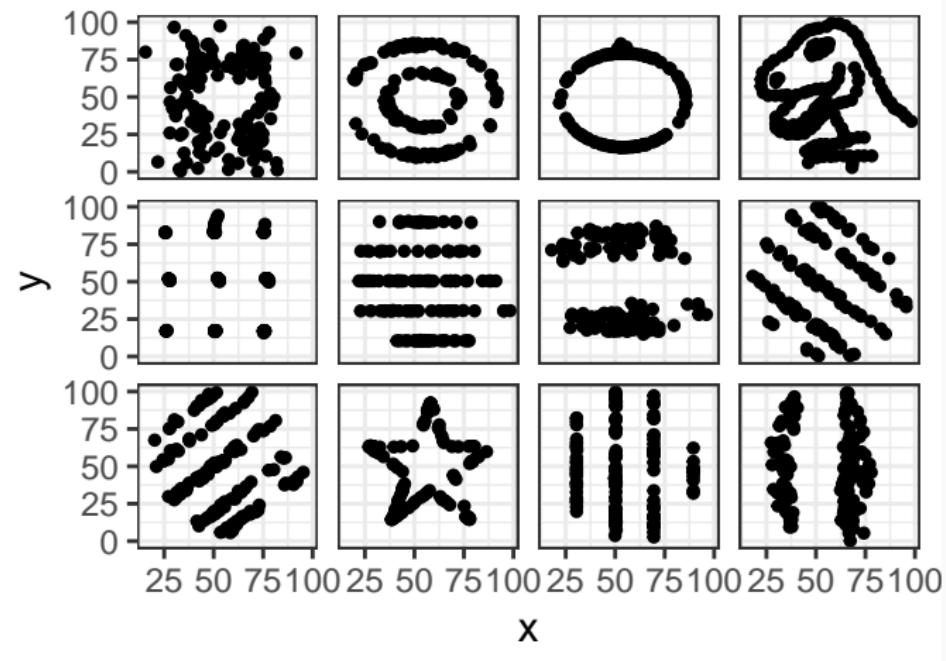


Data visualisation with ggplot2

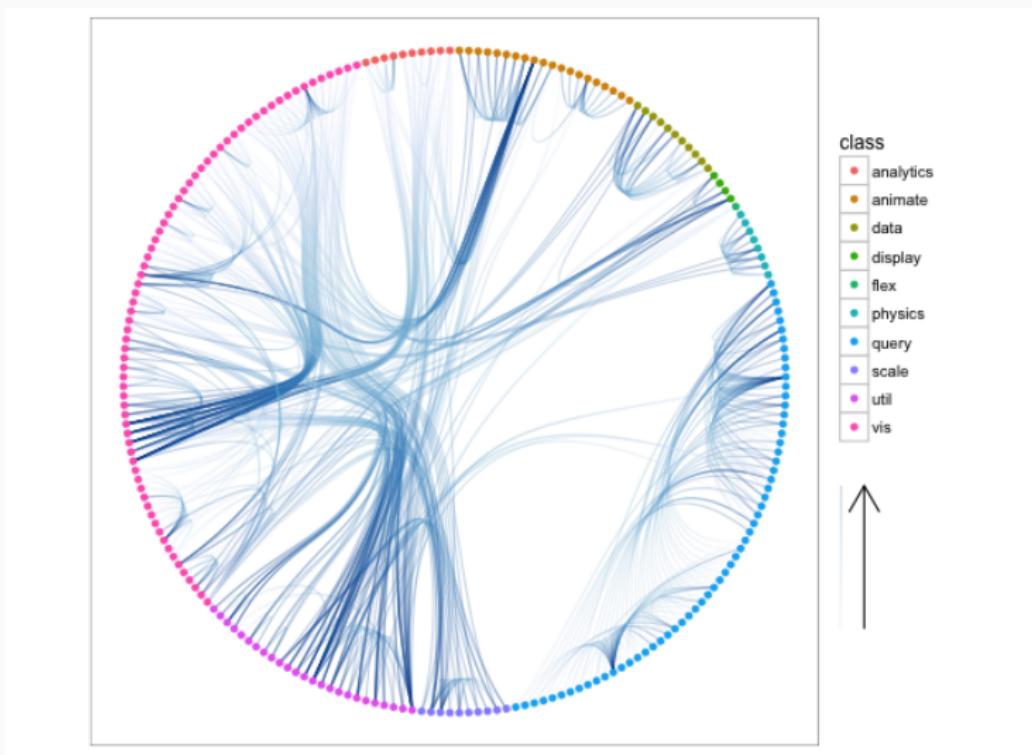
Francisco Rodriguez-Sanchez (@frod_san)

Always plot data!



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

Made with ggplot



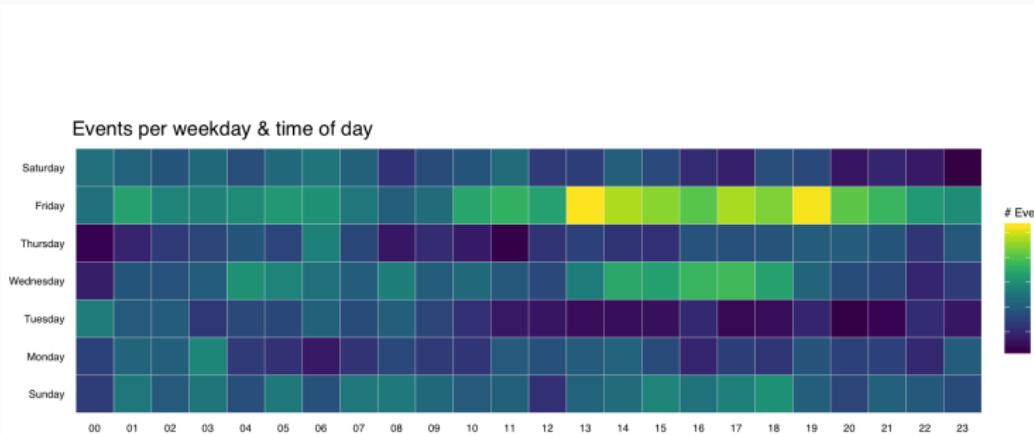
<https://github.com/thomasp85/ggraph>

Made with ggplot



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

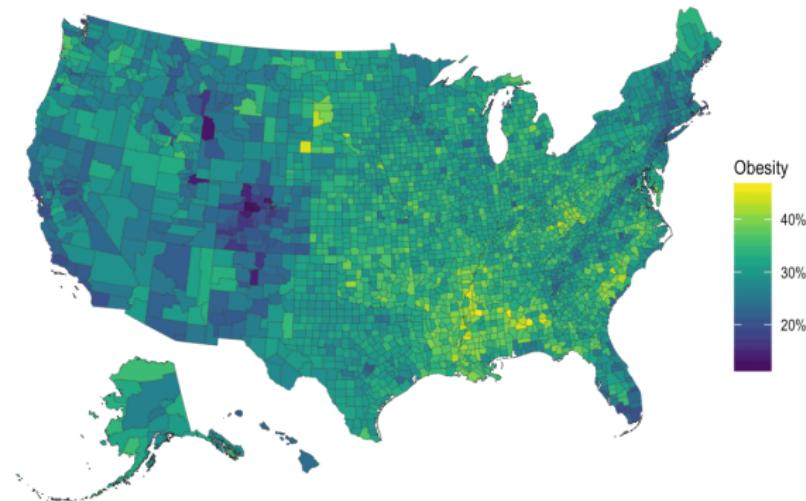
Made with ggplot



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

U.S. Obesity Rate by County (2012)

Content source: Centers for Disease Control and Prevention

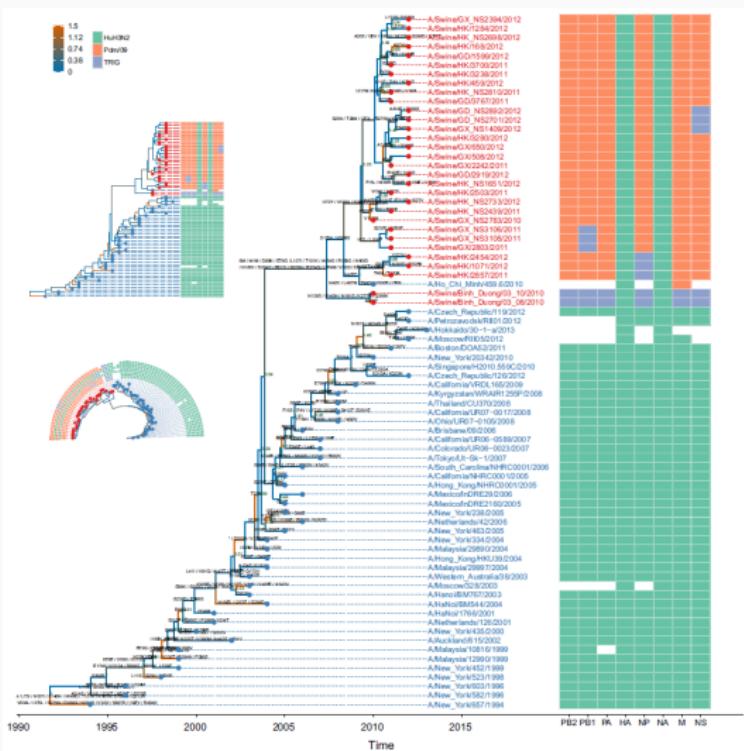


Data from http://www.cdc.gov/diabetes/atlas/countydata/County_ListofIndicators.html

[https:](https://rud.is/b/2016/03/29/easier-composite-u-s-choropleths-with-albersusa/)

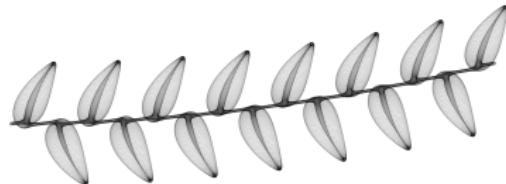
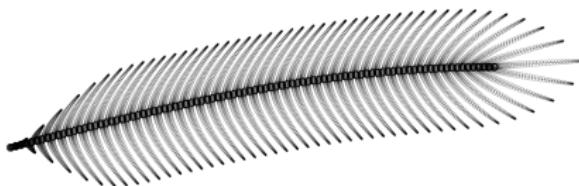
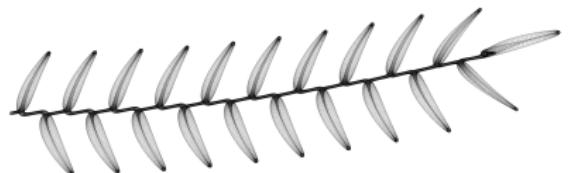
//rud.is/b/2016/03/29/easier-composite-u-s-choropleths-with-albersusa/

Made with ggplot



<https://guangchuangyu.github.io/ggtree/>

Made with ggplot



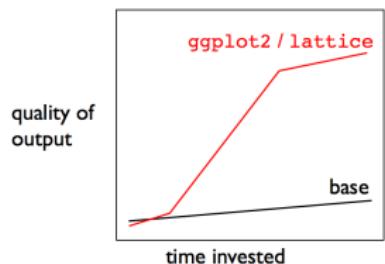
<https://github.com/marcusvolz/mathart>

Why ggplot

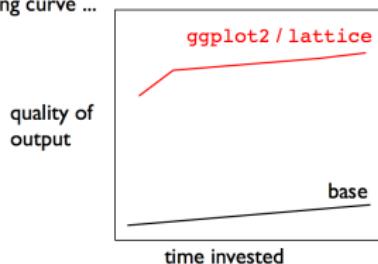
- Extremely powerful and flexible
- Consistent (grammar of graphics)
- Very powerful user base and active development

At the beginning it's hard, but then it pays off

week one



after you've climbed the steepest part of the learning curve ...



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

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

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

Very good documentation and tutorials

- Official `ggplot2` documentation
- `ggplot2` book
- `R graphics cookbook` and `Cookbook for R`
- Beautiful plotting in R: A `ggplot2` cheatsheet
- Introduction to `ggplot2`
- Tutorial: `ggplot2`
- How to format plots for publication using `ggplot2`
- Visualising data with `ggplot2`
- Data Visualization with R and `ggplot2`
- `ggplot2` tutorial
- Data visualisation chapter in R for Data Science
- The complete `ggplot2` tutorial
- Data visualization: a practical introduction (K. Healy)
- Fundamentals of data visualization (C. Wilke)

Cheatsheet

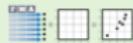
Data Visualization with ggplot2

Cheat Sheet



Basics

`ggplot2` is based on the **grammar of graphics**. The idea is 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's aesthetics (like size, color, and x/y locations).



Complete the template below to build a graph.

```
ggplot([data = mpg], aes(x = cyl, y = hwy))  
  + [ ]  
  + stat = [ ]  
  + position = [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]
```

Required
Not required, sensible defaults supplied

Begin a plot that you finish by adding layers to. Add one geom function per layer.

```
ggplot([x = cyl, y = hwy, data = mpg], geom = "point")  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]
```

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

```
last_plot()  
  Returns the last plot.  
ggplot("plot.png", width = 5, height = 5)  
  Saves last plot as 5" x 5" file named "plot.png" in working directory. Makes file type to file extension.
```

RStudio® is a trademark of RStudio, Inc. | www.rstudio.com | 844-448-2222 | rstudio.com

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 `geom_abline`(*intercept*, *slope*)
- b `geom_bar`(*group*)
- c `geom_blank()` (useful for expanding limits)
- d `geom_curve`(*angle*=0, *x*, *y*, *xend*=*x*, *yend*=*y*, *arrow*, *color*, *curvature*, *length*, *size*)
- e `geom_path`(*method*="butt", *linejoin*=c("round", "miter"))
- f `geom_polygon`(*group*)
- g `geom_rect`(*xmin*=*x*, *xmax*=*x*, *ymin*=*y*, *ymax*=*y*, *alpha*, *color*, *fill*, *linetype*, *size*)
- h `geom_ribbon`(*lower*=*unempoly* - 90, *upper*=*unempoly* + 90, *alpha*, *color*, *fill*, *group*, *linetype*, *size*)

Line Segments

- common aesthetics: *x*, *y*, *alpha*, *color*, *linetype*, *size*
- i `geom_abline`(*intercept*, *slope*)
 - j `geom_hline`(*intercept*)
 - k `geom_vline`(*intercept*)
 - l `geom_segment`(*x*, *xend*=*x*, *y*, *yend*=*y*)
 - m `geom_spline`(*control*=c(1.15, 1))

One Variable

- Continuous
- c `geom_area`(*hsewheight*)
 - d `geom_area`(*stat*="area")
 - e `geom_density`(*kernel*="gaussian")
 - f `geom_densplot`
 - g `geom_freqpoly`
 - h `geom_hex`
 - i `geom_histogram`(*binwidth*=5)
 - j `geom_qqplot`(*sample*=*hwy*)
 - k `geom_violin`(*scale*="area")

Discrete

- l `geom_bar`
- m `geom_count`
- n `geom_dotplot`(*binwidth*=2)
- o `geom_hex`
- p `geom_map`(*map*=*state*, *map*=*map* + *expand_limits*(*x*=*map\$lon*, *y*=*map\$lat*))

Two Variables

- Continuous X, Continuous Y
- e `geom_abline`(*intercept*, *slope*)
 - f `geom_jitter`(*height*=2, *width*=2)

- g `geom_point`
- h `geom_quartile`
- i `geom_rug`(*side*="l")
- j `geom_smooth`(*method*="loess")
- k `geom_text`(*label*=*label*, *method*="nudge", *x*=*x*, *y*, *alpha*, *color*, *fill*, *group*, *linetype*, *size*, *weight*)

- Continuous Function
- l `geom_area`
 - m `geom_line`
 - n `geom_step`(*direction*="hv")

- Continuous Bivariate Distribution
- b `geom_bivar2d`(*smooth*=c(0.25, 500))
 - c `geom_hex`

- d `geom_hex`
- e `geom_hex`

Visualizing error

- ```
#<- data frame with 2 columns ("X", "Y"), n = 4, sa = 2.0
j <- ggplot(df, aes(x = X, y = Y))
```

- f `geom_crossbar`(*sa*=2)
- g `geom_errorbar`

- h `geom_errorbarh`
- i `geom_linerange`

- j `geom_pointrange`
- k `geom_pointrandom`

#### Maps

- ```
data <- data frame with 2 columns ("USArrests$Murder",  
state = USArrests$State)
```
- m `geom_map`(*map*=*state*, *map*=*map* + *expand_limits*(*x*=*map\$lon*, *y*=*map\$lat*))
 - n `geom_map`(*map*=*map*, *id*, *alpha*, *color*, *fill*, *group*, *linetype*, *size*)

Three Variables

- ```
sealida <- melt(seals, vars=c("x", "y", id), value=c("z"))
l <- ggplot(seals, aes(x = x, y = y))
```
- o `geom_raster`(*res*=10, *interpolate*=FALSE)
  - p `geom_hex`
  - q `geom_tile`(*res*=10)

Learn more at [ggplot2.org](https://ggplot2.org) and [ggplot2-exts.org](https://ggplot2-exts.org) • ggplot2 2.3.0 • Updated 11/18

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

## Repos of figures + code

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

Find answers for all your questions in Stack Overflow



## Search

ggplot2

36,854 results



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

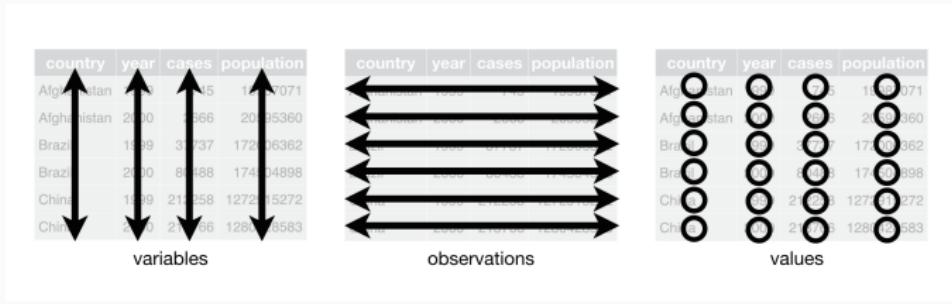
```
library(paperplanes)
head(paperplanes)
```

| id | hour    | person   | gender | age | plane       | paper | distance |
|----|---------|----------|--------|-----|-------------|-------|----------|
| 1  | [17,18) | Roland   | male   | 30  | Standard80  | 80    | 7.8      |
| 2  | [17,18) | Astrid   | female | 30  | Concorde120 | 120   | 2.7      |
| 3  | [17,18) | Roland   | male   | 30  | Standard120 | 120   | 9.2      |
| 4  | [17,18) | Isabella | female | 48  | Standard120 | 120   | 6.0      |
| 5  | [17,18) | Fabienne | female | 17  | Standard120 | 120   | 7.3      |
| 6  | [17,18) | Fabienne | female | 17  | Standard120 | 120   | 7.8      |

Ensuring paper is factor, not numeric

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

# Data must be a tidy data frame



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

The diagram shows the transformation of a wide data frame (`table4`) into a long/tidy data frame using the `tidy::gather` function.

`table4` (wide format):

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

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

| country     | 1999   | 2000   |
|-------------|--------|--------|
| Afghanistan | 745    | 2666   |
| Brazil      | 37737  | 80488  |
| China       | 212258 | 213766 |

<http://r4ds.had.co.nz/tidy-data.html>

## Calling ggplot

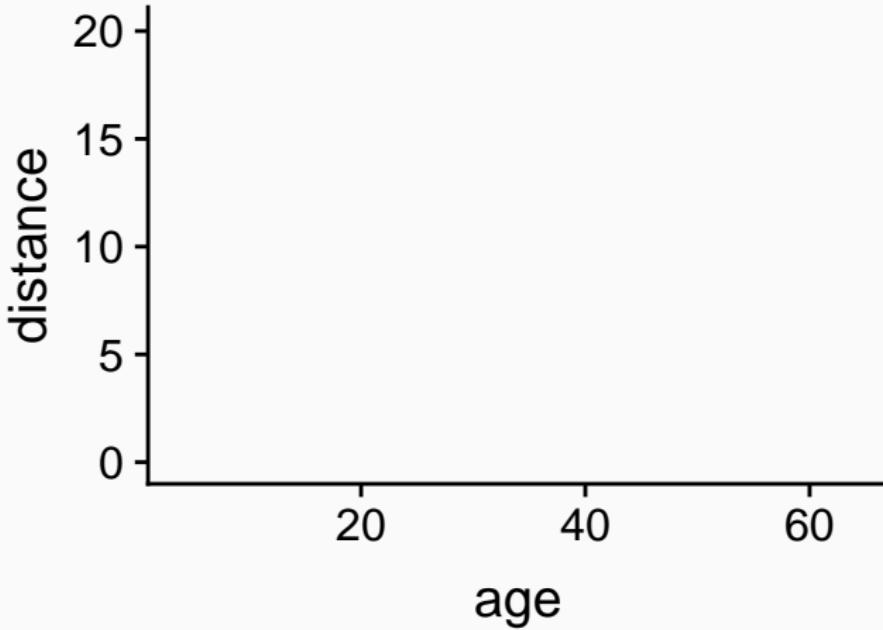
```
library(ggplot2)
ggplot(paperplanes)
```

```
ggplot(paperplanes)
```

First argument is a tidy data frame

## What variables as axes?

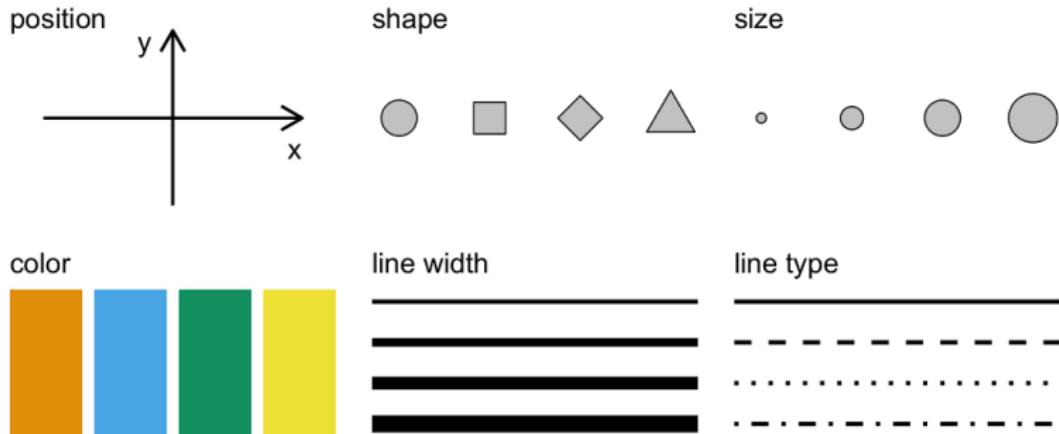
```
ggplot(paperplanes) +
 aes(x = age, y = distance)
```



Note syntax: + followed by new line

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

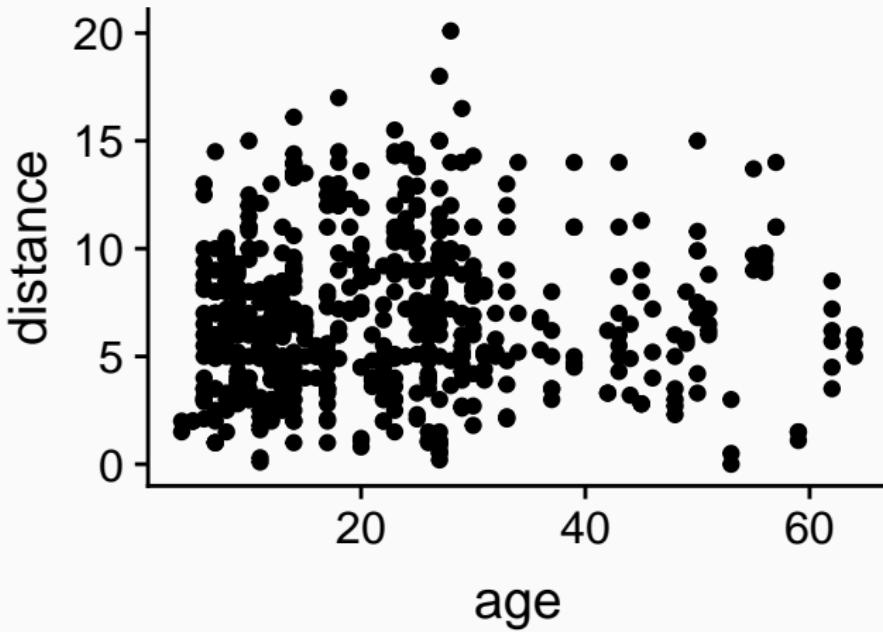
**Aesthetics** (`aes`) map data variables (`age`, `distance`) to graphic elements (`axes`)



<http://serialmentor.com/dataviz/aesthetic-mapping.html>

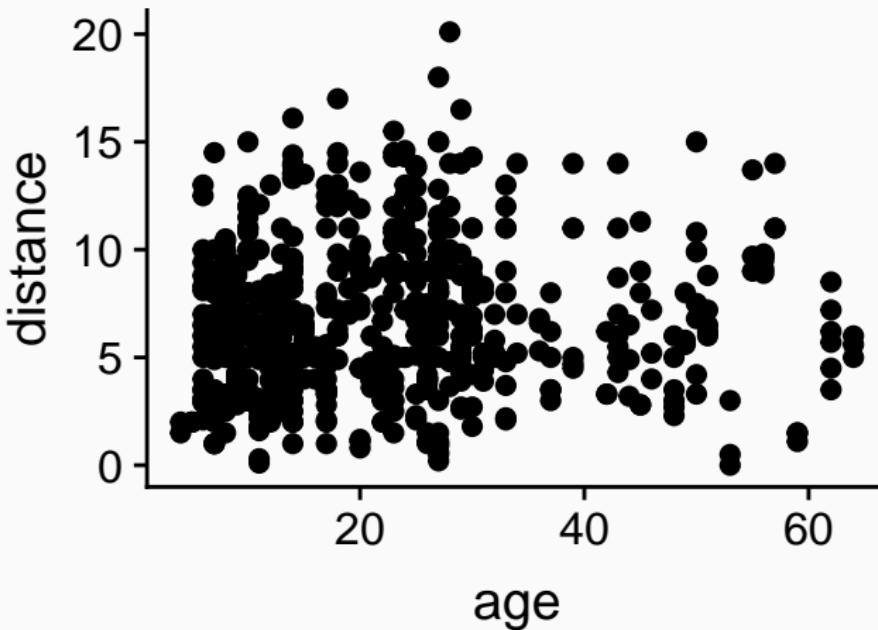
## Adding layers (geoms)

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



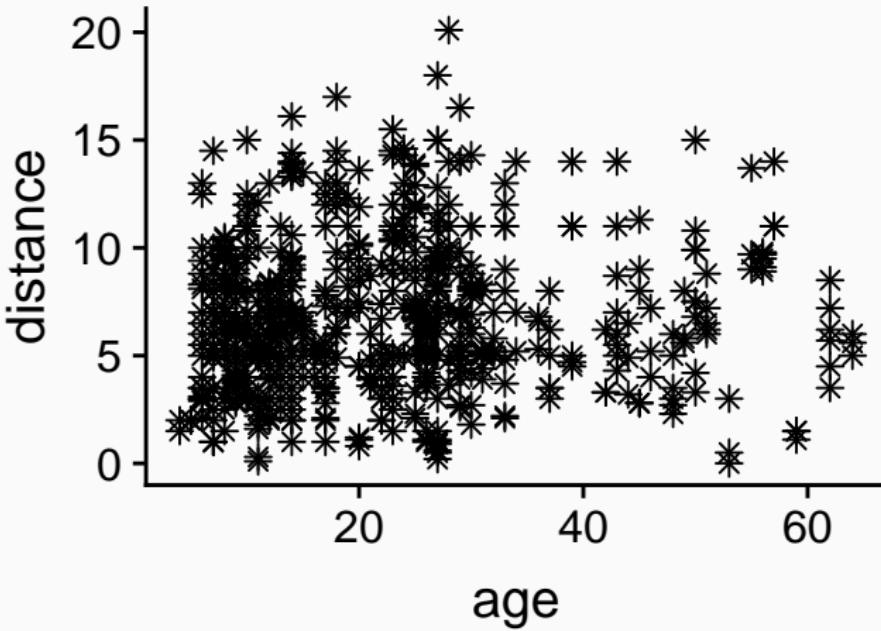
## Changing point size and type

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



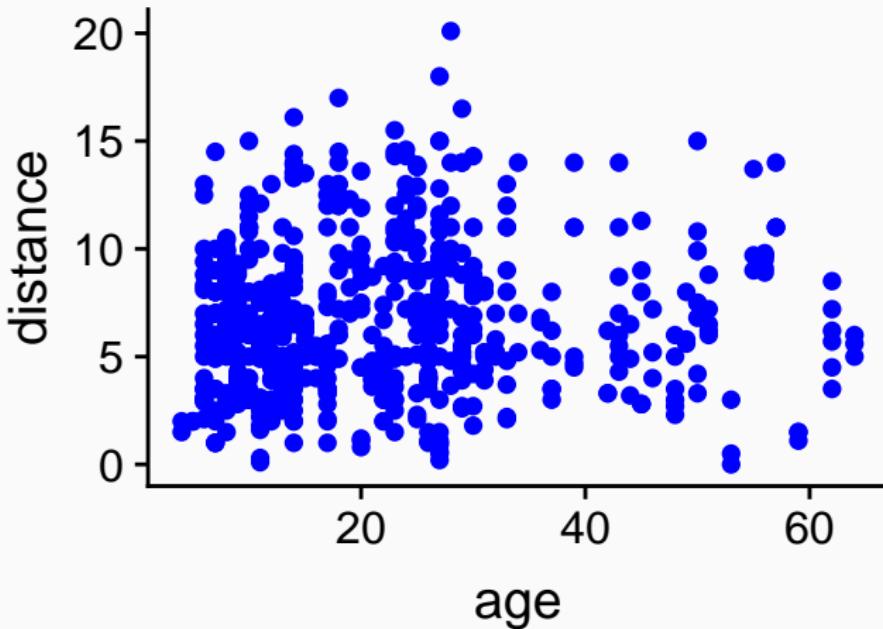
## Changing point size and type

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



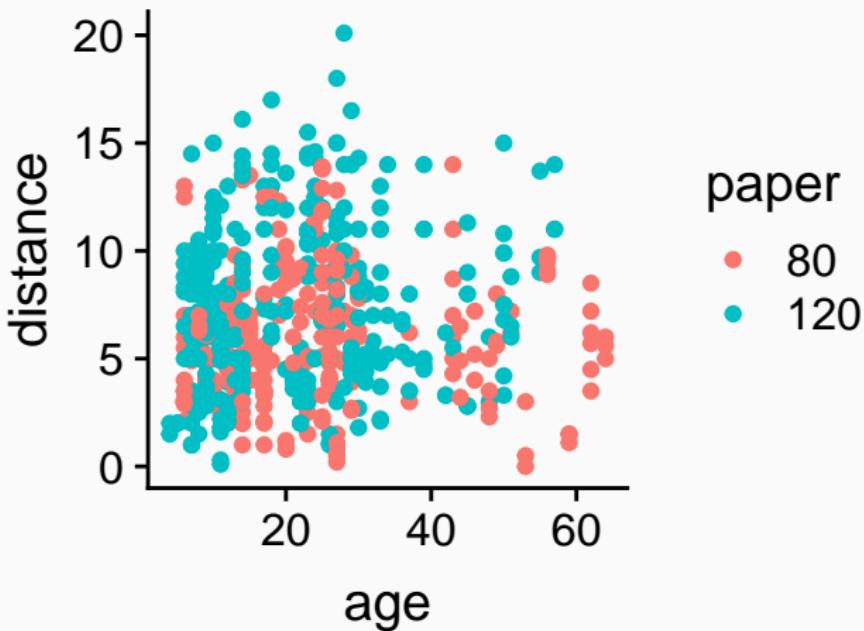
## Changing point size and type

```
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))
```



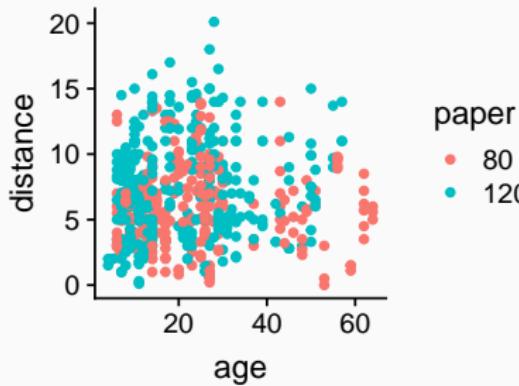
Note difference between

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

```
geom_point(aes(colour = gender))
colour maps a *variable* (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)
```

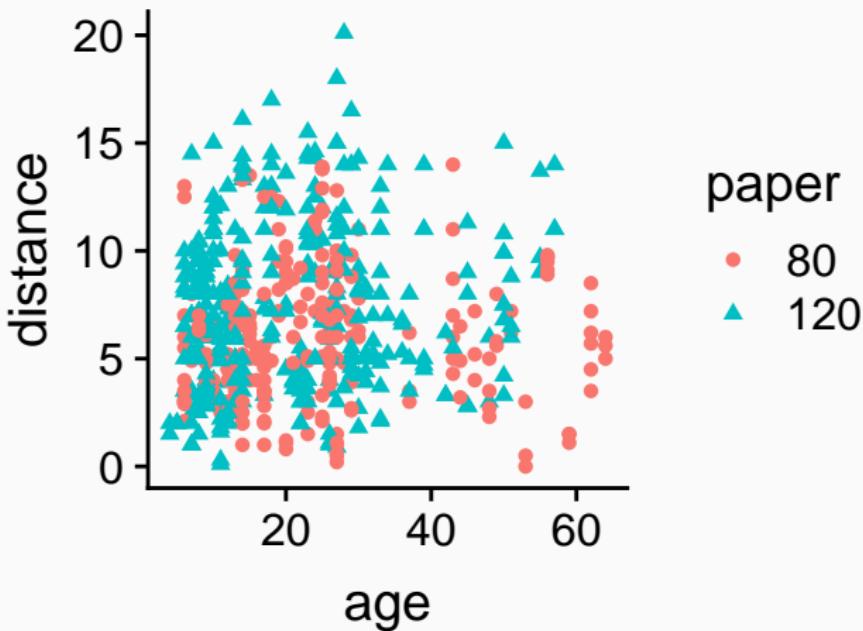
*Error in layer(data = data, mapping =  
mapping, stat = stat, geom =  
GeomPoint, : object 'paper' not found*

'paper' is a variable in dataframe

**Must use aes**

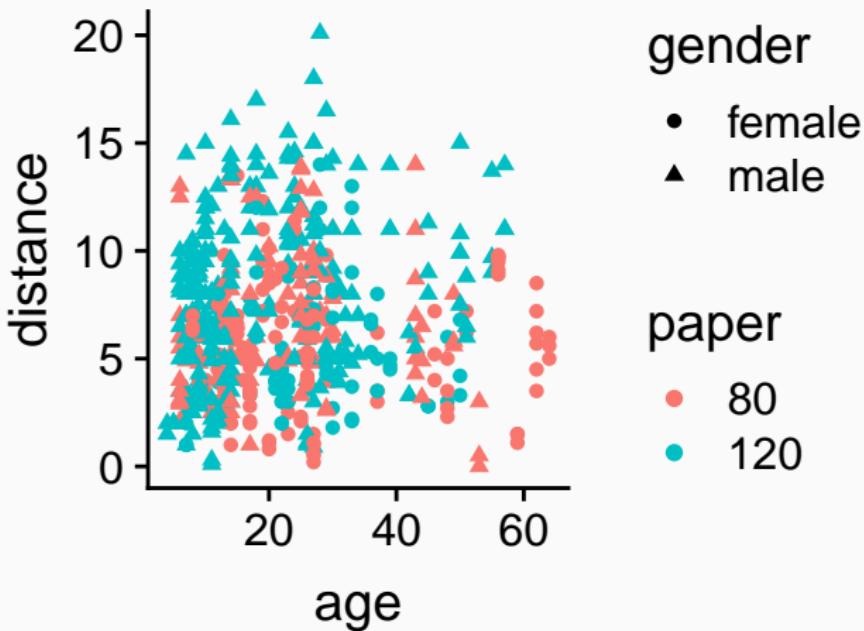
## Map geom aesthetics (colour, shape) to variable

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



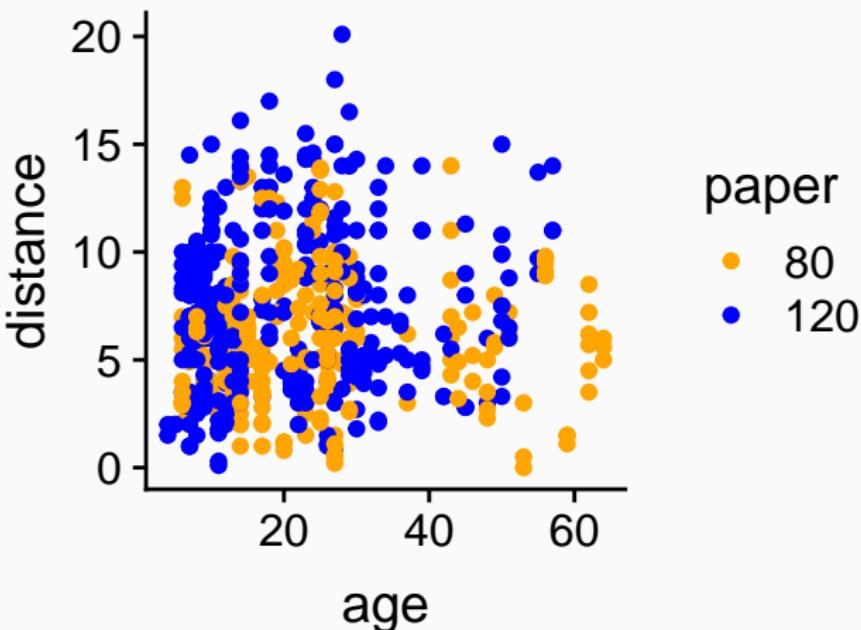
## Map geom aesthetics (colour, shape) to variable

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



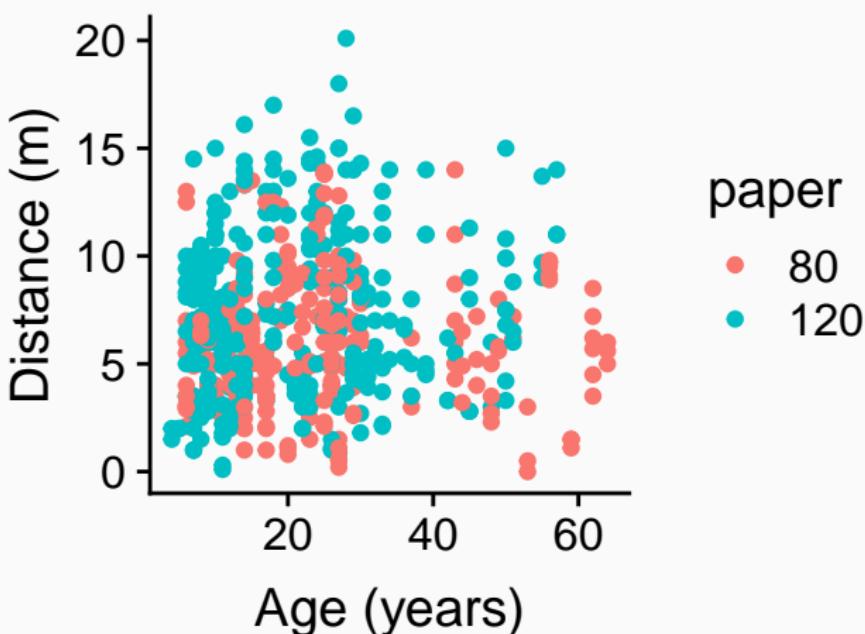
## Change colour scale

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



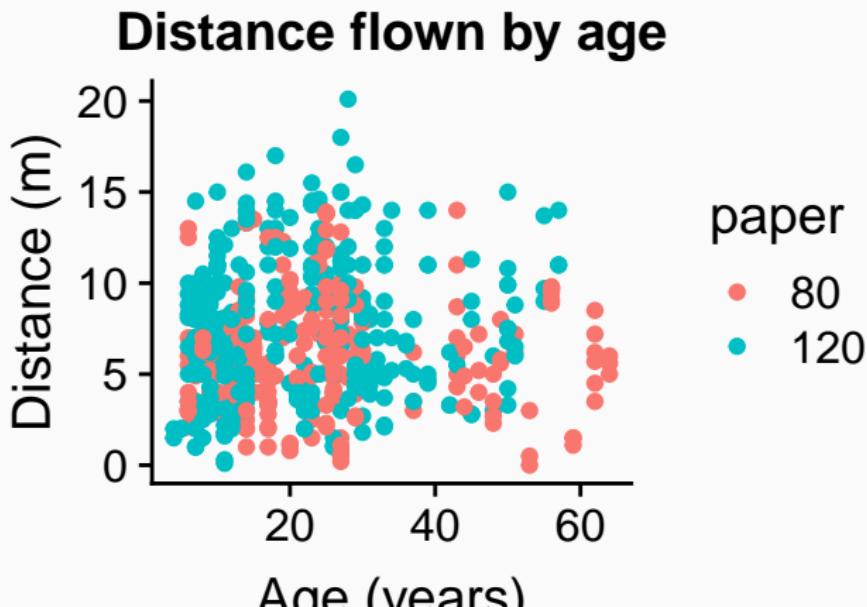
## Change axis labels: `xlab` & `ylab`

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



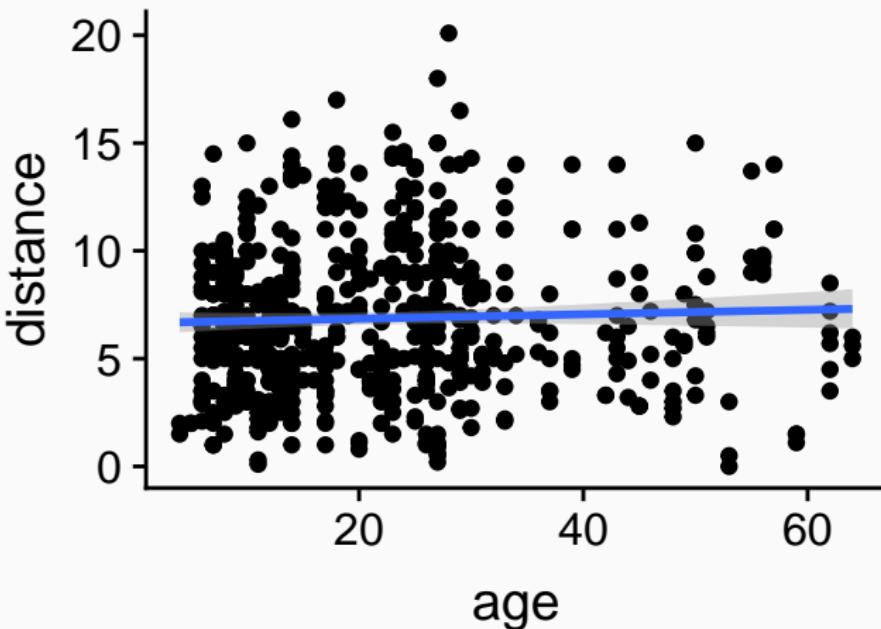
Set title

```
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")
```



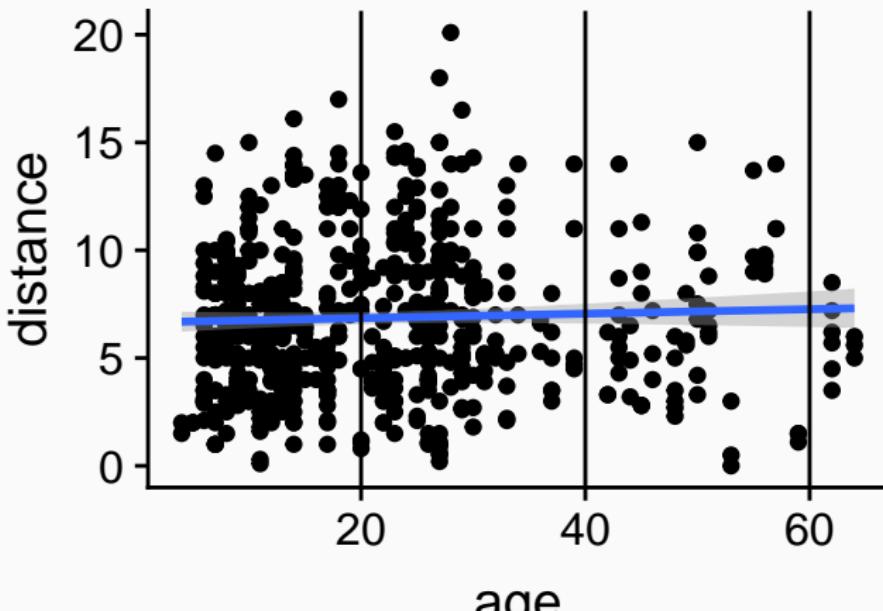
## Adding more layers

```
ggplot(paperplanes) +
 aes(x = age, y = distance) +
 geom_point() +
 geom_smooth(method = "lm")
```



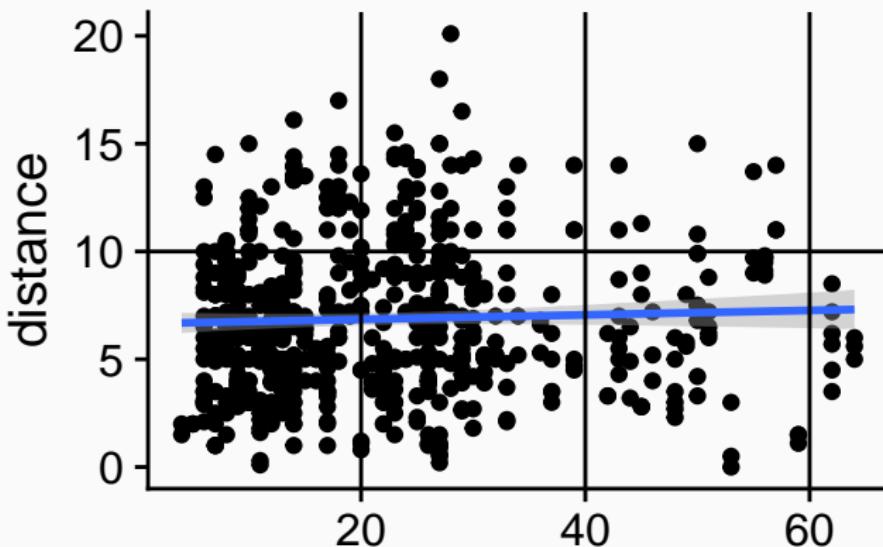
## Adding more layers

```
ggplot(paperplanes) +
 aes(x = age, y = distance) +
 geom_point() +
 geom_smooth(method = "lm") +
 geom_vline(xintercept = c(20, 40, 60))
```



## Adding more layers

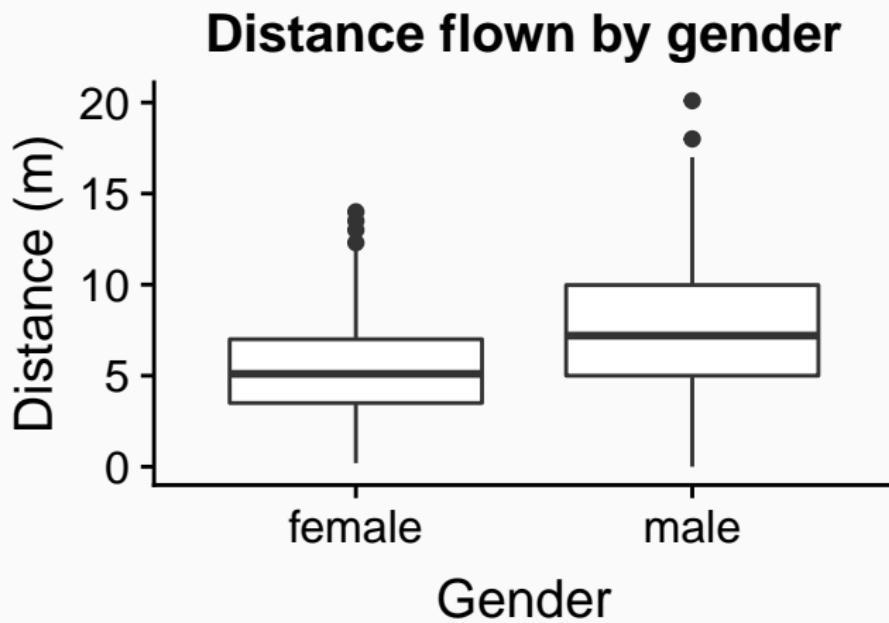
```
ggplot(paperplanes) +
 aes(x = age, y = distance) +
 geom_point() +
 geom_smooth(method = "lm") +
 geom_vline(xintercept = c(20, 40, 60)) +
 geom_hline(yintercept = 10)
```



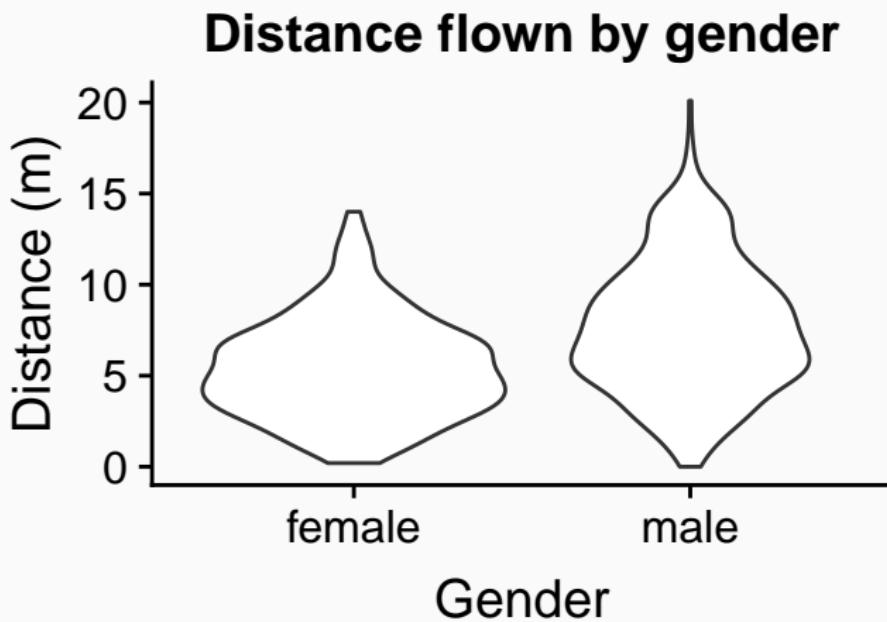
## Summary

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

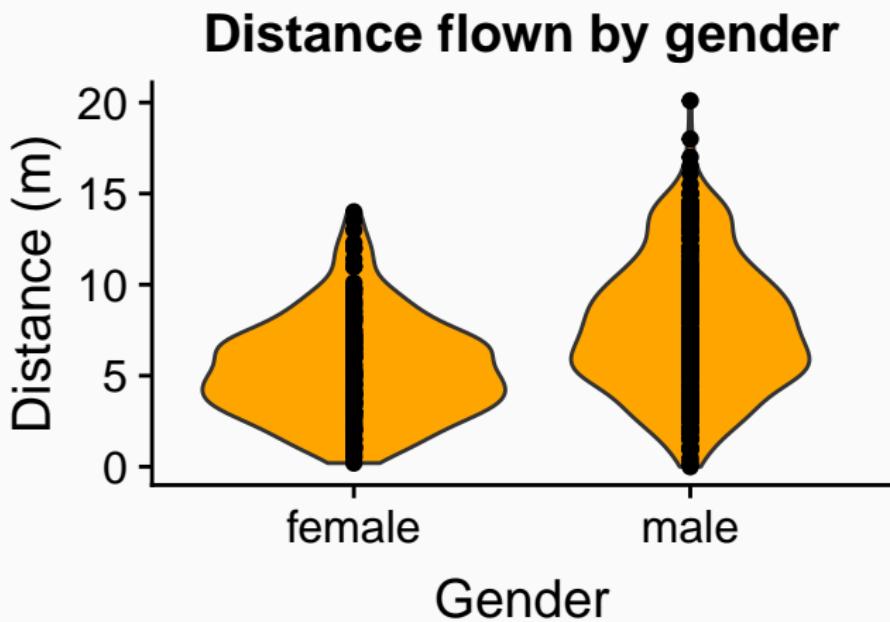
Exercise: Make a plot like this one



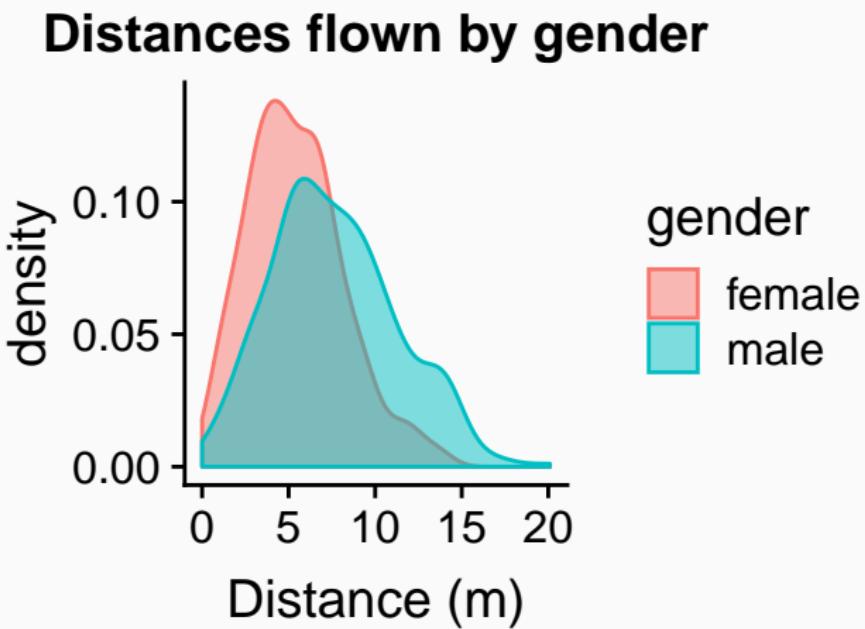
Exercise: Make a plot like this one



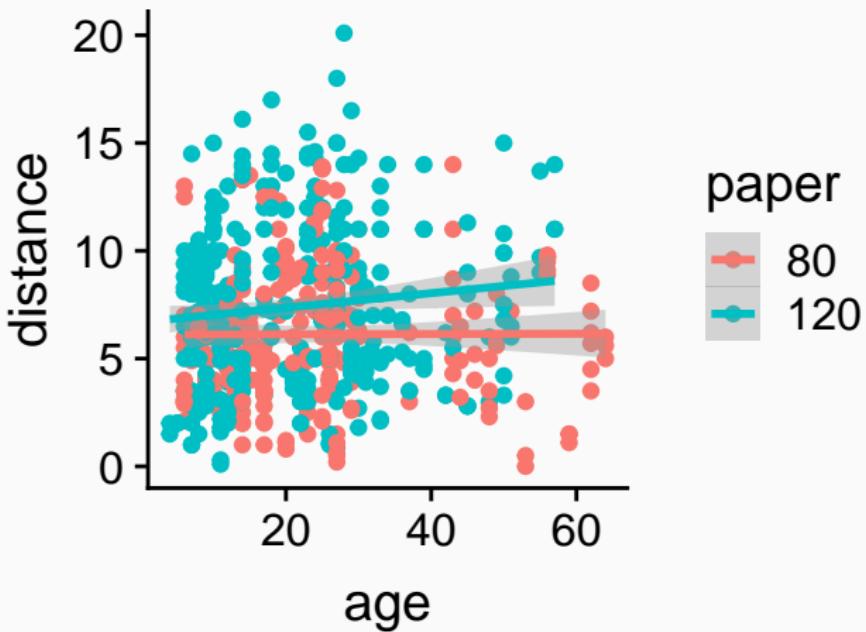
Exercise: Make a plot like this one



Exercise: Make a plot like this one



## Exercise: Make a plot like this one

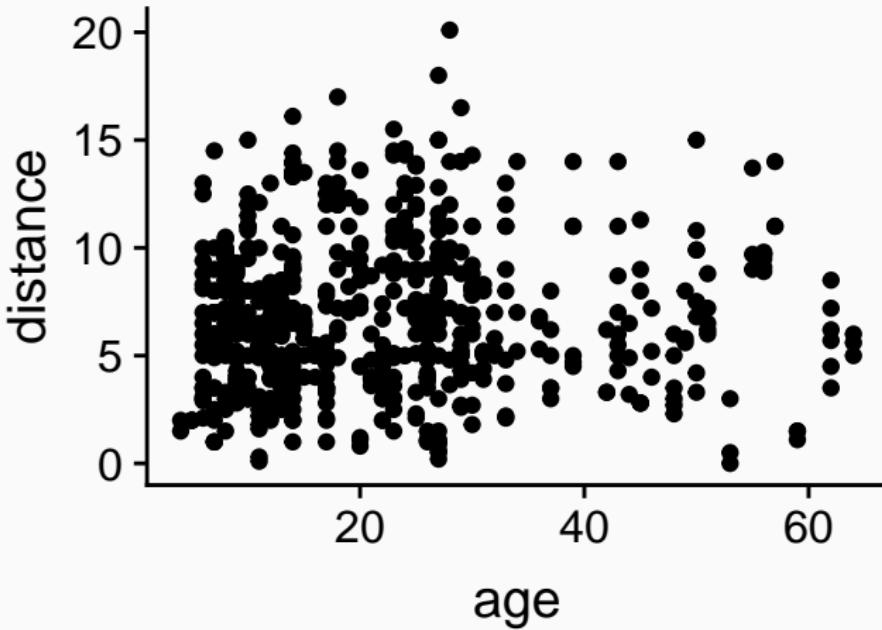


**ggplot2 figures can be assigned to R objects**

---

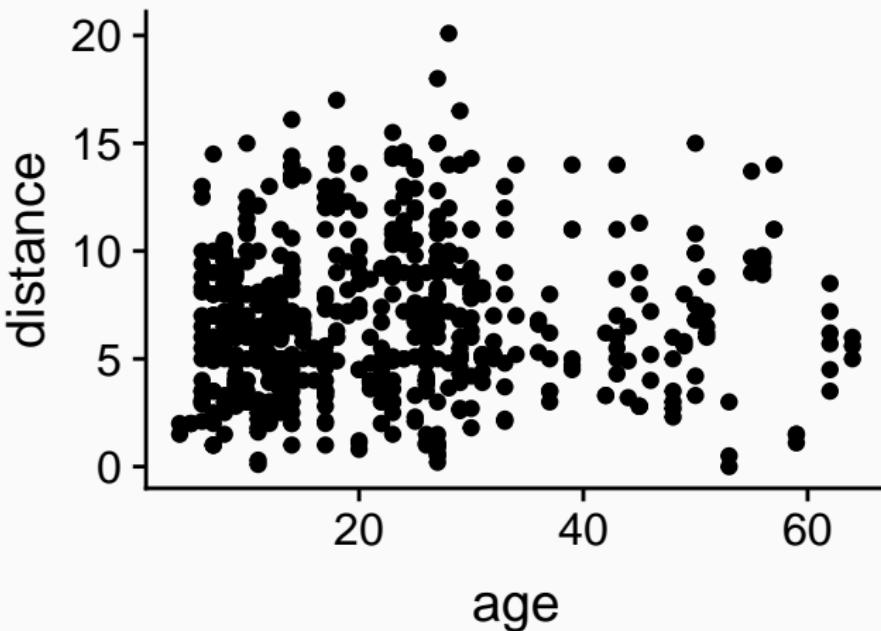
## Assigning ggplot objects

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



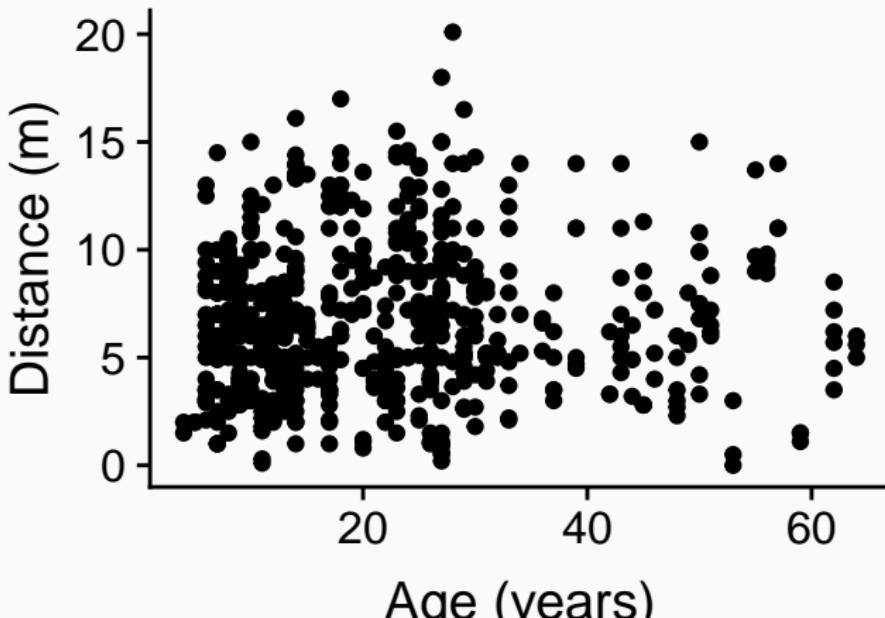
## Assigning ggplot objects

```
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
```

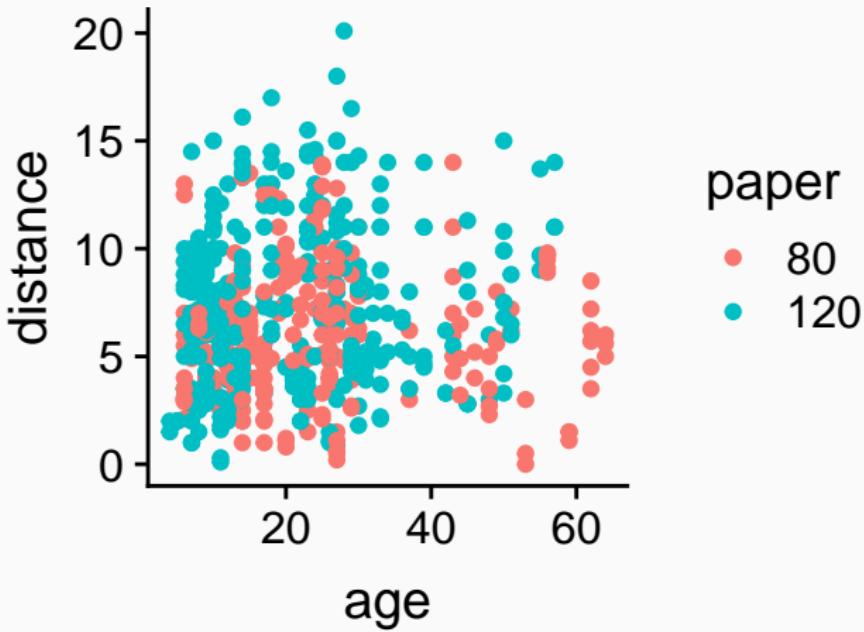


## Themes: changing plot appearance

---

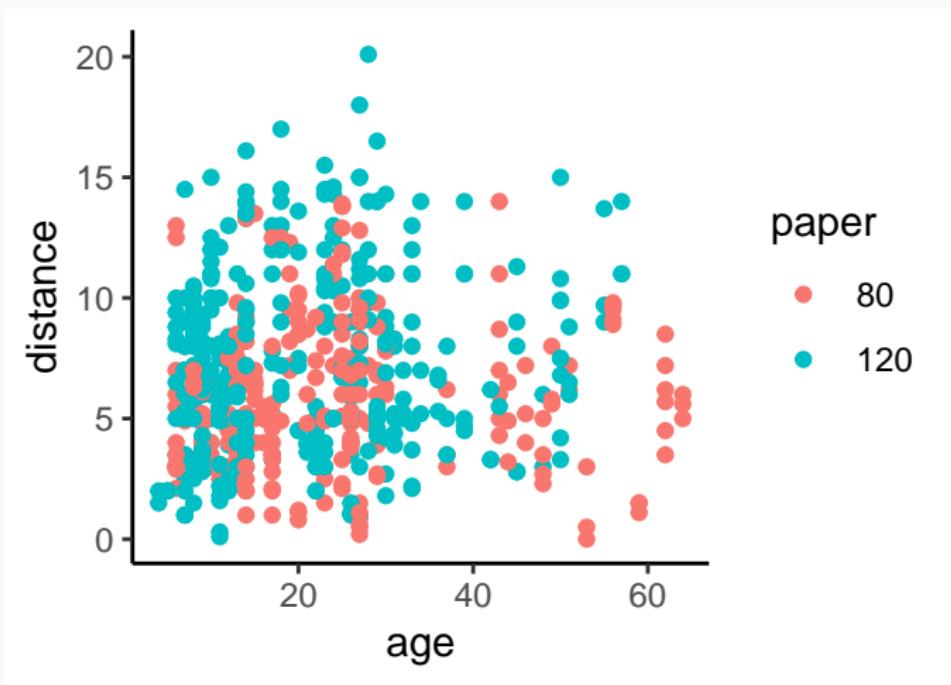
## myplot

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



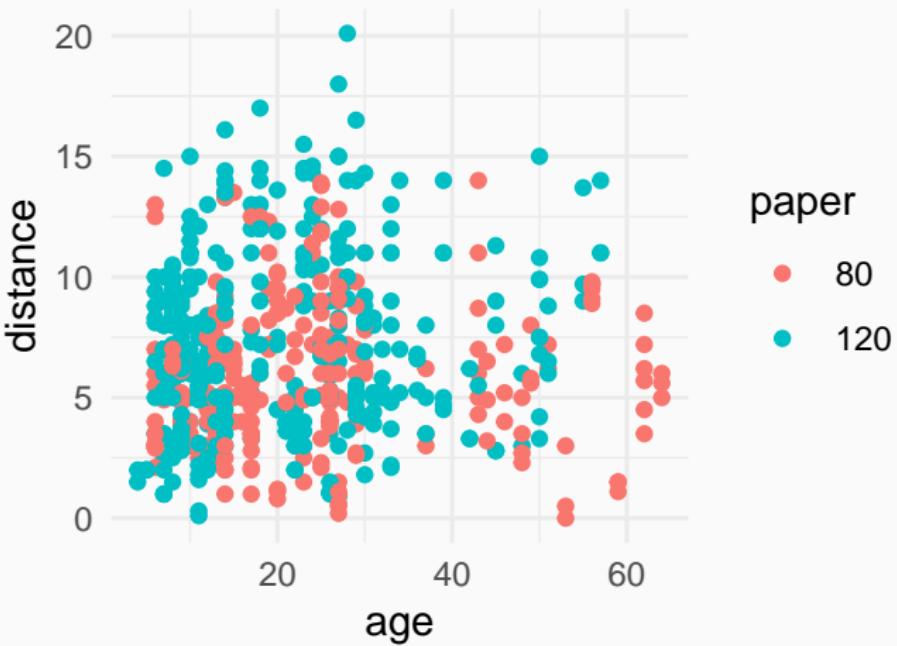
## theme\_classic

```
myplot + theme_classic()
```



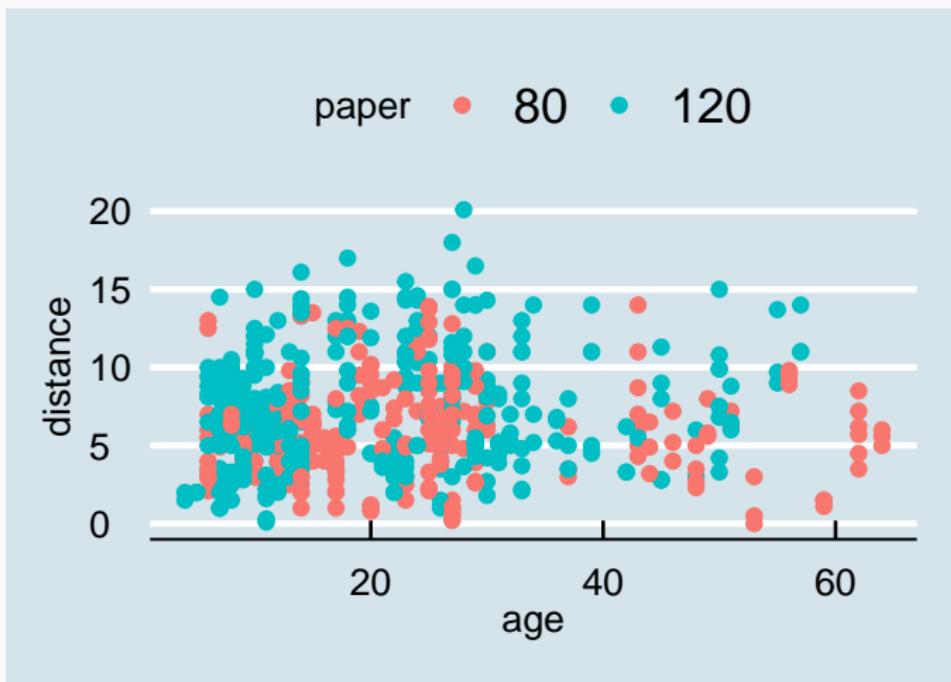
## theme\_minimal

```
myplot + theme_minimal()
```



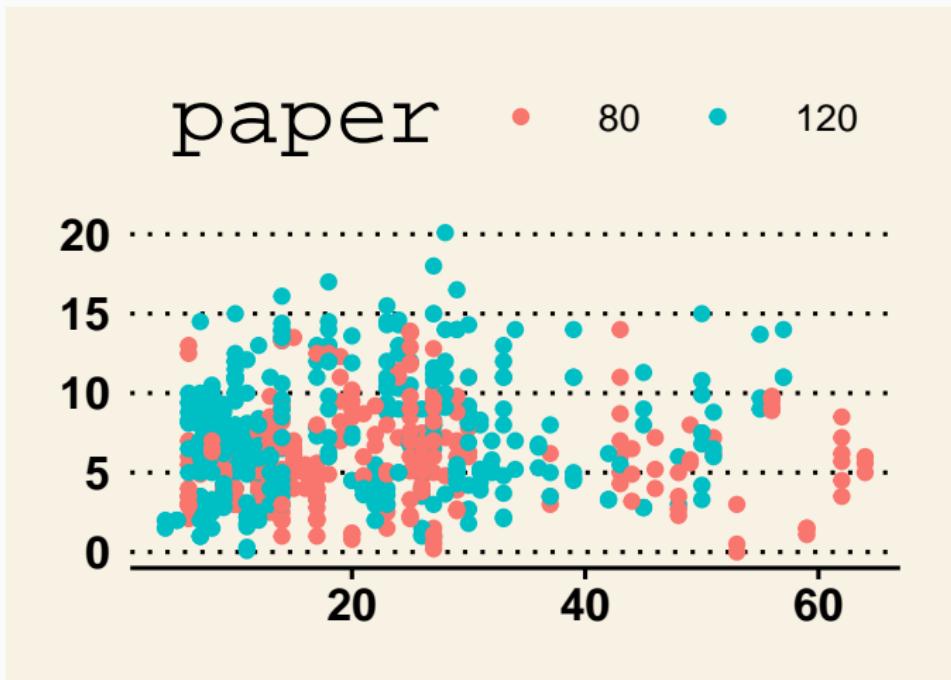
## Lots of themes out there

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



Lots of themes out there

myplot + theme\_wsj()

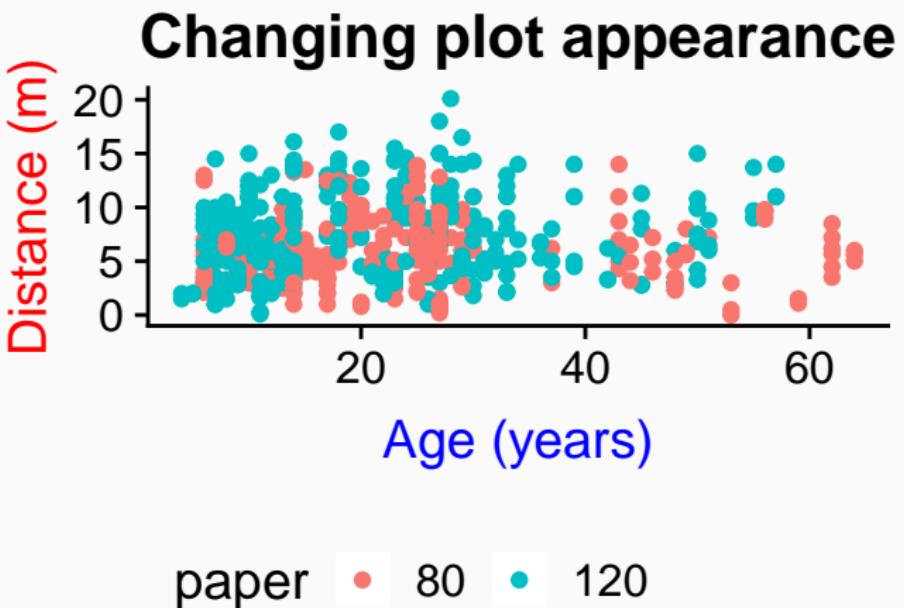


## Editing themes

?theme

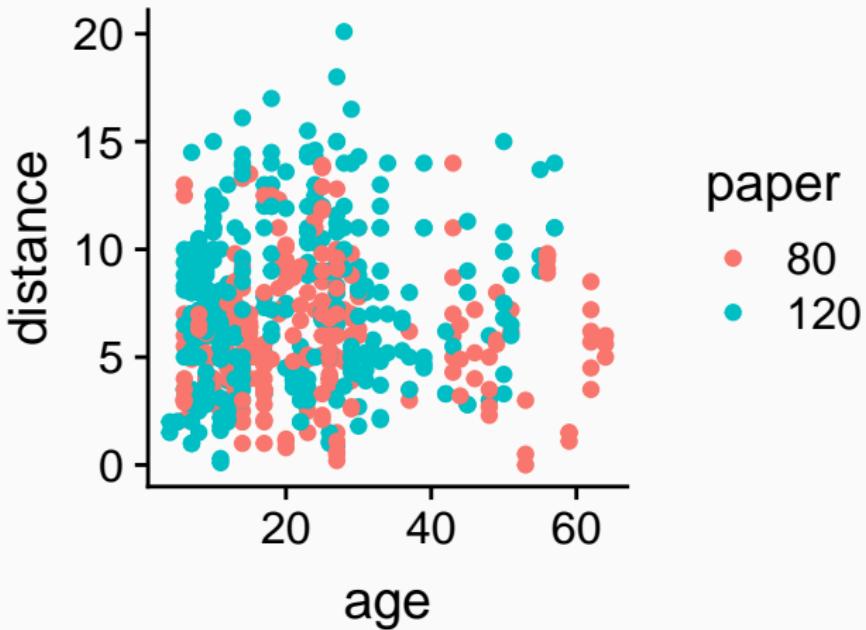
- element\_blank
- element\_text
- element\_line
- element\_rect (borders & backgrounds)

Exercise: make a plot like this one



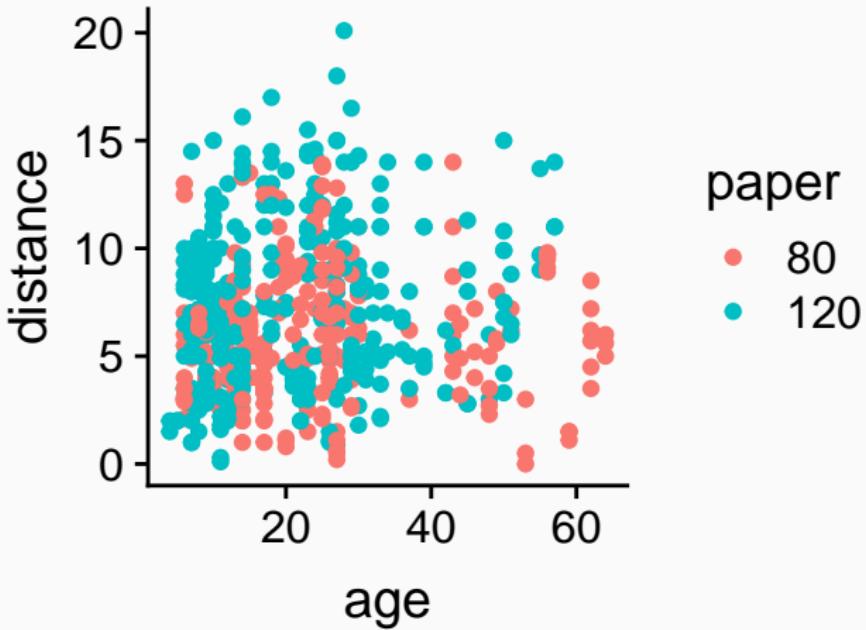
## Easily changing appearance with ggthemeassist (Rstudio addin)

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



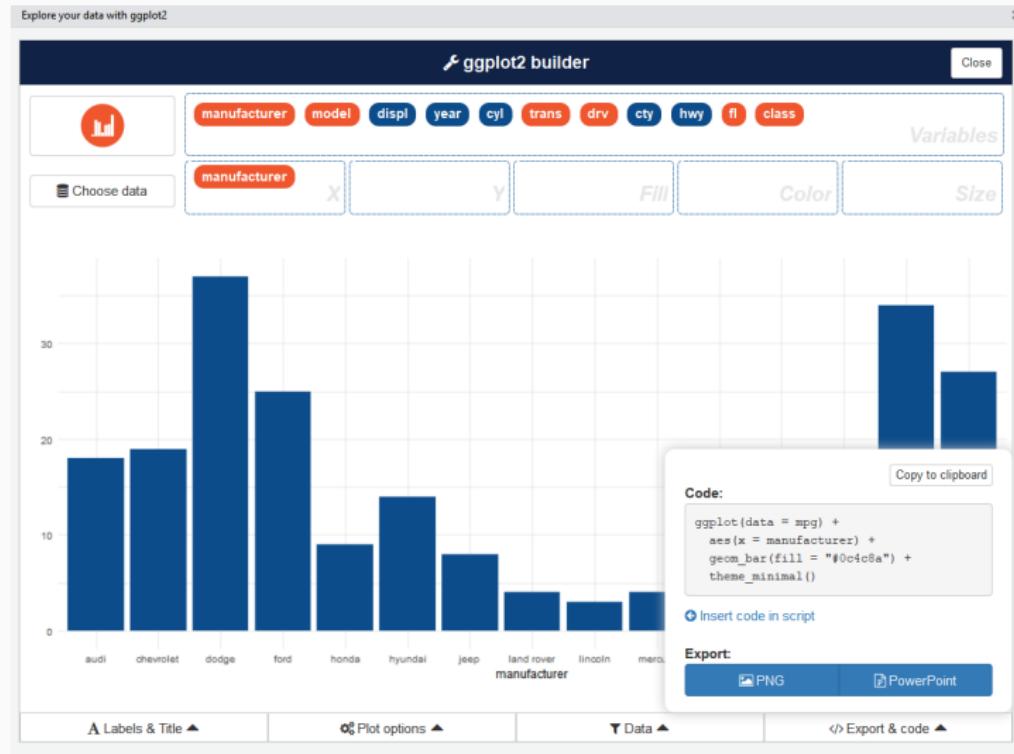
## Easily changing appearance with ggedit

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



# esquisse: ggplot2 builder addin

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





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

---

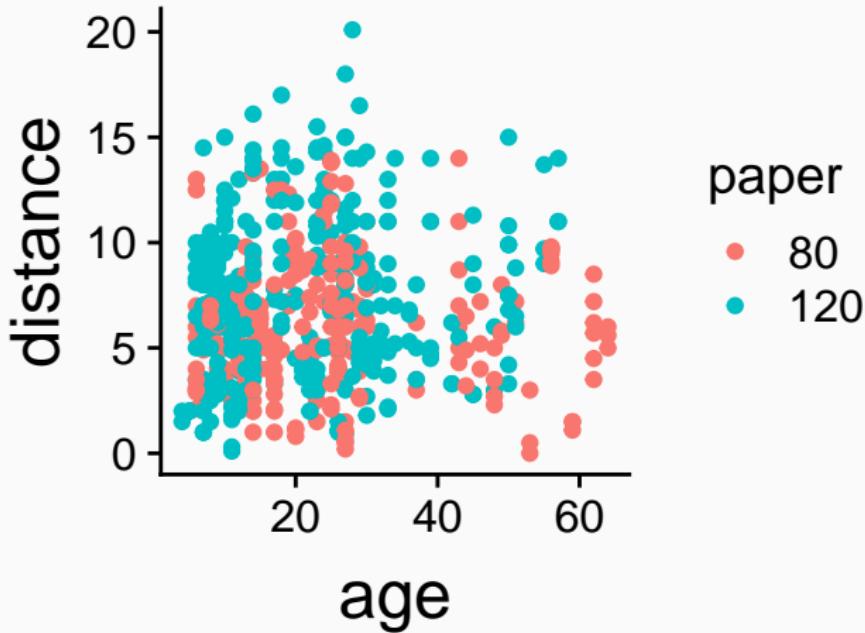
<http://mbjoseph.github.io/2015/02/26/plotting.html>

[serialmentor.com/dataviz/choosing-the-right-visualization-software.html](http://serialmentor.com/dataviz/choosing-the-right-visualization-software.html)

## Think twice before editing plots out of R

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

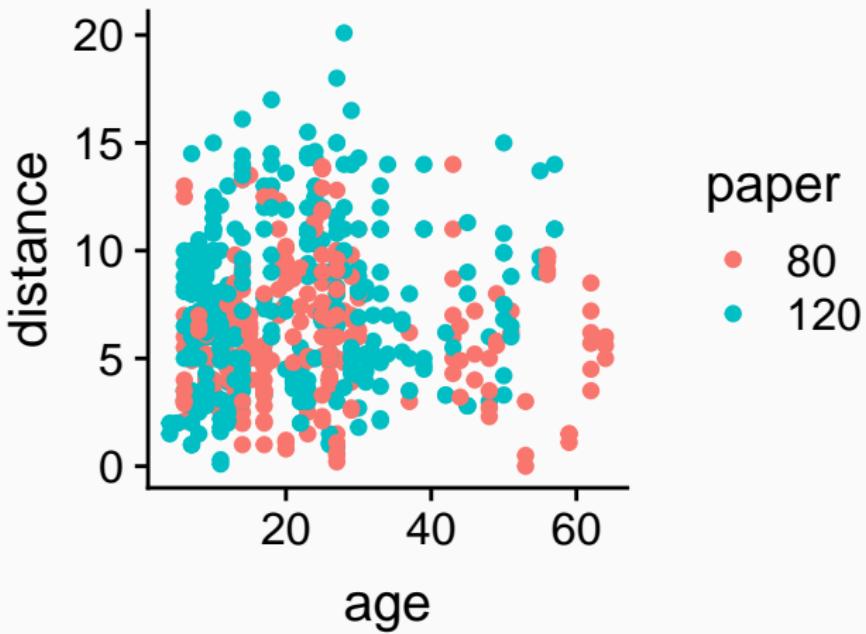
```
myplot +
 theme(axis.title = element_text(size = 18))
```



## Publication-quality plots

```
library(cowplot)
```

```
myplot
```



Some publication themes:

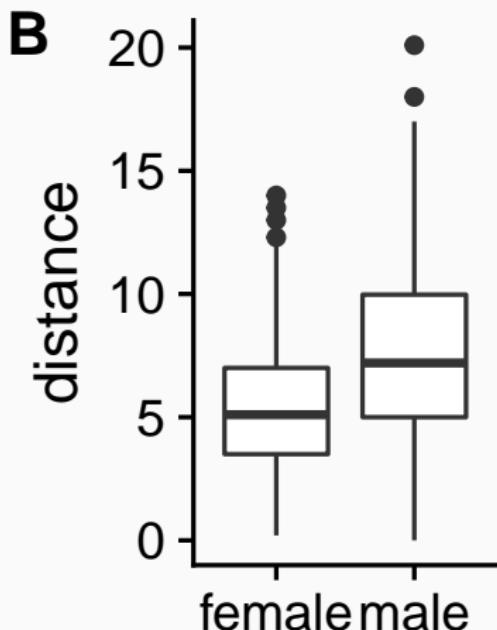
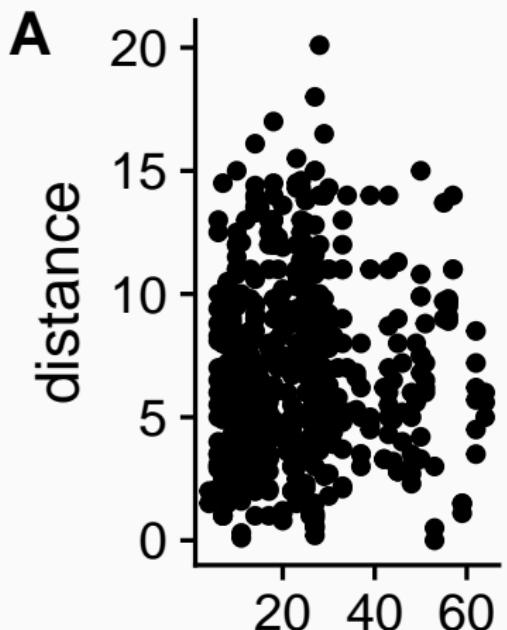
<https://gist.github.com/Pakillo/c2c7ea11c528cc2ee20f#themes>

## **Composite figures**

---

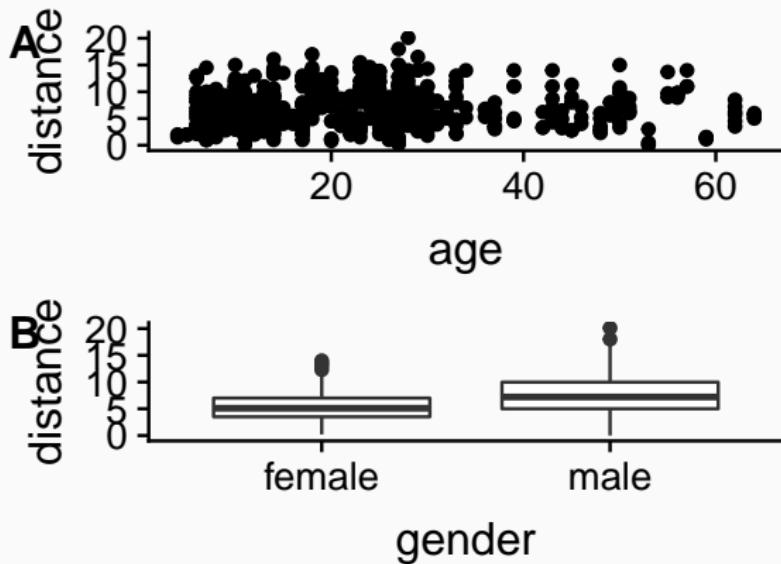
## Composite figures: cowplot

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

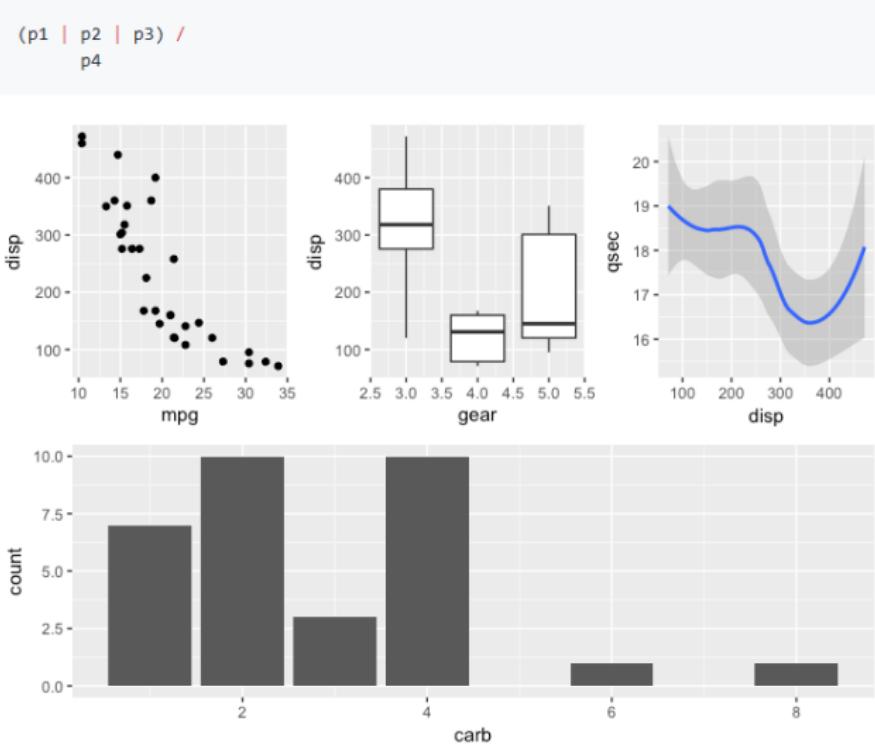


## Composite figures

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

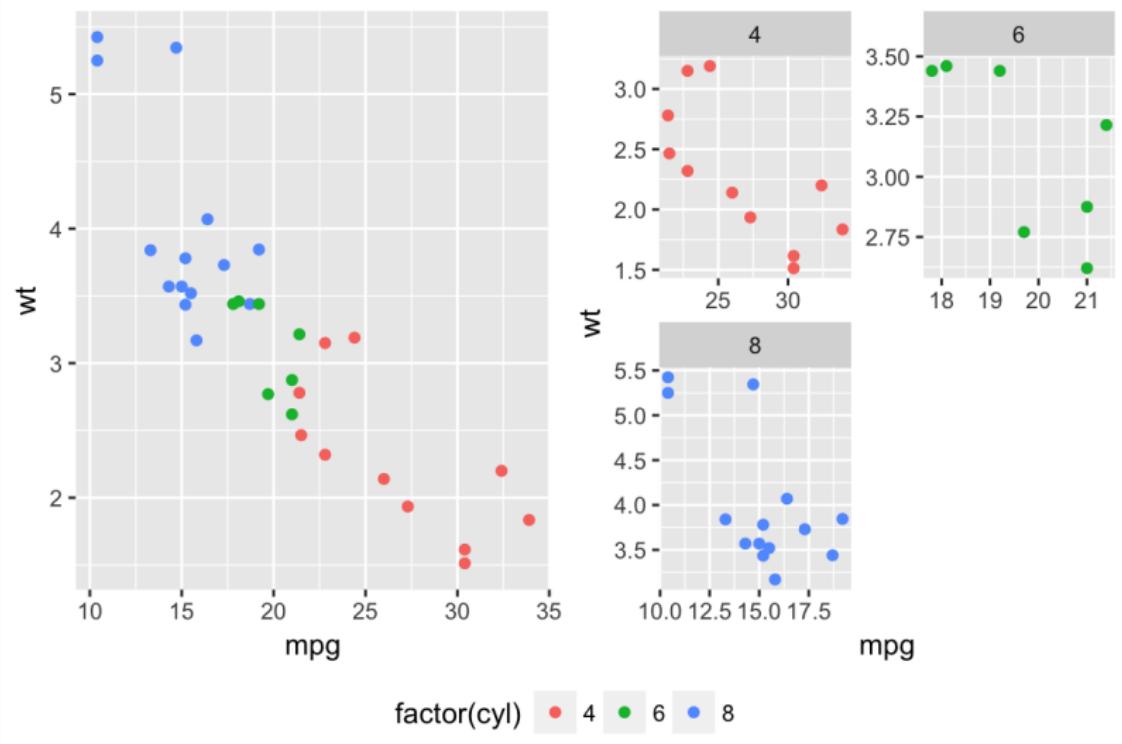


# Composite figures: patchwork



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

## Composite figures: egg



<https://cran.r-project.org/web/packages/egg/index.html>

## Saving plot

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

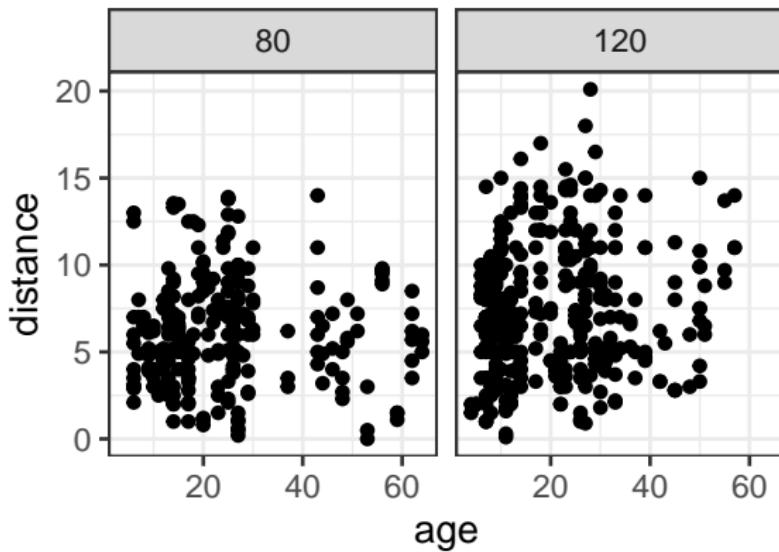
```
save_plot("myplot.pdf")
```

## Facetting (small multiples)

---

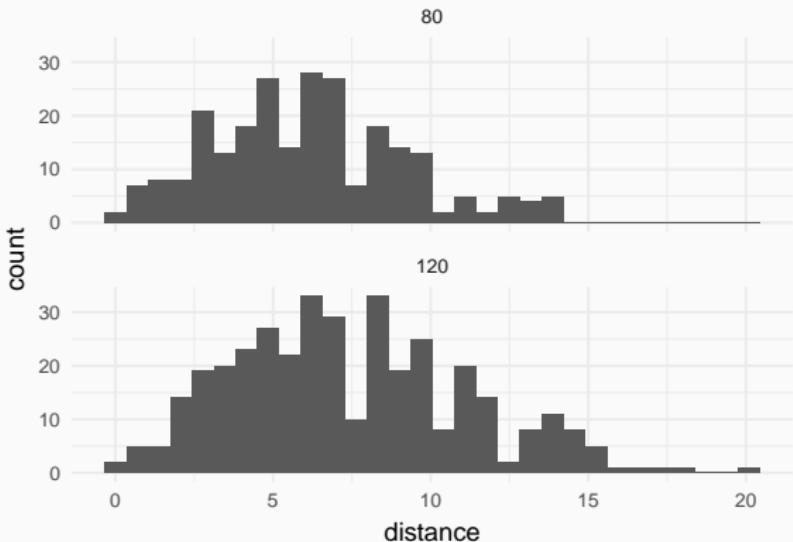
## Facetting

```
ggplot(paperplanes) + aes(age, 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)
```

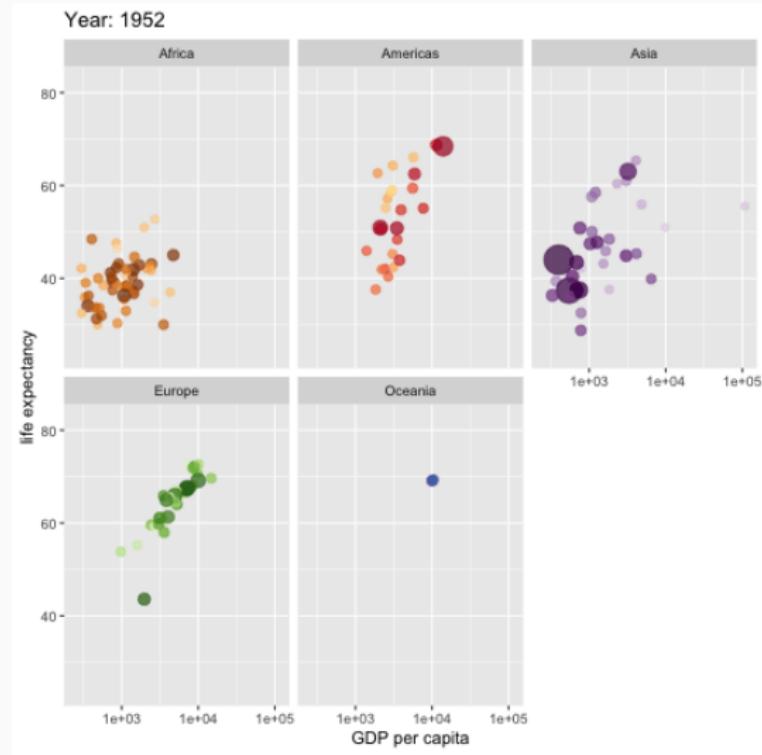


## Interactivity: plotly

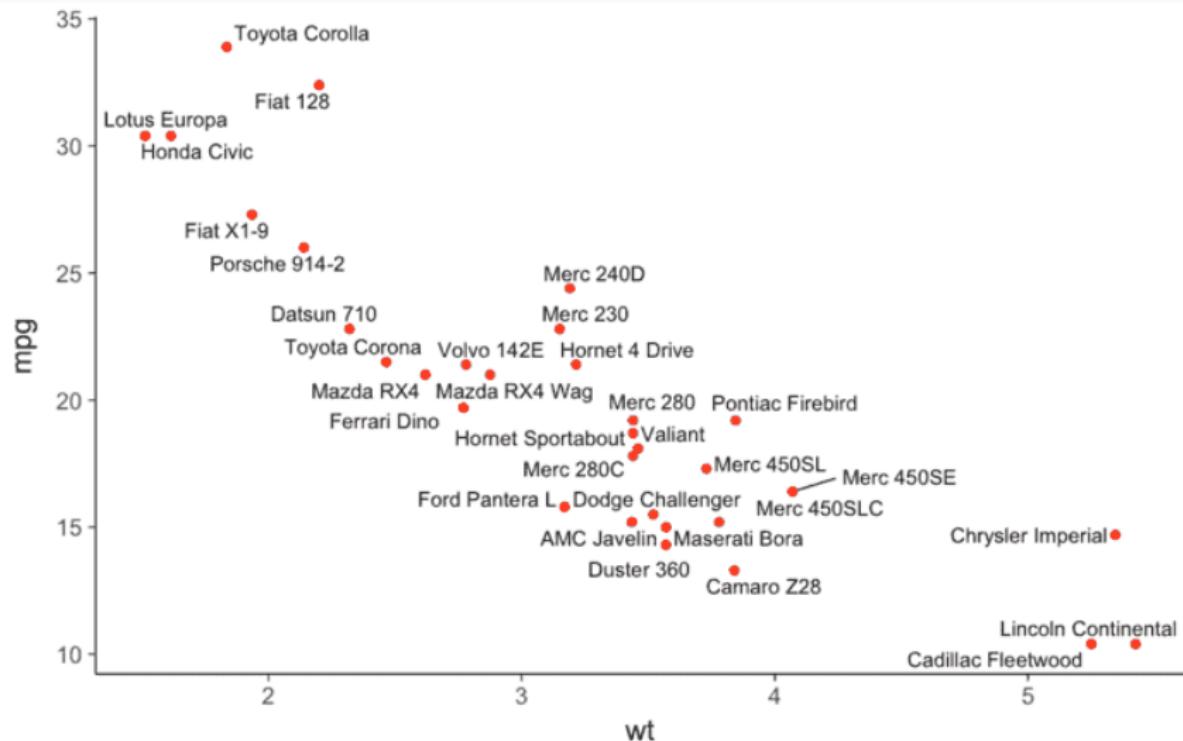
```
library(plotly)
myplot <- ggplot(paperplanes) +
 aes(age, distance)) +
 geom_point()
ggplotly(myplot)
```

# Animated graphs

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



## Automatic label placement



# Many extensions!

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

40 registered extensions available to explore

Sort: Github stars    Filter: search name, author, desc...    Author Filter    Tag Filter    CRAN Only

Showing 40 of 40

**gg3D** (star)   
3D perspective plots for ggplot2.  
• **author:** Daniel Adler  
• **tags:** 3D, Visualization  
• **js libraries:**

**ggQC** (star)   
Use ggQC to plot single, faceted and multi-layered quality control charts.  
• **author:** Kenneth Gray  
• **tags:** QC, XMR, XbarR, SixSigma, Visualization  
• **js libraries:**

**ggedit** (star)   
ggedit is aimed to interactively edit ggplot layers, scales and themes aesthetics  
• **author:** jordic  
• **tags:** visualization, interactive, shiny, general  
• **js libraries:**

**ggforce** (star)   
ggforce is aimed at providing missing functionality to ggplot2 through the extension system introduced with ggplot2 v2.0.0.  
• **author:** thomasp85  
• **tags:** visualization, general  
• **js libraries:**

**ggalt** (star)   
A compendium of 'geom's, 'coords' and 'stats' for 'ggplot2'.  
• **author:** hrbrmstr  
• **tags:** visualization, general  
• **js libraries:**

**ggraph** (star)   
Homologous to make...  
• **author:** denigrider  
• **tags:** visualization, general  
• **js libraries:**

**ggstance** (star)   
ggstance implements horizontal versions of common ggplot2 geoms.  
• **author:** forbesl  
• **tags:** visualization, general  
• **js libraries:**

**ggrepel** (star)   
label overlapping text labels away from each other.  
• **author:** slowflow  
• **tags:** visualization, general  
• **js libraries:**

**ggraph** (star)   
ggraph is tailored at plotting graph-like data structures (graphs, networks, trees, hierarchies,...).  
• **author:** thomasp85  
• **tags:** visualization, general  
• **js libraries:**

**ggpmisc** (star)   
Miscellaneous Extensions to 'ggplot2'.  
• **author:**  
• **tags:** visualization, general  
• **js libraries:**

**geomnet** (star)   
geomnet implements network visualizations in ggplot2 via geom.net.  
• **author:** schmitt  
• **tags:** visualization, general  
• **js libraries:**

**ggExtra** (star)   
ggExtra lets you add histograms to ggplot2.  
• **author:** damian  
• **tags:** visualization, general  
• **js libraries:**

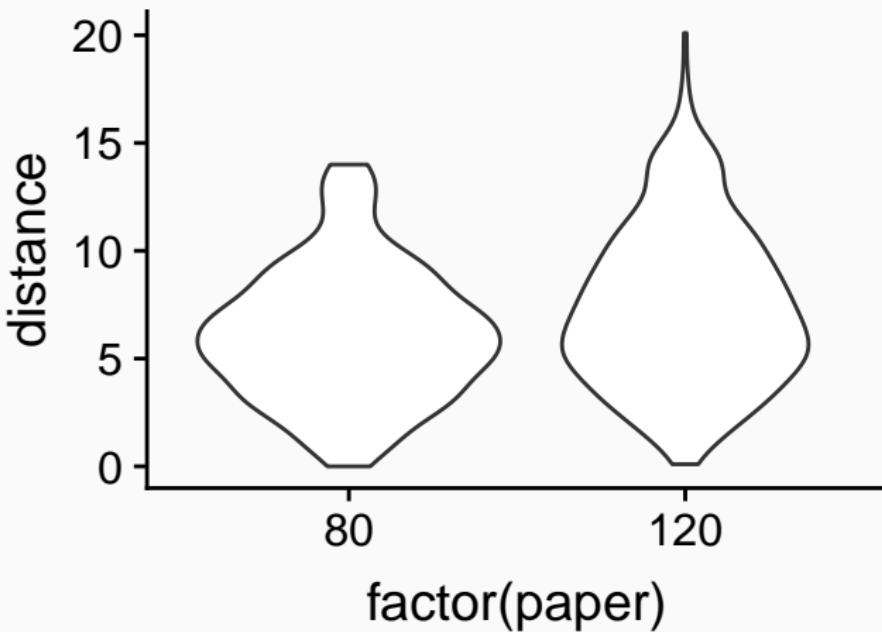
## Summary

---

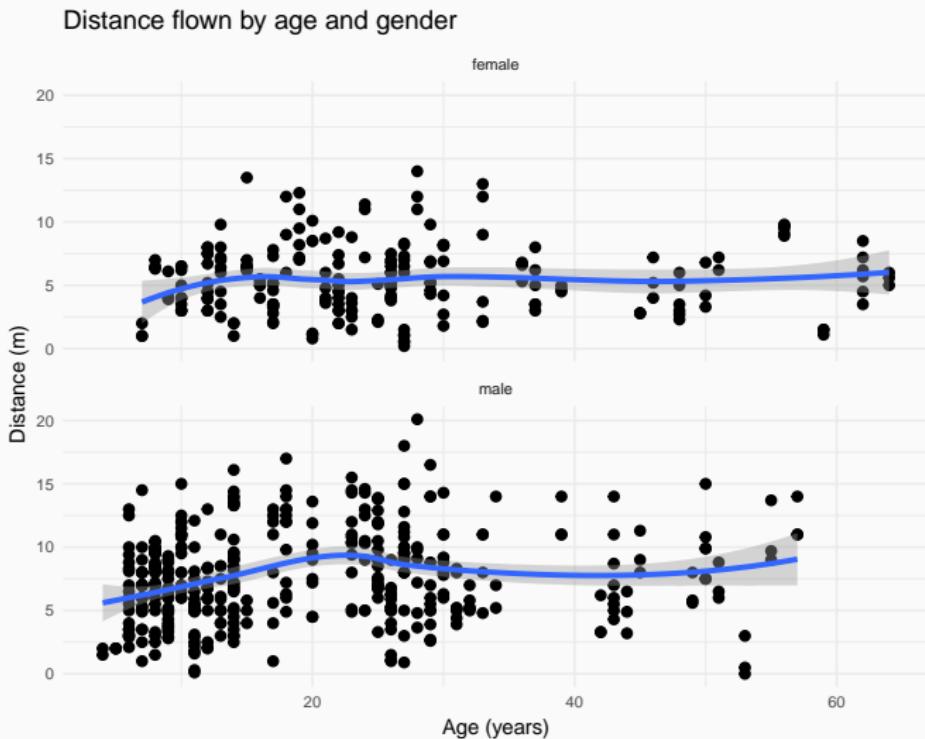
## Grammar of graphics

- **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...)

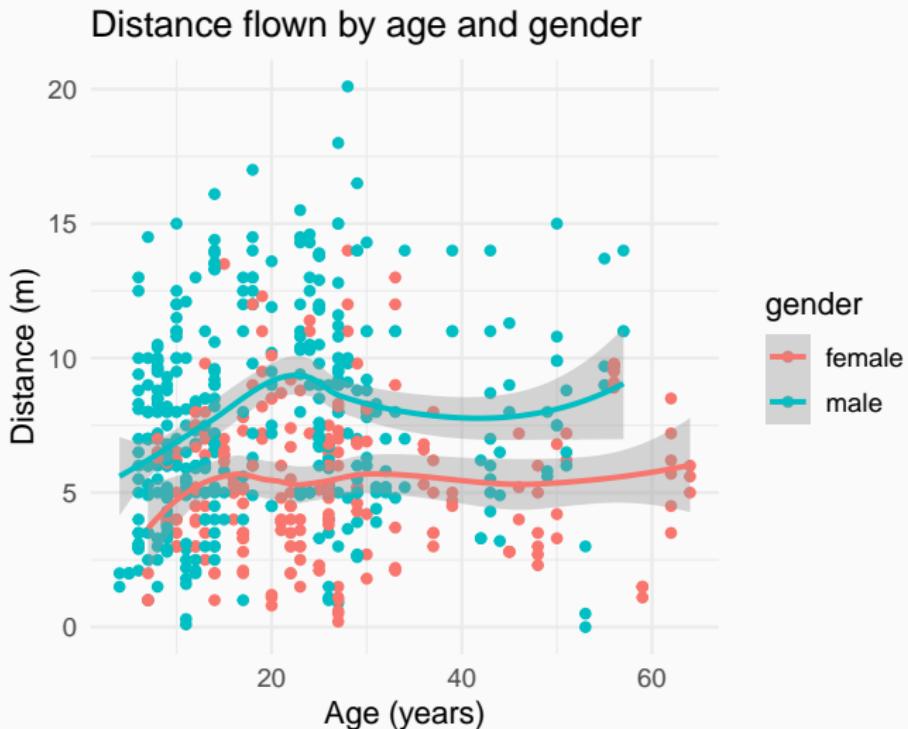
Exercise: make a plot like this one



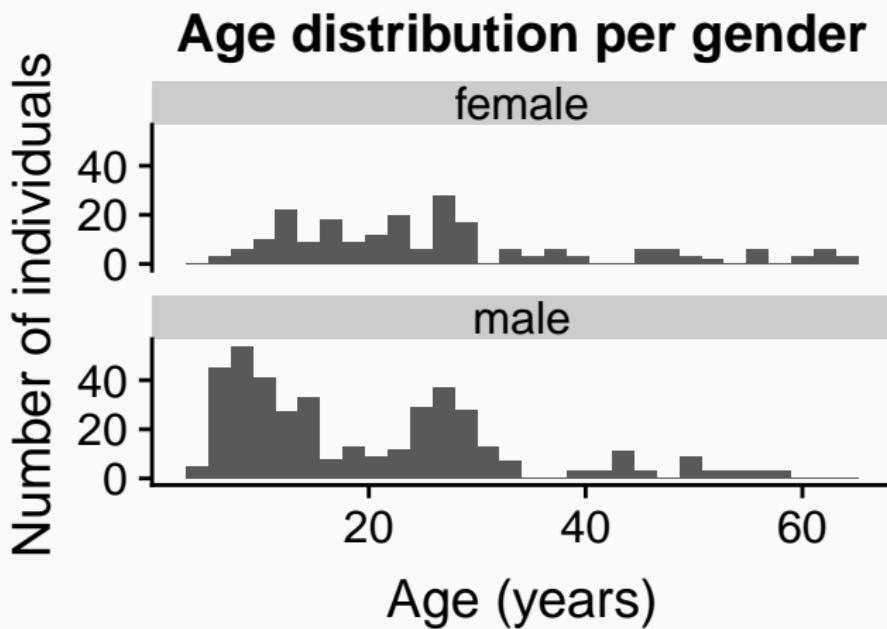
## Exercise: make a plot like this one



## Exercise: make a plot like this one

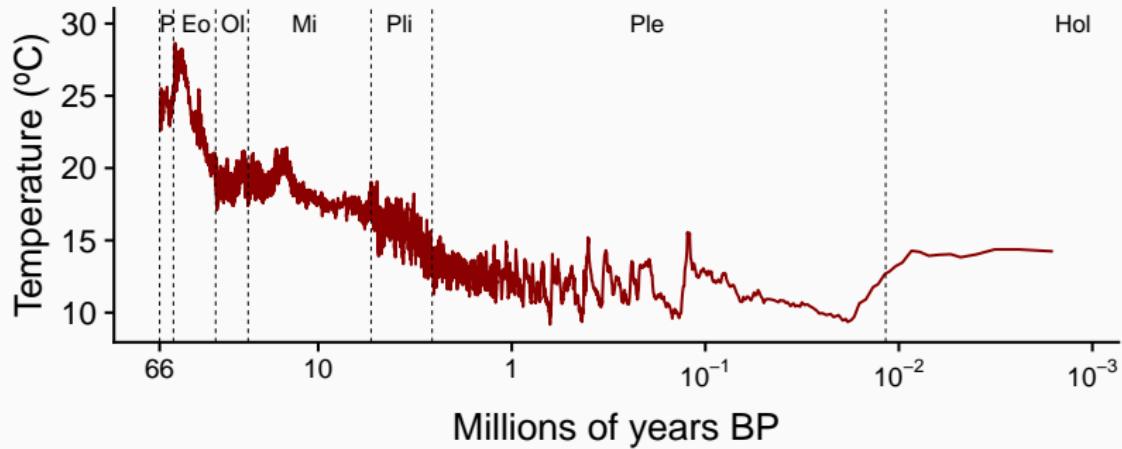


Exercise: make a plot like this one

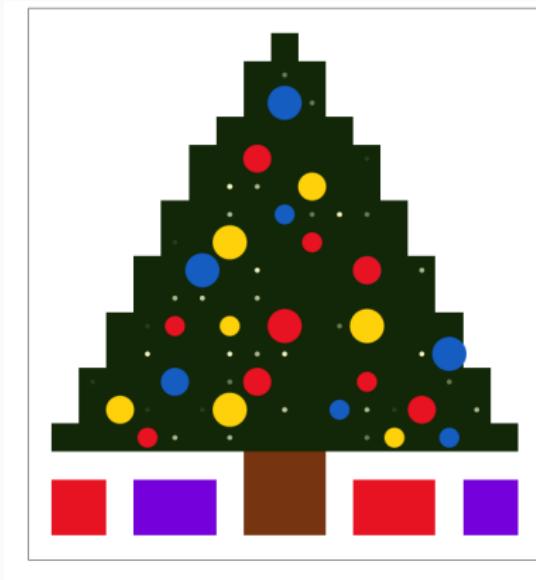


## Exercise: make a plot like this one

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>