

# Visualising data with ggplot2

## Some guiding principles...

1. You should visualise your data early and often.
2. Visualising = understanding.

# This session focuses on the `ggplot2` package

There are several `other` ways of creating plots in R.

- Base graphics e.g. `plot(...)`
- The `lattice` package
- The `vcd` package

# This session focuses on the `ggplot2` package

There are several `other` ways of creating plots in R.

- Base graphics e.g. `plot(...)`
- The `lattice` package
- The `vcd` package

They are all ~~terrible~~ *different*.

## Histograms

### Description

The generic function `hist` computes a histogram of the given data values. If `plot = TRUE`, the resulting object of [class](#) "histogram" is plotted by [plot.histogram](#), before returned.

### Usage

```
hist(x, ...)
```

```
## Default S3 method:
```

```
hist(x, breaks = "Sturges",  
     freq = NULL, probability = !freq,  
     include.lowest = TRUE, right = TRUE,  
     density = NULL, angle = 45, col = NULL, border = NULL,  
     main = paste("Histogram of" , xname),  
     xlim = range(breaks), ylim = NULL,  
     xlab = xname, ylab,  
     axes = TRUE, plot = TRUE, labels = FALSE,  
     nclass = NULL, warn.unused = TRUE, ...)
```

## Forest plot to display the result of a meta-analysis

## Description

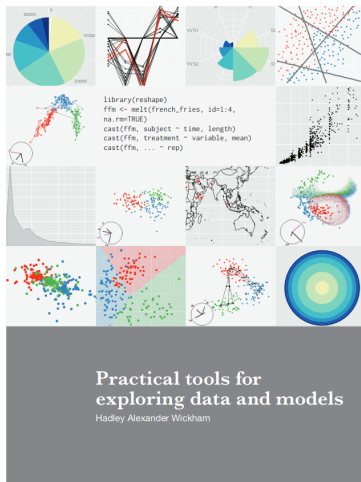
Draws a forest plot in the active graphics window (using grid graphics system).

## Usage

```
forest(x, ...)
```

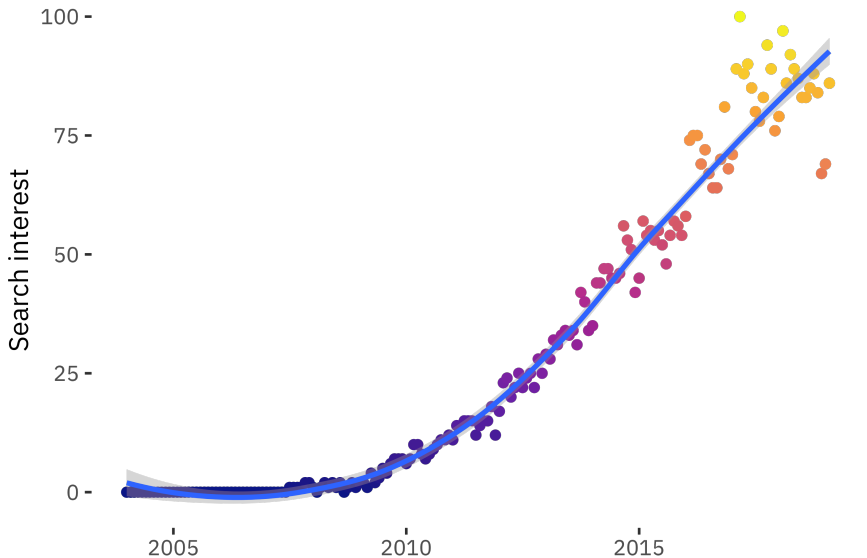
```
## S3 method for class 'meta'  
forest(x, sortvar, studlab=TRUE,  
       layout=gs("layout"),  
       comb.fixed=x$comb.fixed, comb.random=x$comb.random,  
       overall=TRUE,  
       text.fixed=NULL,  
       text.random=NULL,  
       lty.fixed=2, lty.random=3, col.fixed="black", col.random="black",  
       prediction=x$prediction,  
       text.predict=NULL,  
       subgroup=TRUE,  
       print.subgroup.labels=TRUE,  
       bylab=x$bylab, print.byvar=x$print.byvar,  
       byseparator=x$byseparator,  
       text.fixed.w=text.fixed, text.random.w=text.random, bysort=FALSE,  
       pooled.totals=comb.fixed|comb.random, pooled.events=FALSE,  
       pooled.times=FALSE, study.results=TRUE,  
       xlab="", xlab.pos,  
       smlab=NULL, smlab.pos, xlim="symmetric",  
       allstudies=TRUE,  
       weight.study, weight.subgroup,  
       pscale=x$pscale, irscale=x$irscale, irunit=x$irunit,  
       ref=ifelse(backtransf & is.relative.effect(x$sm), 1, 0),  
       leftcols=NULL, rightcols=NULL,  
       leftlabs=NULL, rightlabs=NULL,  
       lth=col.labels, rth=col.labels,
```

# The origins of ggplot2



Hadley Wickham (2008)

# Google Search Trends for ggplot2



<https://trends.google.com/trends/explore?date=all&q=ggplot2>

## Where to learn more

1. The ggplot2 website

`https://ggplot2.tidyverse.org`

2. “Visualize” section in R for Data Science

`https://r4ds.hadley.nz/visualize.html`

3. Data Visualization: A Practical Introduction (2e)

`https://socviz.co/`

4. “ggplot2: elegant graphics for data analysis”

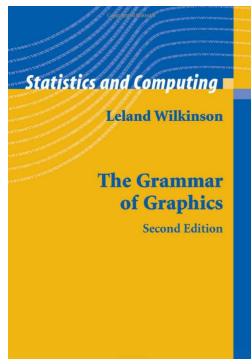
`https://ggplot2-book.org`

ggplot2 is 18 years old.

# What is ggplot2?

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

[ggplot2.tidyverse.org](http://ggplot2.tidyverse.org)





# Essential components of a ggplot2 plot

## 1. Data

# Essential components of a ggplot2 plot

1. Data
2. Aesthetics (or mappings)

# Essential components of a ggplot2 plot

1. Data

2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

# Essential components of a ggplot2 plot

1. Data

2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

3. Coordinates (or geoms)

# Essential components of a ggplot2 plot

## 1. Data

## 2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

## 3. Coordinates (or geoms)

How are the data plotted? Points, lines, bars?

# Essential components of a ggplot2 plot

## 1. Data

## 2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

## 3. Coordinates (or geoms)

How are the data plotted? Points, lines, bars?

## 4. Facets

# Essential components of a ggplot2 plot

## 1. Data

## 2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

## 3. Coordinates (or geoms)

How are the data plotted? Points, lines, bars?

## 4. Facets

Do I want subplots, in a grid?

# Essential components of a ggplot2 plot

## 1. Data

## 2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

## 3. Coordinates (or geoms)

How are the data plotted? Points, lines, bars?

## 4. Facets

Do I want subplots, in a grid?

## 5. Theme

# Essential components of a ggplot2 plot

## 1. Data

## 2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

## 3. Coordinates (or geoms)

How are the data plotted? Points, lines, bars?

## 4. Facets

Do I want subplots, in a grid?

## 5. Theme

Titles, labels, colors, etc.

# Essential components of a ggplot2 plot

## 1. Data

## 2. Aesthetics (or mappings)

How do the data 'map onto' the x-axis and y-axis.

## 3. Coordinates (or geoms)

How are the data plotted? Points, lines, bars?

## 4. Facets

Do I want subplots, in a grid?

## 5. Theme

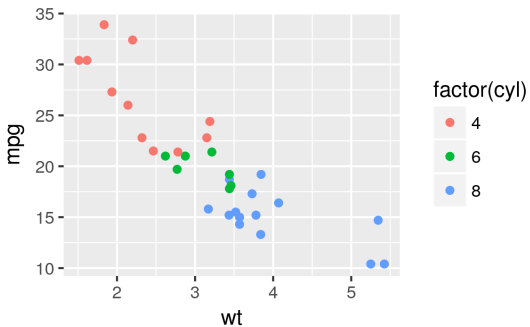
Titles, labels, colors, etc.



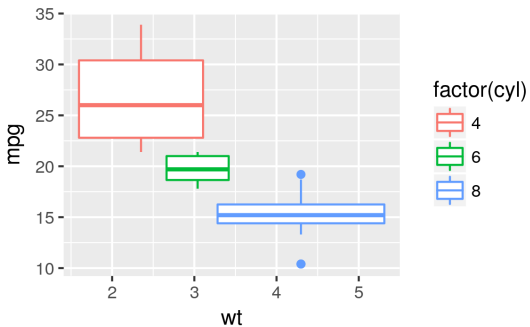
You need to load the `ggplot2` library before using these functions.

It is included in the `tidyverse`, so if you have already loaded that, you do not need to do anything else.

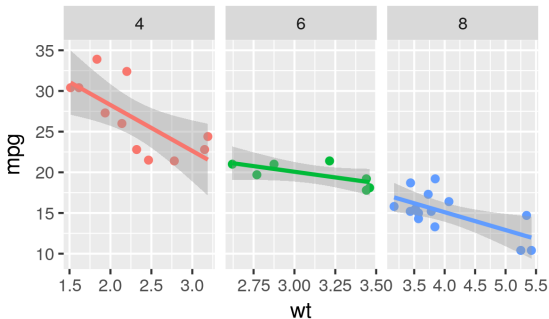
```
ggplot(mtcars,  
  aes(x = wt,  
      y = mpg,  
      color = factor(cyl))) +  
geom_point()
```



```
ggplot(mtcars,  
  aes(x = wt,  
      y = mpg,  
      color = factor(cyl))) +  
geom_boxplot()
```



```
ggplot(mtcars,
  aes(x = wt,
      y = mpg,
      color = factor(cyl))) +
  geom_point() +
  geom_smooth(method = "lm") +
  facet_wrap(~ cyl, scales = "free_x")
```



# Why do we care?

1. Consistency

2. Flexibility

Also nice to have:

- Widely used, well supported, big and friendly community.
- Better defaults, themes, extensibility.

So far we've covered:

- ggplot2 and how it compares to 'base' R.
- The core components of a ggplot2 plot.

So far we've covered:

- ggplot2 and how it compares to 'base' R.
- The core components of a ggplot2 plot.

The remainder of this session...

1. The different components of a ggplot2 plot.
  1. Data
  2. Aesthetics (or mappings)
  3. Coordinates (or geoms)
  4. Facets
  5. Theme
2. Some examples (scatterplot, histogram, bar plots)...
3. Things you should know.
4. PRACTICE.

Preparing your data  
for plotting...

# 1. Data

More important than it sounds...

- Most difficulties with `ggplot2` arise because the source data are not in the correct format.
- Your data should be **tidy**.

country	year	cases	population
Afghanistan	2000	266	1995360
Afghanistan	2000	266	2095360
Brazil	1999	3737	17206362
Brazil	2000	488	17404898
China	1999	21258	127215272
China	2000	276	12802583

**Variables**

country	year	cases	population
Afghanistan	2000	266	1995360
Afghanistan	2000	266	2095360
Brazil	1999	3737	17206362
Brazil	2000	488	17404898
China	1999	21258	127215272
China	2000	276	12802583

**Observations**

country	year	cases	population
Afghanistan	2000	266	1995360
Afghanistan	2000	266	2095360
Brazil	1999	3737	17206362
Brazil	2000	488	17404898
China	1999	21258	127215272
China	2000	276	12802583

**Values**

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Once your data are `tidy`, they are added to the plot as the first argument.

```
ggplot(data, ...)
```

Alternatively, using the pipe...

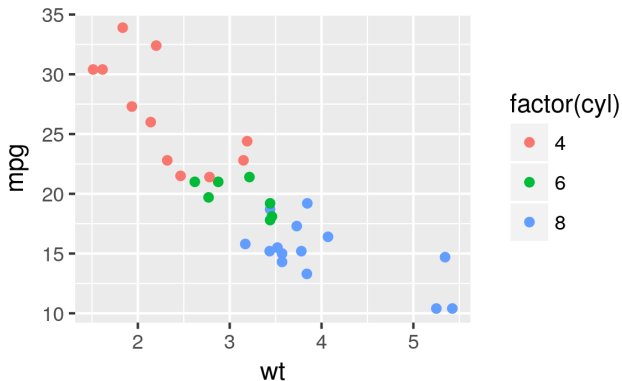
```
data |>  
  ggplot(...)
```

Mapping data onto  
components of your  
plot...

# Aesthetics (or mappings)

These describe how your data map onto different components of the plot.

- Which variable goes on the **x-axis**?
- Should the points be **grouped**?
- Which variable should determine the **colour** of the points?



- The required aesthetics depend on the type of plot.
- Some are essential; others are optional.
- They are specified using `aes`.

```
data |>  
  ggplot() +  
  aes(x = weight,  
      y = height)
```

NOTE: We're 'adding' components; no pipes here.

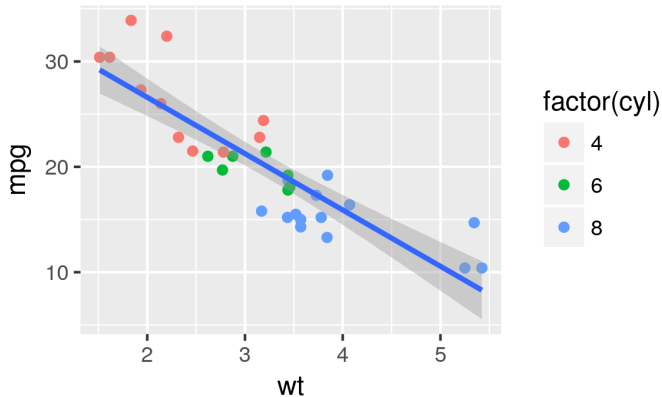
# Geoms

- Geoms specify how the data should be plotted.  
Do we want a point, a line, a histogram?
- There are some basic geoms that you'll use regularly.
  - `geom_point()`
  - `geom_line()`
  - `geom_histogram()`

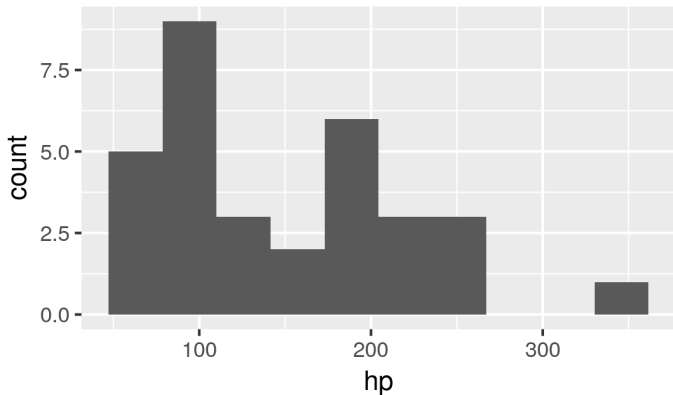
```
mtcars |>  
ggplot() +  
  aes(x = wt, y = mpg) +  
  geom_point()
```



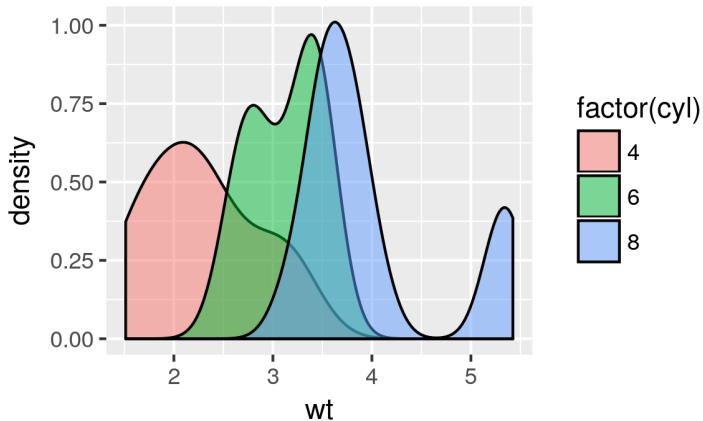
```
mtcars |>
  ggplot() +
    aes(x = wt, y = mpg) +
    geom_point(aes(color = factor(cyl))) +
    geom_smooth(method = "lm")
```



```
mtcars |>
  ggplot() +
    aes(x = hp) +
    geom_histogram(bins = 10)
```



```
mtcars |>
  ggplot() +
    aes(x = wt,
        fill = factor(cyl)) +
    geom_density(alpha = 0.5)
```





geom_abline	geom_histogram	geom_segment
geom_bar	geom_hline	geom_smooth
geom_bin2d	geom_jitter	geom_spoke
geom_boxplot	geom_label	geom_step
geom_col	geom_line	geom_text
geom_contour	geom_linerange	geom_tile
geom_count	geom_map	geom_violin
geom_crossbar	geom_path	geom_vline
geom_curve	geom_point	geom_freqpoly
geom_density	geom_pointrange	geom_hex
geom_density_2d	geom_polygon	geom_histogram
geom_dotplot	geom_qq	geom_rect
geom_errorbar	geom_quantile	geom_ribbon
geom_errorbarh	geom_raster	geom_rug

[exts.ggplot2.tidyverse.org](https://exts.ggplot2.tidyverse.org)

## Remember

- You can store plots as **objects**.

```
p <- ggplot(...)
```

- You will find it easier to write ggplot2 code in the **script editor**, not the console.

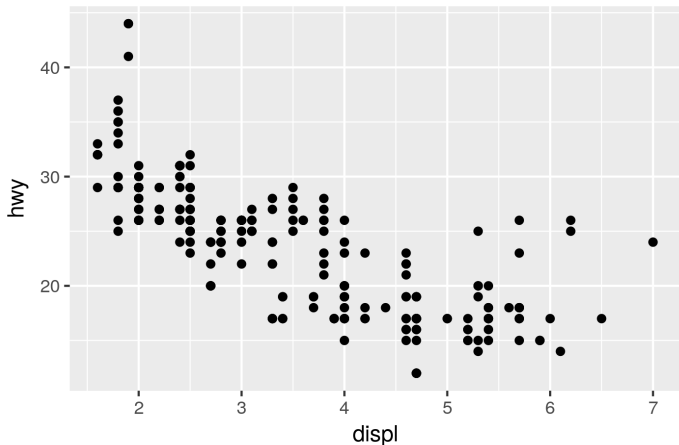
Dividing the plot into  
subplots...

# Facets

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

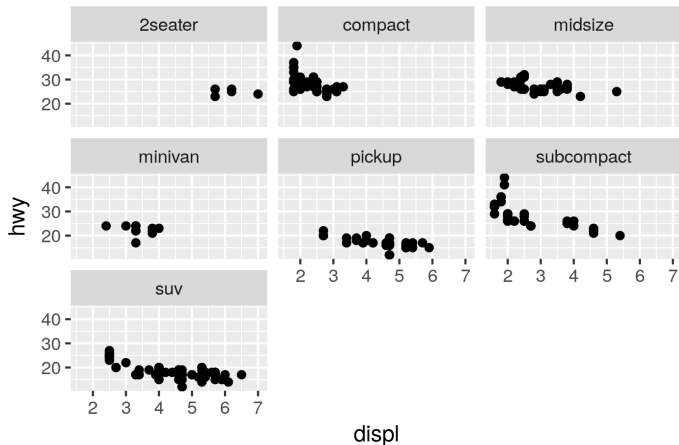
# Facets

Facets [divide a plot into subplots](#) based on the values of one or more discrete variables.

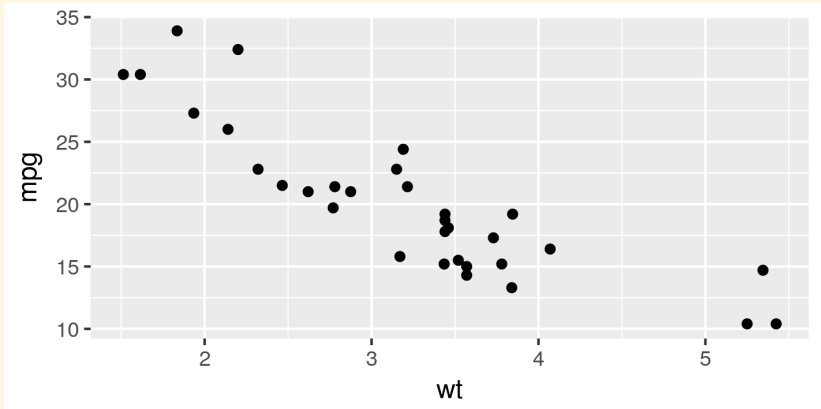


# Facets

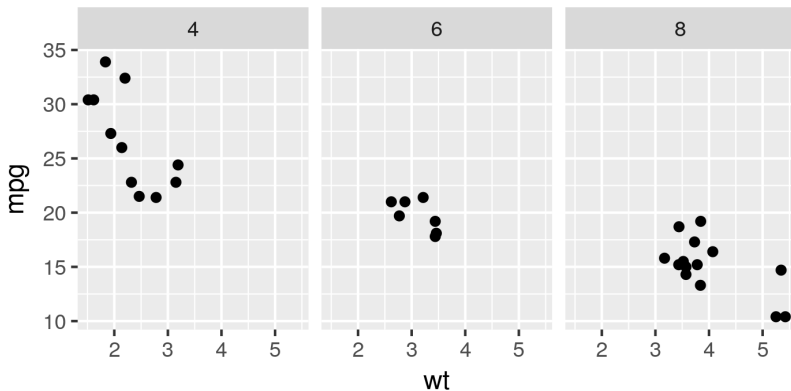
Facets divide a plot into subplots based on the values of one or more discrete variables.



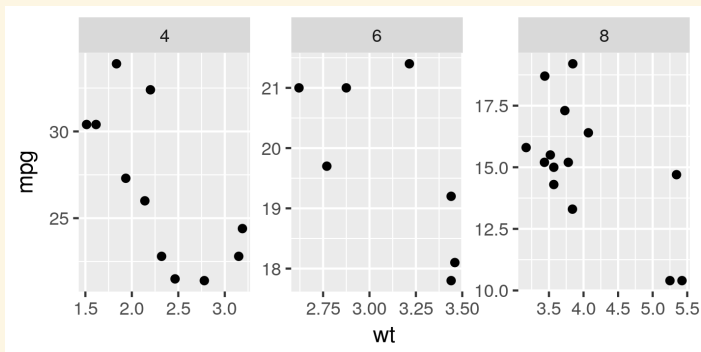
```
ggplot(mtcars),  
  aes(x = wt, y = mpg) +  
  geom_point()
```



```
mtcars |>  
  ggplot() +  
    aes(x = wt, y = mpg) +  
    geom_point() +  
    facet_wrap(~ cyl)
```



```
mtcars |>  
  ggplot() +  
    aes(x = wt, y = mpg) +  
    geom_point() +  
    facet_wrap(~ cyl, scales = "free")
```



Things you should  
know...

## Save plots with ggsave

```
p <- ggplot(df) +  
  aes(x = weight,  
       y = height) +  
  geom_point()  
  
# Save as a PNG image  
ggsave(  
  filename = "scatterplot.png",  
  plot = p,  
  device = "png",  
  dpi = 300,  
  width = 7,  
  height = 5  
)
```

## Label plots with `labs()`

```
ggplot(df) +  
  aes(x = weight,  
      y = height) +  
  geom_point() +  
  labs(title = "My plot title",  
       x = "A label for the x-axis",  
       y = "A label for the y-axis")
```

See `?labs` for more.

## Change axis ranges with `coord_cartesian`

```
ggplot(df) +  
  aes(x = weight,  
      y = height) +  
  geom_point() +  
  coord_cartesian(xlim = c(0, 100),  
                 ylim = c(10, 20))
```

Read this section of the `ggplot2` book to learn more:

<https://ggplot2-book.org/scales-guides.html>

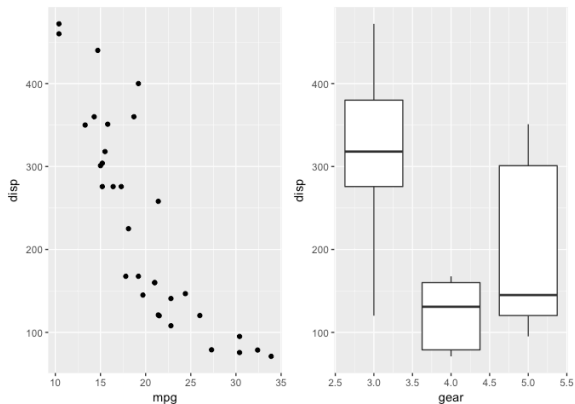
# Write functions that create plots

Suppose we have 10 datasets; we want a plot for each one.

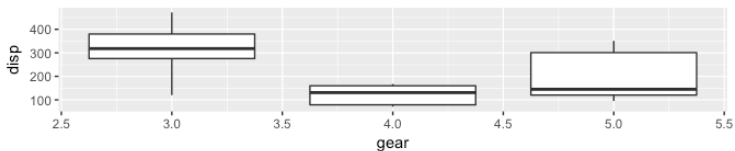
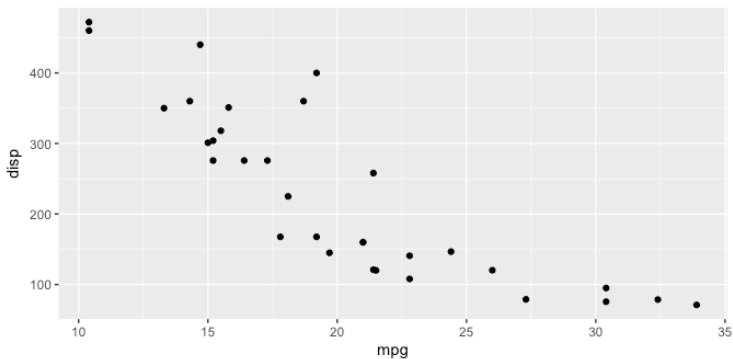
```
draw_the_plot <- function(data) {  
  data |>  
    ggplot() +  
    aes(x = weight,  
        y = height) +  
    geom_point()  
}  
  
for (d in many_datasets) {  
  p <- draw_the_plot(d)  
  ggsave(p, filename = "myplot.png")  
}
```

# Combine multiple plots with patchwork

```
library(patchwork)
p1 <- ggplot(mtcars) + ... + geom_point()
p2 <- ggplot(mtcars) + ... + geom_boxplot()
p1 + p2
```



```
p1 + p2 +  
  plot_layout(ncol = 1, heights = c(3, 1))
```



<https://patchwork.data-imaginist.com>

# Wrapping up

1. Construct your plot by combining different components.
  - a. Data
  - b. Aesthetics
  - c. Coordinates
  - d. Facets
  - e. Theme
2. Make sure your data are tidy first.
3. Save with `ggsave`.

ggplot2:

Build a data  
MASTERPIECE

