

# Towards publication-quality graphics

Melbourne Statistical Consulting Platform

University of Melbourne

April 2024

## Exploratory graphics

Only looked at by you, maybe immediate colleagues.

To help you understand your data better.

Make lots of them. Make them quickly.

Throw most of them away.

Don't worry too much about labels or formatting.

## Publication-quality graphics

Included in your paper, thesis, report, web site, etc.

To show your results to world.

Be prepared to spend hours on each graphic. Each one should have a reason to exist and a story to tell.

Labels must be clear (ideal: impossible to misunderstand).  
Formatting may have to conform to a style guide.

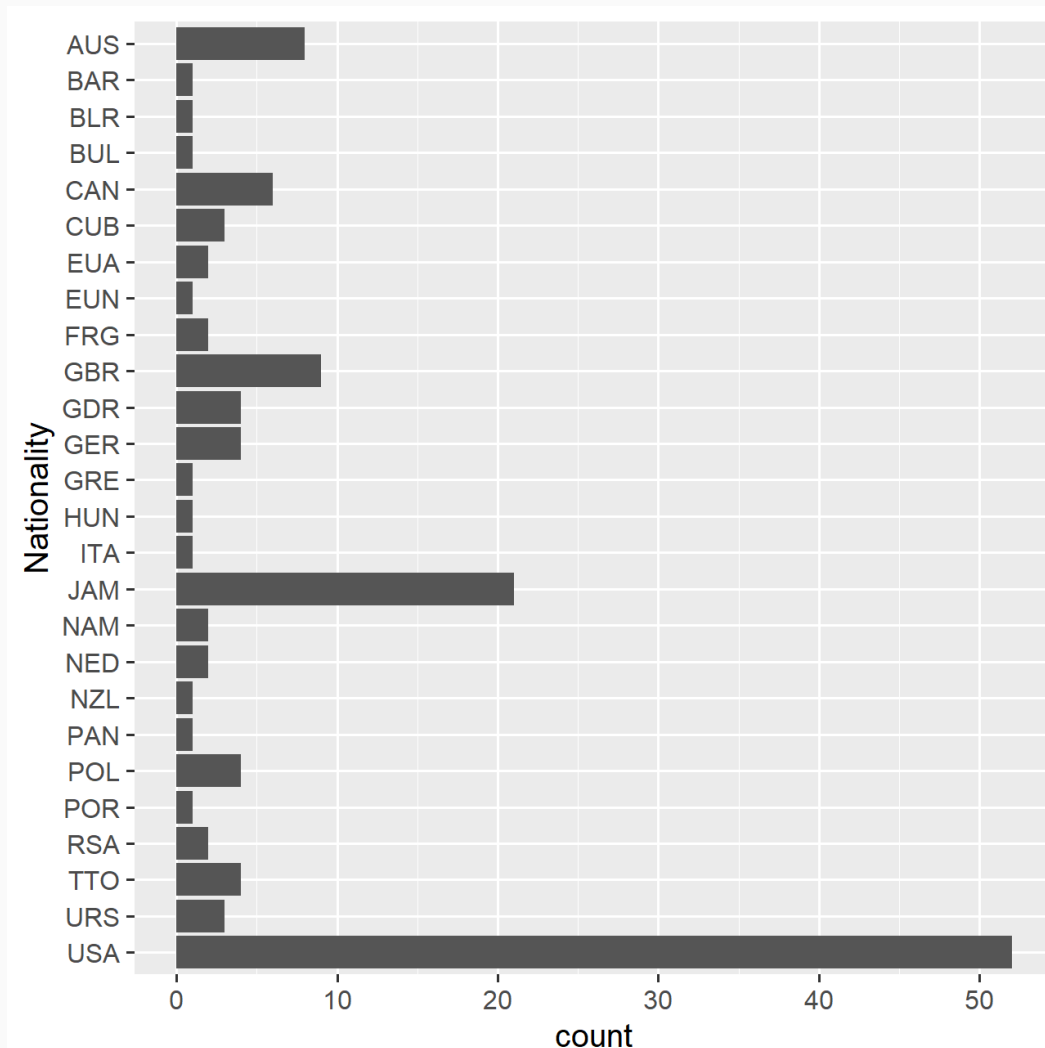
Image by Desirée De Leon ([Twitter](#)).



# Better bar charts: sorting factor levels

```
ggplot(olympic_100m_data,  
       aes(y = Nationality)) +  
  geom_bar() +  
  scale_y_discrete(limits = rev)
```

Our bar chart of Olympic sprint medalists is back. Let's improve it.

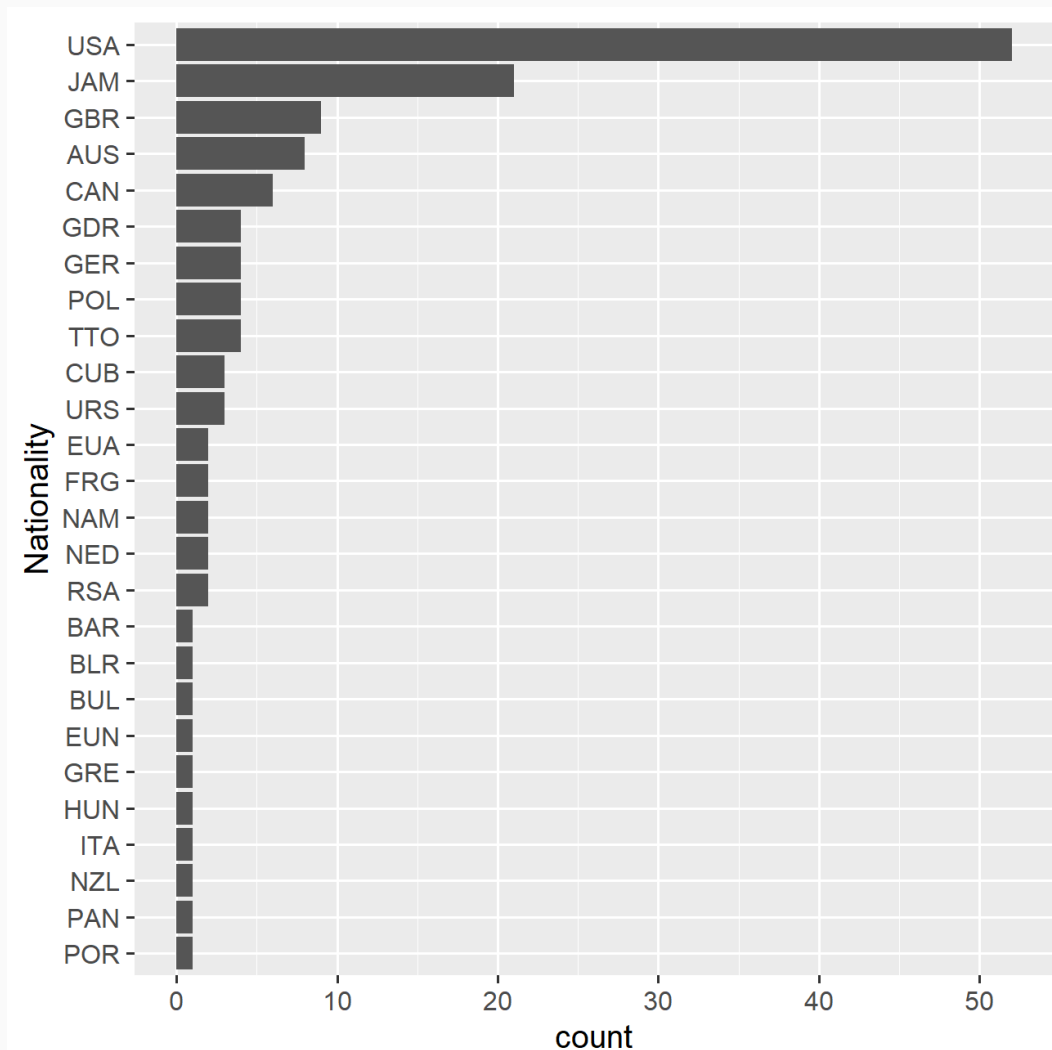


# Better bar charts: sorting factor levels

```
olympic_100m_clean <- olympic_100m_data %>%  
  mutate(Nationality = fct_infreq(Nationality))  
ggplot(olympic_100m_clean, aes(y = Nationality)) +  
  geom_bar() +  
  scale_y_discrete(limits = rev)
```

Some new things here!

- We're creating a new data frame with a 'fixed up' version of the data, with the nationalities in the order we want. This is a type of data cleaning, which you'll see more of throughout the course.
- What do you think `mutate()` and `fct_infreq()` do?



# Diversion: modifying data with mutate

The `mutate()` function does operations on columns within a data frame. You can make new columns or replace existing ones.

You can do more than one calculation in a single `mutate()`. Operations are done in the order written.

Do arithmetic or use any R function that operates on vectors (columns).

```
olympic_100m_extra_vars <- olympic_100m_data %>%  
  mutate(Time_minutes = Time / 60,  
         log_Time_minutes = log10(Time_minutes))  
glimpse(olympic_100m_extra_vars)
```

Rows: 138

Columns: 12

\$ Gender	<chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M"...
\$ Event	<chr> "100M Men", "100M Men", "100M Men", "100M M..."
\$ Location	<chr> "Rio", "Rio", "Rio", "Beijing", "Beijing", "Beij..."
\$ Year	<dbl> 2016, 2016, 2016, 2008, 2008, 2008, 2000, 2000, 2...
\$ Medal	<chr> "G", "S", "B", "G", "S", "B", "G", "S", "B", "G"...
\$ Name	<chr> "Usain BOLT", "Justin GATLIN", "Andre DE GR..."
\$ Nationality	<chr> "JAM", "USA", "CAN", "JAM", "TTO", "USA", "..."
\$ Result	<dbl> 9.81, 9.89, 9.91, 9.69, 9.89, 9.91, 9.87, 9...
\$ Wind	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
\$ Time	<dbl> 9.81, 9.89, 9.91, 9.69, 9.89, 9.91, 9.87, 9...
\$ Time_minutes	<dbl> 0.1635000, 0.1648333, 0.1651667, 0.1615000, ...
\$ log_Time_minutes	<dbl> -0.7864822, -0.7829550, -0.7820776, -0.7918...

A single value will be repeated down the column.

```
olympic_100m_extra_vars <- olympic_100m_data %>%  
  mutate(Sport = "Running")  
glimpse(olympic_100m_extra_vars)
```

Rows: 138

Columns: 11

\$ Gender	<chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M"...
\$ Event	<chr> "100M Men", "100M Men", "100M Men", "100M Men", ...
\$ Location	<chr> "Rio", "Rio", "Rio", "Beijing", "Beijing", "Beij..."
\$ Year	<dbl> 2016, 2016, 2016, 2008, 2008, 2008, 2000, 2000, ...
\$ Medal	<chr> "G", "S", "B", "G", "S", "B", "G", "S", "B", "G"...
\$ Name	<chr> "Usain BOLT", "Justin GATLIN", "Andre DE GRASSE"...
\$ Nationality	<chr> "JAM", "USA", "CAN", "JAM", "TTO", "USA", "USA", ...
\$ Result	<dbl> 9.81, 9.89, 9.91, 9.69, 9.89, 9.91, 9.87, 9.99, ...
\$ Wind	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
\$ Time	<dbl> 9.81, 9.89, 9.91, 9.69, 9.89, 9.91, 9.87, 9.99, ...
\$ Sport	<chr> "Running", "Running", "Running", "Running", "Run..."

# Diversion: functions for working with factors

These are all in the `forcats` package, which is part of the Tidyverse.

## Reordering categories

- `fct_inorder()`: order categories by first appearance
- `fct_infreq()`: order categories by frequency (how common they are in the data)
- `fct_inseq()`: order categories by numerical sequence
- `fct_reorder()`: order categories by a different variable (e.g. sort countries by GDP)
- `fct_relevel()`: order categories manually
- `fct_rev()`: reverse the order of categories

R jargon alert: "factor level" and "level of a factor" are other names for a category. You'll see these terms used a lot if you read the documentation for these functions.

## Renaming and combining categories

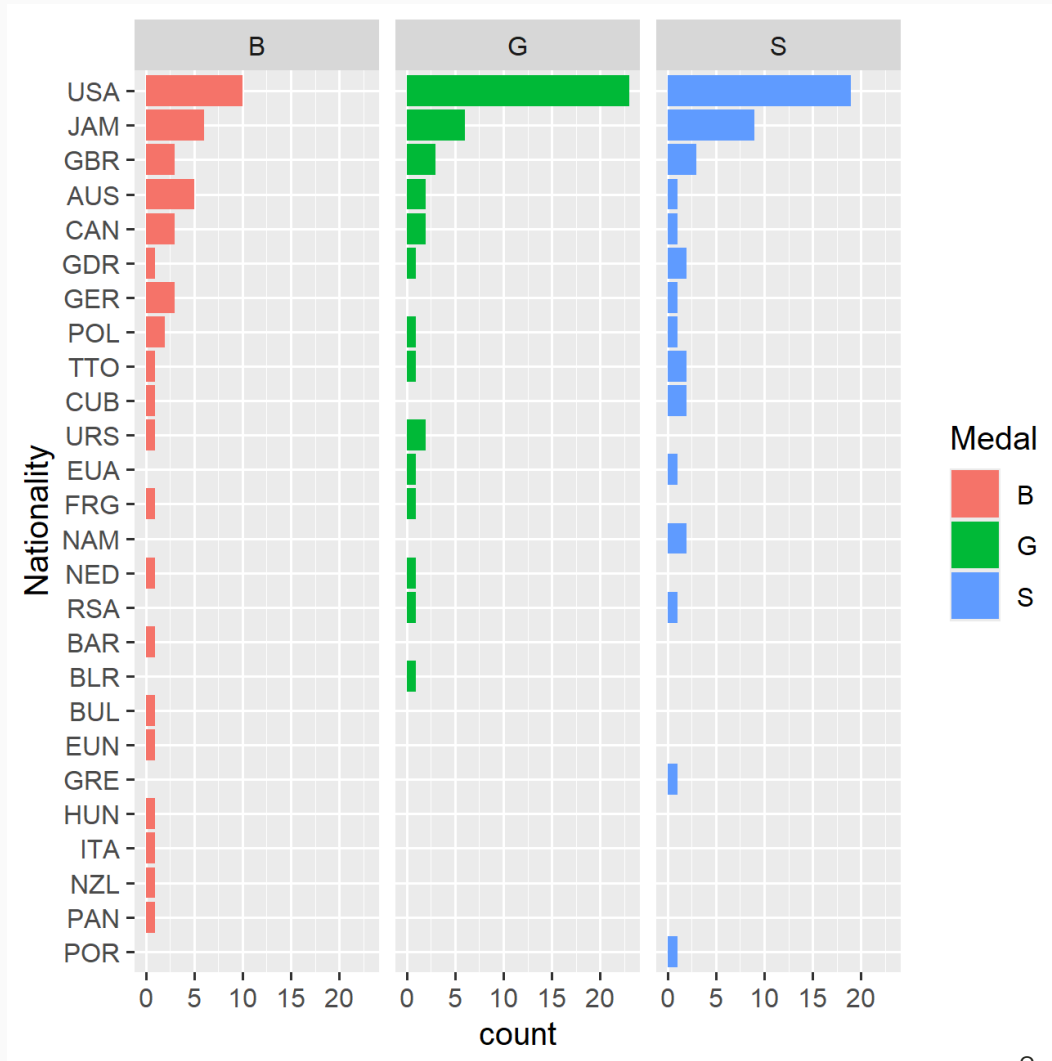
- `fct_recode()`: manually change category names
- `fct_collapse()`: combine groups of related categories
- `fct_lump_n()`, `fct_lump_min()`, `fct_lump_prop()`, `fct_lump_lowfreq()`: combine uncommon categories into "Other"
- `fct_other()`: combine specific categories into "Other"

# Better bar charts: facets and colours

```
olympic_100m_clean <- olympic_100m_data %>%  
  mutate(Nationality = fct_infreq(Nationality))  
ggplot(olympic_100m_clean,  
  aes(y = Nationality, fill = Medal)) +  
  geom_bar() +  
  facet_wrap(vars(Medal), ncol = 3) +  
  scale_y_discrete(limits = rev)
```

Change the colour of bars using the `fill` aesthetic (not `colour`, which is for lines and points, not solid colour).

Have the medals been presented in a sensible order?

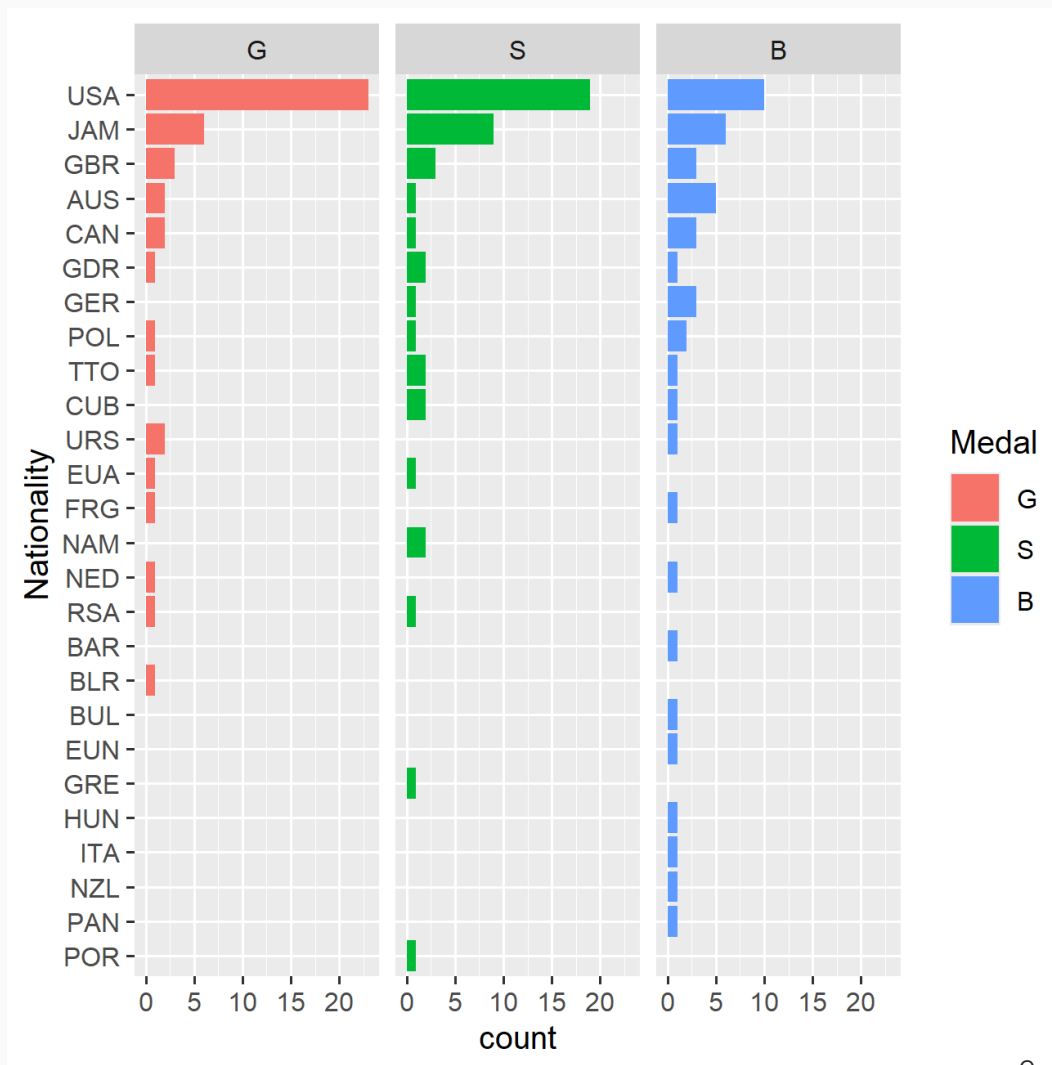




# Better bar charts: facets and colours

```
olympic_100m_clean <- olympic_100m_data %>%  
  mutate(Nationality = fct_infreq(Nationality),  
         Medal = fct_relevel(Medal, "G", "S", "B"))  
ggplot(olympic_100m_clean,  
       aes(y = Nationality, fill = Medal)) +  
  geom_bar() +  
  facet_wrap(vars(Medal), ncol = 3) +  
  scale_y_discrete(limits = rev)
```

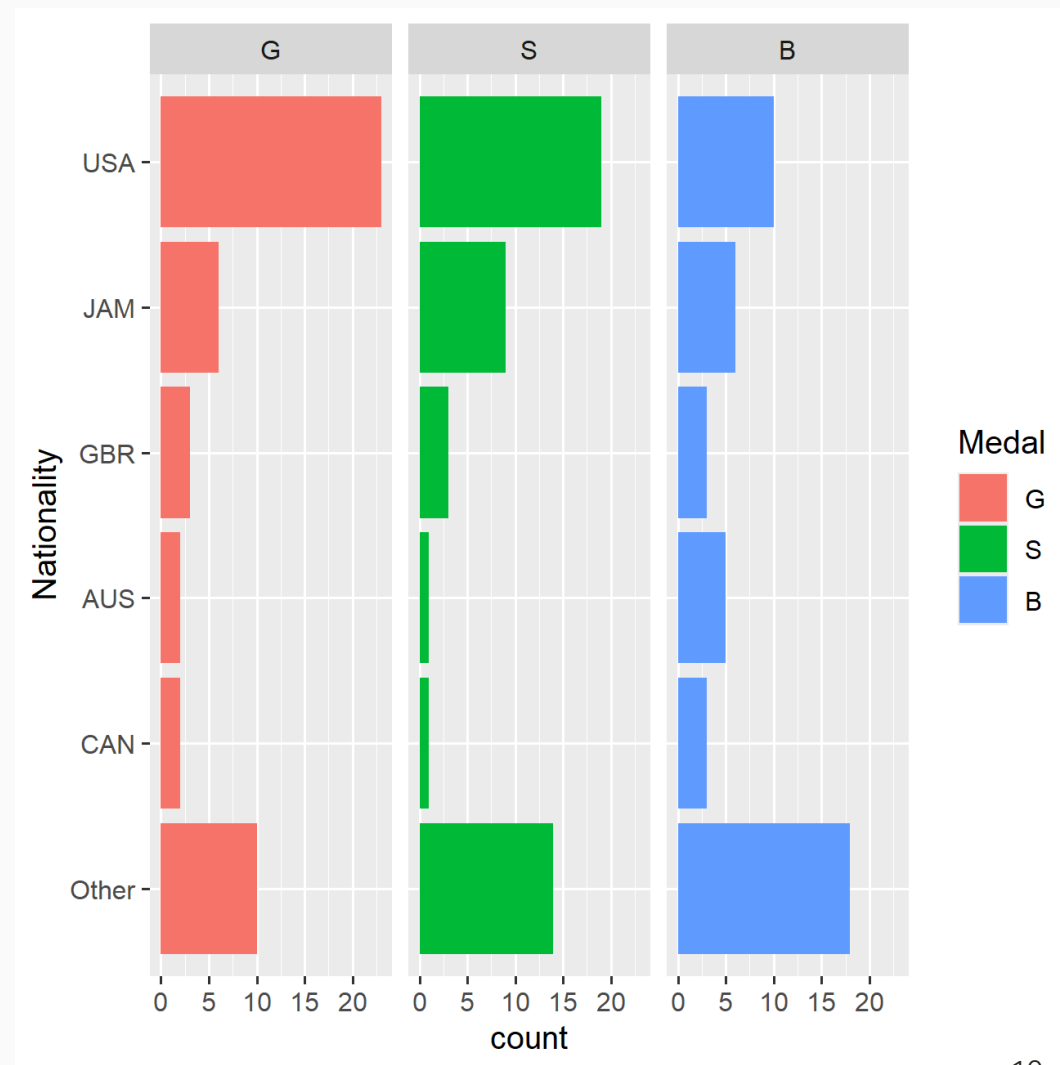
Maybe we don't need to see **every** country in this plot.



# Better bar charts: combining rare categories

```
olympic_100m_clean <- olympic_100m_data %>%  
  mutate(  
    Nationality = fct_lump_n(fct_infreq(Nationality), 5),  
    Medal = fct_relevel(Medal, "G", "S", "B")  
  )  
ggplot(olympic_100m_clean,  
  aes(y = Nationality, fill = Medal)) +  
  geom_bar() +  
  facet_wrap(vars(Medal), ncol = 3) +  
  scale_y_discrete(limits = rev)
```

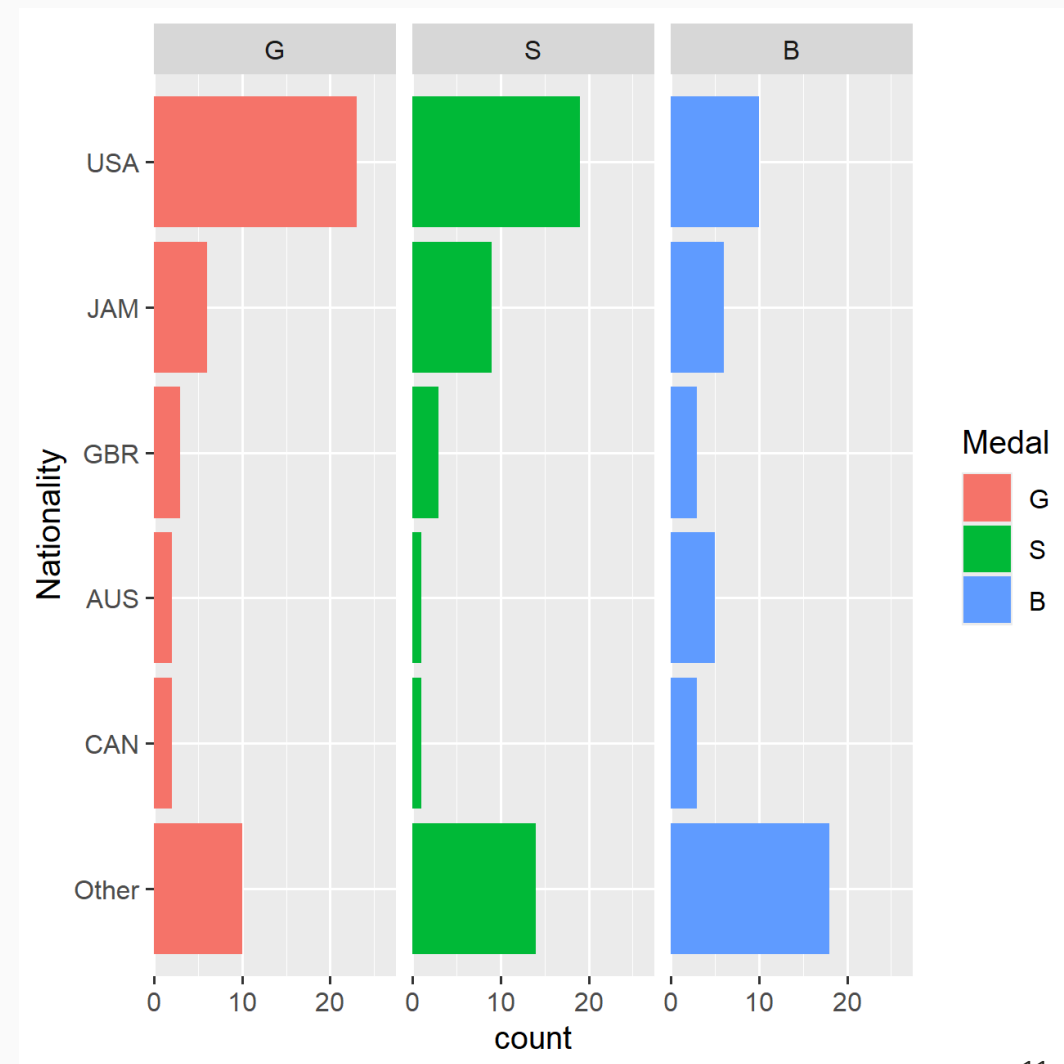
Use `fct_lump_n()` to reduce to 5 countries and "Other".



# Better bar charts: moving the bars to the edge

```
olympic_100m_clean <- olympic_100m_data %>%  
  mutate(  
    Nationality = fct_lump_n(fct_infreq(Nationality), 5),  
    Medal = fct_relevel(Medal, "G", "S", "B")  
  )  
ggplot(olympic_100m_clean,  
  aes(y = Nationality, fill = Medal)) +  
  geom_bar() +  
  facet_wrap(vars(Medal), ncol = 3) +  
  scale_x_continuous(expand = expansion(mult = c(0, 0.1)),  
    limits = c(0, 25)) +  
  scale_y_discrete(limits = rev)
```

Normally ggplot extends the axis for a continuous variable 10% on either side of the range of the data. You can control this using `scale_x_continuous()` or `scale_y_continuous()`.

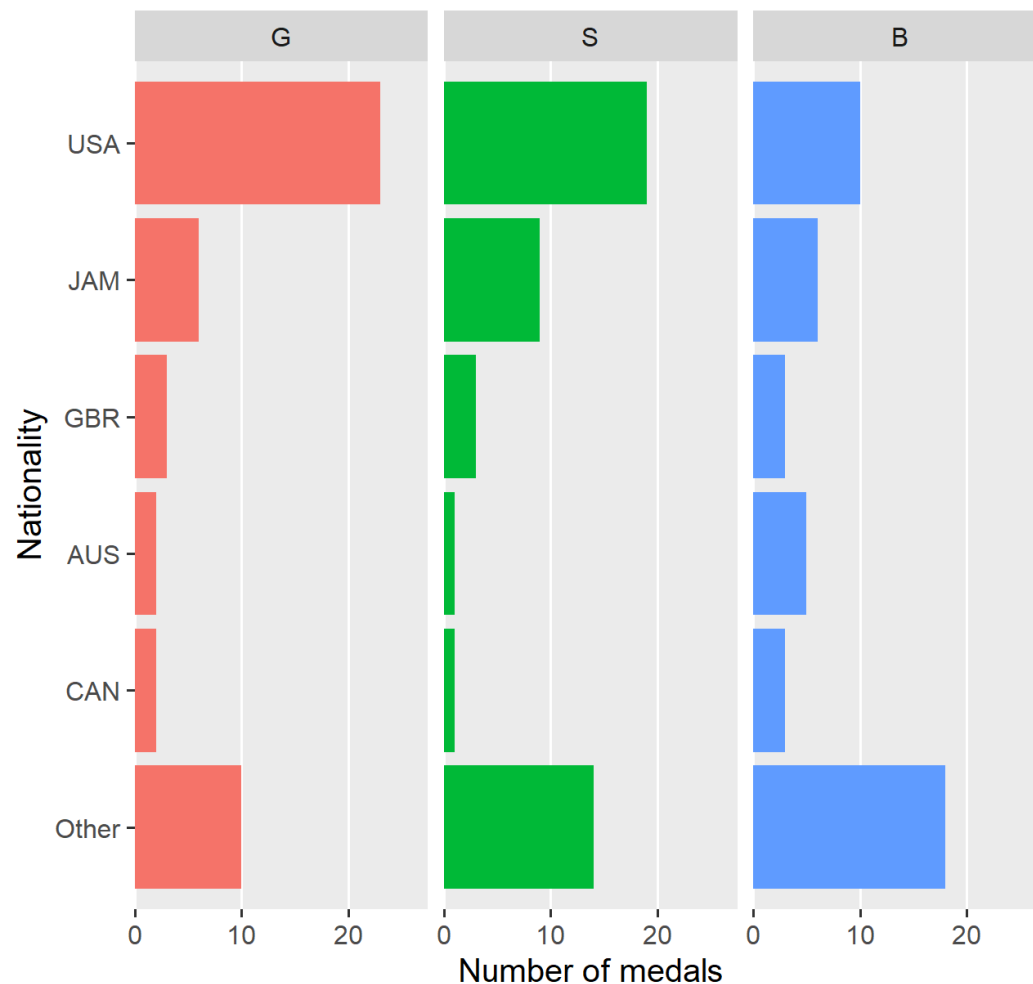


# Finishing touches: less grid lines

```
olympic_100m_clean <- olympic_100m_data %>%
  mutate(
    Nationality = fct_lump_n(fct_infreq(Nationality), 5),
    Medal = fct_relevel(Medal, "G", "S", "B")
  )
ggplot(olympic_100m_clean,
  aes(y = Nationality, fill = Medal)) +
  geom_bar() +
  facet_wrap(vars(Medal), ncol = 3) +
  scale_x_continuous(expand = expansion(mult = c(0, 0.1)),
    limits = c(0, 25)) +
  scale_y_discrete(limits = rev) +
  theme(panel.grid.major.y = element_blank(),
    panel.grid.minor = element_blank(),
    plot.title.position = "plot",
    legend.position = "off") +
  labs(x = "Number of medals",
    title = "Olympic 100m sprint medals, 1896 to 2016")
```

Categorical variables usually don't require gridlines. I also often remove ggplot's minor grid lines for a less cluttered look. The `theme()` function allows you to change almost any aspect of a plot's appearance. I've also added a title and better axis labels, and removed the redundant legend.

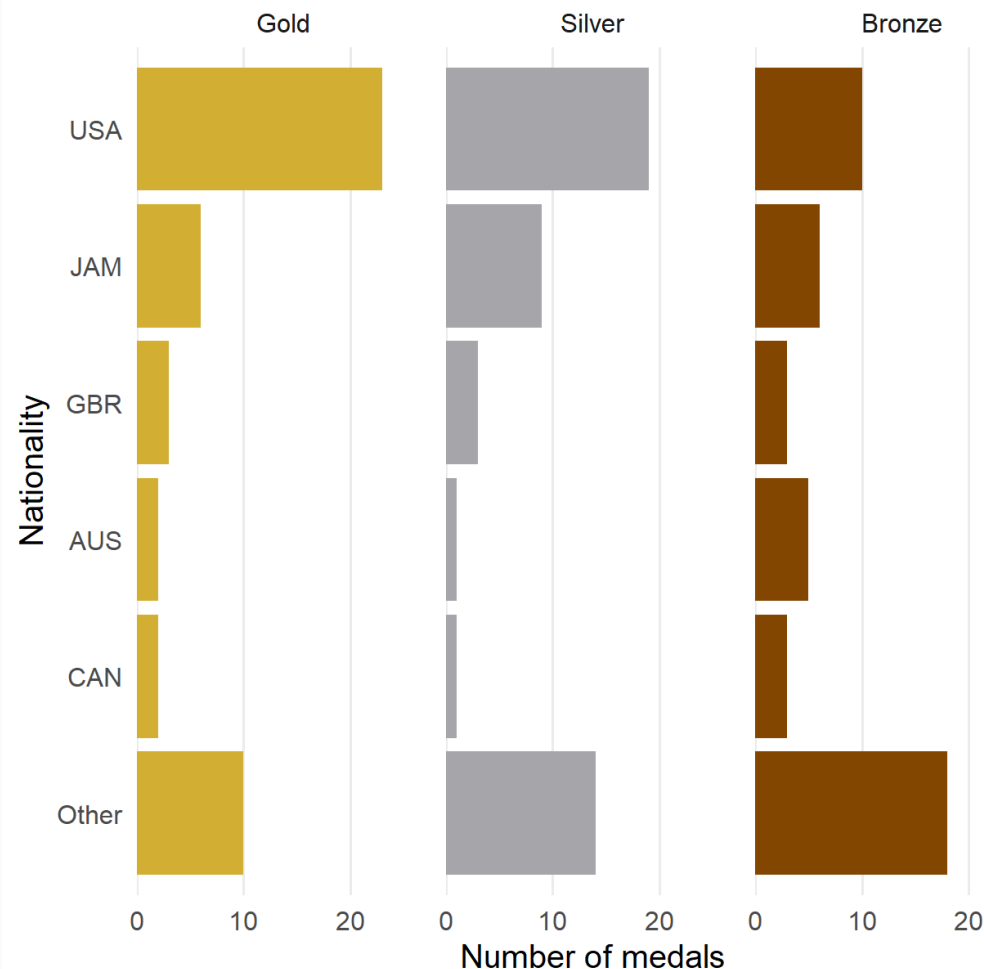
Olympic 100m sprint medals, 1896 to 2016



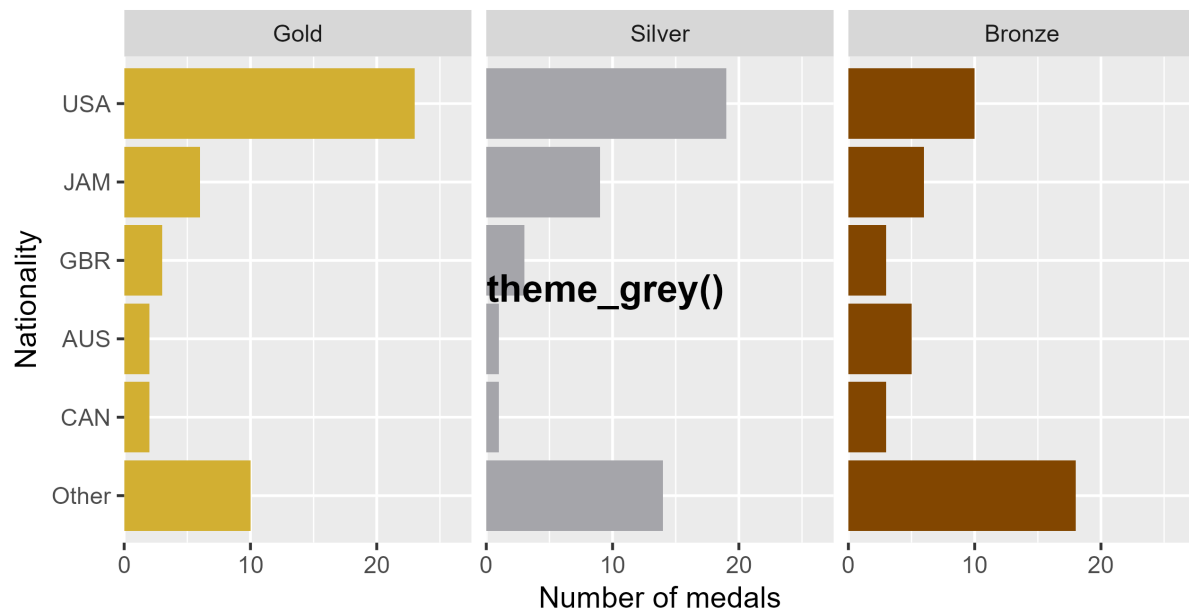
# Finishing touches: colours and labels

```
olympic_100m_clean <- olympic_100m_data %>%
  mutate(
    Nationality = fct_lump_n(fct_infreq(Nationality), 5),
    Medal = fct_relevel(Medal, "G", "S", "B"),
    Medal = fct_recode(Medal, "Gold" = "G", "Silver" = "S",
                      "Bronze" = "B")
  )
ggplot(olympic_100m_clean,
  aes(y = Nationality, fill = Medal)) +
  geom_bar() +
  facet_wrap(vars(Medal), ncol = 3) +
  scale_x_continuous(expand = expansion(mult = c(0, 0.1)),
    limits = c(0, 25)) +
  scale_y_discrete(limits = rev) +
  scale_fill_manual(
    values = c("#D6AF36", "#A7A7AD", "#824A02")) +
  theme_minimal() +
  theme(panel.grid.major.y = element_blank(),
    panel.grid.minor = element_blank(),
    plot.title.position = "plot",
    legend.position = "off") +
  labs(x = "Number of medals",
    title = "Olympic 100m sprint medals, 1896 to 2016")
```

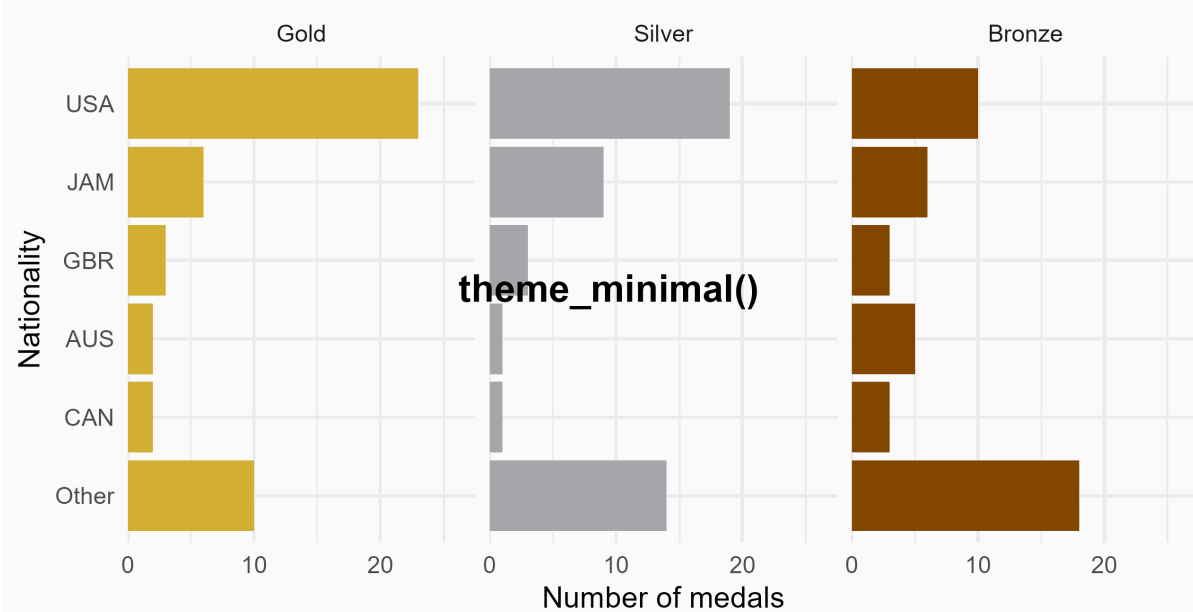
Olympic 100m sprint medals, 1896 to 2016



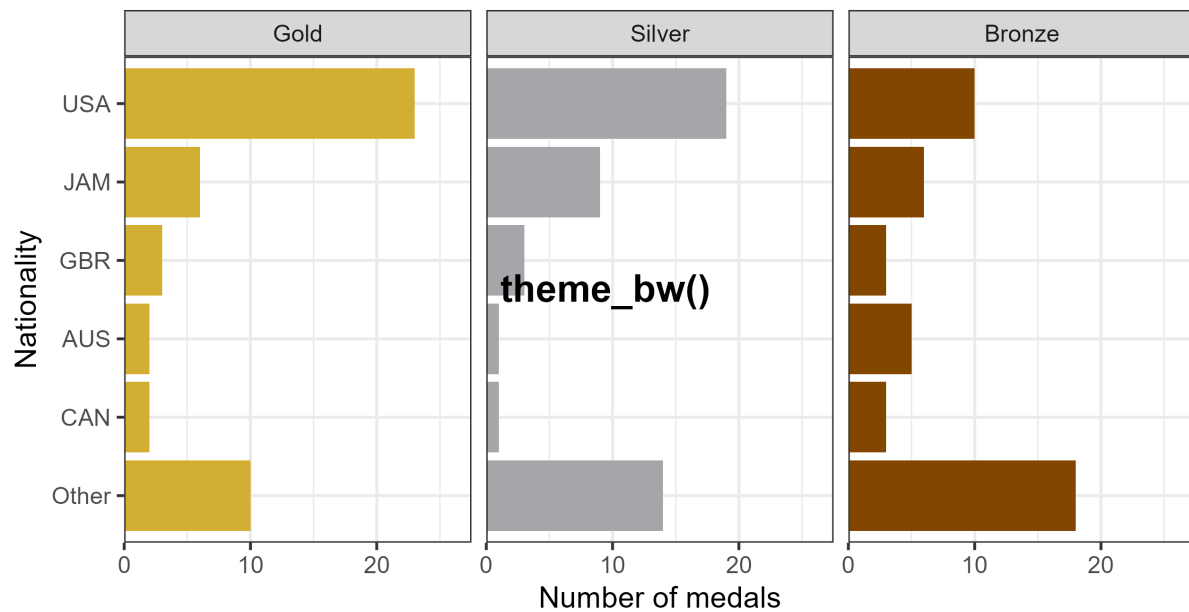
Olympic 100m sprint medals, 1896 to 2016



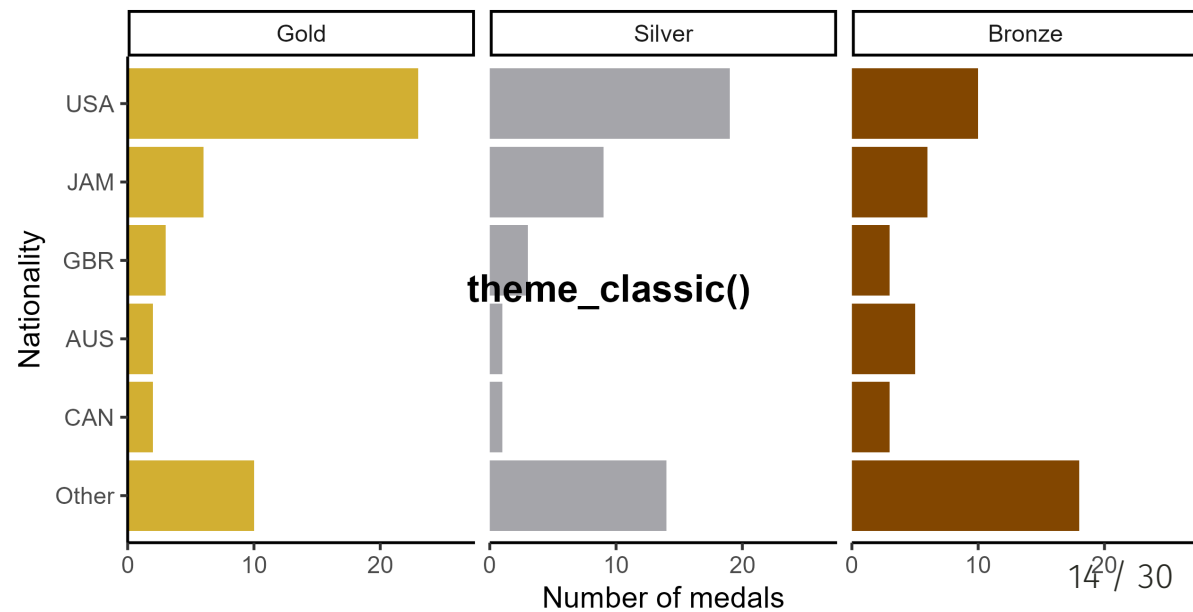
Olympic 100m sprint medals, 1896 to 2016



Olympic 100m sprint medals, 1896 to 2016



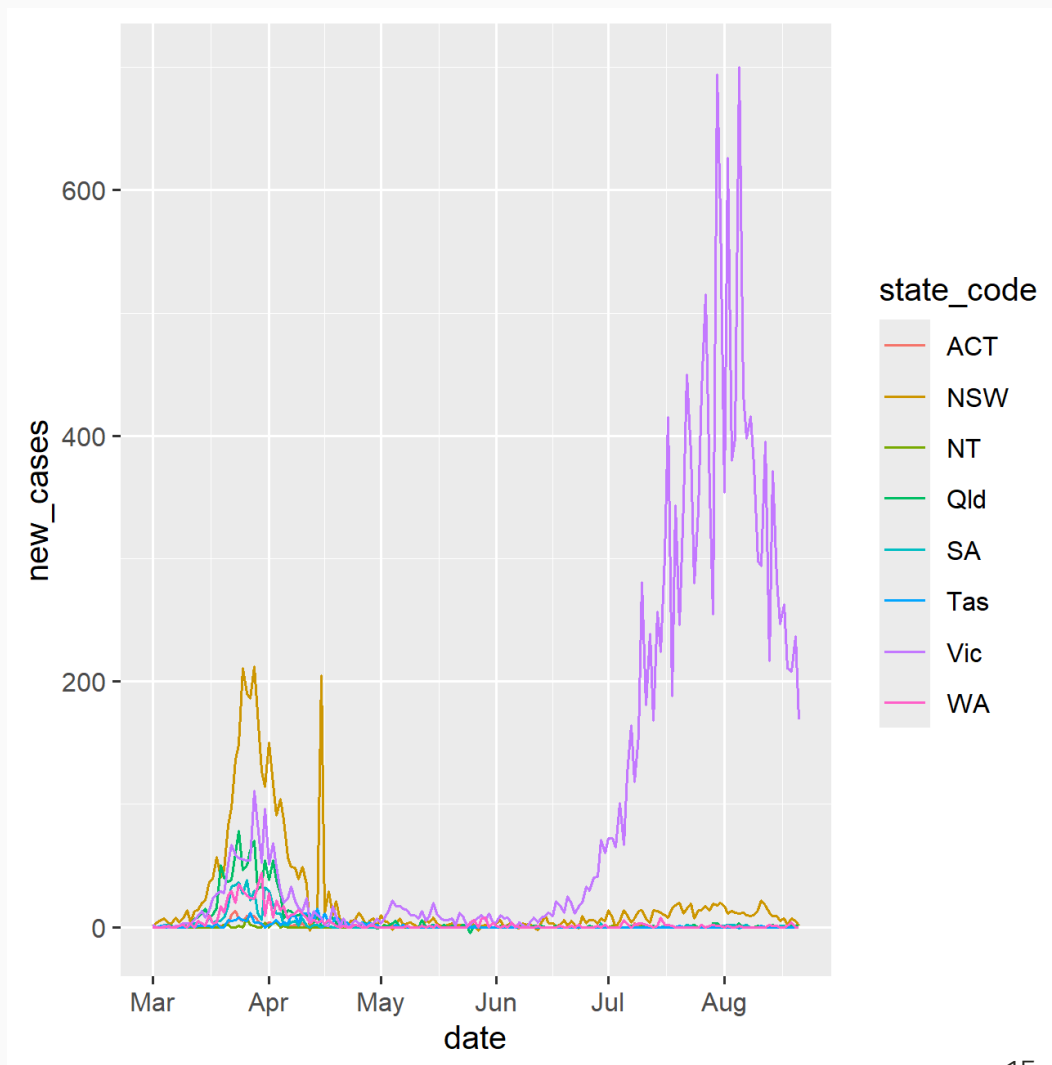
Olympic 100m sprint medals, 1896 to 2016



# Better line graphs: sorting the legend

```
covid_by_state <- read_csv("covid_by_state_20200821.csv")
ggplot(covid_by_state,
      aes(x = date, y = new_cases, colour = state_code)) +
  geom_line()
```

Here's one you've probably seen before. The states are listed in alphabetical order, so the legend doesn't line up with the data.



# Better line graphs: sorting the legend

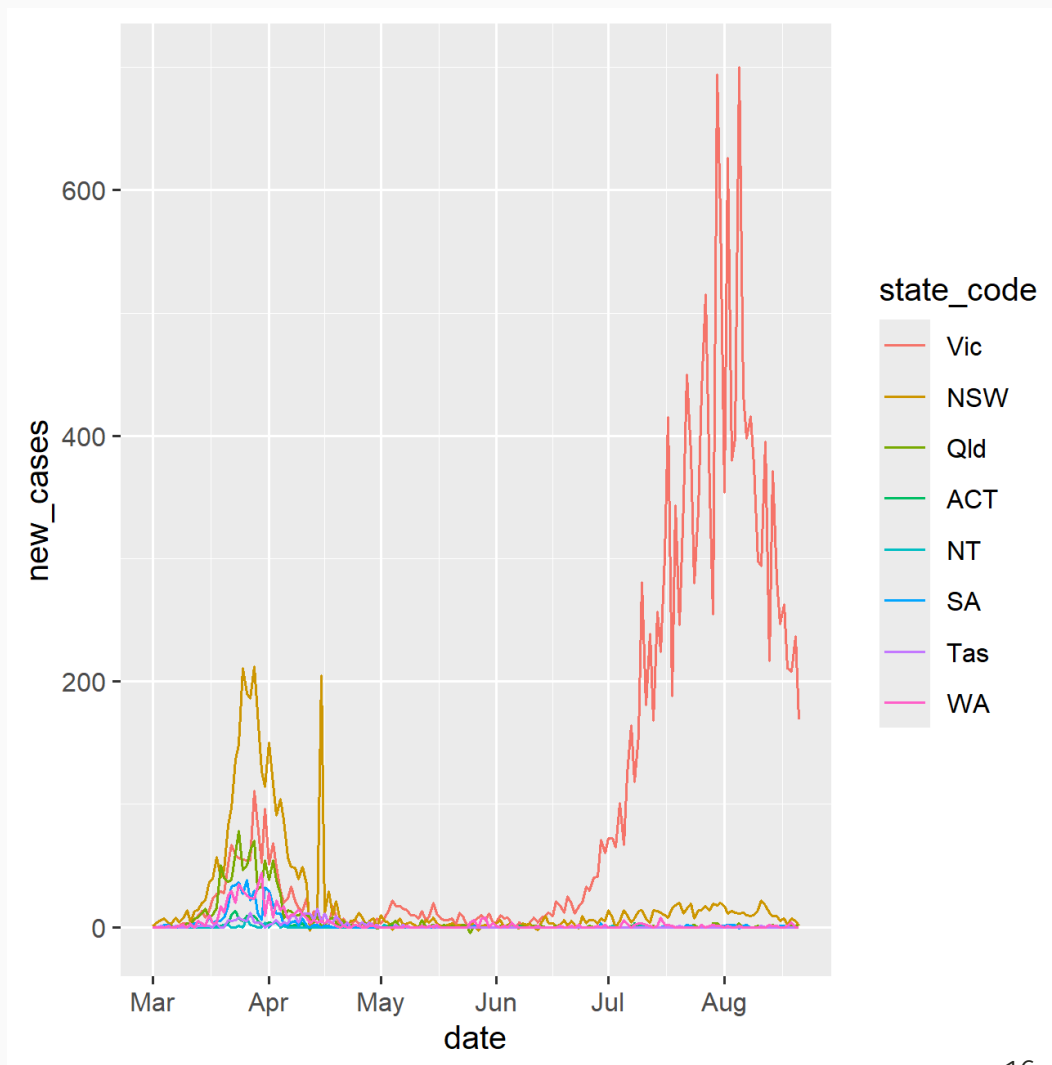
```
covid_by_state %>%  
  mutate(state_code = fct_reorder2(state_code, date, new_cases)) %>%  
  ggplot(aes(x = date, y = new_cases, colour = state_code)) +  
  geom_line()
```

You can pipe data through mutate and into ggplot! Beware: lines end with `%>%` before ggplot and `+` after ggplot.

`fct_reorder2()` sorts factor levels by the number of cases on the last date.

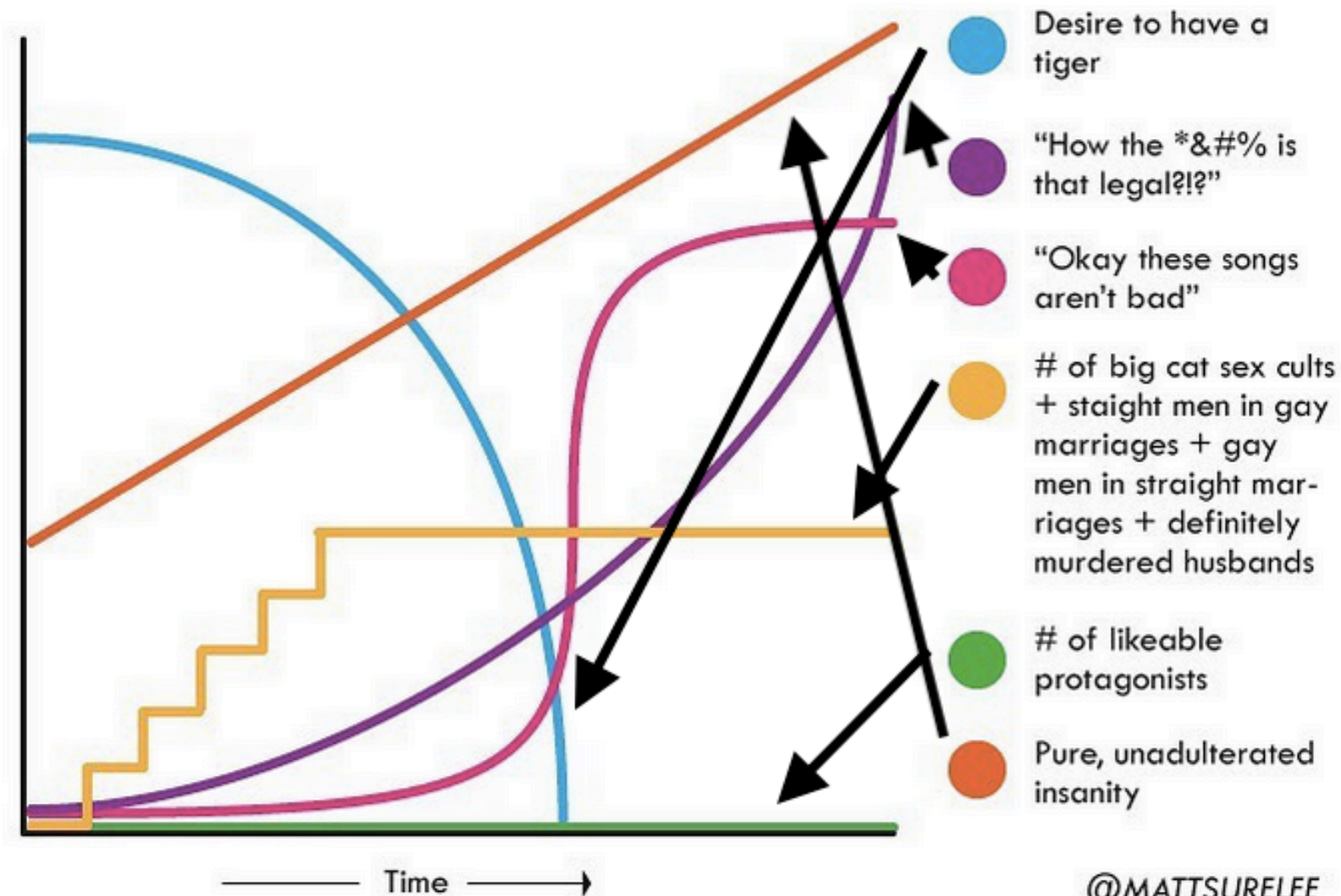
Better again: adding text to directly label lines. (We'll get back to that later.)

The standard ggplot colours aren't particularly attractive, and adjacent colours look quite similar when there are a lot of categories.





# WATCHING TIGER KING



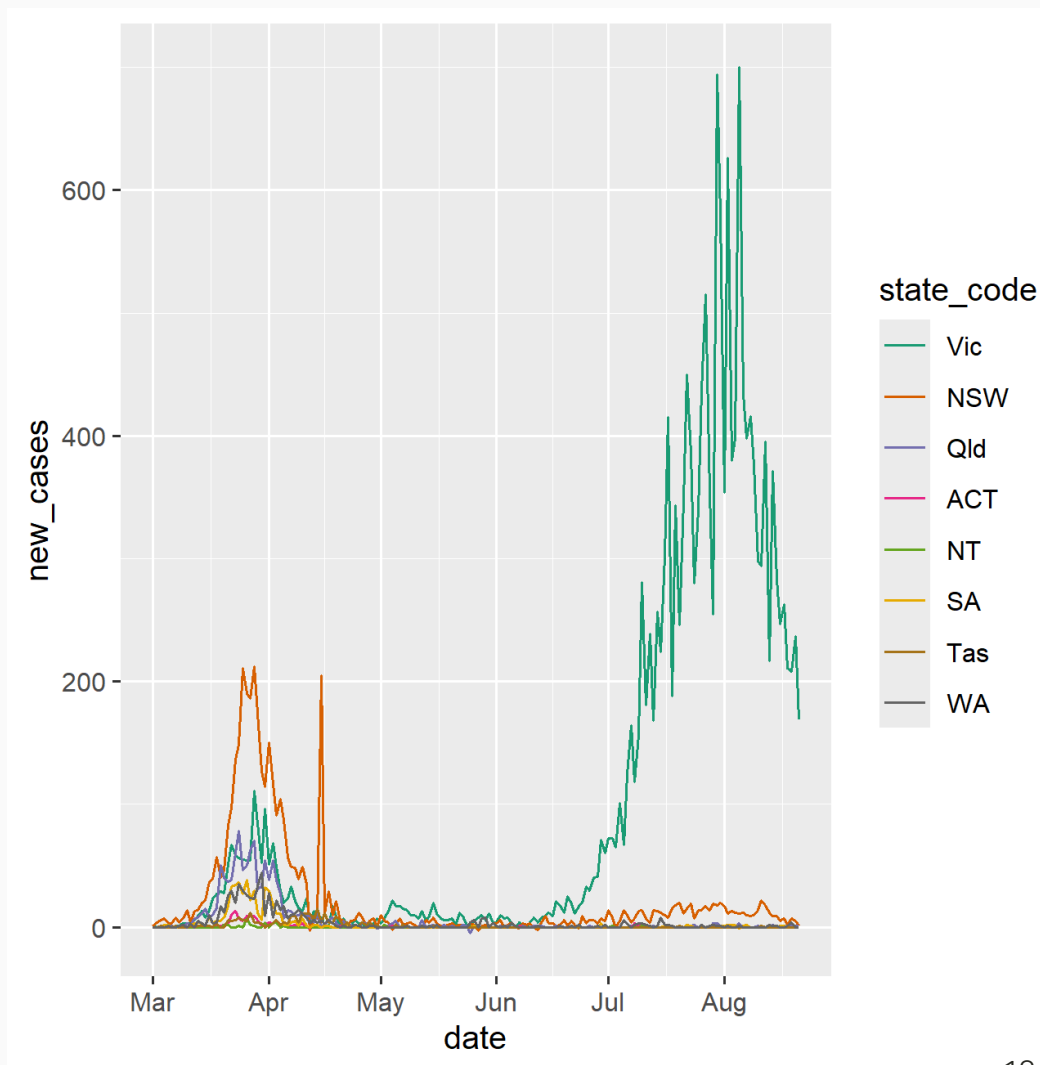
# Better line graphs: colour palettes

```
covid_by_state %>%  
  mutate(state_code = fct_reorder2(state_code, date, new_cases)) %>%  
  ggplot(aes(x = date, y = new_cases, colour = state_code)) +  
  geom_line() +  
  scale_colour_brewer(palette = "Dark2")
```

Color Brewer is a set of palettes designed for discrete data. I often use their palettes in my plots. More about Color Brewer: <https://www.r-graph-gallery.com/38-rcolorbrewers-palettes.html>

More R colour tips here:

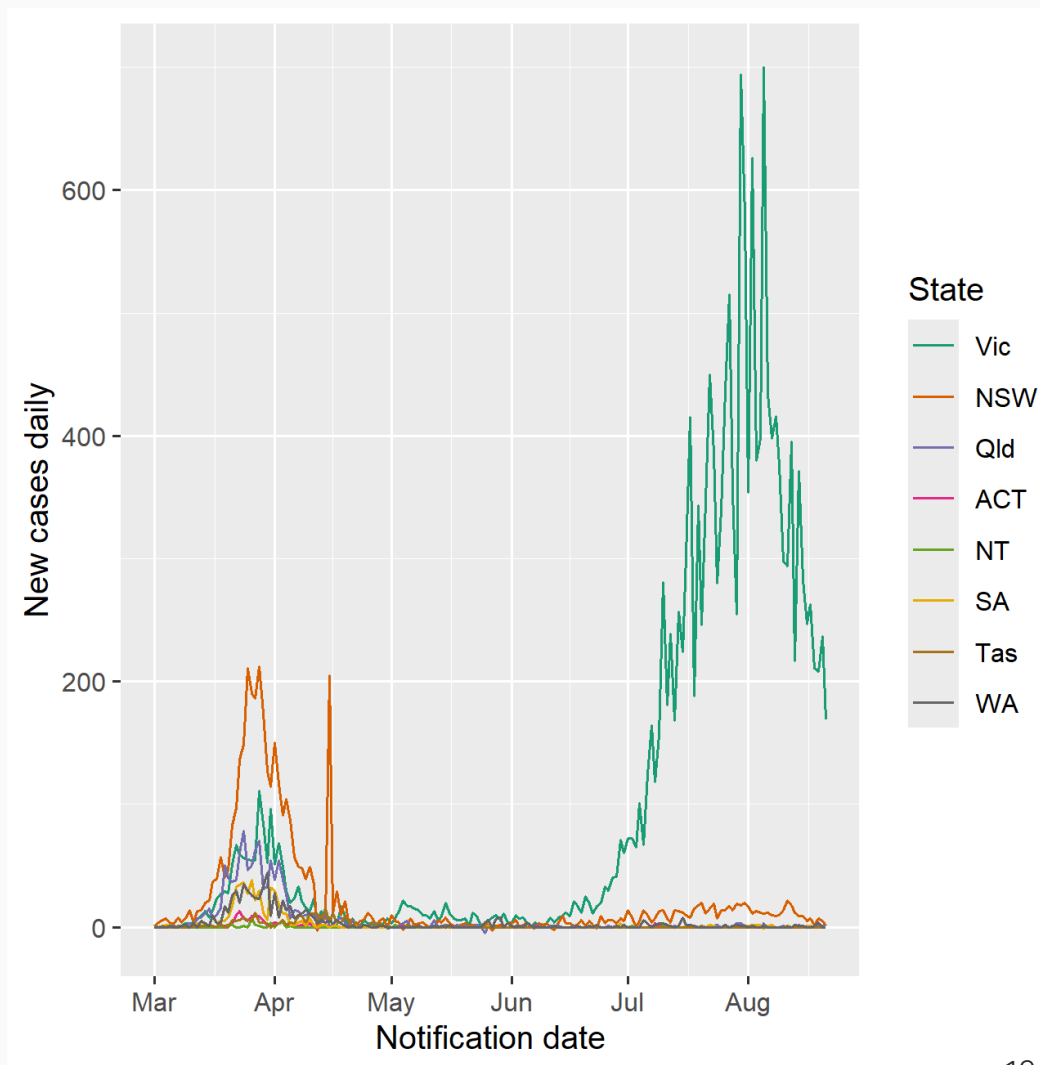
<https://www.datanovia.com/en/blog/ggplot-colors-best-tricks-you-will-love/>



# Better line graphs: think about axis labels

```
covid_by_state %>%  
  mutate(state_code = fct_reorder2(state_code, date, new_cases)) %>%  
  ggplot(aes(x = date, y = new_cases, colour = state_code)) +  
  geom_line() +  
  scale_colour_brewer(palette = "Dark2") +  
  labs(x = "Notification date",  
       y = "New cases daily",  
       colour = "State")
```

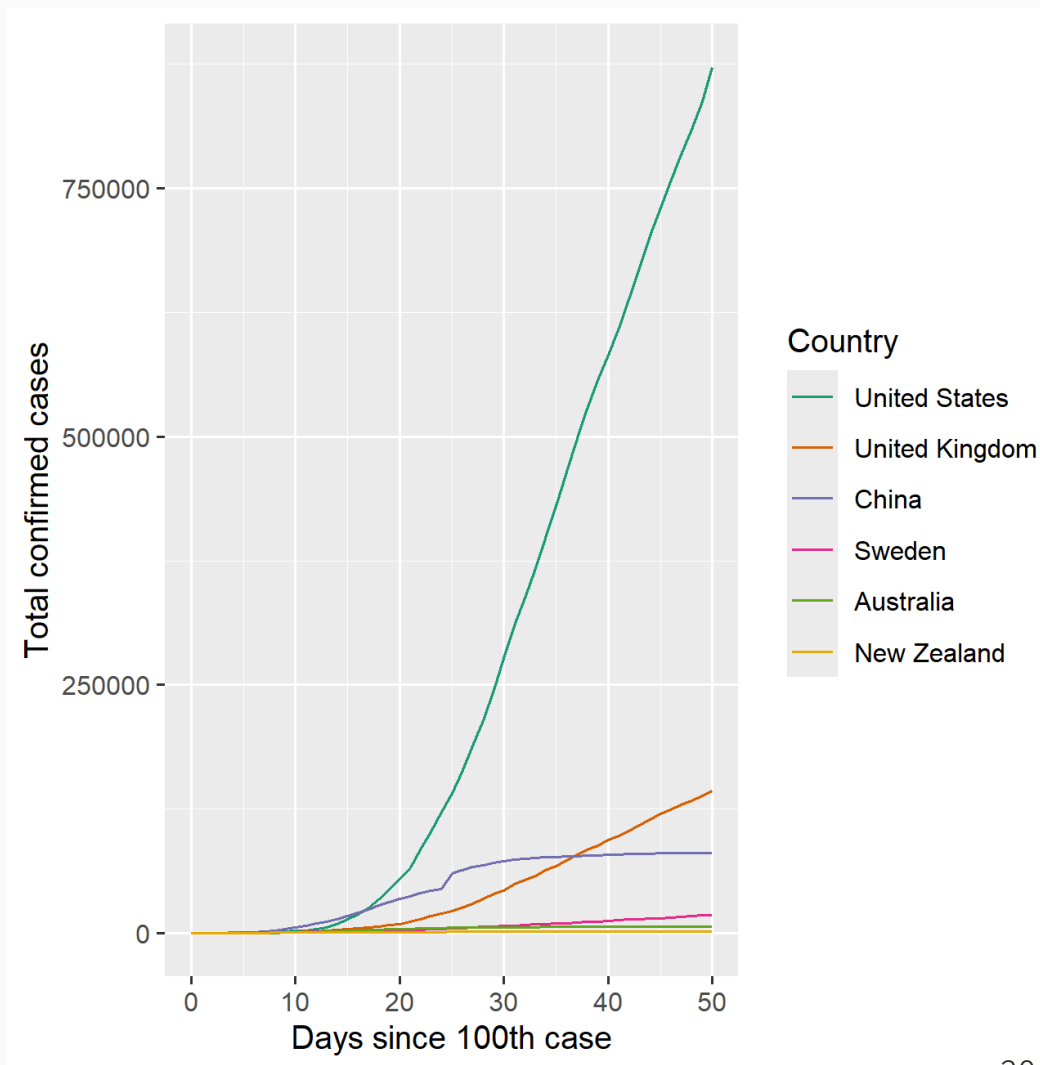
We can usually be more specific than just 'date': in this case, the date on the x axis represents the date at which authorities were notified of the new cases, not necessarily the date at which COVID-19 was acquired.



# Better line graphs: think about scales

```
covid_by_country_50days %>%  
  mutate(country = fct_reorder2(  
    country, days_since_100_case, confirmed)) %>%  
  ggplot(aes(x = days_since_100_case, y = confirmed,  
            colour = country)) +  
  geom_line() +  
  scale_colour_brewer(palette = "Dark2") +  
  labs(x = "Days since 100th case",  
       y = "Total confirmed cases",  
       colour = "Country")
```

This is a different COVID-19 dataset, showing total cases from different countries around the world.

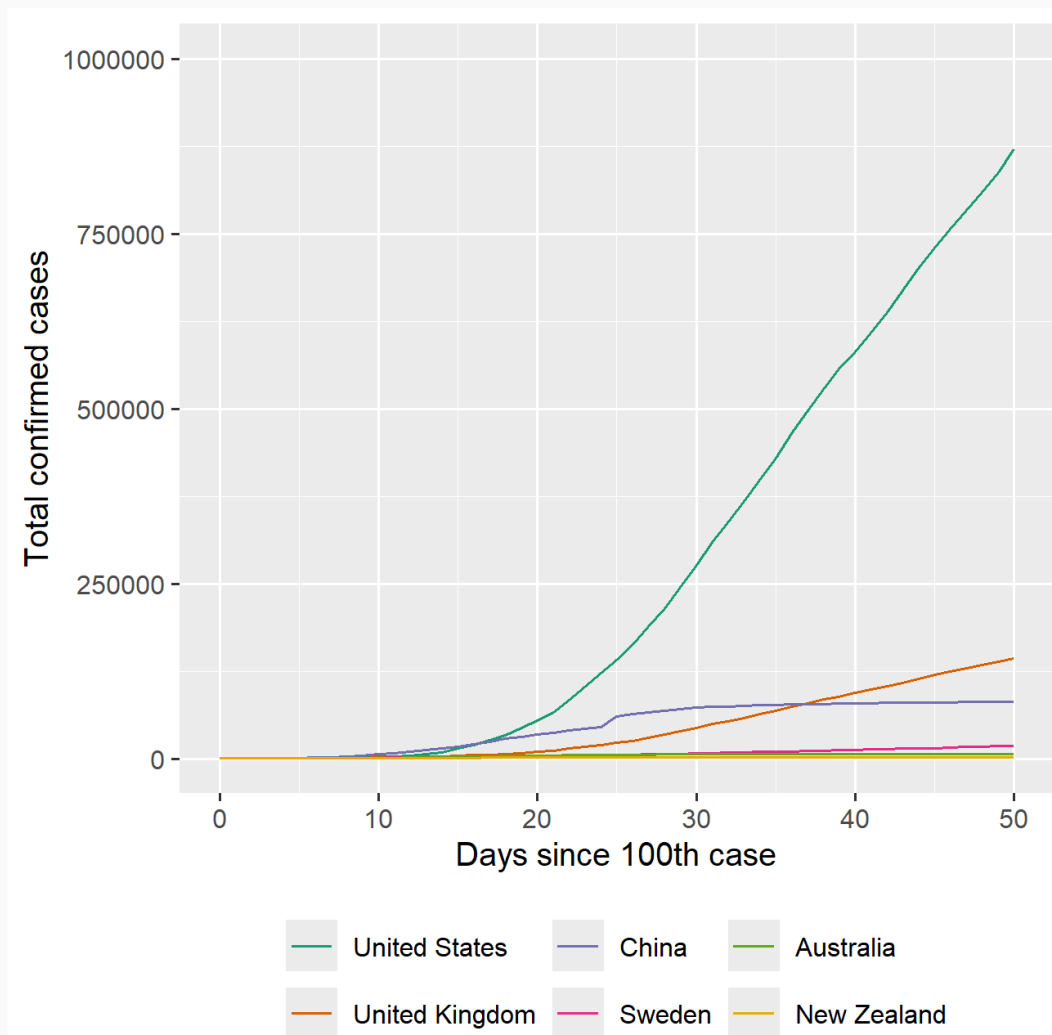


# Better line graphs: think about scales

```
covid_by_country_50days %>%  
  mutate(country = fct_reorder2(  
    country, days_since_100_case, confirmed)) %>%  
  ggplot(aes(x = days_since_100_case, y = confirmed,  
    colour = country)) +  
  geom_line() +  
  scale_colour_brewer(palette = "Dark2") +  
  scale_y_continuous(limits = c(0, 1000000)) +  
  labs(x = "Days since 100th case",  
    y = "Total confirmed cases",  
    colour = NULL) +  
  theme(legend.position = "bottom")
```

I've also moved the legend to the bottom to make better use of space, and changed the axis scale so it shows a round 1,000,000.

Changing the axis labels doesn't fix the big problem here: the countries with fewer cases are hard to see.

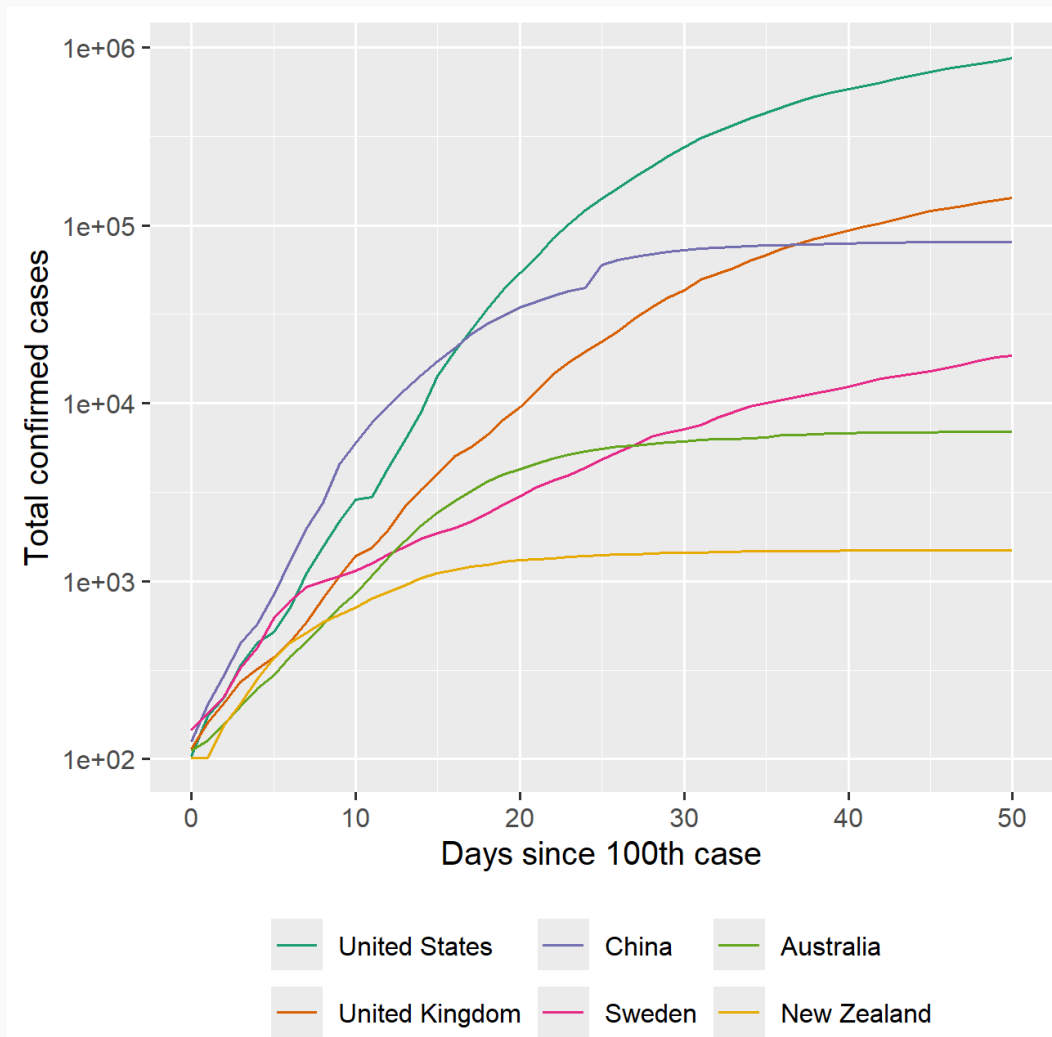


# Better line graphs: think about scales

```
covid_by_country_50days %>%  
  mutate(country = fct_reorder2(  
    country, days_since_100_case, confirmed)) %>%  
  ggplot(aes(x = days_since_100_case, y = confirmed,  
    colour = country)) +  
  geom_line() +  
  scale_colour_brewer(palette = "Dark2") +  
  scale_y_log10() +  
  labs(x = "Days since 100th case",  
    y = "Total confirmed cases",  
    colour = NULL) +  
  theme(legend.position = "bottom")
```

ggplot has decided to use scientific notation for the axis labels here. It's ugly!

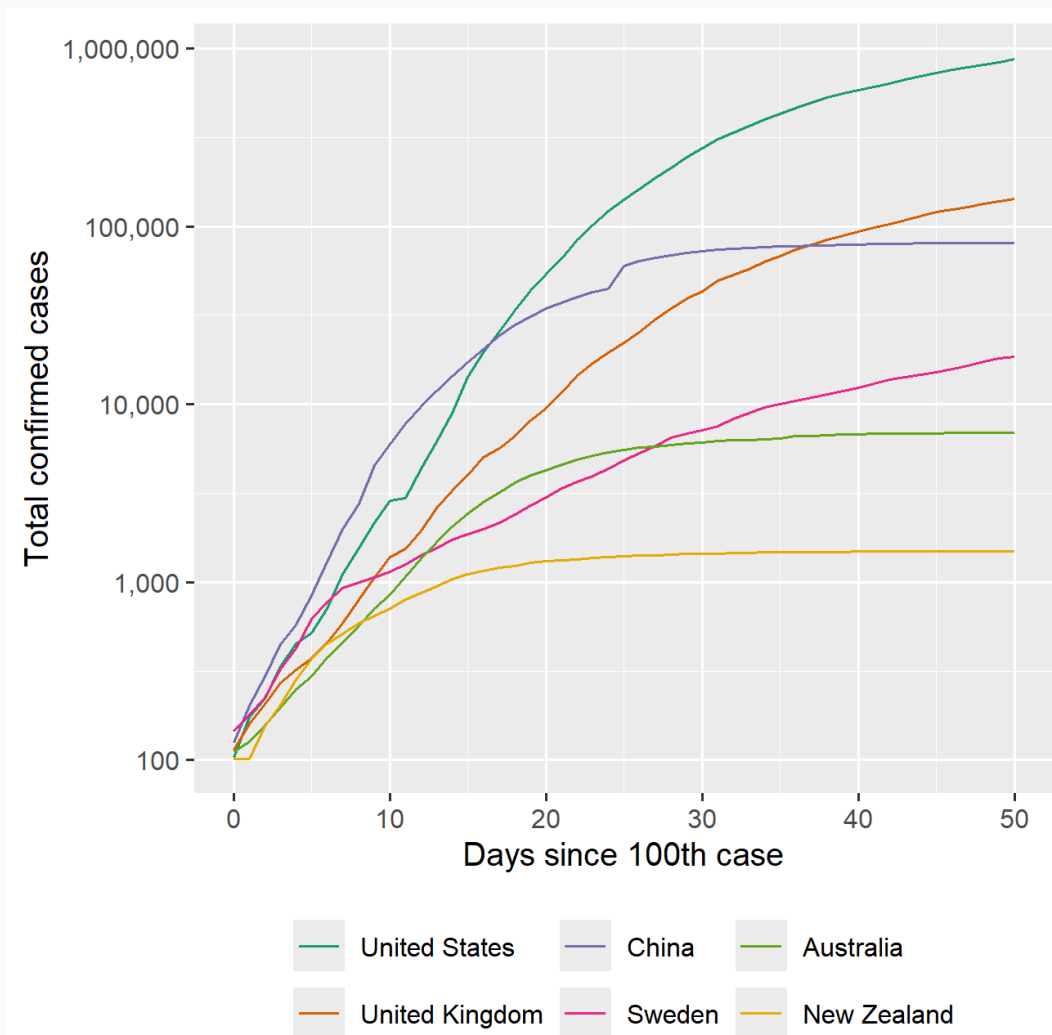
A log scale is useful when there is exponential growth and is also commonly used to display ratios (one quantity divided by another quantity). In this case, the log scale improves the legibility of the data from the smaller countries, but hides just how much worse the situation was in the US. As always, think about what you're trying to achieve



# Better line graphs: think about scales

```
covid_by_country_50days %>%  
  mutate(country = fct_reorder2(  
    country, days_since_100_case, confirmed)) %>%  
  ggplot(aes(x = days_since_100_case, y = confirmed,  
    colour = country)) +  
  geom_line() +  
  scale_colour_brewer(palette = "Dark2") +  
  scale_y_log10(labels = scales::label_comma()) +  
  labs(x = "Days since 100th case",  
    y = "Total confirmed cases",  
    colour = NULL) +  
  theme(legend.position = "bottom")
```

New R notation: `package::function()` refers to a function in a particular package. Here, `label_comma()` is one of a number of labelling functions in the `scales` package.

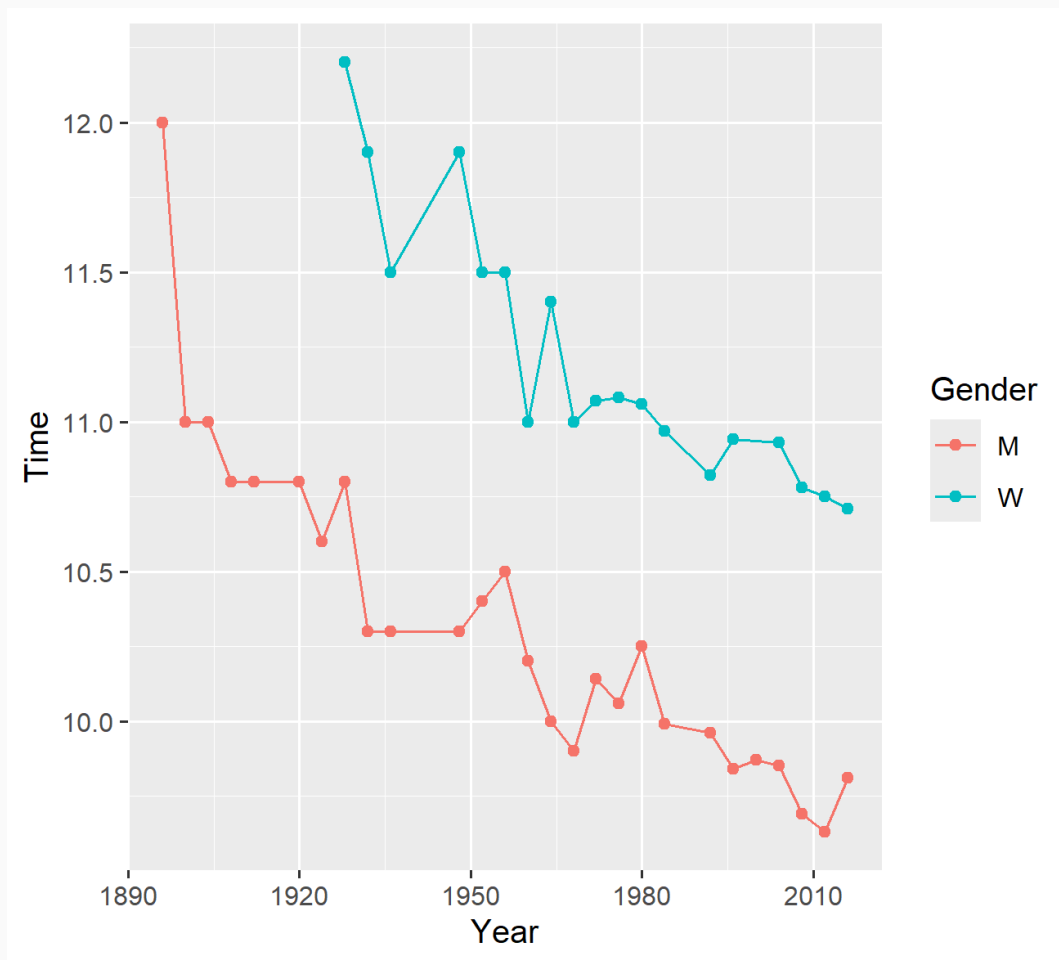


# Adding lines of best fit

```
olympic_100m_gold <- olympic_100m_data %>%  
  filter(Medal == "G")  
ggplot(olympic_100m_gold,  
  aes(x = Year, y = Time, colour = Gender)) +  
  geom_point() +  
  geom_line()
```

Back to the Olympic 100m data. Here's a scatter plot you saw earlier.

`filter()` selects rows matching a condition. This time we're only looking at gold medals. We'll see more about `filter()` later.





# Adding lines of best fit

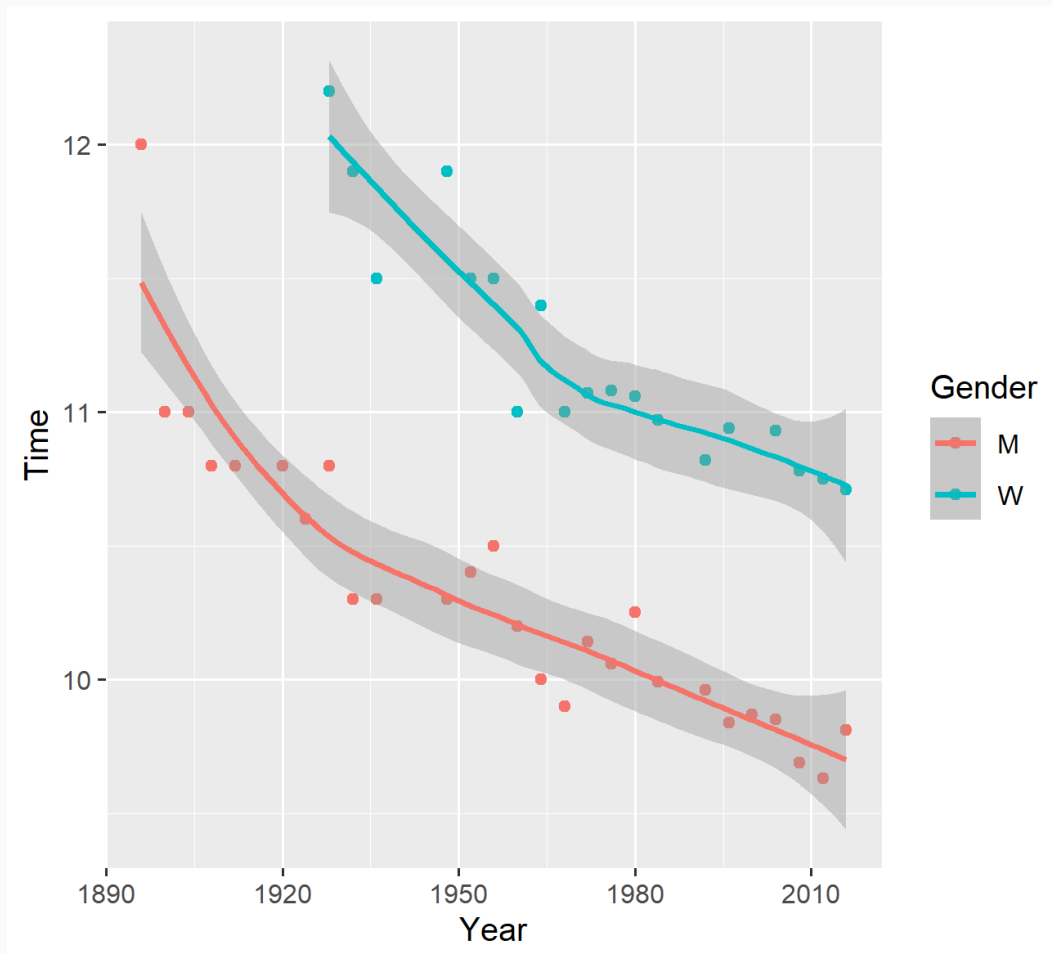
```
ggplot(olympic_100m_gold,  
  aes(x = Year, y = Time, colour = Gender)) +  
  geom_point() +  
  geom_smooth()
```

While `geom_line()` connects each consecutive observation with point, `geom_smooth()` fits a smooth curve. There are a few possible curves you can fit here; by default we get a LOESS (locally estimated scatterplot smoother) and a warning.

You've seen this before in ggplot, e.g. with histograms: often there are defaults which do *something*, but you are encouraged to make your own decisions.

What do you think the shaded area signifies?

`geom_smooth()` using method = 'loess' and formula = 'y ~ x'

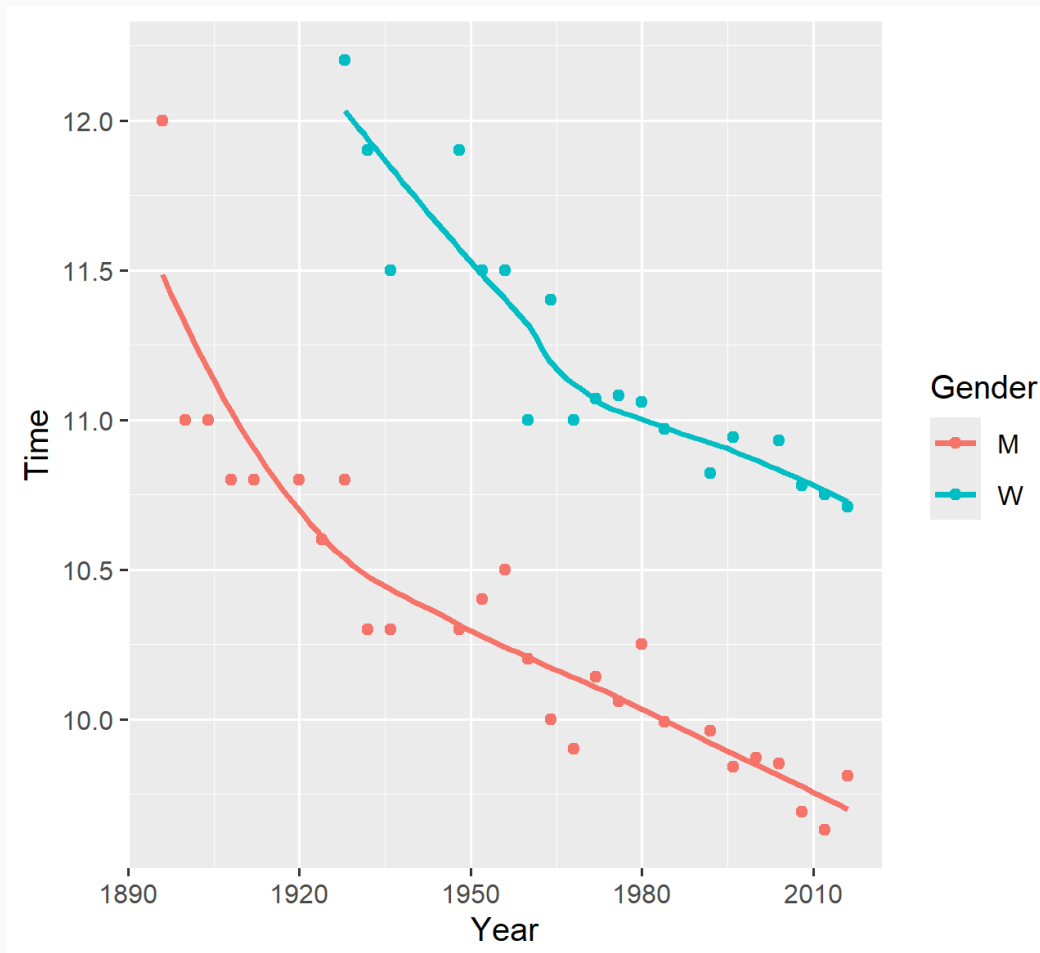


# Adding lines of best fit

```
ggplot(olympic_100m_gold,  
  aes(x = Year, y = Time, colour = Gender)) +  
  geom_point() +  
  geom_smooth(method = "loess", se = FALSE)
```

Here I've explicitly specified the smoothing method, and turned off the 95% confidence interval around the curve.

``geom_smooth()`` using formula = `'y ~ x'`



# Adding lines of best fit

```
ggplot(olympic_100m_gold,  
  aes(x = Year, y = Time, colour = Gender)) +  
  geom_point() +  
  geom_smooth(method = "lm", se = FALSE)
```

There are a few other methods accepted by `geom_smooth()`, but the only other one I use commonly is "lm" (linear model) which fits a straight line.

``geom_smooth()`` using formula = 'y ~ x'

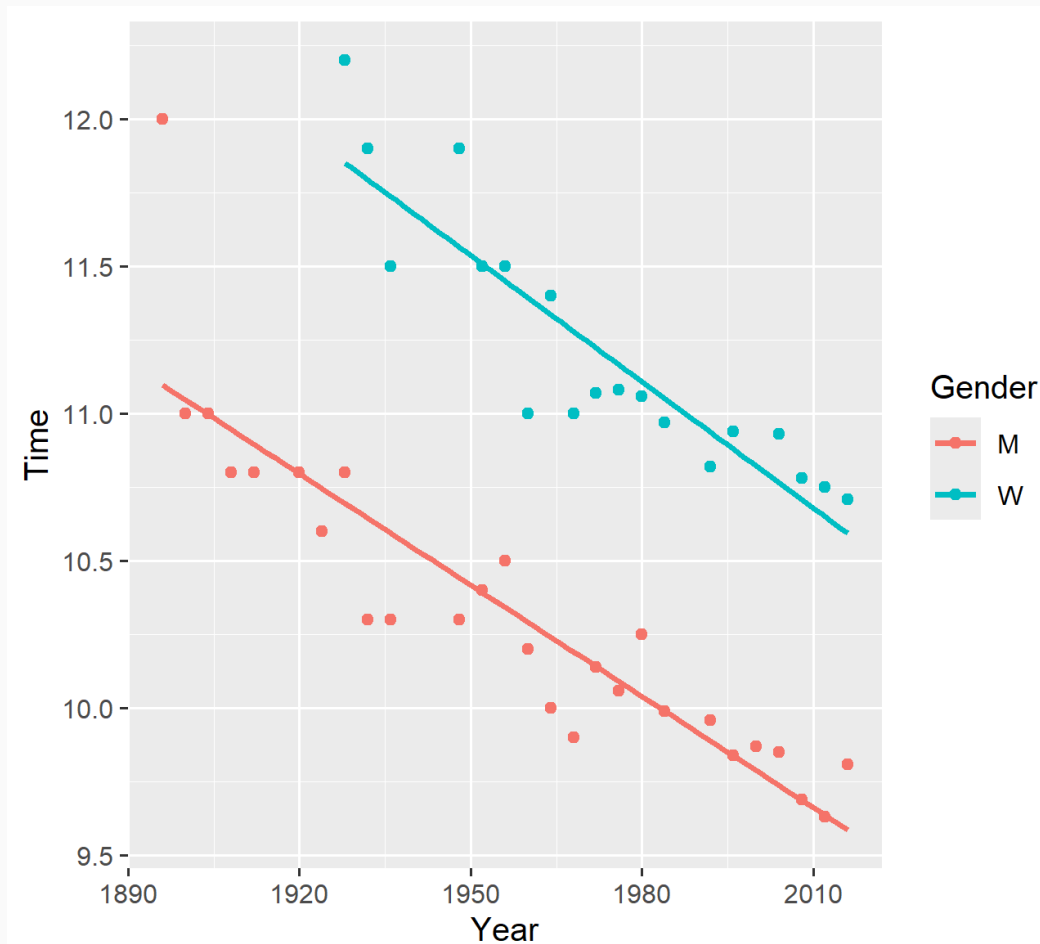
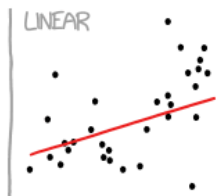


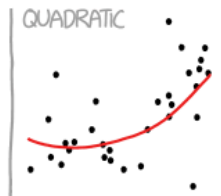
Image by Randall Munroe.

<https://xkcd.com/2048/>

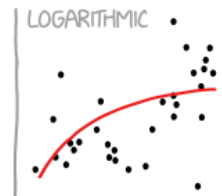
## CURVE-FITTING METHODS AND THE MESSAGES THEY SEND



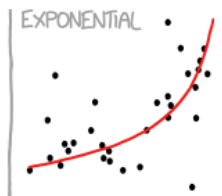
"HEY, I DID A  
REGRESSION."



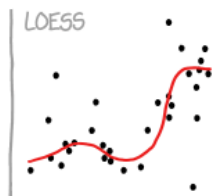
"I WANTED A CURVED  
LINE, SO I MADE ONE  
WITH MATH."



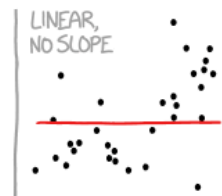
"LOOK, IT'S  
TAPERING OFF!"



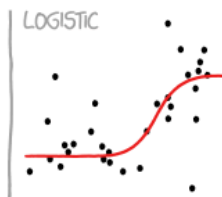
"LOOK, IT'S GROWING  
UNCONTROLLABLY!"



"I'M SOPHISTICATED, NOT  
LIKE THOSE BUMBLING  
POLYNOMIAL PEOPLE."



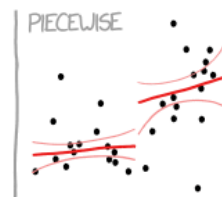
"I'M MAKING A  
SCATTER PLOT BUT  
I DON'T WANT TO."



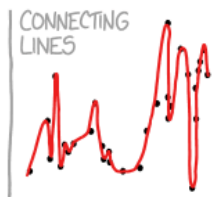
"I NEED TO CONNECT THESE  
TWO LINES, BUT MY FIRST IDEA  
DIDN'T HAVE ENOUGH MATH."



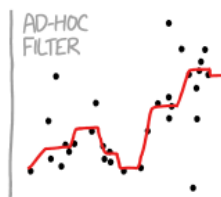
"LISTEN, SCIENCE IS HARD.  
BUT I'M A SERIOUS  
PERSON DOING MY BEST."



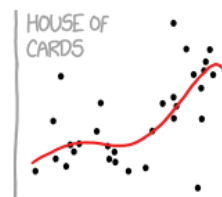
"I HAVE A THEORY,  
AND THIS IS THE ONLY  
DATA I COULD FIND."



"I CLICKED 'SMOOTH  
LINES' IN EXCEL."



"I HAD AN IDEA FOR HOW  
TO CLEAN UP THE DATA.  
WHAT DO YOU THINK?"



"AS YOU CAN SEE, THIS  
MODEL SMOOTHLY FITS  
THE— WAIT NO NO DON'T  
EXTEND IT AAAAAA!!!"

# Are we there yet?

Coming up next...

- Plotting group means and error bars
- Controlling the size of your plots
- Exporting plots to a file
- More faceted plots
- Adding annotations (labels, lines)

## Exercise 2.3.