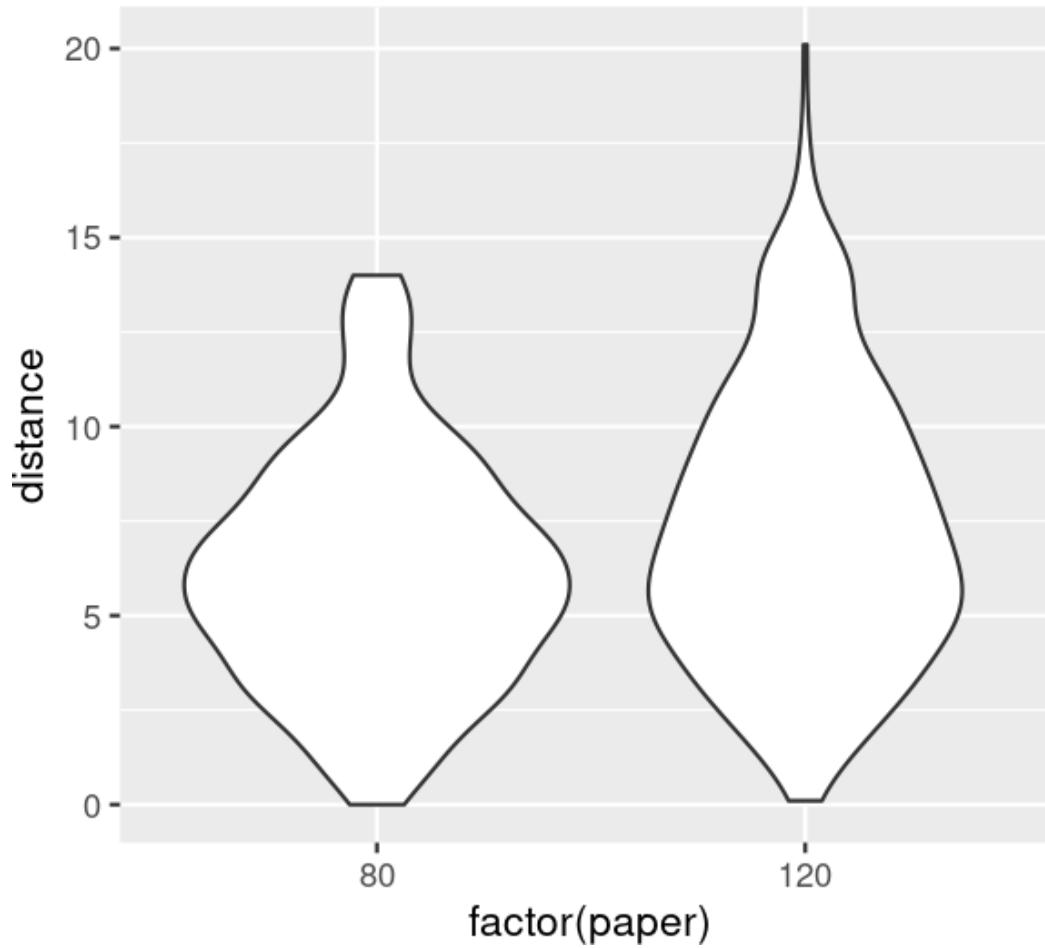


Summary

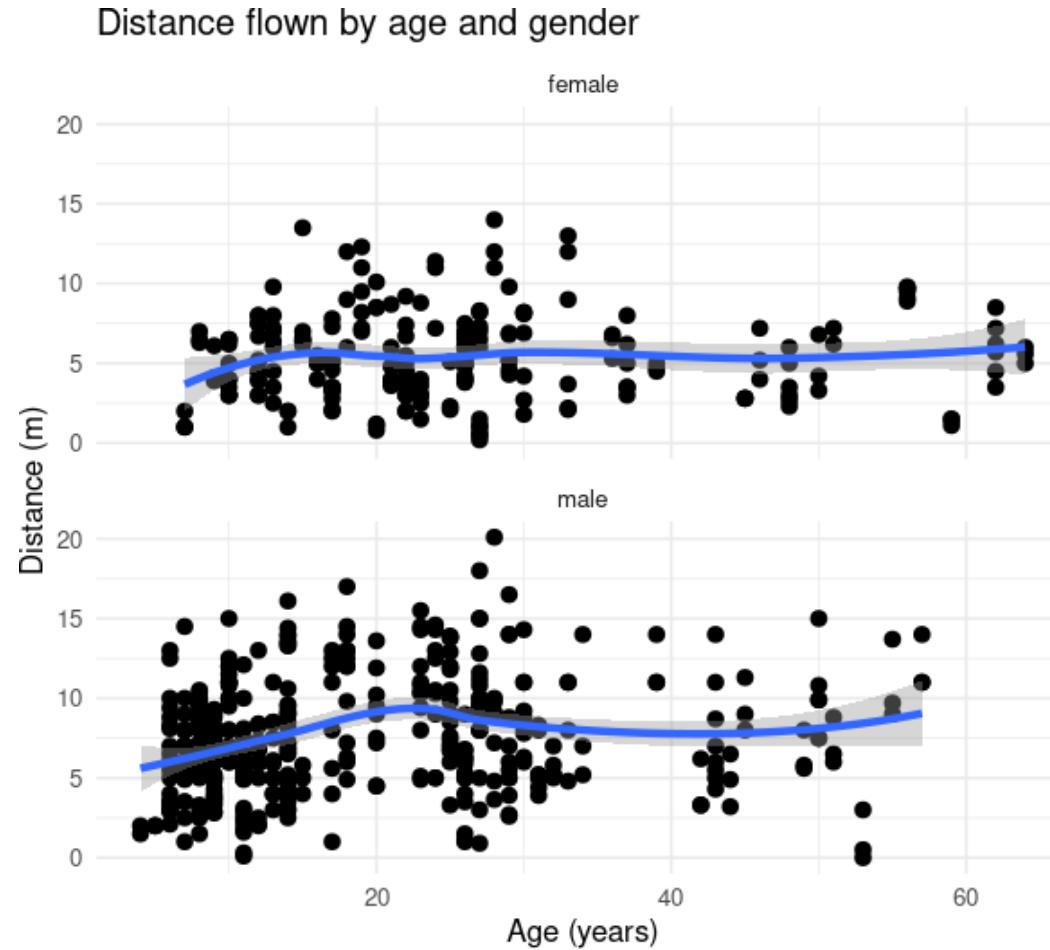
Grammar of graphics

- **Data** (tidy data frame)
- **Layers** (*geoms*: points, lines, polygons...)
- **Aesthetics** mappings (x, y, size, colour...)
- **Scales** (colour, size, shape...)
- **Facets** (small multiples)
- **Themes** (appearance)
- **Coordinate system** (Cartesian, polar, map projections...)

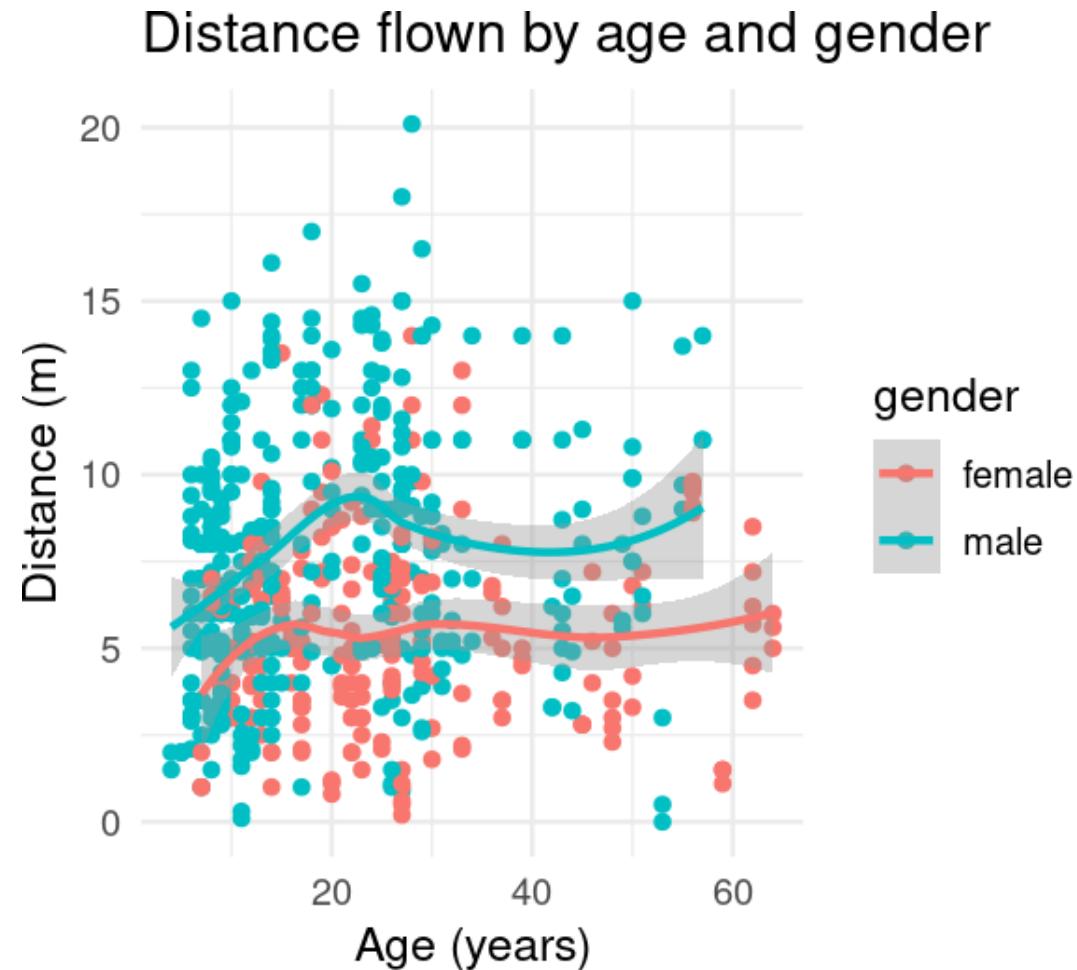
Exercise: make a plot like this one



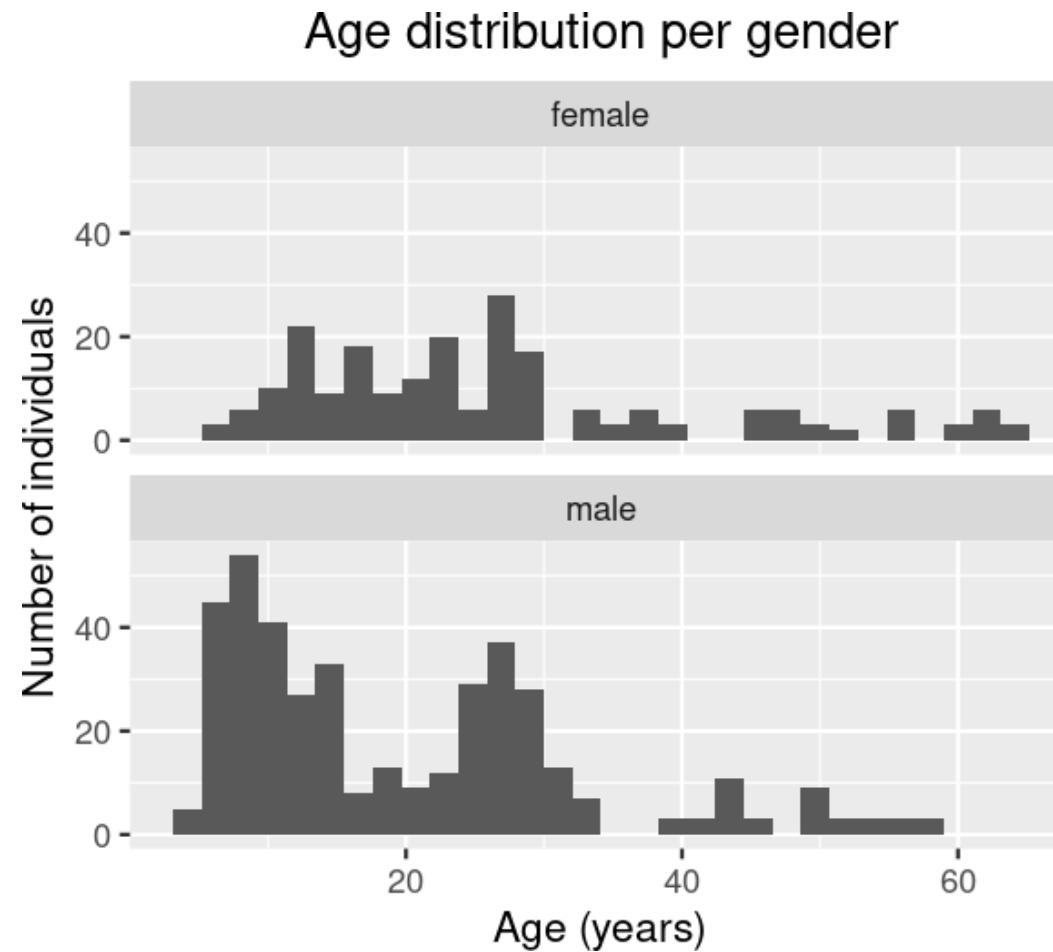
Exercise: make a plot like this one



Exercise: make a plot like this one

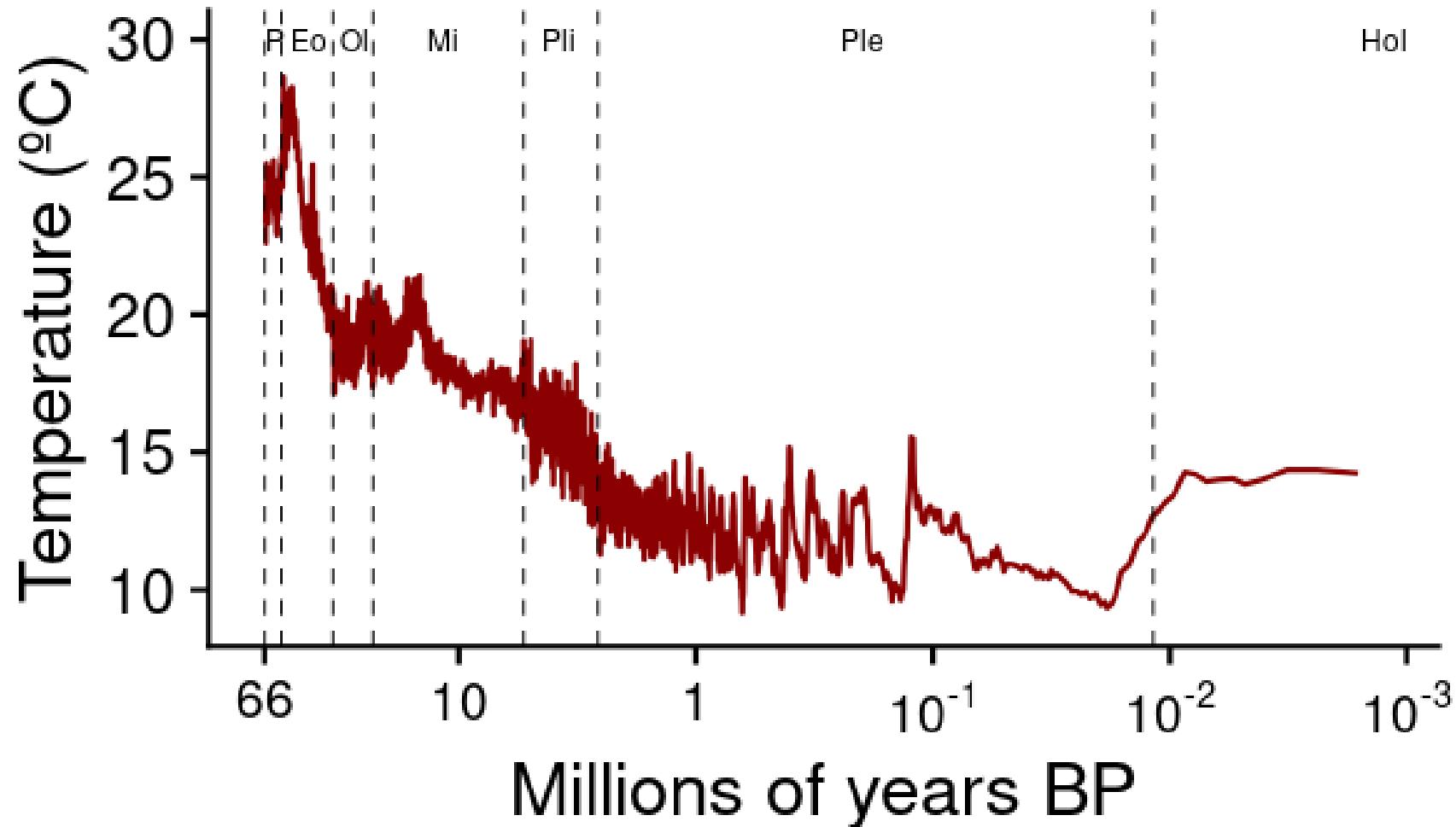


Exercise: make a plot like this one



Exercise: make a plot like this one

Data from <http://www.columbia.edu/~mhs119/Sensitivity+SL+CO2/Table.txt>



Exercise: make a plot like this one

END



Slides and source code available at <https://github.com/Pakillo/ggplot-intro>