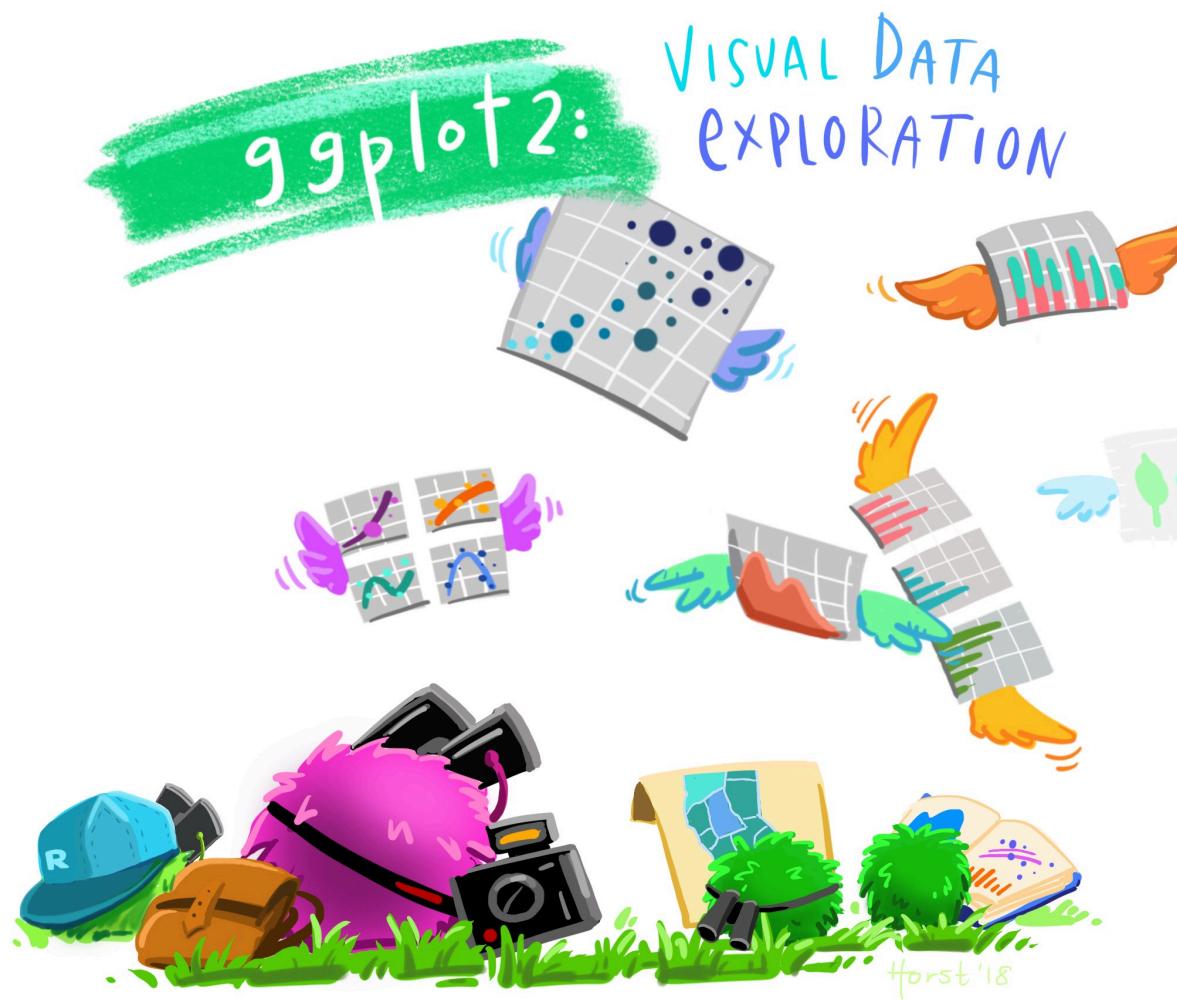


Towards publication-quality graphics (continued)

Melbourne Statistical Consulting Platform

University of Melbourne

April 2024



Plotting group means and error bars

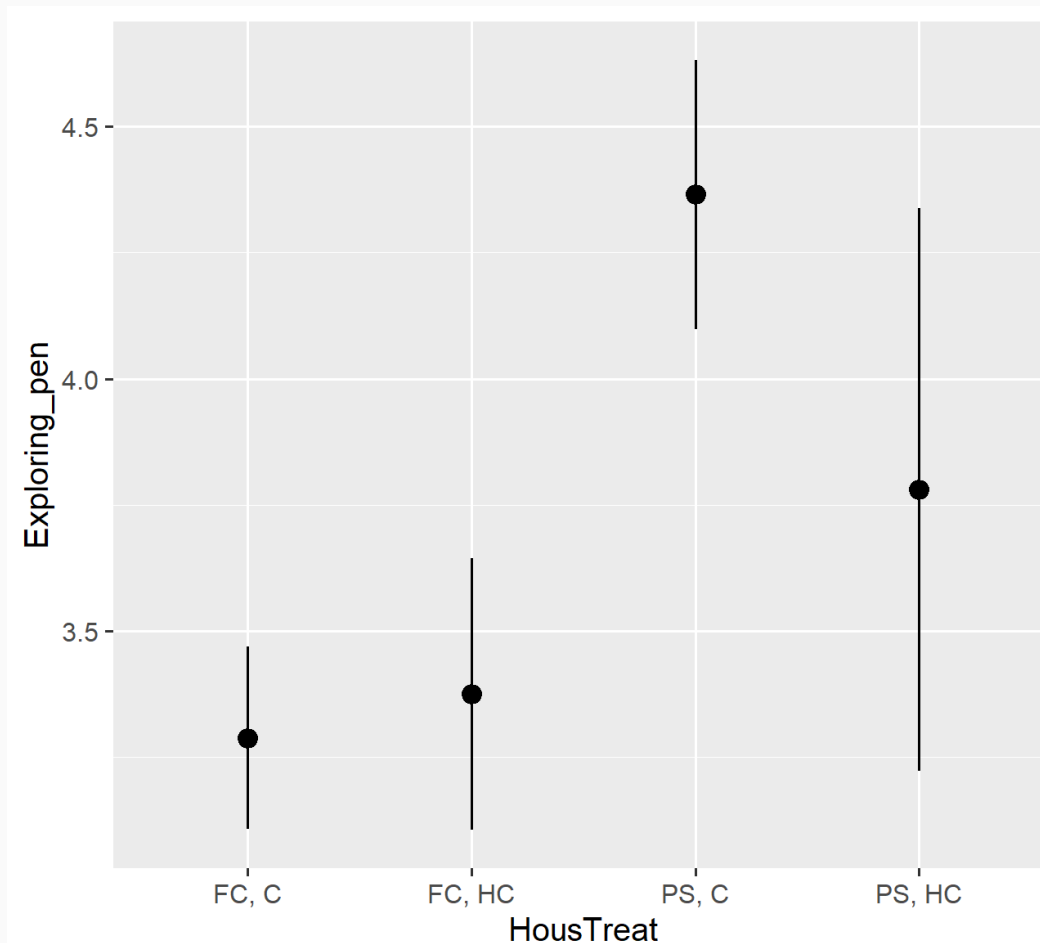
The data shown here come from an experiment by Dr Megan Lucas of the Animal Welfare Research Centre at The University of Melbourne. Piglets were kept in different types of pens (FC was a confined pen, PS was a loose pen) and either given human contact or not (C means control, HC means human contact).

After weaning, the pigs were exposed to novel stimuli and the amount of time spent engaging in different behaviours was recorded.

```
pig_behaviour_means <-  
  read_csv("pig_behaviour_means.csv")
```

```
pig_behaviour_means %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = HousTreat, y = Exploring_pen)) +  
  stat_summary()
```

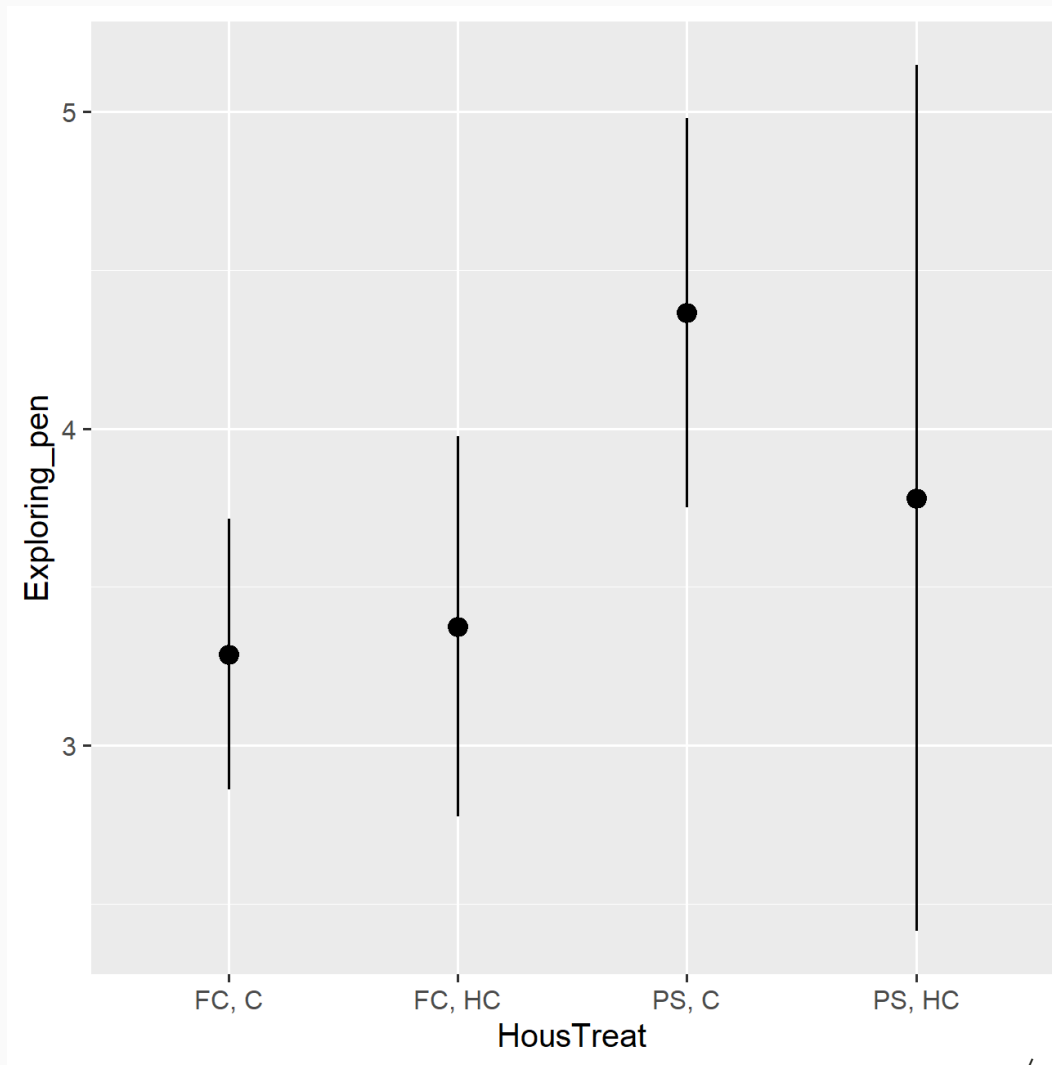
No summary function supplied, defaulting to `mean_se()`



Plotting group means and error bars

```
pig_behaviour_means %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = HousTreat, y = Exploring_pen)) +  
  stat_summary(fun.data = "mean_cl_normal")
```

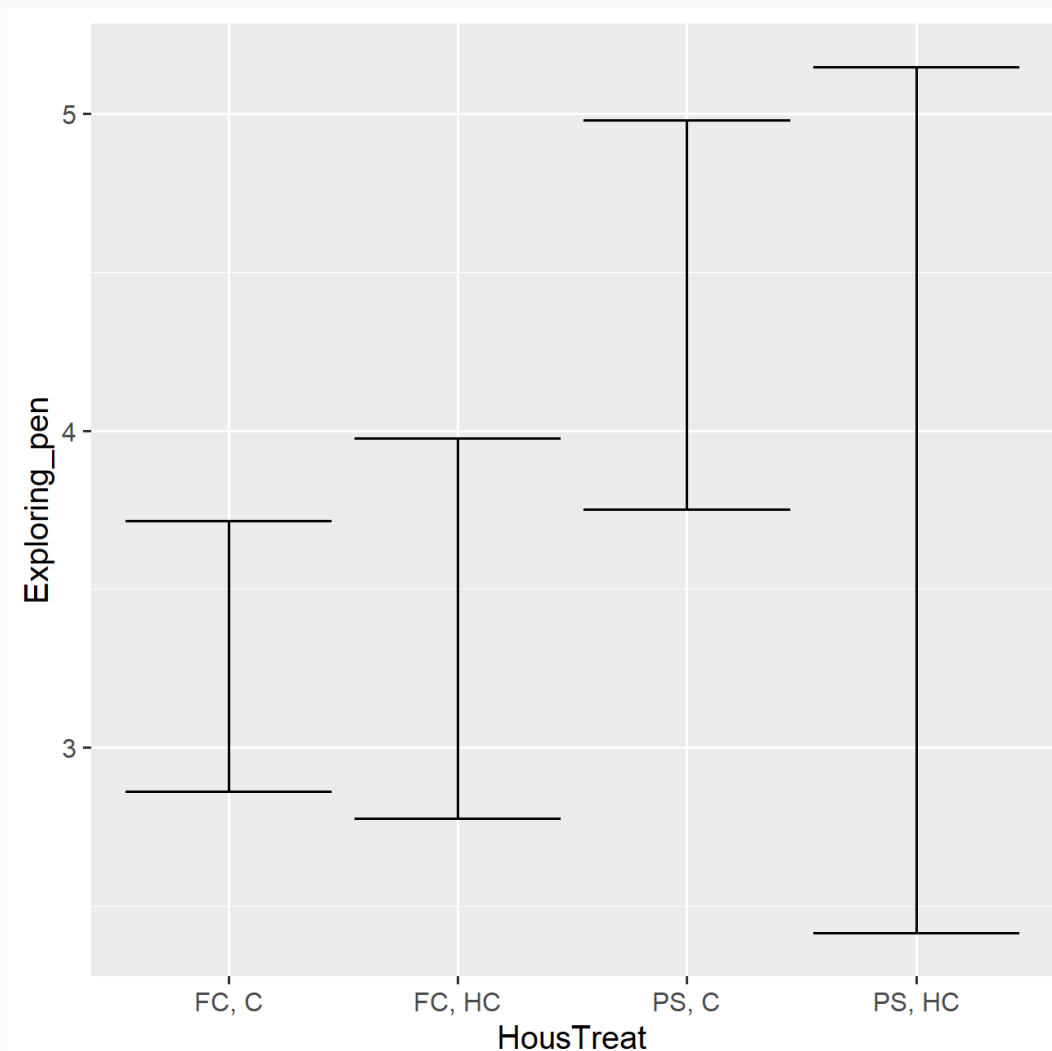
The function `mean_cl_normal` supplied with `ggplot` computes the mean and 95% confidence interval.



Plotting group means and error bars

```
pig_behaviour_means %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = HousTreat, y = Exploring_pen)) +  
  stat_summary(fun.data = "mean_cl_normal", geom = "errorbar")
```

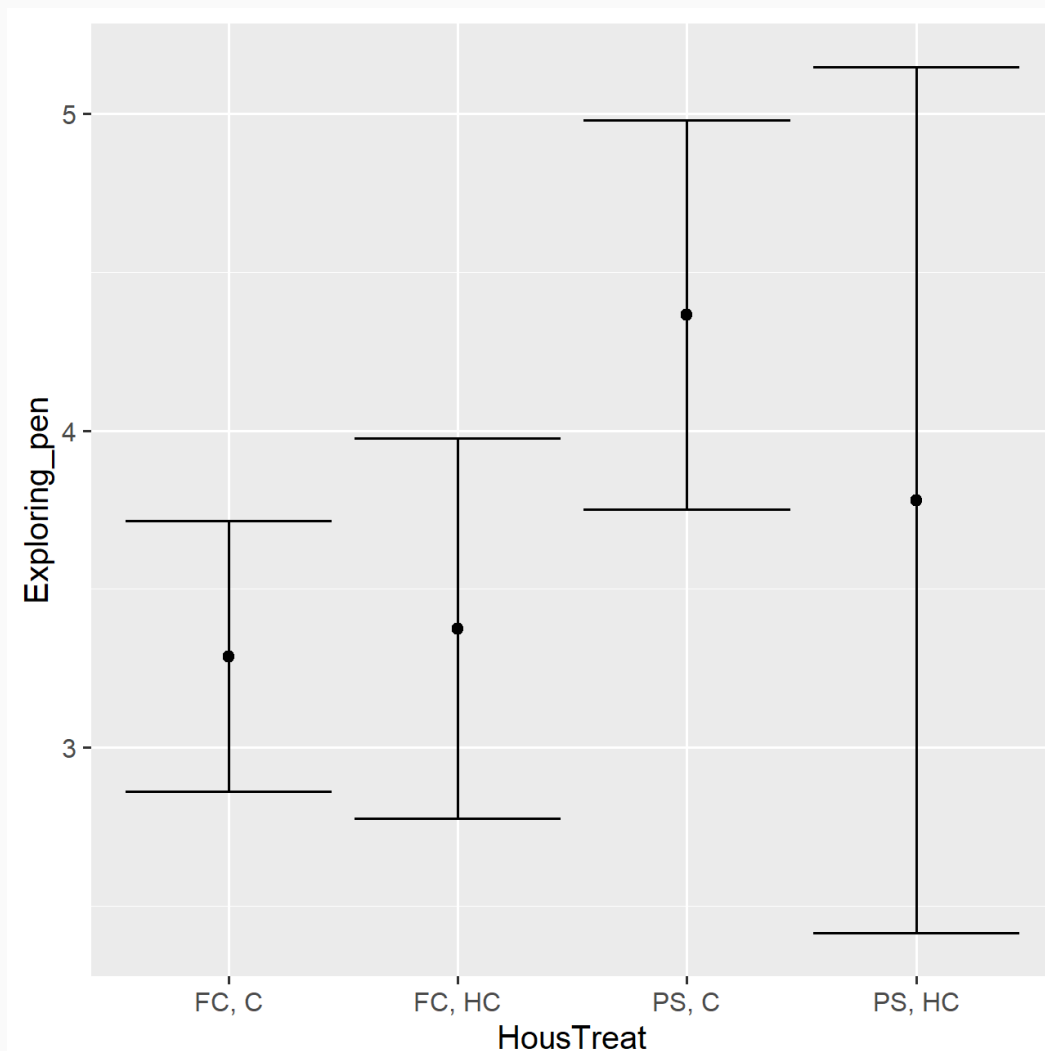
By default, `stat_summary()` uses `geom_pointrange()`, but you can change the geom. In this case, we see `geom_errorbar()` which plots the error bars but no point in the middle.



Plotting group means and error bars

```
pig_behaviour_means %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = HousTreat, y = Exploring_pen)) +  
    stat_summary(fun.data = "mean_cl_normal", geom = "errorbar") +  
    stat_summary(fun.data = "mean_cl_normal", geom = "point")
```

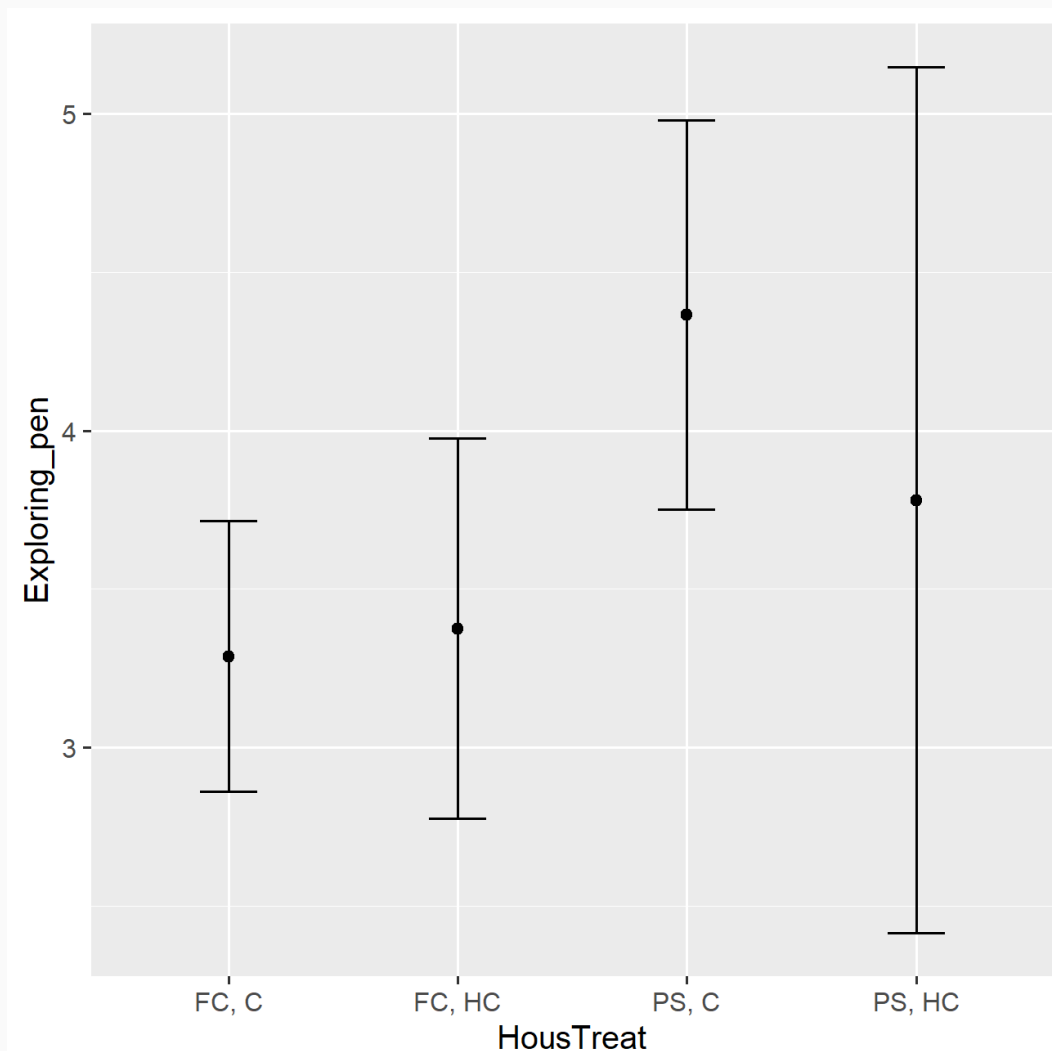
We need to use `stat_summary()` twice to get both a centre point for the mean and an error bar for the confidence interval.



Plotting group means and error bars

```
pig_behaviour_means %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = HousTreat, y = Exploring_pen)) +  
    stat_summary(fun.data = "mean_cl_normal", geom = "errorbar",  
                width = 0.25) +  
    stat_summary(fun.data = "mean_cl_normal", geom = "point")
```

Error bars in ggplot default to taking up the entire width between categories. You can change that with the `width =` option.

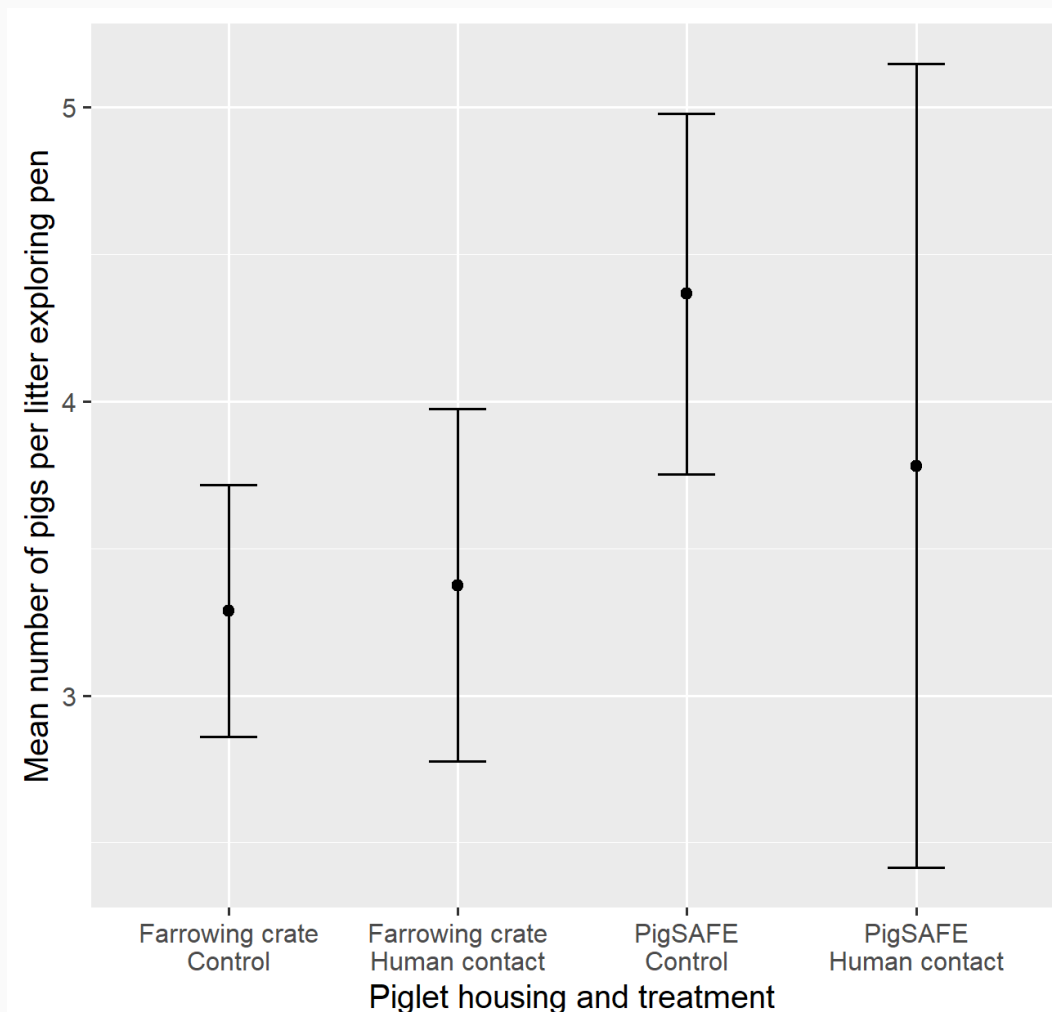


Plotting group means and error bars

```
pig_behaviour_means %>%  
  drop_na(Exploring_pen) %>%  
  mutate(HousTreat = fct_recode(HousTreat,  
    "Farrowing crate\nControl" = "FC, C",  
    "Farrowing crate\nHuman contact" = "FC, HC",  
    "PigSAFE\nControl" = "PS, C",  
    "PigSAFE\nHuman contact" = "PS, HC")) %>%  
  ggplot(aes(x = HousTreat, y = Exploring_pen)) +  
  stat_summary(fun.data = "mean_cl_normal", geom = "errorbar",  
    width = 0.25) +  
  stat_summary(fun.data = "mean_cl_normal", geom = "point") +  
  labs(x = "Piglet housing and treatment",  
    y = "Mean number of pigs per litter exploring pen",  
    caption =  
      "Error bars show 95% confidence interval for the mean.")
```

Error bars get used to show a lot of different things. It's good practice to make it very clear what your error bars represent.

Note the use of `\n` ("new line") to insert a line break in the labels.



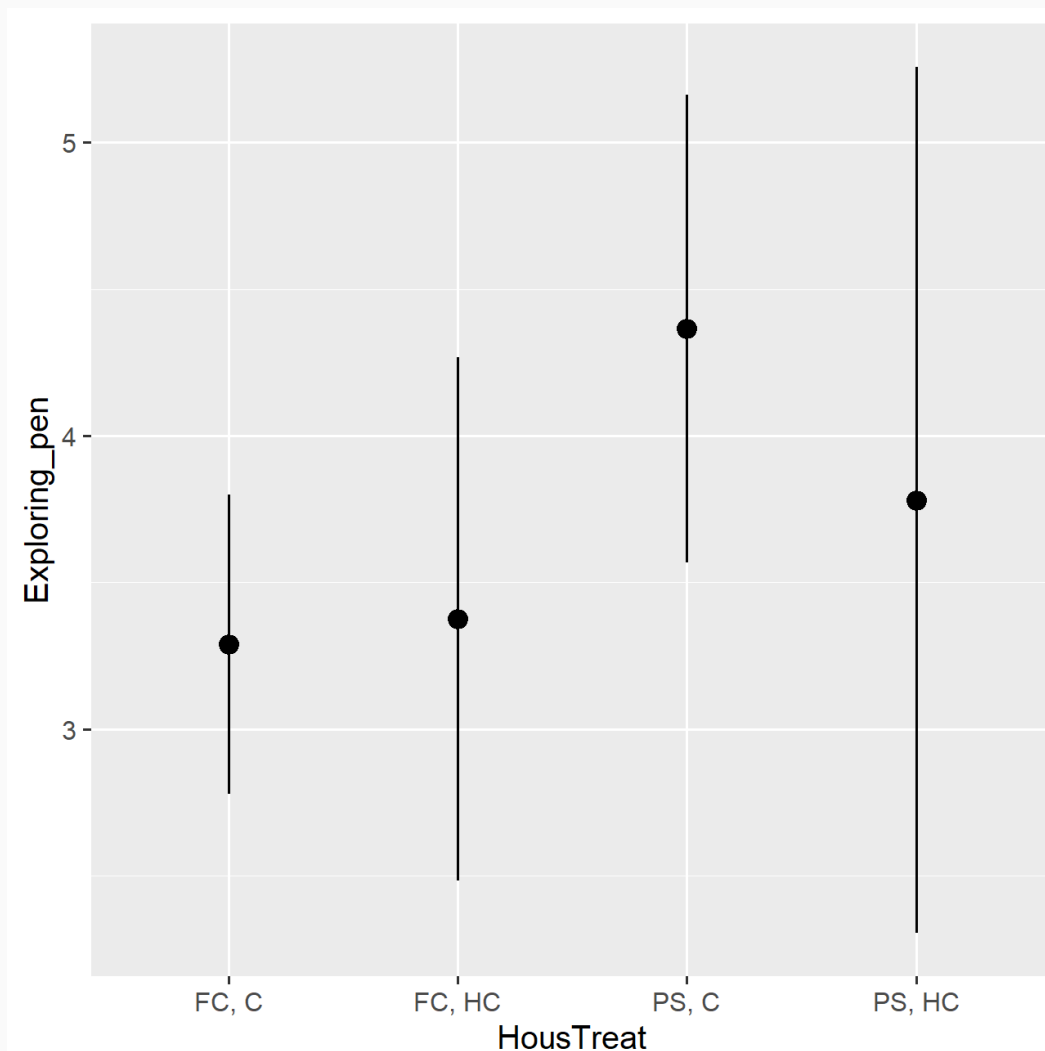
Error bars show 95% confidence interval for the mean.

Plotting group means and error bars

```
pig_behaviour_means %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = HousTreat, y = Exploring_pen)) +  
    stat_summary(fun = ~ mean(.),  
                 fun.min = ~ mean(.) - sd(.),  
                 fun.max = ~ mean(.) + sd(.))
```

There's no built-in option to plot mean and standard deviation with `stat_summary()`, but you can specify your own summaries of the data like this. It's not elegant, but it works.

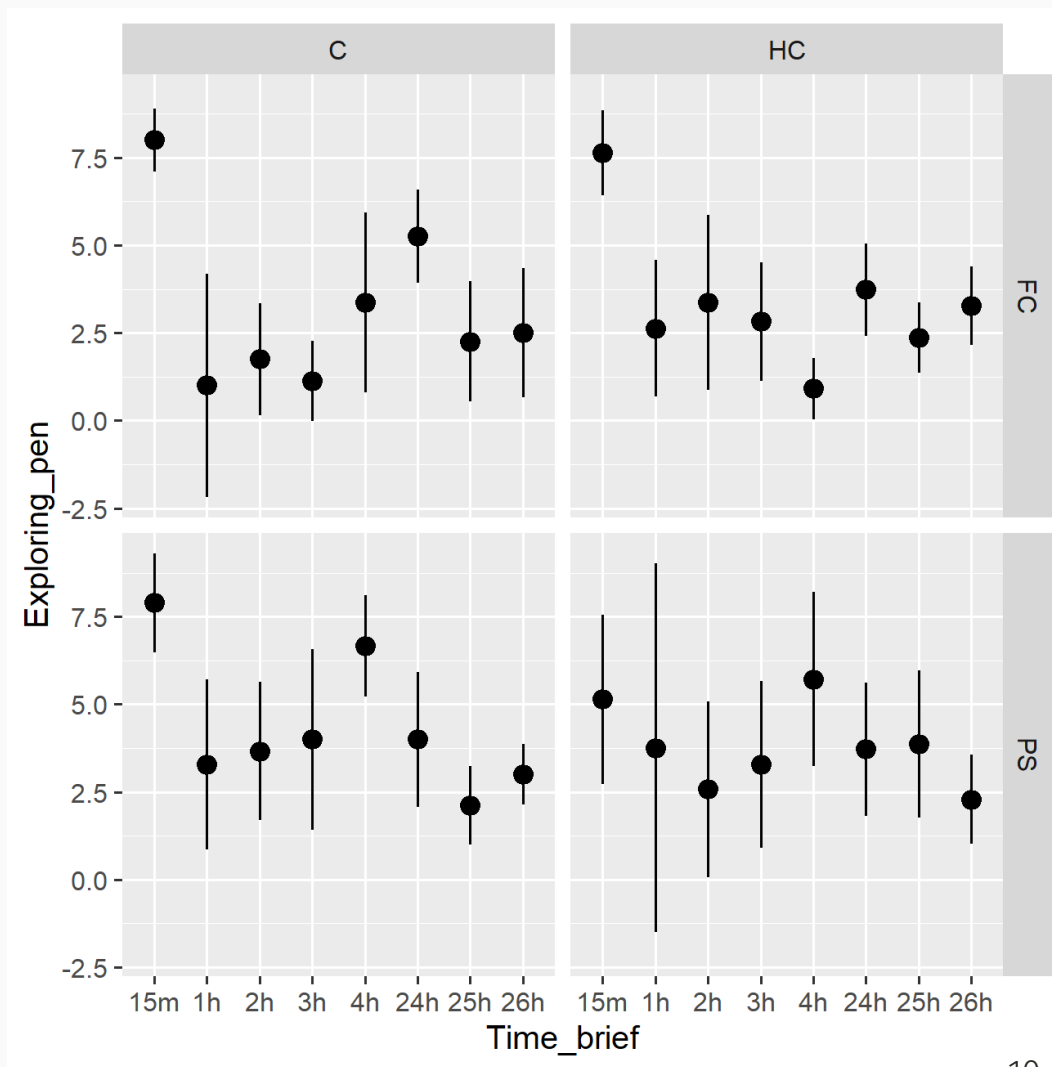
For more complex summary statistics, it's better to separately summarise the data and then use `ggplot` to plot the data once it's been transformed.



Facetting with multiple variables

```
pig_behaviour_data %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = Time_brief, y = Exploring_pen)) +  
    stat_summary(fun.data = "mean_cl_normal") +  
    facet_grid(rows = vars(Housing),  
              cols = vars(Treatment))
```

It is possible to facet on more than one variable. Using `facet_grid()`, you assign one variable to rows and one variable to columns.

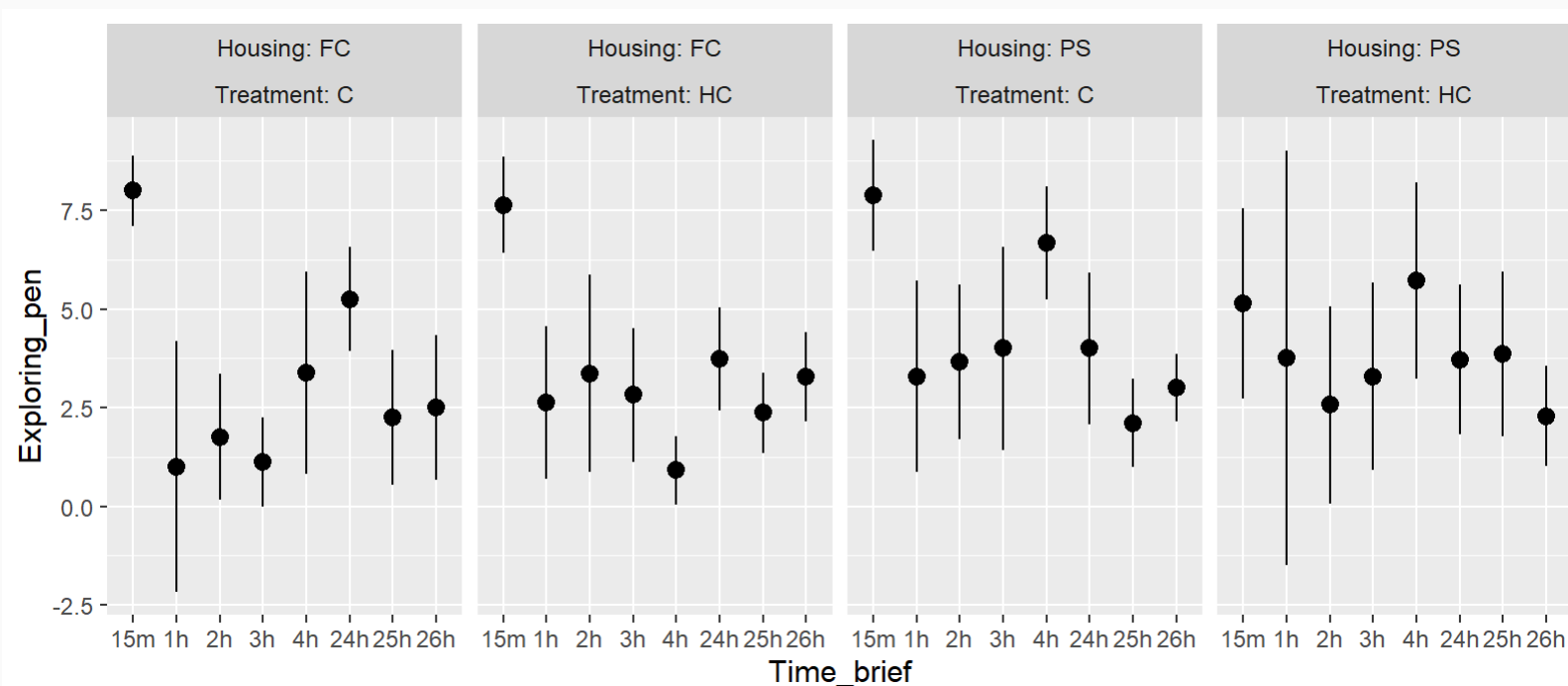


Facetting with multiple variables

Using `facet_wrap()`, the facet variables don't have to be assigned to an axis.

Setting `labeller = "label_both"` makes it clear which part of the label belongs to which variable.

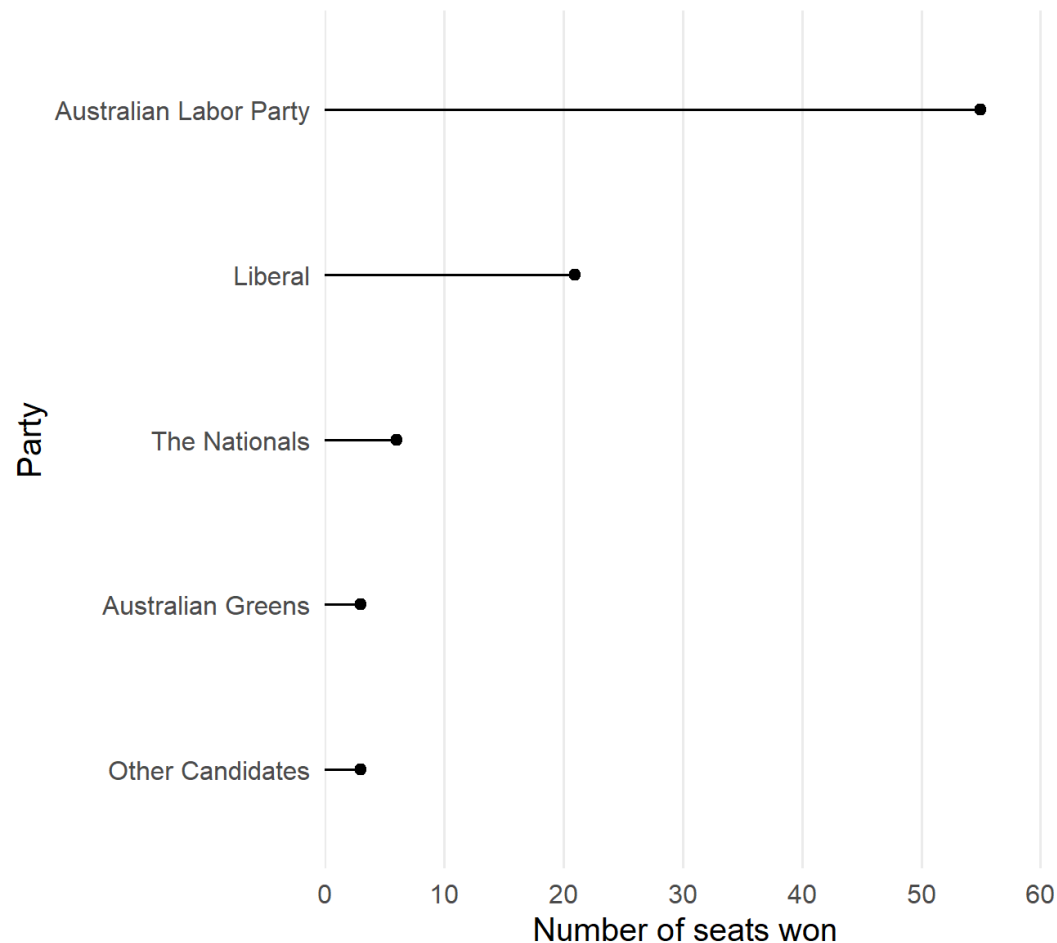
```
pig_behaviour_data %>%  
  drop_na(Exploring_pen) %>%  
  ggplot(aes(x = Time_brief, y = Exploring_pen)) +  
  stat_summary(fun.data = "mean_cl_normal") +  
  facet_wrap(vars(Housing, Treatment), ncol = 4, labeller = "label_both")
```



Adding annotations

```
election_data %>%
  mutate(party = fct_reorder(party, seats_won, .desc = TRUE)) %>%
  ggplot(aes(x = seats_won, y = party)) +
  geom_segment(aes(xend = 0, yend = party)) +
  geom_point() +
  scale_x_continuous(expand = expansion(mult = c(0, 0.1))) +
  scale_y_discrete(limits = rev) +
  labs(x = "Number of seats won",
       y = "Party",
       title = "Victorian state election 2018 lower house results",
       caption = "Data source: Victorian Electoral Commission") +
  theme_minimal() +
  theme(plot.title.position = "plot",
        panel.grid.minor.x = element_blank(),
        panel.grid.major.y = element_blank())
```

Victorian state election 2018 lower house results



Data source: Victorian Electoral Commission

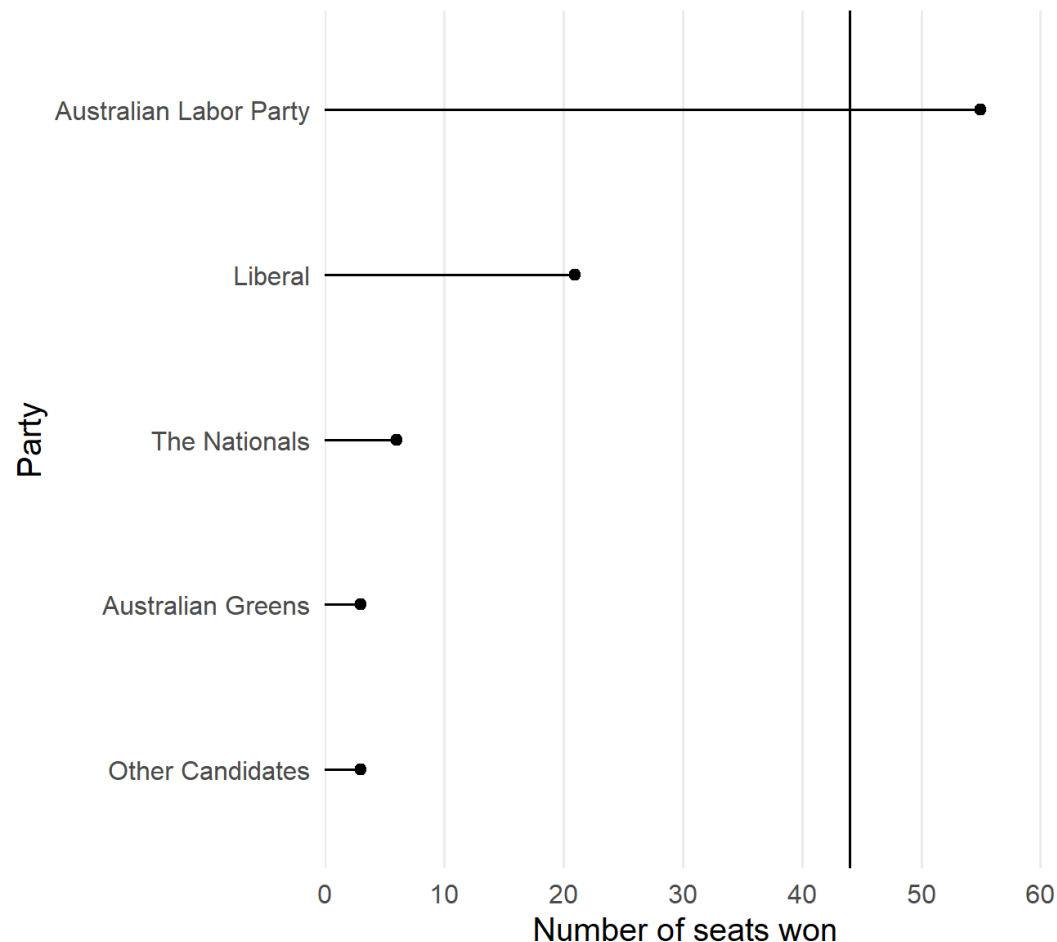
Adding annotations

```
election_data %>%  
  mutate(party = fct_reorder(party, seats_won, .desc = TRUE)) %>%  
  ggplot(aes(x = seats_won, y = party)) +  
  geom_segment(aes(xend = 0, yend = party)) +  
  geom_point() +  
  geom_vline(xintercept = 44) +  
  scale_x_continuous(expand = expansion(mult = c(0, 0.1))) +  
  scale_y_discrete(limits = rev) +  
  labs(x = "Number of seats won",  
       y = "Party",  
       title = "Victorian state election 2018 lower house results",  
       caption = "Data source: Victorian Electoral Commission") +  
  theme_minimal() +  
  theme(plot.title.position = "plot",  
        panel.grid.minor.x = element_blank(),  
        panel.grid.major.y = element_blank())
```

There are 88 seats in Victorian parliament, so a majority is achieved by a party with more than 44 seats.

`geom_vline()` adds a vertical line. There is also `geom_hline()` (horizontal), `geom_abline()` (provide slope and intercept) and `geom_segment()` (provide start and end coordinates).

Victorian state election 2018 lower house results



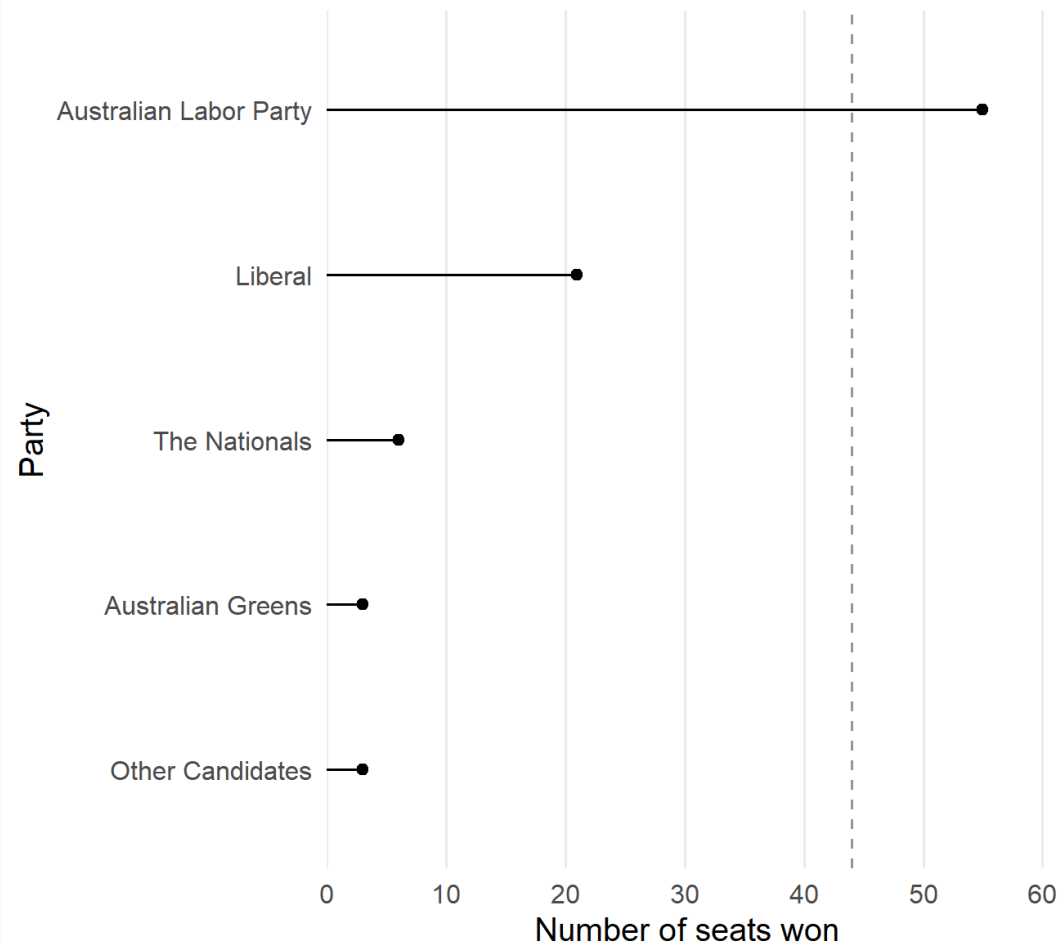
Data source: Victorian Electoral Commission

Adding annotations

```
election_data %>%
  mutate(party = fct_reorder(party, seats_won, .desc = TRUE)) %>%
  ggplot(aes(x = seats_won, y = party)) +
  geom_segment(aes(xend = 0, yend = party)) +
  geom_point() +
  geom_vline(xintercept = 44, linetype = "dashed",
            colour = "grey60") +
  scale_x_continuous(expand = expansion(mult = c(0, 0.1))) +
  scale_y_discrete(limits = rev) +
  labs(x = "Number of seats won",
       y = "Party",
       title = "Victorian state election 2018 lower house results",
       caption = "Data source: Victorian Electoral Commission") +
  theme_minimal() +
  theme(plot.title.position = "plot",
        panel.grid.minor.x = element_blank(),
        panel.grid.major.y = element_blank())
```

Change the colour of the line and made it dashed, so that it is distinct from the gridlines but doesn't dominate the display. (You can also set the width of the line, using `size`.)

Victorian state election 2018 lower house results



Data source: Victorian Electoral Commission

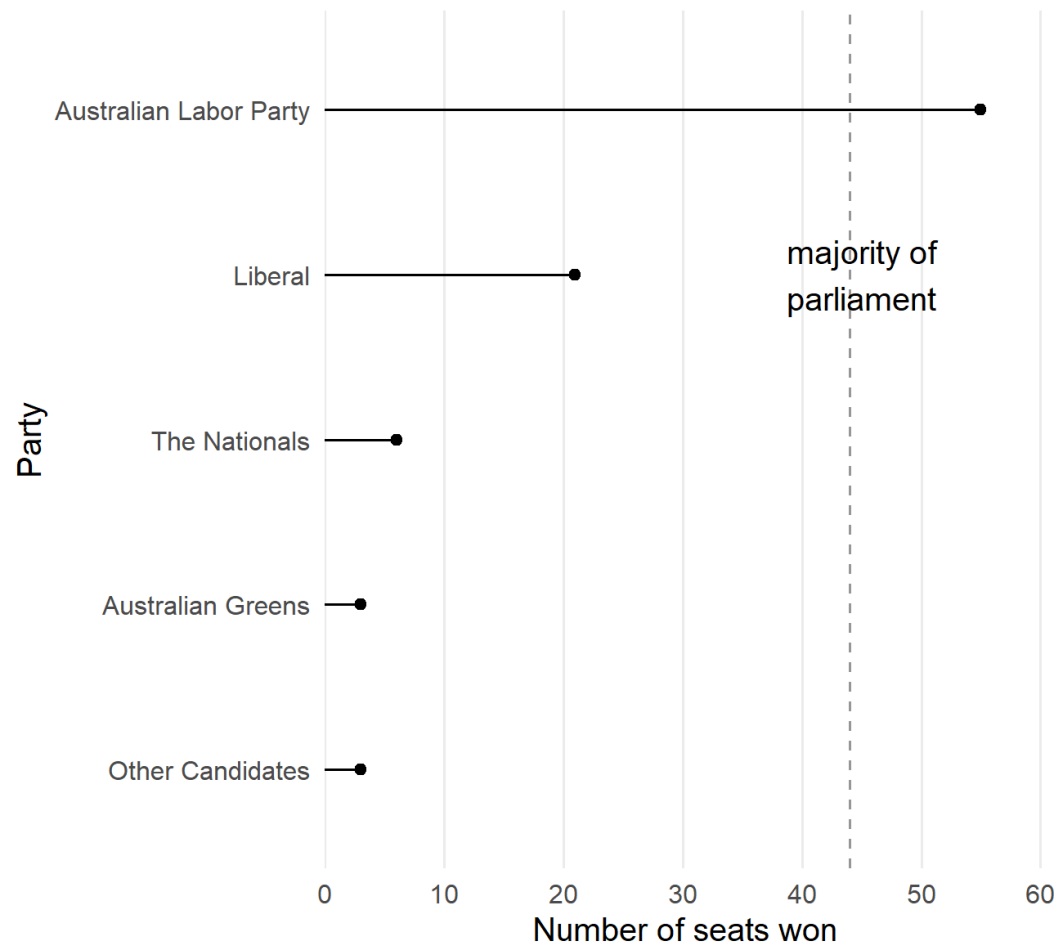
Adding annotations

```
election_data %>%  
  mutate(party = fct_reorder(party, seats_won, .desc = TRUE)) %>%  
  ggplot(aes(x = seats_won, y = party)) +  
  geom_segment(aes(xend = 0, yend = party)) +  
  geom_point() +  
  geom_vline(xintercept = 44, linetype = "dashed",  
            colour = "grey60") +  
  annotate("text", label = "majority of\nparliament",  
         x = 45, y = 4) +  
  scale_x_continuous(expand = expansion(mult = c(0, 0.1))) +  
  scale_y_discrete(limits = rev) +  
  labs(x = "Number of seats won",  
       y = "Party",  
       title = "Victorian state election 2018 lower house results",  
       caption = "Data source: Victorian Electoral Commission") +  
  theme_minimal() +  
  theme(plot.title.position = "plot",  
        panel.grid.minor.x = element_blank(),  
        panel.grid.major.y = element_blank())
```

`geom_text()` is used for adding text to a plot. Normally it plots data from rows in the data frame. `annotate()` allows us to make a `geom_text()` with manually entered data.

The label doesn't look quite right - it's been centred on the coordinates we've given, and is a bit large.

Victorian state election 2018 lower house results



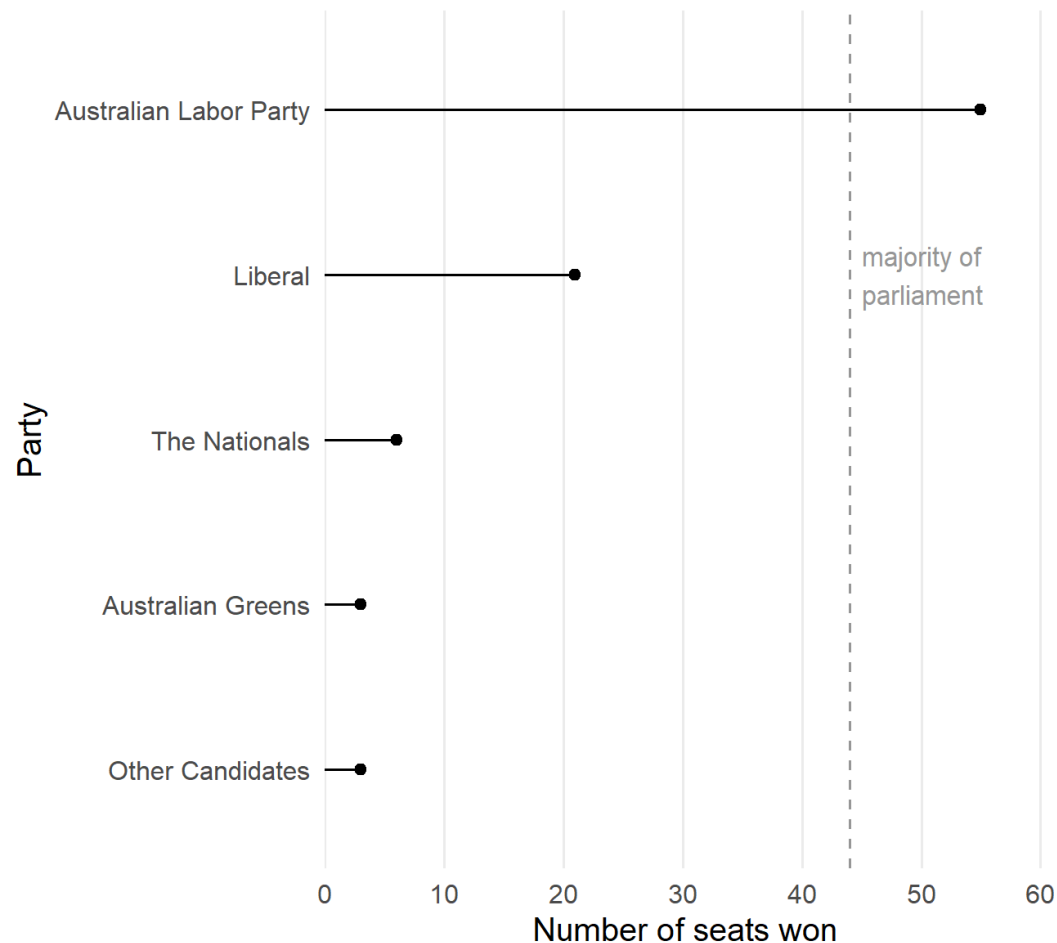
Data source: Victorian Electoral Commission

Adding annotations

```
election_data %>%
  mutate(party = fct_reorder(party, seats_won, .desc = TRUE)) %>%
  ggplot(aes(x = seats_won, y = party)) +
  geom_segment(aes(xend = 0, yend = party)) +
  geom_point() +
  geom_vline(xintercept = 44, linetype = "dashed",
            colour = "grey60") +
  annotate("text", label = "majority of\nparliament",
         x = 45, y = 4,
         hjust = 0, size = 9 / .pt, colour = "grey60") +
  scale_x_continuous(expand = expansion(mult = c(0, 0.1))) +
  scale_y_discrete(limits = rev) +
  labs(x = "Number of seats won",
       y = "Party",
       title = "Victorian state election 2018 lower house results",
       caption = "Data source: Victorian Electoral Commission") +
  theme_minimal() +
  theme(plot.title.position = "plot",
        panel.grid.minor.x = element_blank(),
        panel.grid.major.y = element_blank())
```

The `size` parameter for `geom_text()` is given in millimetres. Dividing by the special value `.pt` allows us to specify a size in points, as is more conventional for text.

Victorian state election 2018 lower house results

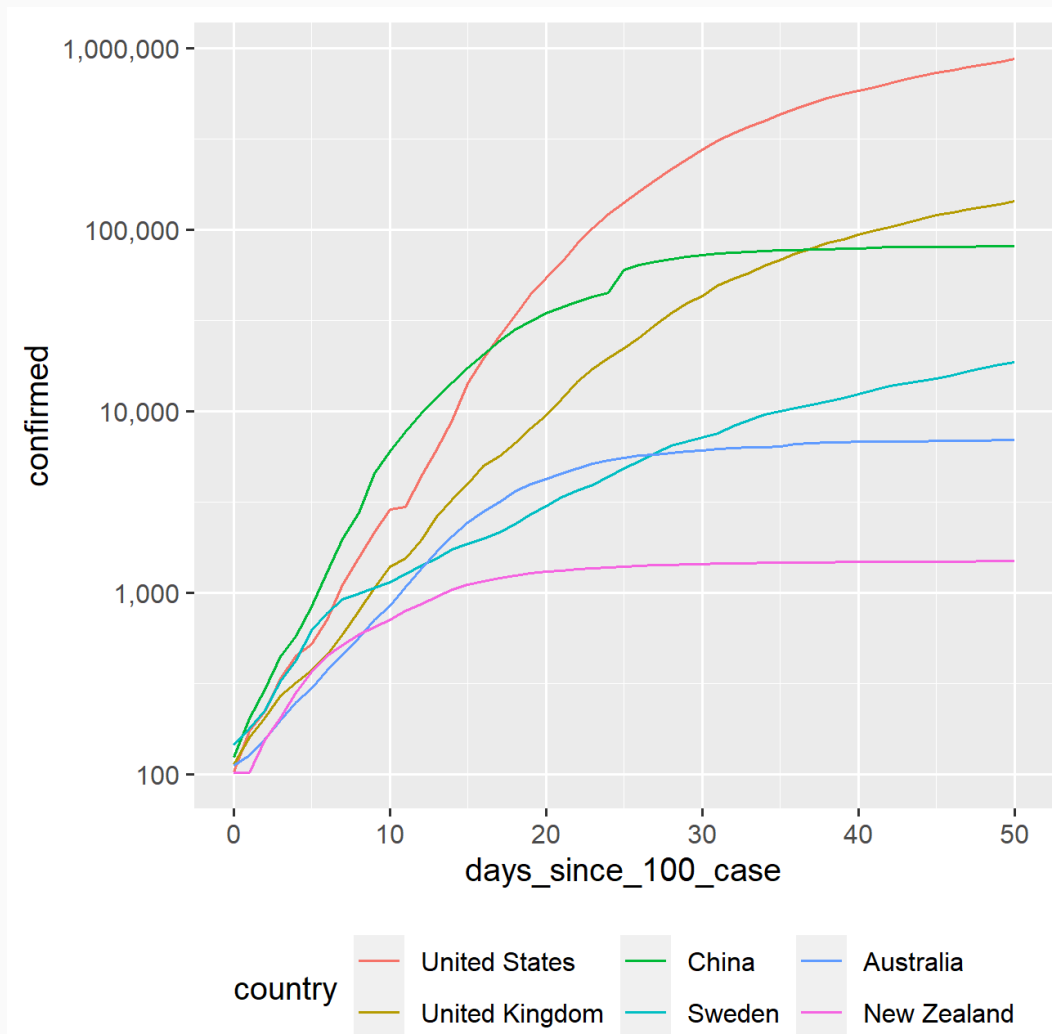


Data source: Victorian Electoral Commission

Extension: Direct annotations to replace a legend

```
ggplot(covid_by_country_50days,  
       aes(x = days_since_100_case, y = confirmed,  
           colour = country)) +  
  geom_line() +  
  scale_y_log10(labels = scales::label_comma()) +  
  theme(legend.position = "bottom")
```

Old mate is back again.



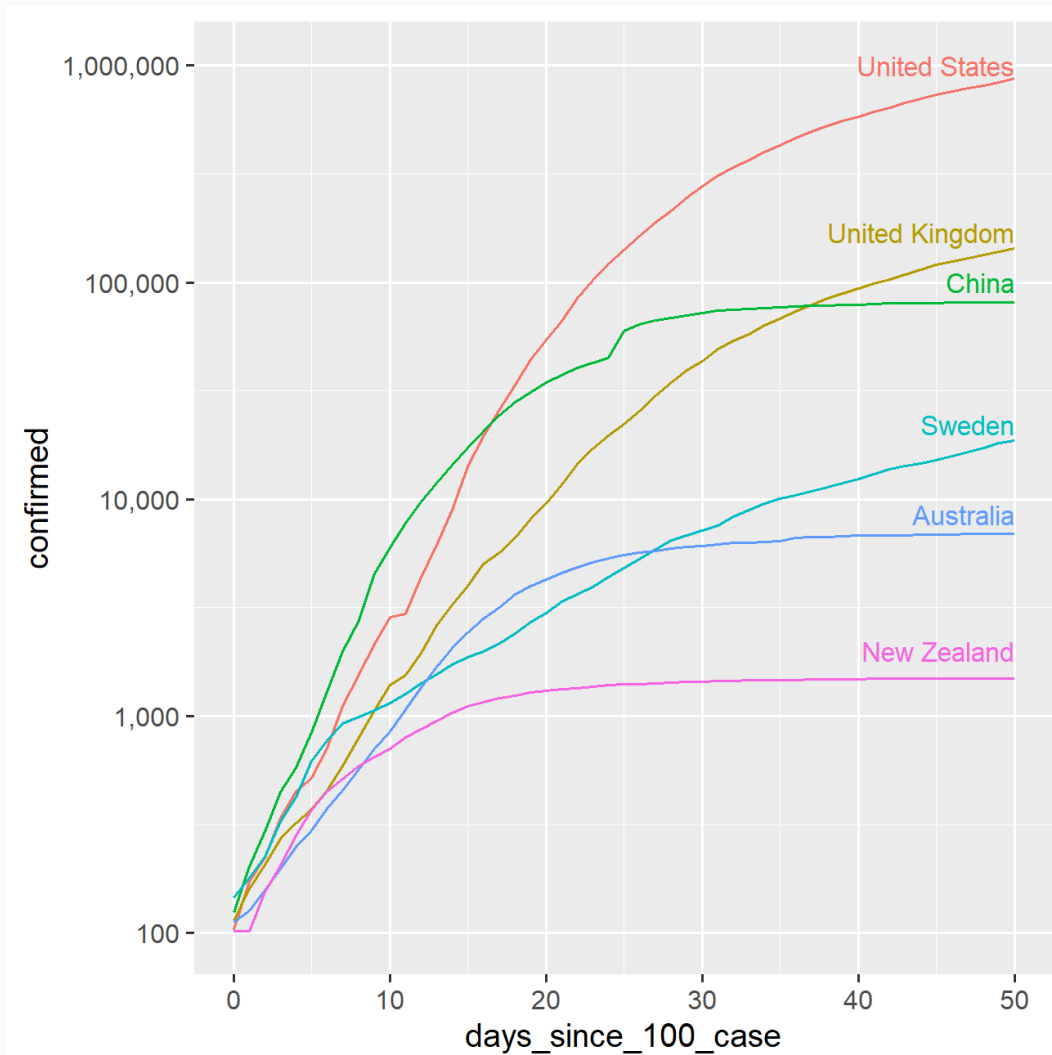
Extension: Direct annotations to replace a legend

```
covid_country_labels <- tribble(
  ~country,      ~days, ~confirmed,
  "United States", 50,    1000000,
  "United Kingdom", 50,    170000,
  "China",         50,    100000,
  "Sweden",        50,    22000,
  "Australia",     50,    8500,
  "New Zealand",   50,    2000,
)

ggplot(covid_by_country_50days,
  aes(x = days_since_100_case, y = confirmed,
    colour = country)) +
  geom_line() +
  geom_text(aes(x = days, y = confirmed, label = country),
    data = covid_country_labels,
    hjust = 1, vjust = 0.5,
    size = 9 / .pt) +
  scale_y_log10(labels = scales::label_comma()) +
  theme(legend.position = "off")
```

It is difficult to label lines close together in this way!

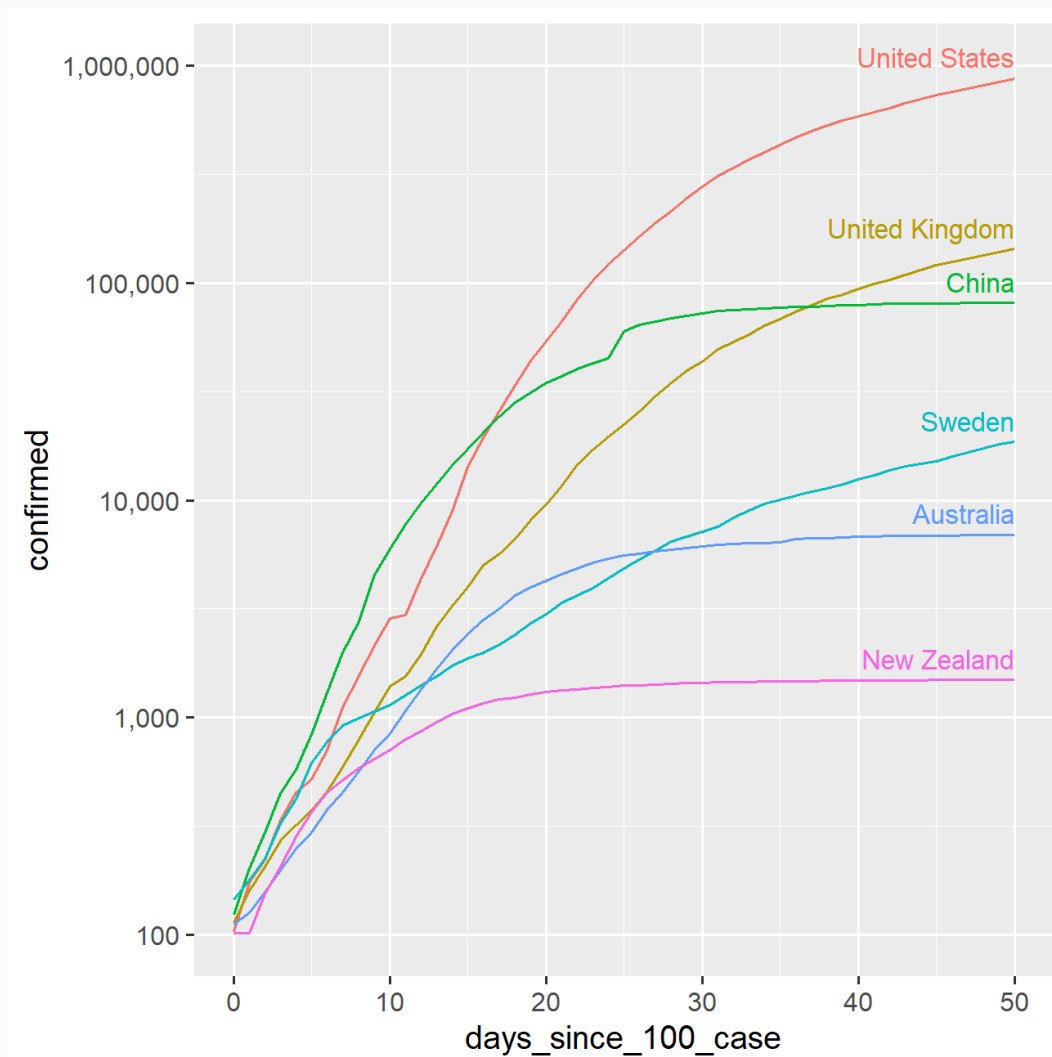
Here I have manually specified a location for each label.



Extension: Direct annotations to replace a legend

```
covid_country_labels <- covid_by_country_50days %>%
  group_by(country) %>%
  filter(days_since_100_case == max(days_since_100_case)) %>%
  ungroup()
ggplot(covid_by_country_50days,
  aes(x = days_since_100_case, y = confirmed,
    colour = country)) +
  geom_line() +
  geom_text(aes(x = days_since_100_case, y = confirmed,
    label = country),
    data = covid_country_labels,
    hjust = 1, vjust = 0, nudge_y = 0.05,
    size = 9 / .pt) +
  scale_y_log10(labels = scales::label_comma()) +
  theme(legend.position = "off")
```

This time we've used `group_by()` and `filter()` to extract the location of the rightmost point (highest value of `days`) and placed the labels there. The `nudge_y` option to `geom_text()` offsets the label slightly from the exact location given.



Controlling the size of your plots

You can control the appearance of the plots in R Markdown using the `fig.width`, `fig.height` and `dpi` chunk options. Plot sizes are given in inches. An appropriate size depends on where your plot will be displayed. For example a 10 inch wide plot may look fine full-screen on your computer; for printed output, 4 to 6 inches wide is likely to work better.

`dpi` (dots per inch) controls the size in pixels of the rendered image. Higher `dpi` will result in greater magnification of the plot, until the maximum width of the column is reached (910 pixels in default R Markdown).

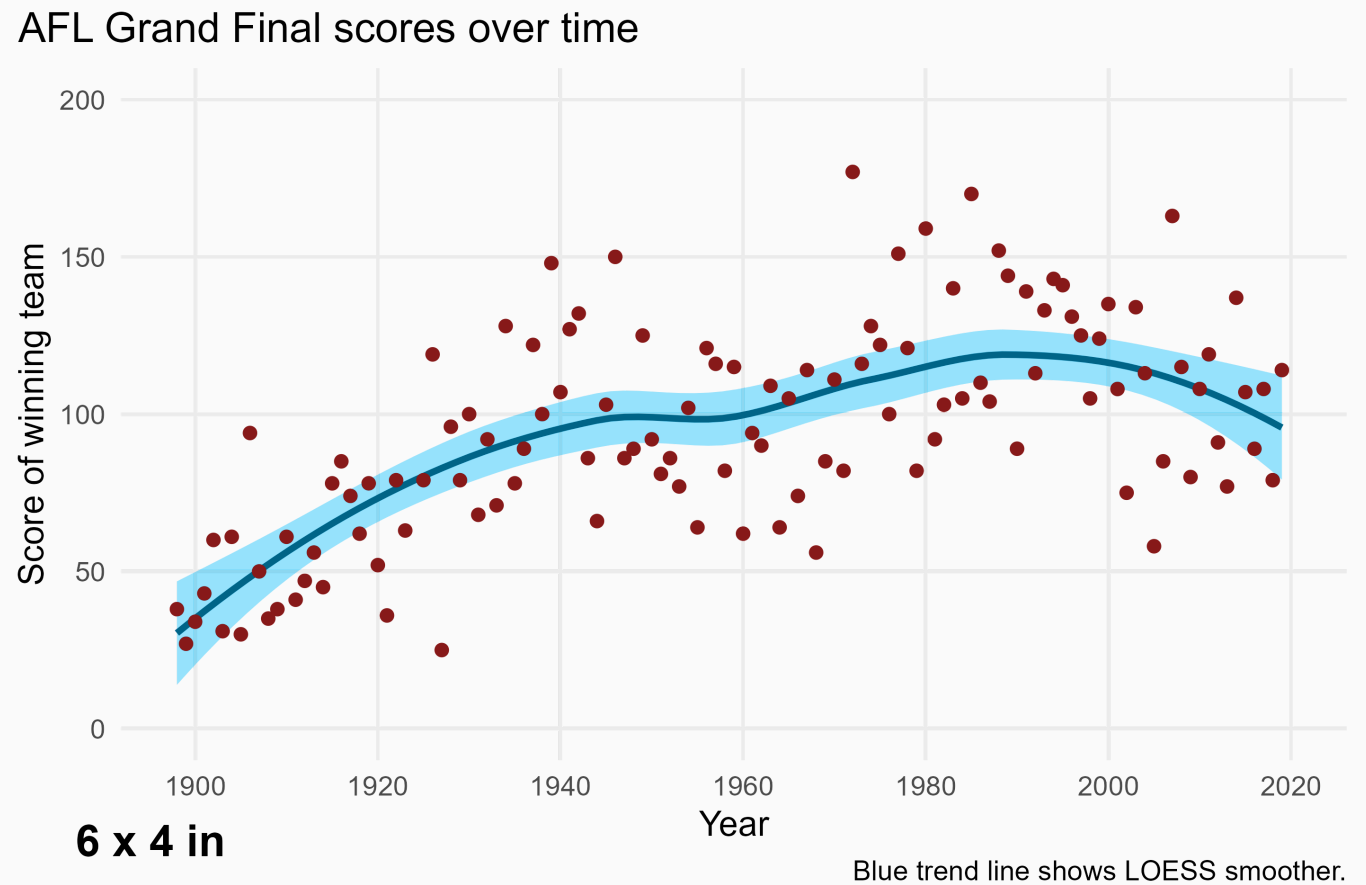
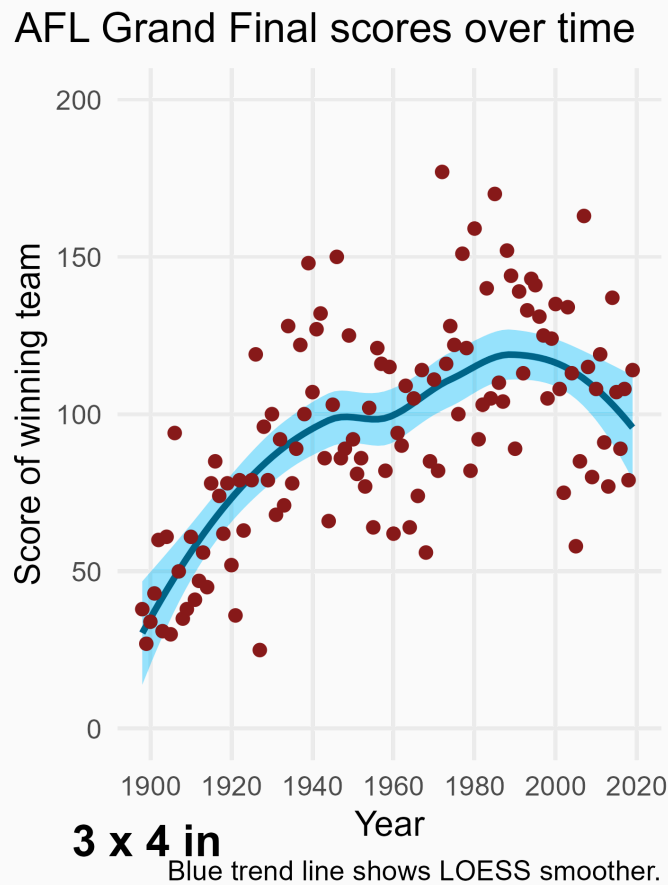
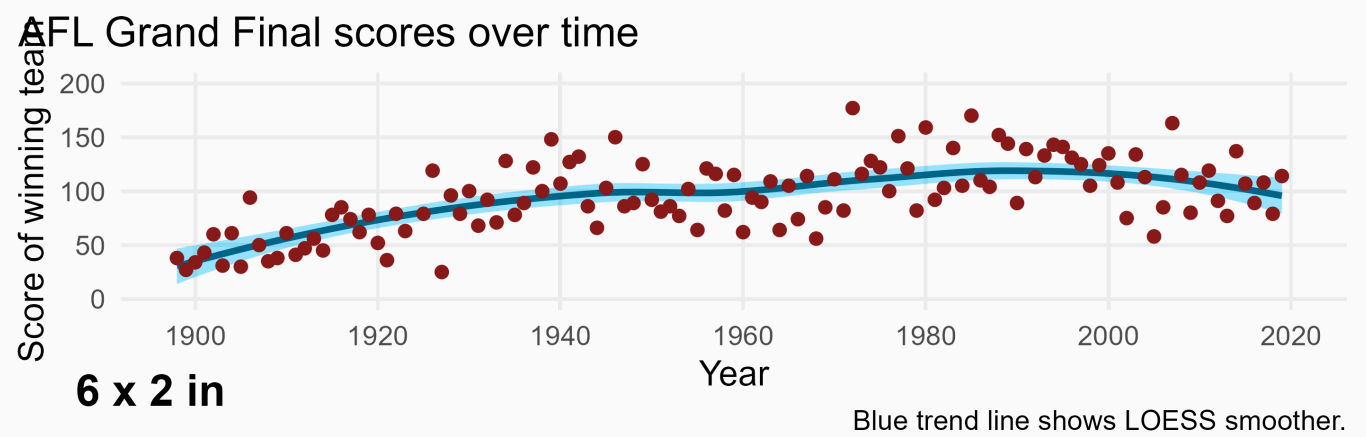
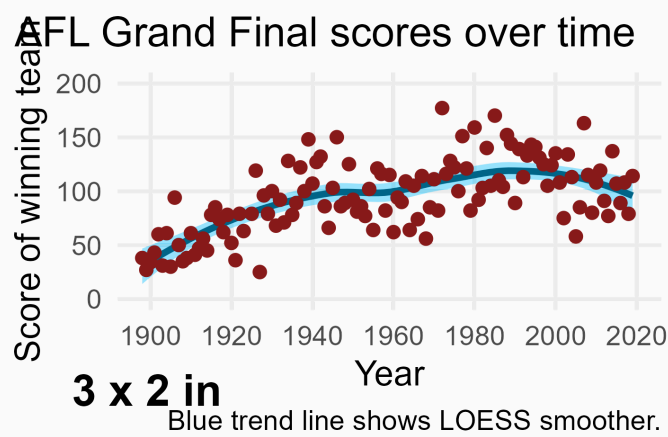
Small plot sizes demand careful consideration of axis labels and title, and possible change of font size (using `theme()`).

Defaults for your document can be set in your setup chunk.

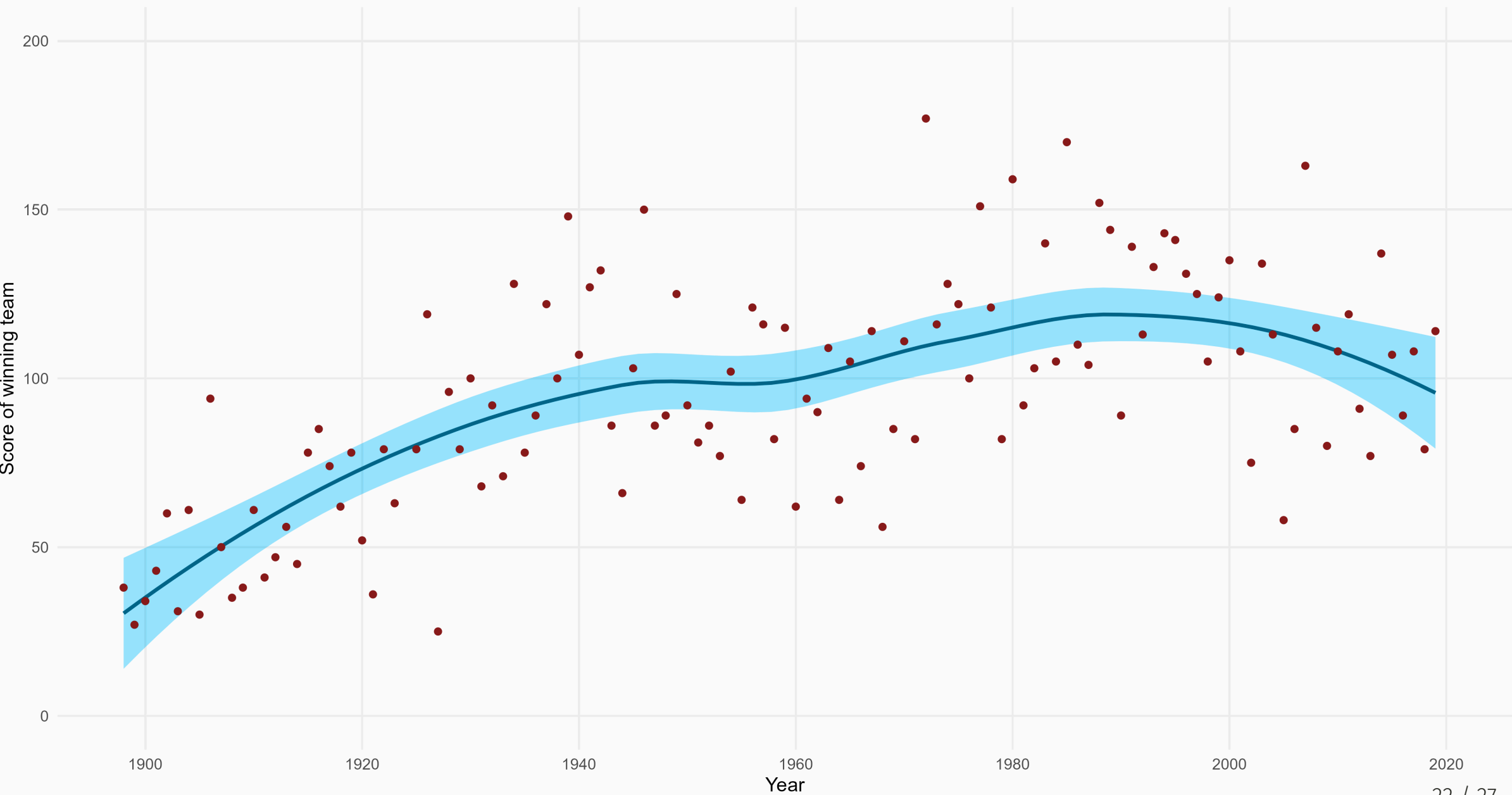
```
```{r setup, include = FALSE}
knitr::opts_chunk$set(
 echo = TRUE,
 comment = NA,
 fig.width = 7,
 fig.height = 4,
 dpi = 130
),,,
```

You can also set the size of a particular plot.

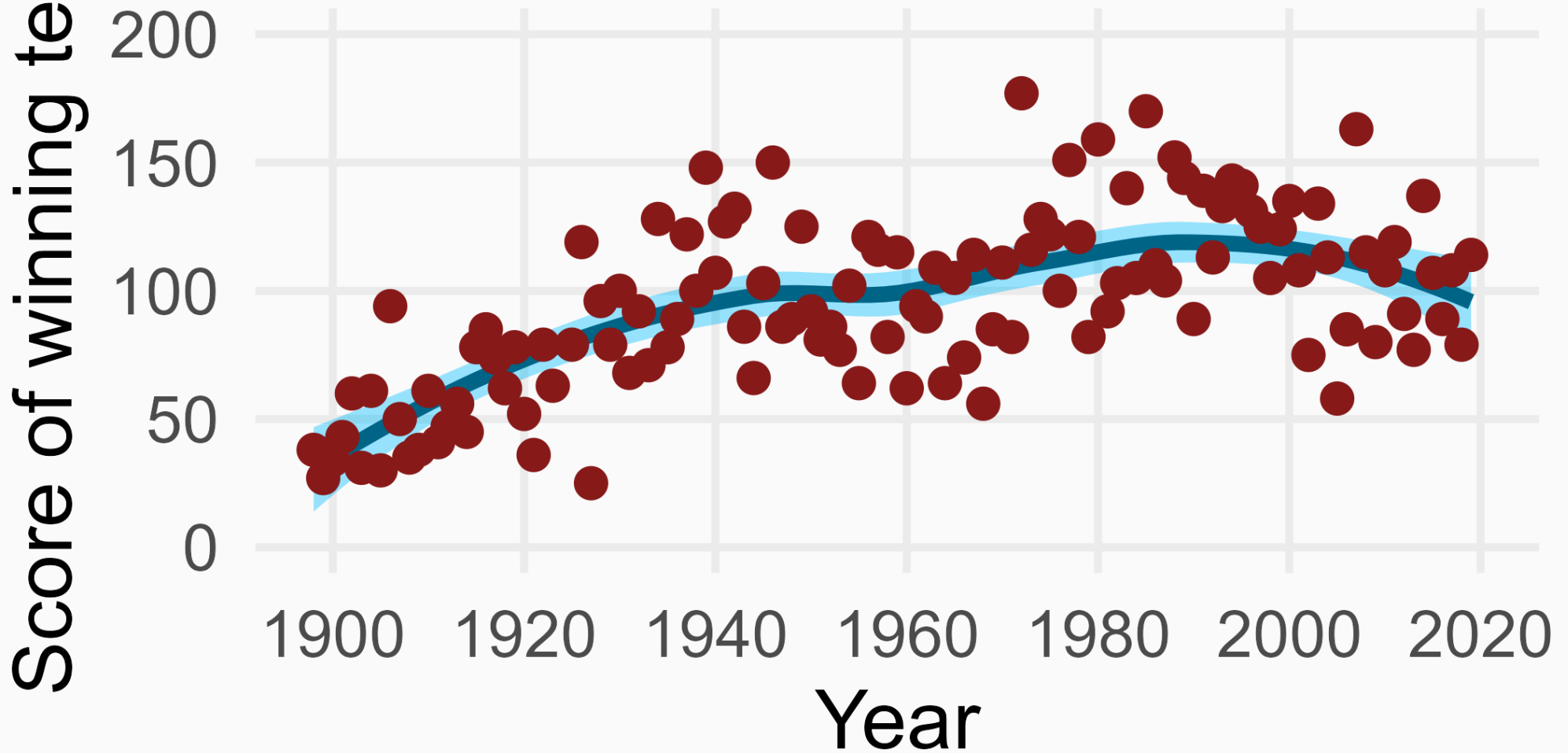
```
```{r, fig.width = 5, fig.height = 2.5}
election_data %>%
  mutate(party = fct_reorder(party, seats
```



AFL Grand Final scores over time



AFL Grand Final scores over time



Blue trend line shows LOESS smoother.

Score of
winning team

200

100

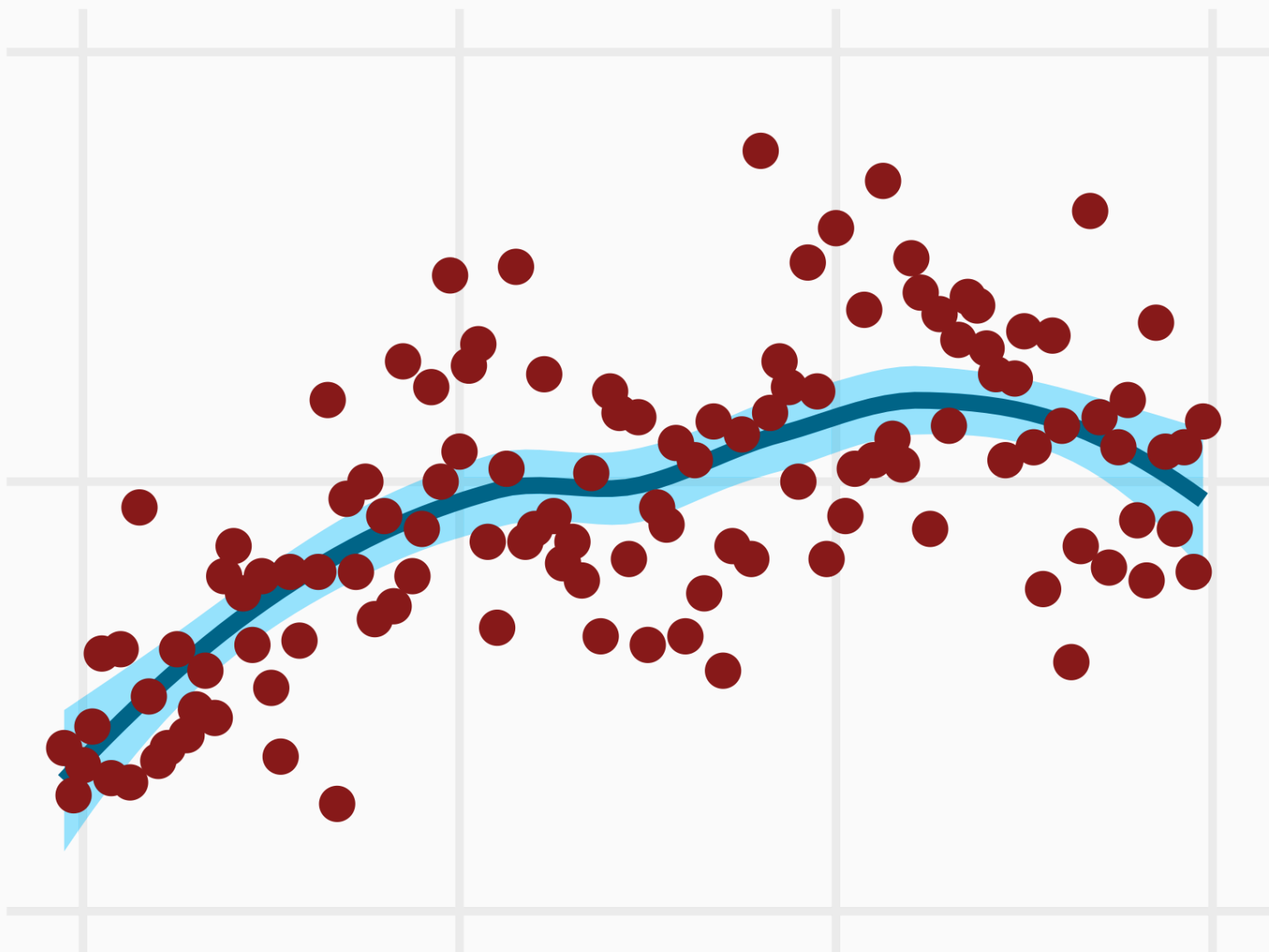
0

1900

1940

1980

2020



Saving your plots to a file

We've kept the easiest part for last!

PNG is the simplest format to include in e.g. Microsoft Word documents. A high `dpi` (e.g. 300 or 600) is needed to get acceptable print quality. Journals usually specify a required dpi for figures, often 600.

```
ggplot(...) +  
  ...  
ggsave("my_favourite_plot.png",  
        width = 6, height = 4, dpi = 600)
```

You can also save as PDF (or many other formats). Note that `dpi` does not apply to PDF since PDF is a scalable document format, not an image format.

```
ggplot(...) +  
  ...  
ggsave("my_favourite_plot.pdf", width = 6, height = 4)
```

Further reading

Fundamentals of Data Visualization

Free online book by Claus O. Wilke

General principles for showing data clearly, ggplot code available for all examples, focus on "why"

<https://serialmentor.com/dataviz/>

Data Visualization: A practical introduction

Free online book by Kieran Healy

Detailed guide to using ggplot, focus on "how"

<http://socviz.co/>

Pies and doughnuts are off the menu

Blog post by Sue Finch

Books by Alberto Cairo:

The Functional Art, The Truthful Art, How Charts Lie.

Books by Edward Tufte, particularly

The Visual Display of Quantitative Information.

The Glamour of Graphics

Talk by Will Chase

General principles for making attractive graphics

<https://resources.rstudio.com/rstudio-conf-2020/the-glamour-of-graphics-william-chase>

Take a Sad Plot and Make It Better

Talk by Alison Hill

Case study of improving a plot

<https://alison.netlify.com/rlm-sad-plot-better/unsw>

ggplot2: Elegant Graphics for Data Analysis

Free online book by Hadley Wickham

A comprehensive reference

<https://ggplot2-book.org/>

Exercise 3.2.