Data Visualisation with `ggplot2`

Communicating your data

Daniela Palleschi

2023-04-13

Table of contents

```
knitr::opts_chunk$set(eval = T, # evaluate chunks
                      echo = T, # 'print code chunk?'
                      message = F, # 'print messages (e.g., warnings)?'
                      error = F, # stop when error encountered
                      warning = F) # don't print warnings
```

```
## play sound if error encountered
### from: https://sejohnston.com/2015/02/24/make-r-beep-when-r-markdown-finishes-or-when-i
options(error = function(){    # Beep on error
  beepr::beep(sound = "wilhelm")
  Sys.sleep(2) #
  }
 )
## and when knitting is complete
.Last <- function() {          # Beep on exiting session
  beepr::beep(sound = "ping")
  Sys.sleep(6) # allow to play for 6 seconds
  }
```

```
# Create references.json file based on the citations in this script
# make sure you have 'bibliography: references.json' in the YAML
rbbt::bbt_update_bib("_data_viz.qmd")
```

Data communication

Load packages and data

```
# load tidyverse
library(tidyverse)

# load data
df_lifetime <- readr::read_csv(here::here("data/tidy_data_lifetime_pilot.csv"),
                               # for special characters
                               locale = readr::locale(encoding = "latin1")
                               ) |>
  mutate_if(is.character,as.factor) |> # all character variables as factor
  filter(type == "critical", # only critical trials
         px != "px3") # this participant had lots of 0's for some reason
```

Tables

- we can create summaries of our data

```
# compute summary
summary_ff <- df_lifetime |>
  filter(region=="verb") |>
  group_by(condition,lifetime,tense) %>%
  summarise(N = n(),
            mean.ff = mean(ff, na.rm = T),
            sd = sd(ff, na.rm = T)) %>%
  # compute standard error, confidence intervals, and lower/upper ci bounds
  mutate(se = sd / sqrt(N),
         ci = qt(1 - (0.05 / 2), N - 1) * se,
         lower.ci = mean.ff - qt(1 - (0.05 / 2), N - 1) * se,
         upper.ci = mean.ff + qt(1 - (0.05 / 2), N - 1) * se)
```

- and print the output with the `kable()` function from the `knitr` package
  - for extra customisation you can also use the `kableExtra` package (e.g., with the `kable_styling()` function)

```
# install.packages("knitr") # if not yet installed
knitr::kable(summary_ff, digits=1,
```

```
            caption = "Table with summmary statistics for first-fixation duration at the
```

Table 1: Table with summmary statistics for first-fixation duration at the verb region

condition	lifetime	tense	N	mean.ff	sd	se	ci	lower.ci	upper.ci
deadPP	dead	PP	140	198.9	57.9	4.9	9.7	189.2	208.6
deadSF	dead	SF	139	194.6	67.9	5.8	11.4	183.2	205.9
livingPP	living	PP	140	194.2	77.3	6.5	12.9	181.3	207.1
livingSF	living	SF	140	186.0	57.6	4.9	9.6	176.4	195.6

Exercise

1. install the `knitr` package (`install.packages("knitr")`)
2. create an object with some summary statistics of the variable `rt`

- call it `summary_rt`

3. use `kable()` from `knitr` to print a table

```
knitr::kable(summary_rt, digits=1,
             caption = "Summary of reaction times (ms) per condition")
```

Table 2: Summary of reaction times (ms) per condition

lifetime	tense	condition	N	mean.rt	sd
dead	PP	deadPP	140	3530.5	2915.8
dead	SF	deadSF	139	1747.0	1153.4
living	PP	livingPP	140	2257.7	1346.3
living	SF	livingSF	140	2578.1	1958.7

Plotting reading times

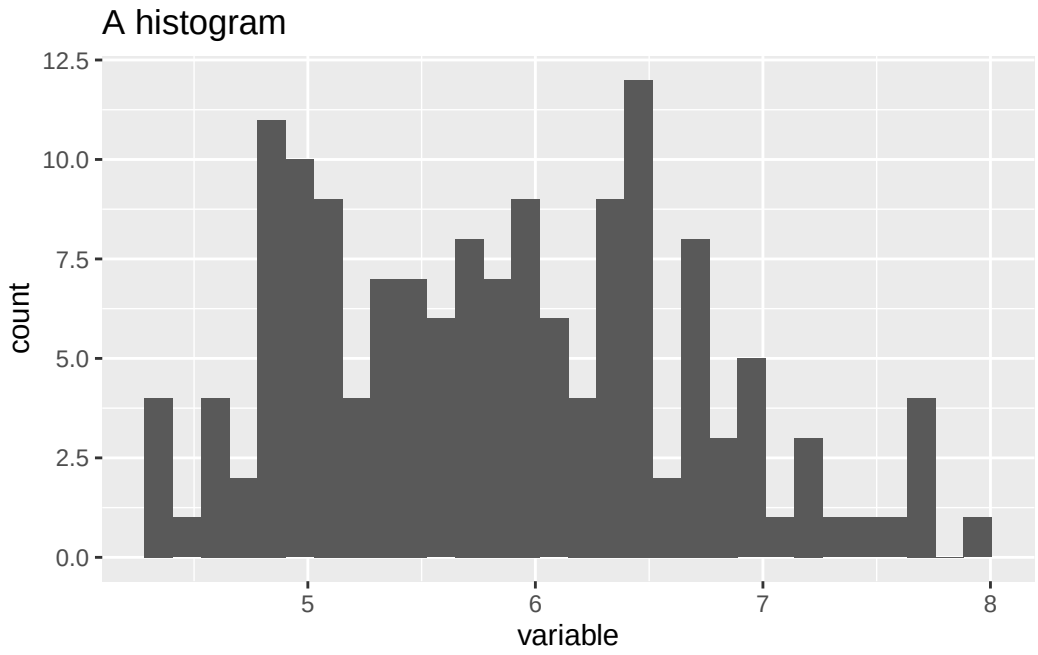
- reading times are (usually) continuous variables
  - as are e.g., reaction times
- they are truncated at 0, meaning they cannot have negative values
  - because of this, they tend to have a *skewed distribution*

Plots with `ggplot2`

- `ggplot2` is part of the tidyverse (like `dplyr`)
  - uses a *layered grammar of graphics*
  - i.e., we build layers

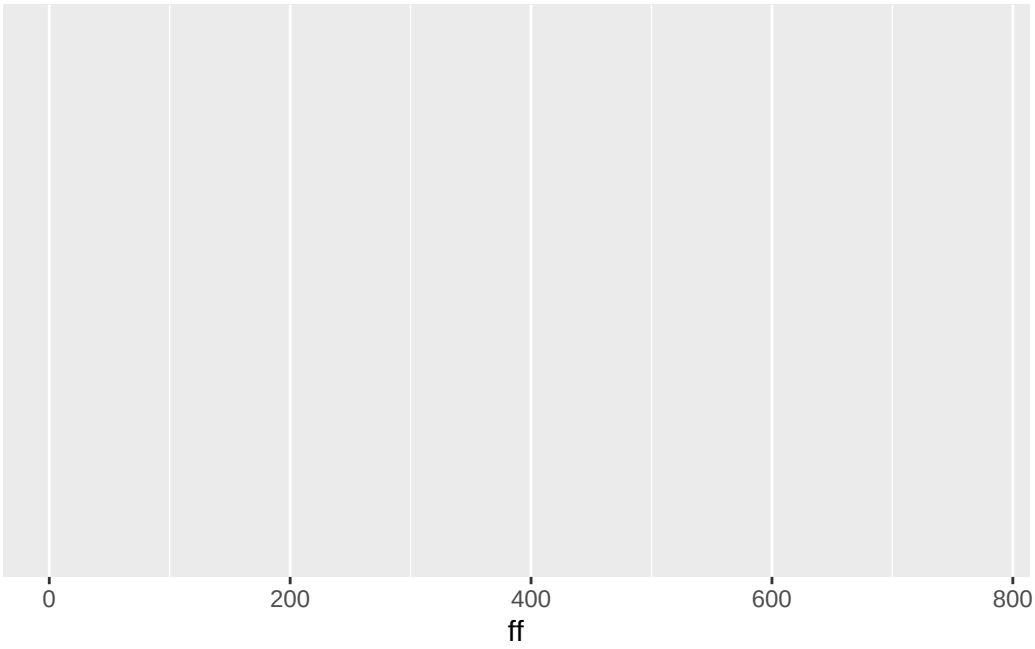
An example: histogram

```
iris |>
  ggplot(aes(x=Sepal.Length)) +
  geom_histogram() +
  labs(title = "A histogram",
       x = "variable")
```



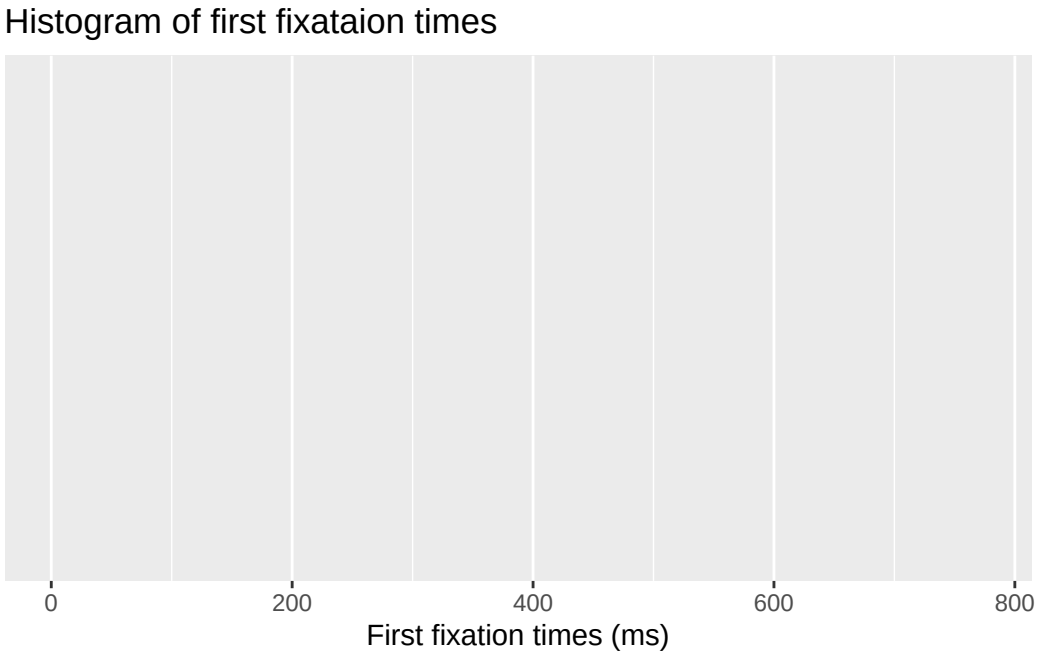
Start layering

```
df_lifetime |> ggplot(aes(ff)) # aes = 'aesthetic'
```



Add labels

```
df_lifetime |> ggplot(aes(ff)) +
  labs(title = "Histogram of first fixataion times",
       x = "First fixation times (ms)")
```



Add

```
df_lifetime |> ggplot(aes(ff)) +
  labs(title = "Histogram of first fixataion times",
       x = "First fixation times (ms)") +
  geom_histogram()
```

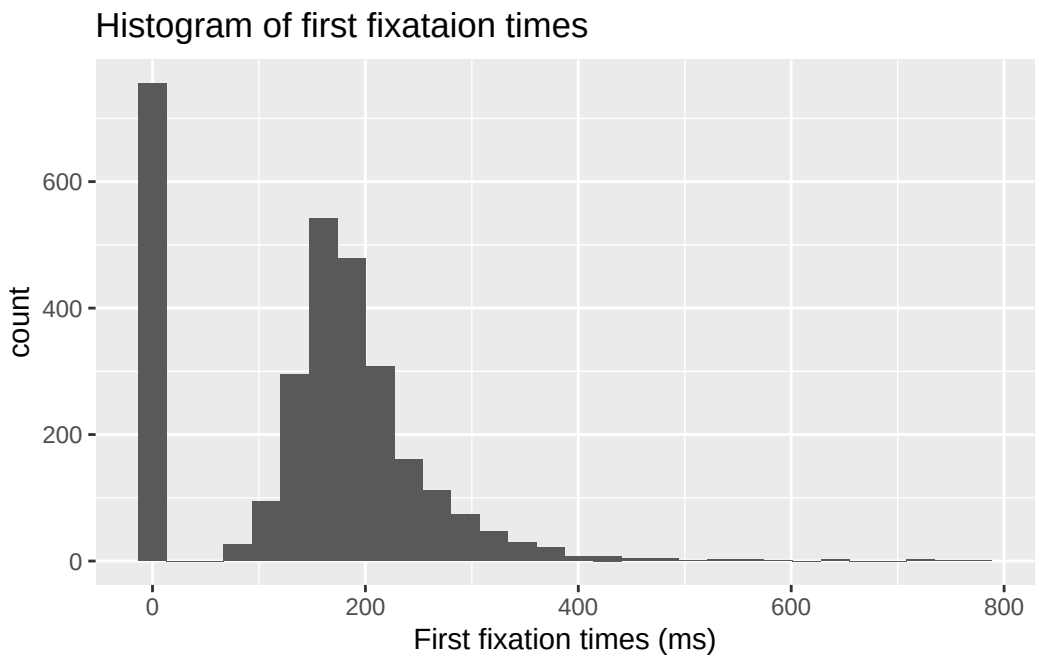


Figure 1: Distribution of first fixation times at the verb region (raw milliseconds)

Add condition

```
df_lifetime |> ggplot(aes(ff, fill = condition)) +
  labs(title = "First fixataion times at the verb region",
       x = "First fixation times (ms)") +
  geom_histogram()
```

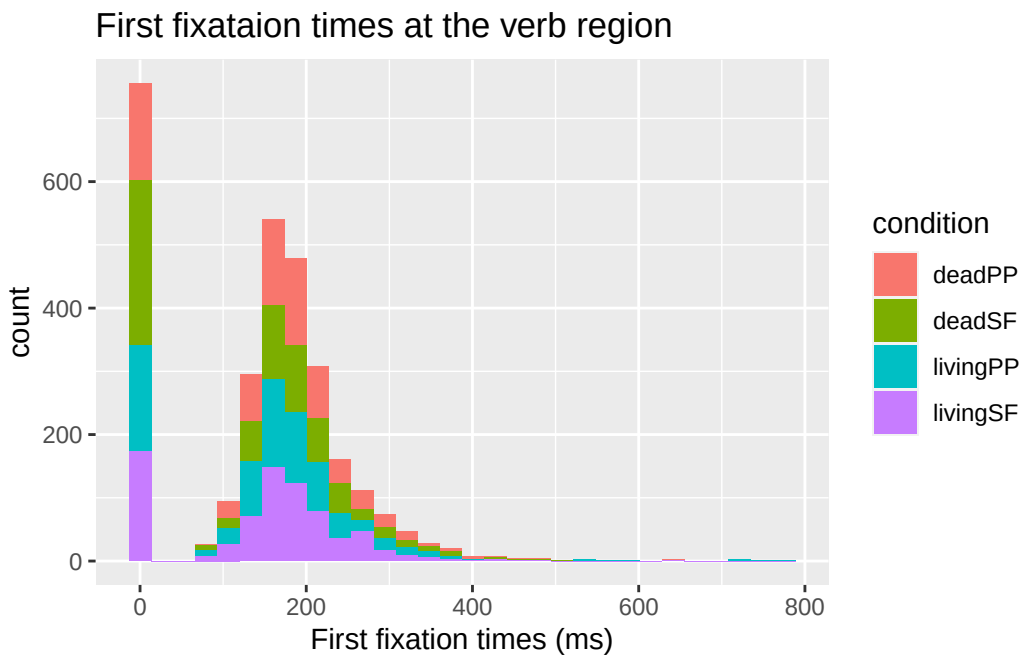


Figure 2: Distribution of first fixation times at the verb region (raw milliseconds)

The colour here is STACKED!! i.e., not layered. Notice the distribution doesn't change from all grey to coloured

Customisation

- we can add arguments to our geoms
  - e.g., transparency: `alpha =` takes a value between 0 to 1
- we can use `theme()` to customise font sizes, legend placement, etc.
- tehre are also popular preset themes, such as `theme_bw()` and `theme_minimal()`

`theme_bw()`

```
df_lifetime |> ggplot(aes(ff, fill = condition)) +
  labs(title = "Histogram of first fixataion times",
       x = "First fixation times (ms)") +
  geom_histogram(alpha=.5) +
  theme_bw()
```

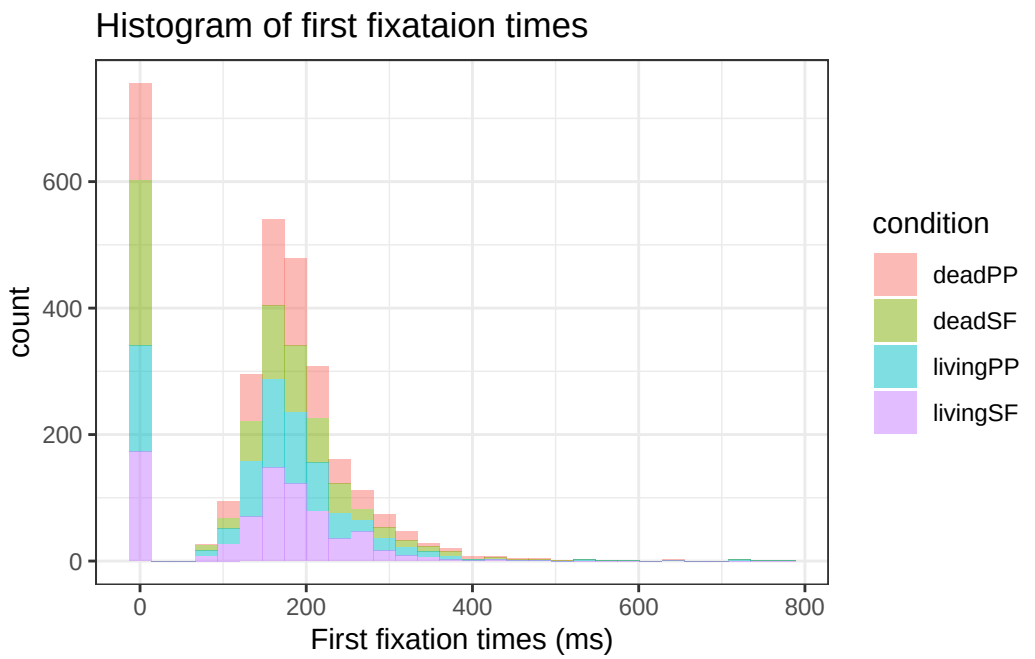


Figure 3: Distribution of first fixation times at the verb region (raw milliseconds).

theme_minimal()

```
df_lifetime |> ggplot(aes(ff, fill = condition)) +
  labs(title = "Histogram of first fixataion times",
       x = "First fixation times (ms)") +
  geom_histogram(alpha=.5) +
  theme_minimal()
```

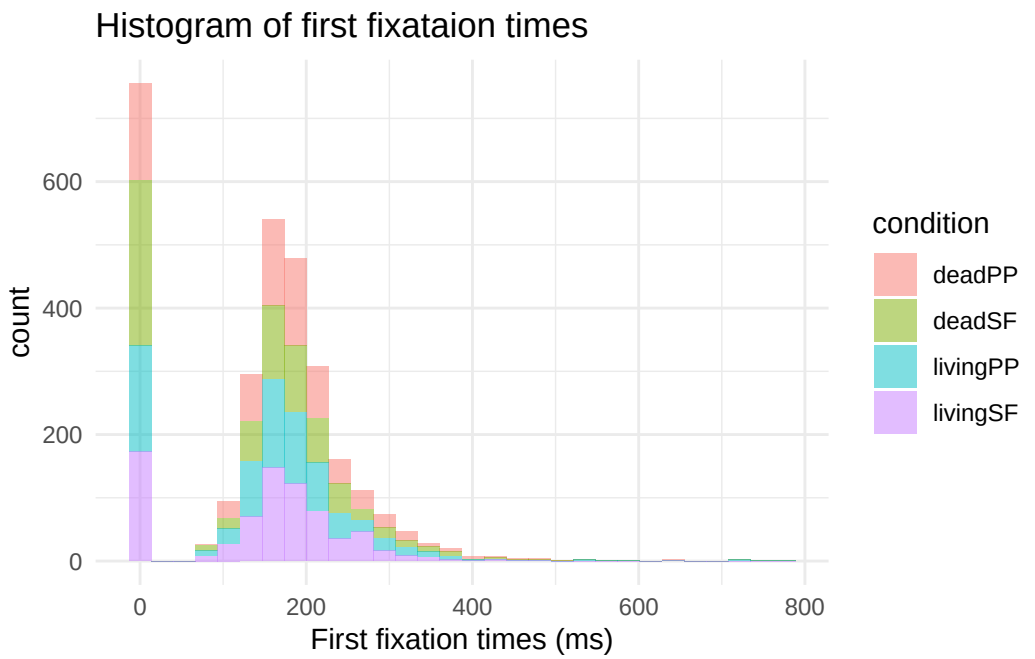
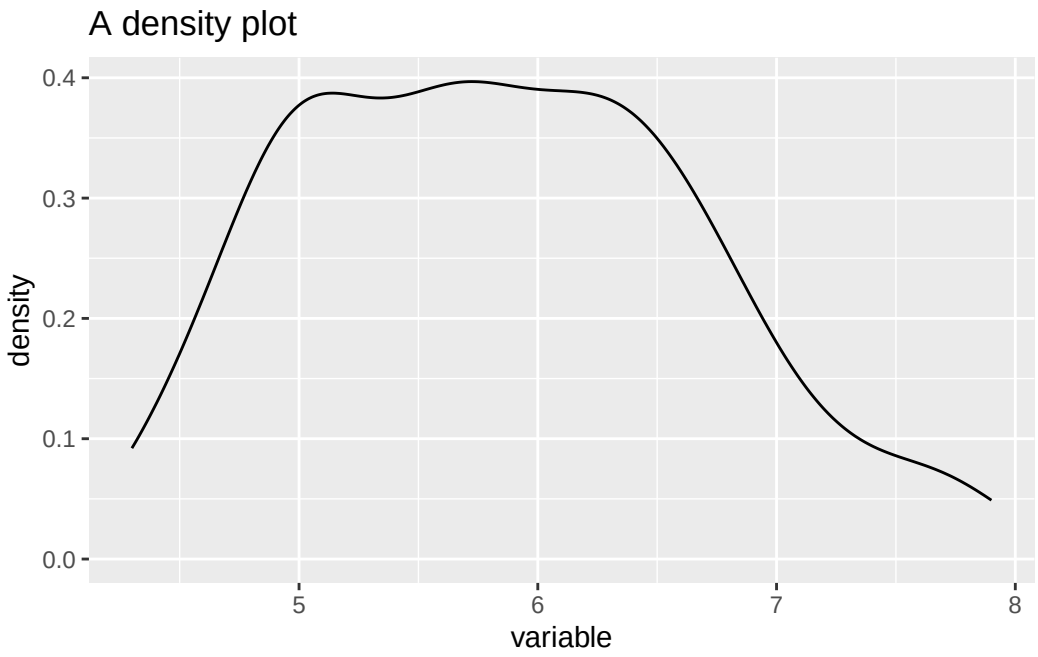


Figure 4: Distribution of first fixation times at the verb region (raw milliseconds).

Distributions

- show the distribution of observations
  - so we can see where the data are clustered
  - and eyeball the shape of the distribution
- we already saw the histogram, which shows the number of observations per variable value
- **density plots** are another useful plot for visualising distributions

```
iris |>
  ggplot(aes(x=Sepal.Length)) +
  geom_density() +
  labs(title = "A density plot",
       x = "variable")
```



Density plots

- below I just replaced `geom_histogram()` with `geom_density()`
  - I also filtered the data to include only values of `ff` above 0
- what is plotted along the y-axis? how does this differ from a histogram?

```
df_lifetime |>
  filter(ff > 0) |>
  ggplot(aes(ff)) +
  labs(title = "Histogram of first fixataion times",
       x = "First fixation times (ms)") +
  geom_density() +
  theme_minimal()
```

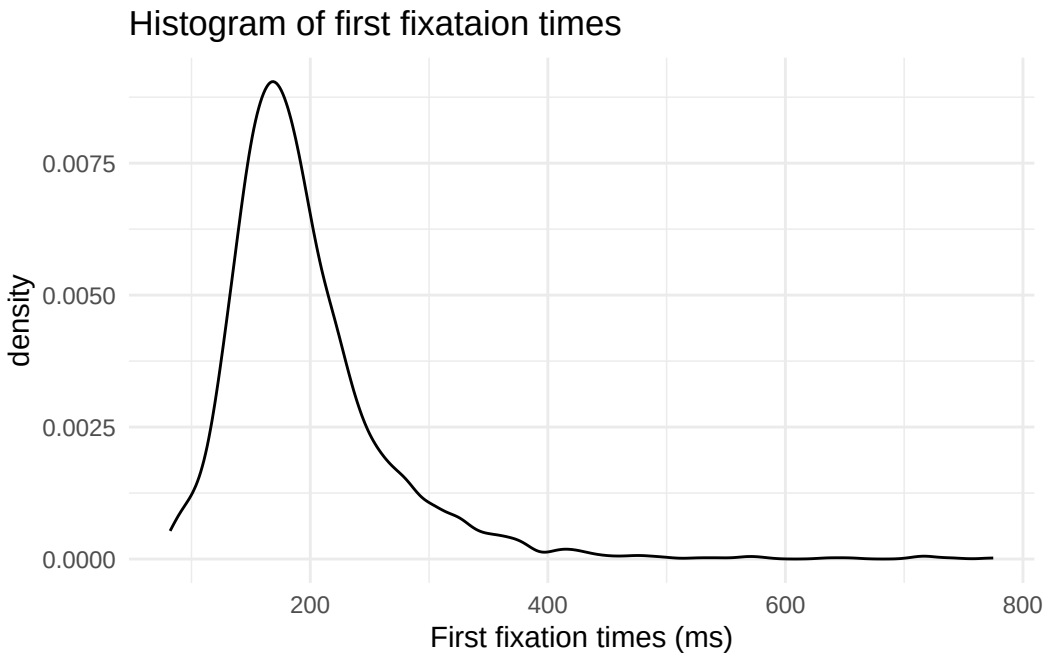


Figure 5: Distribution of first fixation times at the verb region (raw milliseconds).

Grouped density plots

- just like with histograms, we can look at the density plots of different subsets of the data with `aes(fill = )`
  - like region

```
df_lifetime |>
  filter(ff > 0) |>
  ggplot(aes(ff, fill = region)) +
  labs(title = "Histogram of first fixataion times",
       x = "First fixation times (ms)") +
  geom_density(alpha=.5) +
  theme_minimal()
```

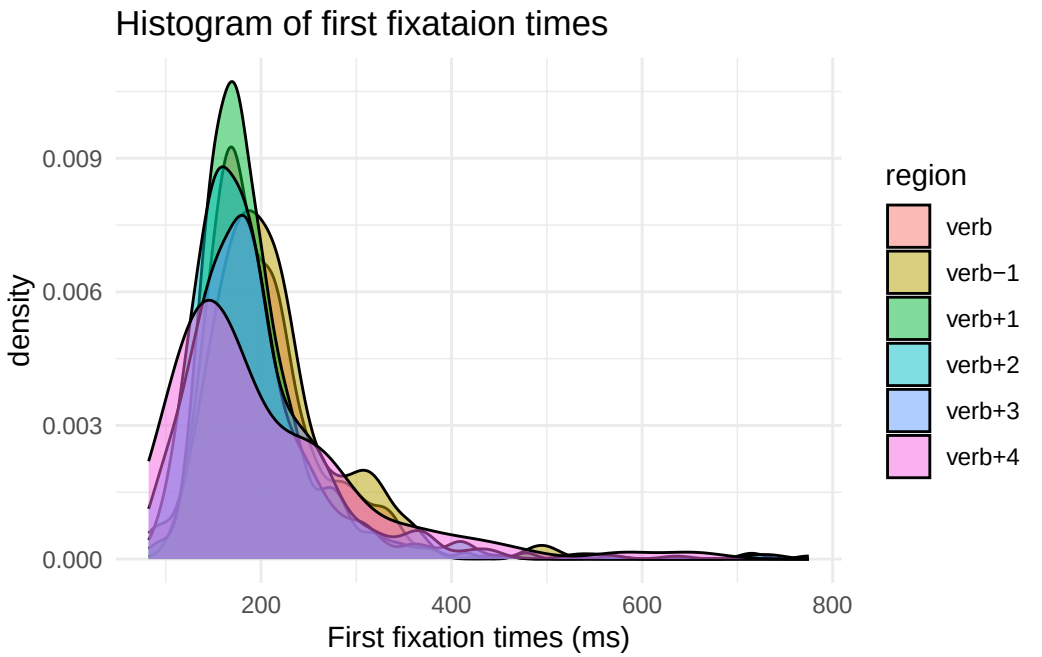


Figure 6: Distribution of first fixation times at the verb region (raw milliseconds).

facet_grid()

- there are a lot of overlapping density curves, let's try to separate them with `facet_grid(x~y)`

```
df_lifetime |>
  filter(ff > 0) |>
  ggplot(aes(ff, fill = region)) +
  facet_grid(.~region) +
  labs(title = "Density plot of first fixataion times by region",
       x = "First fixation times (ms)") +
  geom_density(alpha=.5) +
  theme_bw()
```

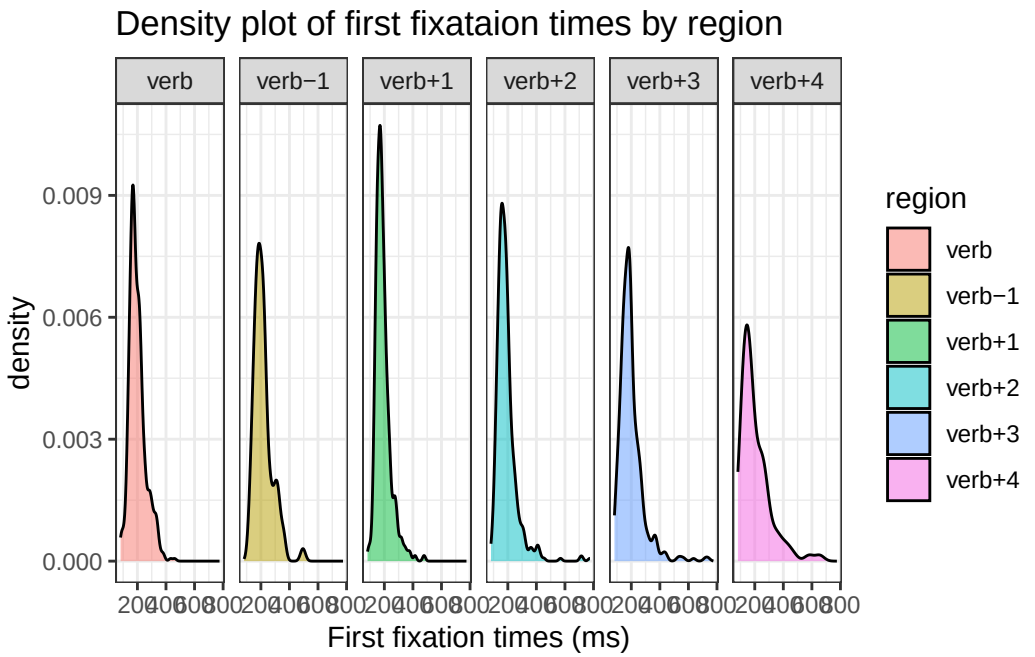


Figure 7: Distribution of first fixation times at the verb region (raw milliseconds).

- how would you describe the density plots of the different regions?

re-ordering factors

- by default, factors will be ordered alphabetically
  - but we don't always want that
  - here, `verb-1` should be before `verb`

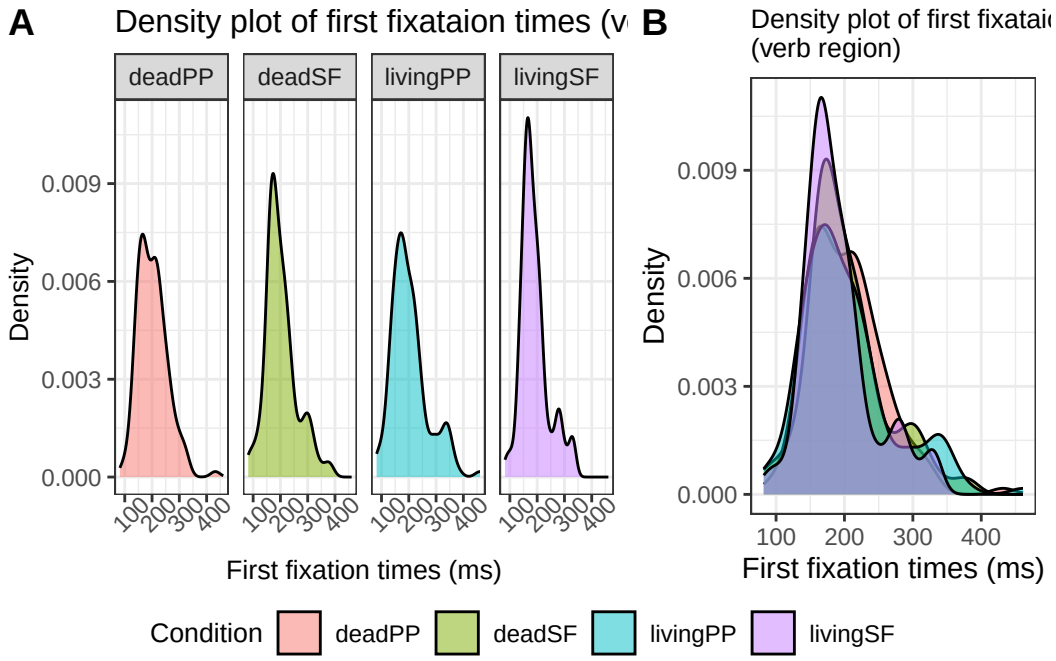
```
df_lifetime <- df_lifetime %>%
  mutate(region = factor(region,
                         levels = c("verb-1","verb","verb+1","verb+2","verb+3","verb+4")))

summary(df_lifetime$region)
```

```
verb-1   verb verb+1 verb+2 verb+3 verb+4
   559    559    559    559    559    182
```

Exercise

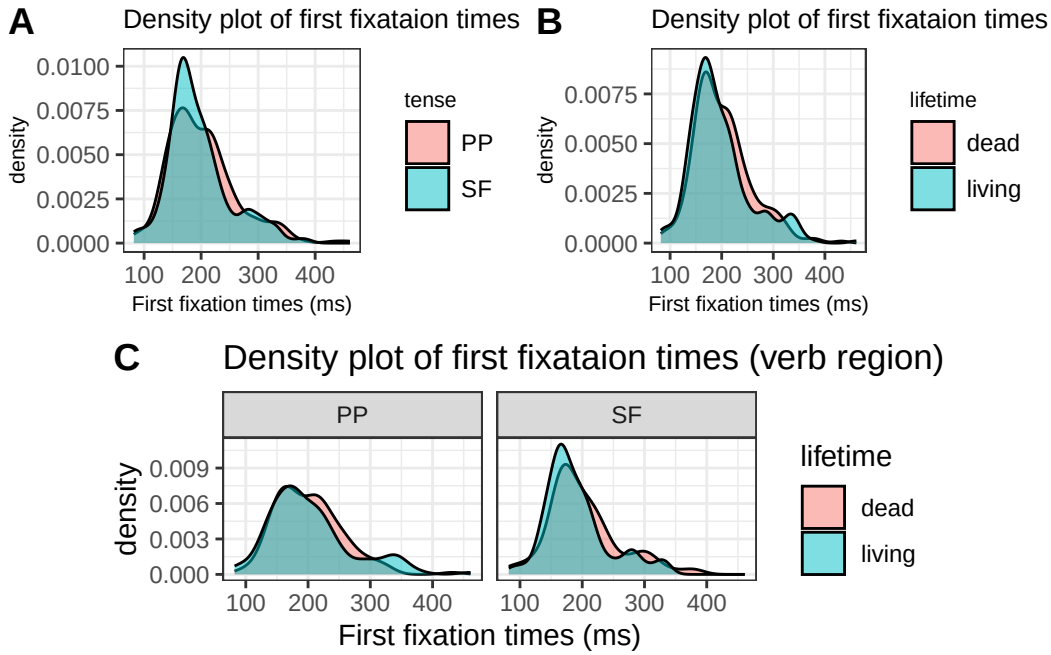
1. create a density plot with the fill colour set to `condition`, but:

- subset the data to only include the verb region
- you can decide if you want to use facets or to have the density curves overlayed
- your plot should look something like A or B:



Extra exercise

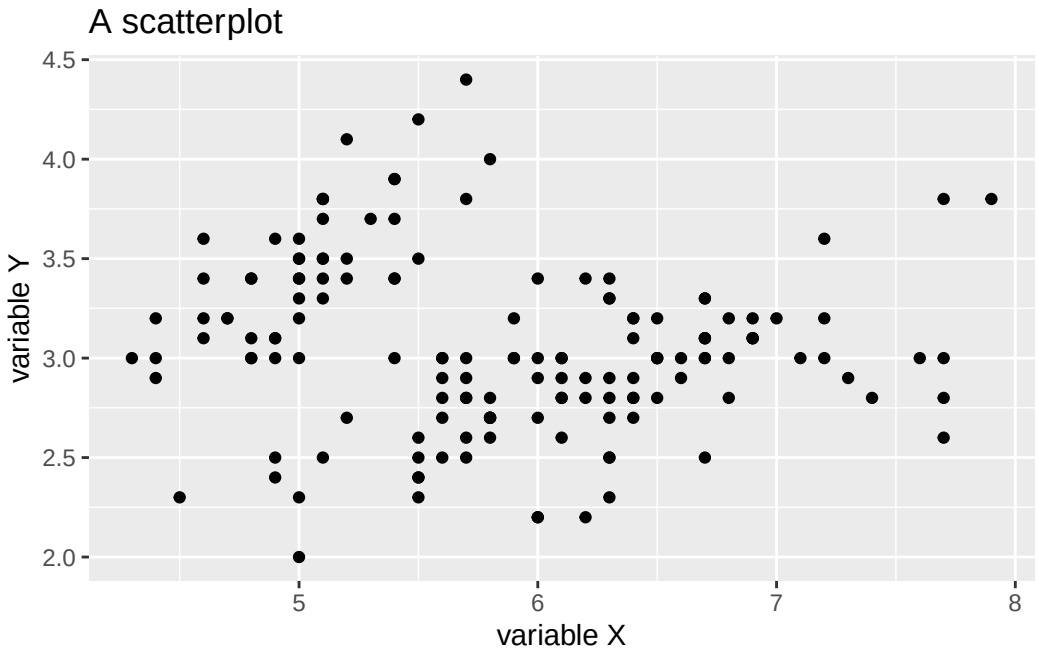
2. Can you produce these plots?



Scatterplots

- histograms and density plots plot a **single variable** along the x-axis
  - in most other plots the dependent (measure) variable is plotted along the y-axis by convention
- scatterplots plot the relationship between **two variables**

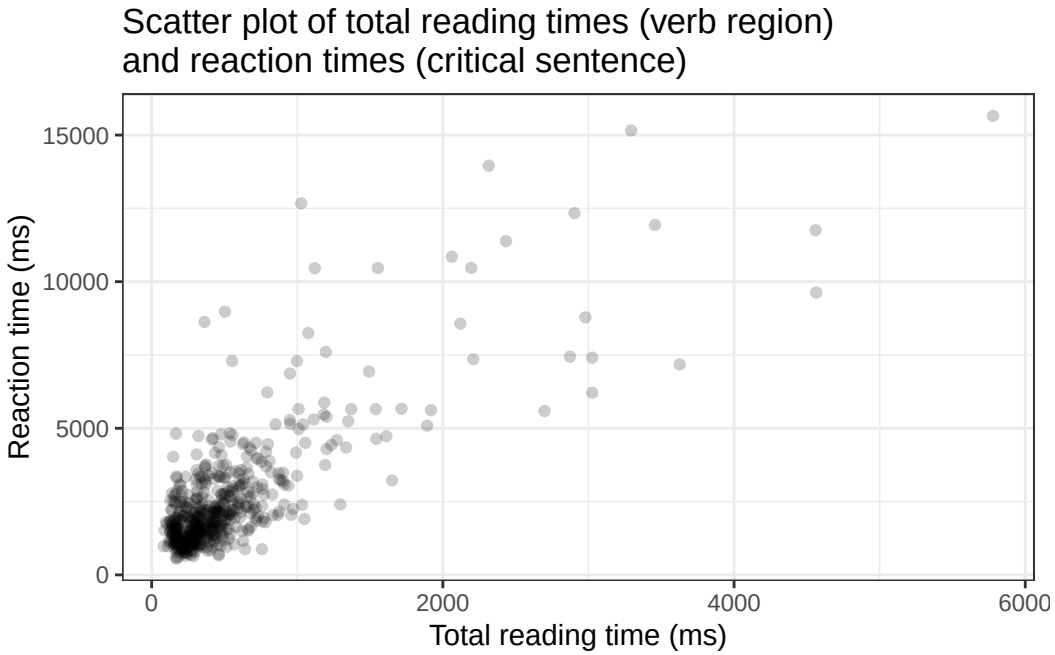
```
iris |>
  ggplot(aes(x=Sepal.Length, y=Sepal.Width)) +
  geom_point() +
  labs(title = "A scatterplot",
       x = "variable X",
       y = "variable Y")
```



Scatterplots

- the figure below plots total reading times (verb region) to the verb region (x-axis) and reaction times to the critical sentence (y-axis)
  - what does each point represent?
  - how would you describe the relationship between the two variables?

```
df_lifetime |>
  filter(ff > 0,
         region == "verb") |>
  ggplot(aes(x = tt, y = rt)) +
  labs(title = "Scatter plot of total reading times (verb region)
and reaction times (critical sentence)",
       x = "Total reading time (ms)",
       y = "Reaction time (ms)") +
  geom_point(alpha = .2) +
  theme_bw()
```



Exercise

1. Generate a scatterplot of total reading times and reaction times, with:
    - colour and shape set to condition
    - tip: these both belong in `aes()`
2. What information does this plot suggest?

Summary statistics

- measures of location: mean, median, mode
- measures of spread: (interquartile) range, standard deviation

Boxplots

- boxplots provide information about the distribution of a *continuous* variable
    - but includes information like *median* (dark line) and *quartiles* (box and whiskers)
    - and *outliers* (dots)
- like scatterplots, require x and y variables

– but one of them needs to be **categorical**

```
iris |>
  ggplot(aes(x = Species, y = Sepal.Length)) +
  labs(title = "A scatterplot",
       x = "Categorical variable",
       y = "Continuous variable") +
  geom_boxplot()
```

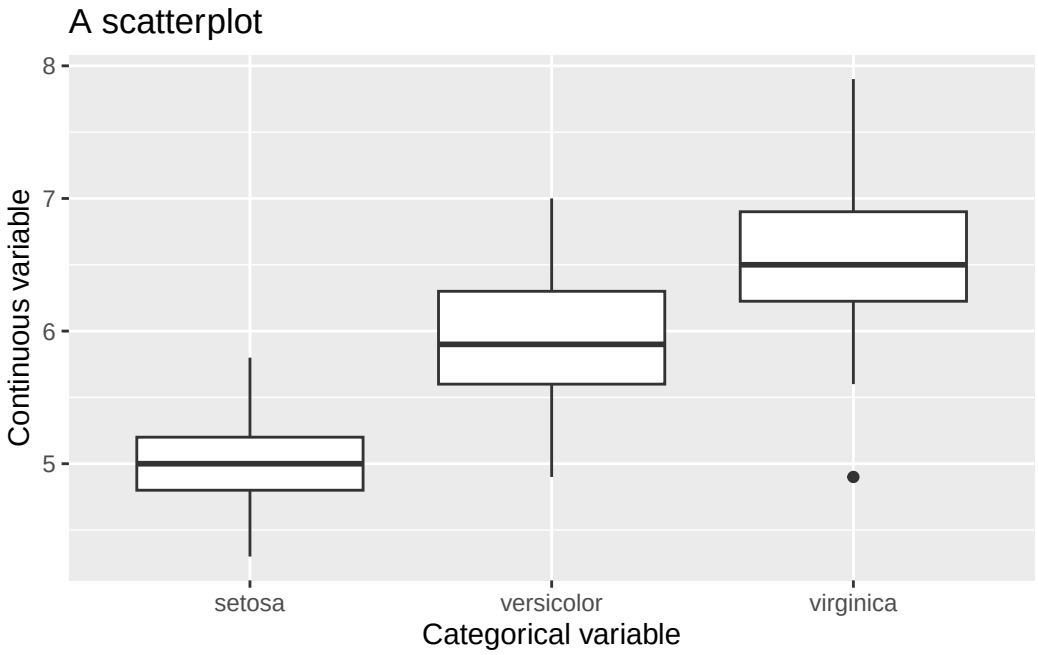


Figure 8: A scatterplot. Median (50th percentile): thick black lines; interquartile range (IQR; 25th and 75th percentile): box limits; minimum (0th percentile) and maximum (100th percentile) excluding outliers: : whiskers; outliers: points

Boxplot explained

Boxplots

- let's change our scatterplot to a boxplot

```
df_lifetime |>
  filter(ff > 0,
```

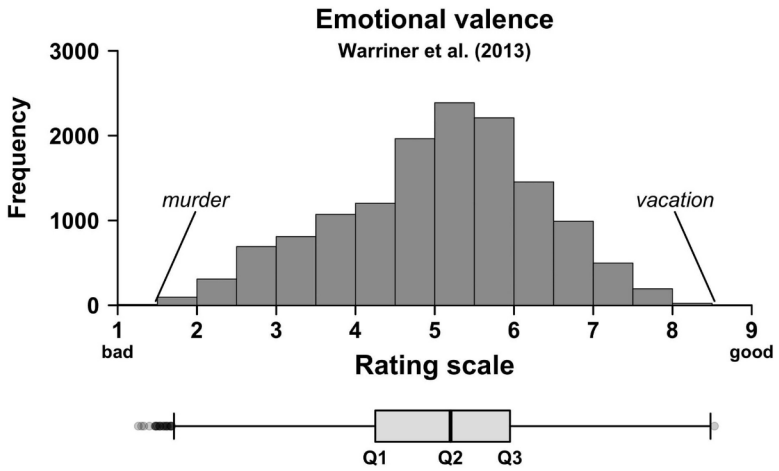
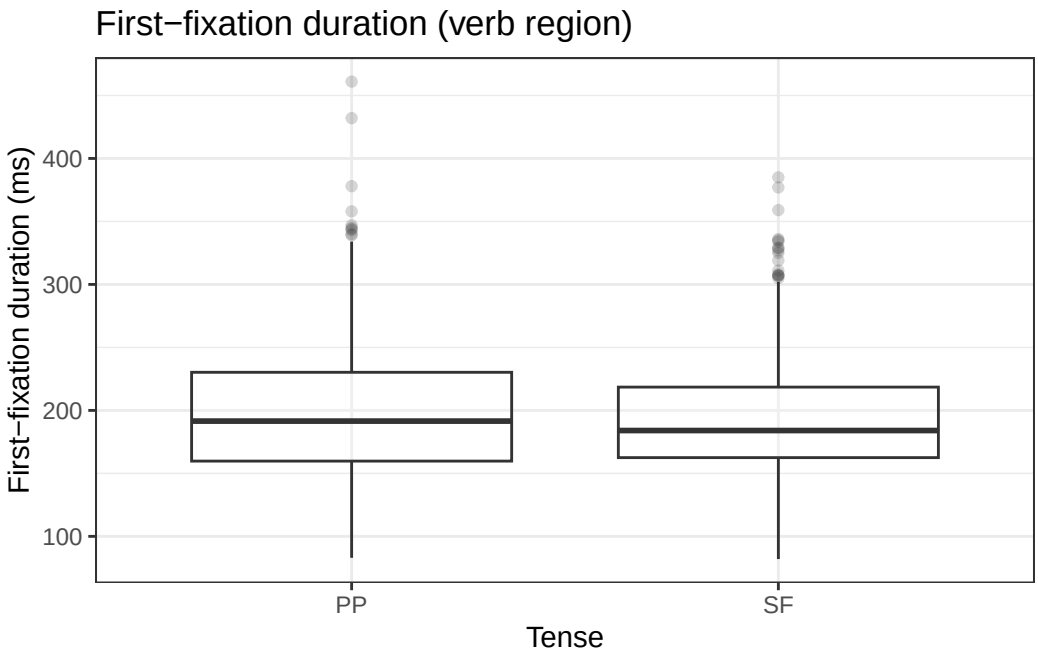


Figure 3.4. A histogram of the emotional valence rating data

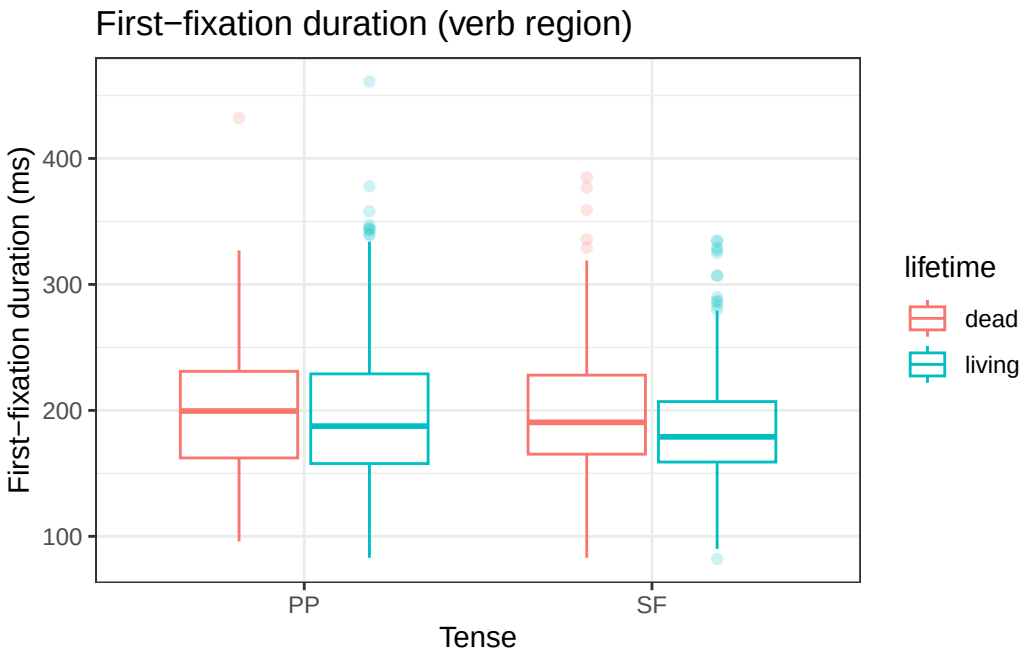
Figure 9: Image source: (**winter_statistics_2019?**) (all rights reserved)

```
        region == "verb") |>
ggplot(aes(x = tense, y = ff)) +
labs(title = "First-fixation duration (verb region)",
     x = "Tense",
     y = "First-fixation duration (ms)") +
geom_boxplot(alpha = .2) +
theme_bw()
```



Grouped boxplots

```
df_lifetime |>
  filter(ff > 0,
         region == "verb") |>
  ggplot(aes(x = tense, y = ff, colour = lifetime)) +
  labs(title = "First-fixation duration (verb region)",
       x = "Tense",
       y = "First-fixation duration (ms)") +
  geom_boxplot(alpha = .2) +
  theme_bw()
```



Exercise

1. Create a group boxplot (x = tense, fill = lifetime) for

- first-pass reading time (verb region)
- regression path duration (verb region)
- total reading time (verb region)
- reaction times (use the `distinct()` verb to have a single observation per participant and per trial)

Violin plots

Violin boxplots

Summary statistics

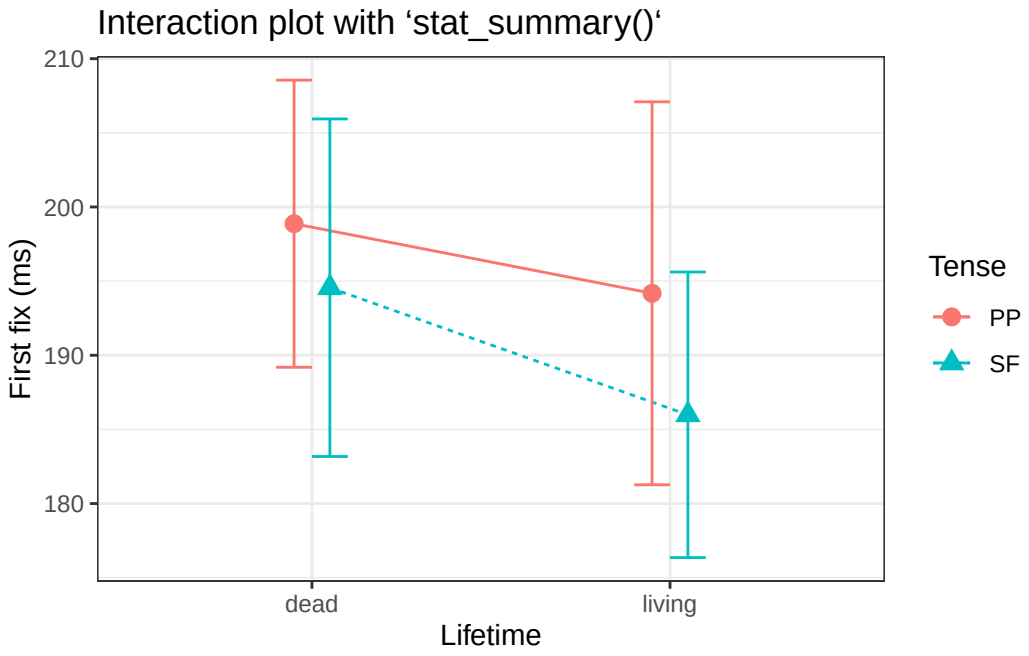
Interaction plots

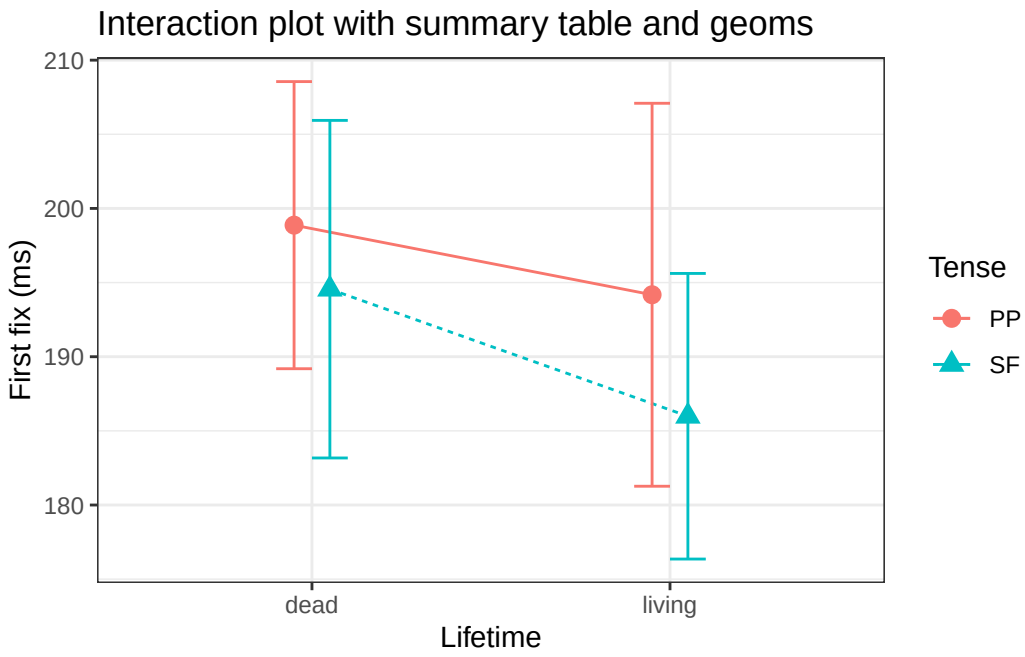
- common for **factorial** designs, i.e., comparing *categorical* predictors
- there are 2 ways of producing them:
  - with your data frame and `stat_summary()`

condition	lifetime	tense	N	mean.ff	sd	se	ci	lower.ci	upper.ci
deadPP	dead	PP	140	198.9	57.9	4.9	9.7	189.2	208.6
deadSF	dead	SF	139	194.6	67.9	5.8	11.4	183.2	205.9
livingPP	living	PP	140	194.2	77.3	6.5	12.9	181.3	207.1
livingSF	living	SF	140	186.0	57.6	4.9	9.6	176.4	195.6

  - or with a summary table and ggplot geoms `geom_point()`, `geom_errorbar()`, and `geom_line()`

- we'll need our summary table to plot an interaction plot





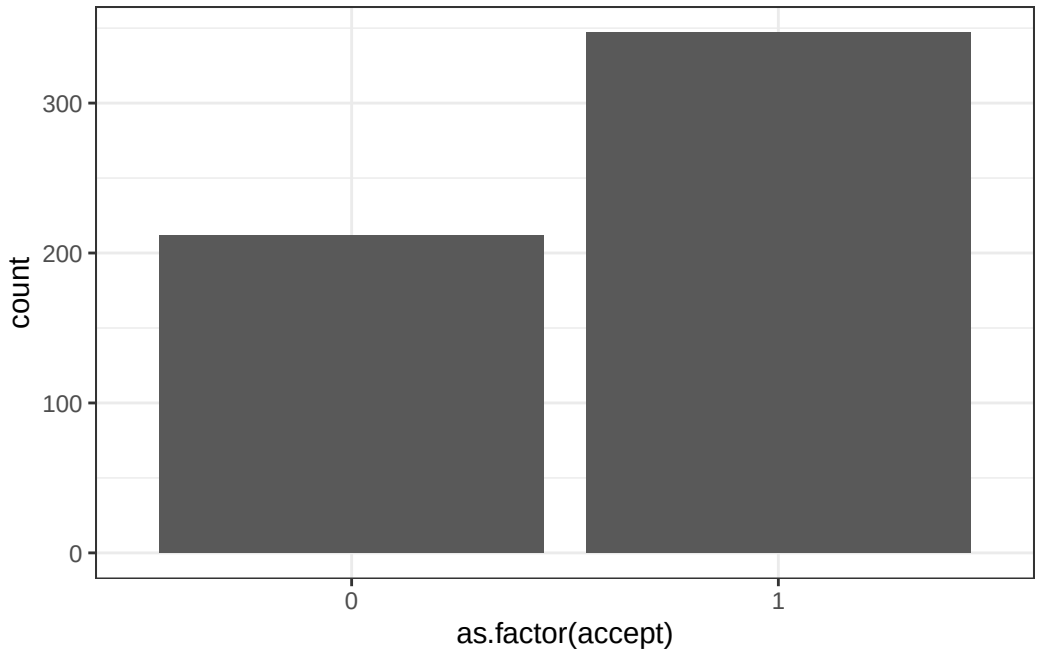
Binomial data

- binomial data are those with 2 categories, for example
  - present, absent
  - yes, no
- in our dataset, each trial ended with a binary naturalness judgement task
  - how might we plot such data?

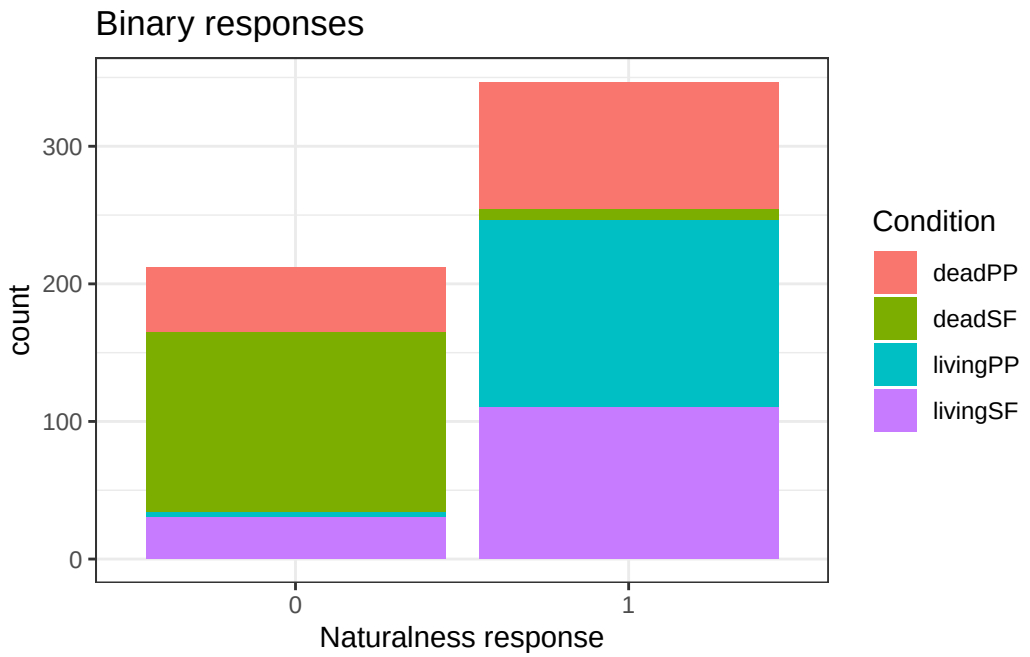
Bar plot

- be sure to read in `accept` as a factor!

```
df_lifetime |>
  distinct(px,trial,.keep_all=T) |>
  ggplot(aes(x = as.factor(accept) )) +
  geom_bar() +
  theme_bw()
```

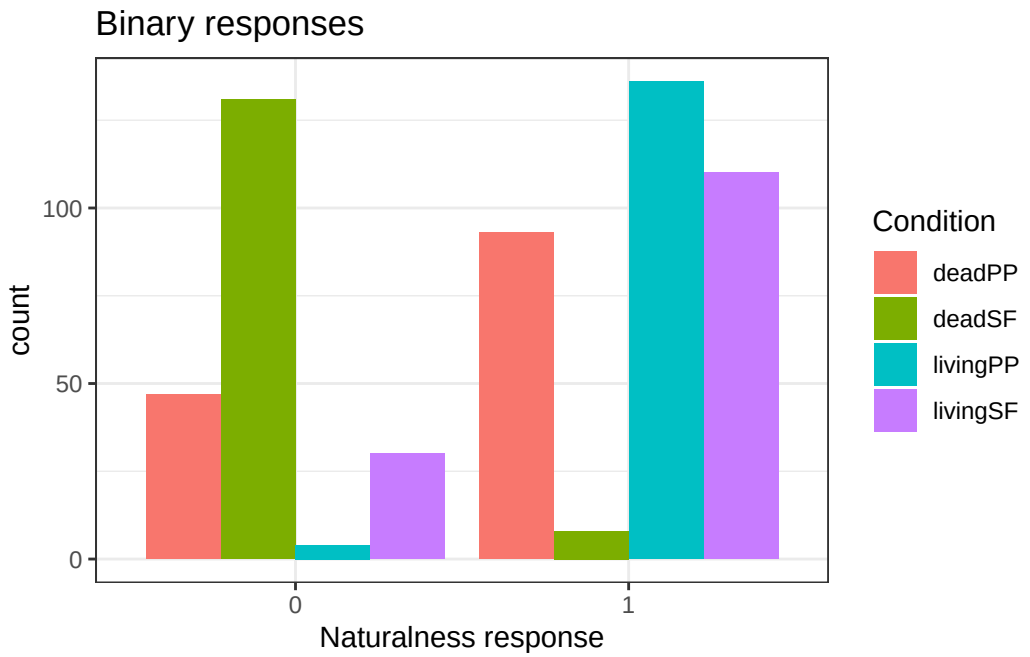



```
df_lifetime |>
  distinct(px,trial,.keep_all=T) |>
  ggplot(aes(x = as.factor(accept), fill = condition)) +
  labs(title = "Binary responses",
       x = "Naturalness response",
       fill = "Condition") +
  geom_bar() +
  theme_bw()
```



Grouped bar plots

```
df_lifetime |>
  distinct(px,trial,.keep_all=T) |>
  ggplot(aes(x = as.factor(accept), fill = condition)) +
  labs(title = "Binary responses",
       x = "Naturalness response",
       fill = "Condition") +
  geom_bar(position = "dodge") +
  theme_bw()
```



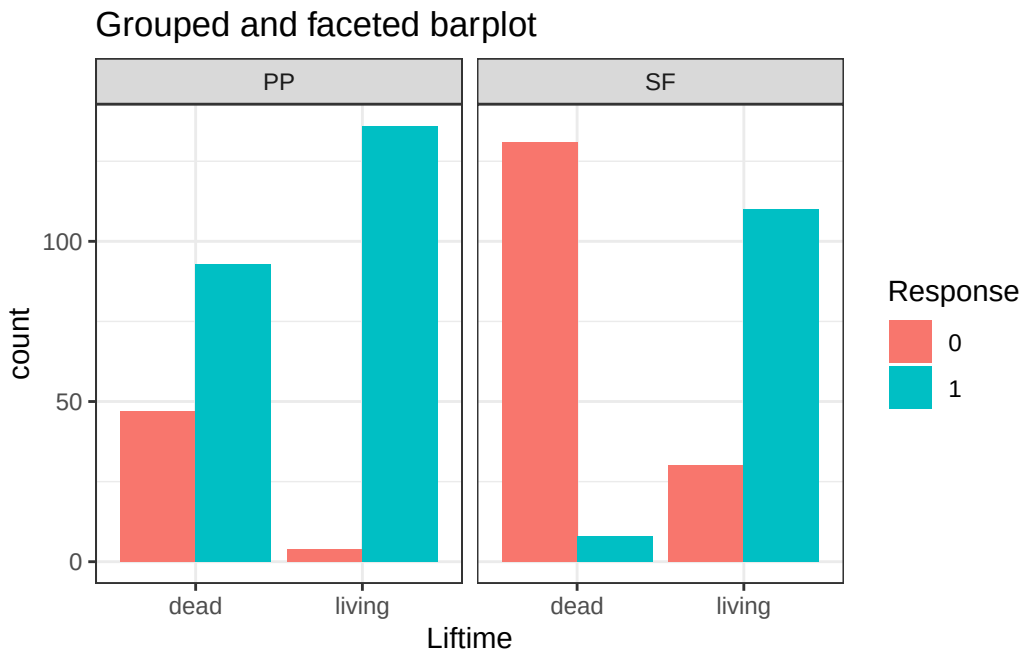
Exercise

1. Generate a grouped bar plot (i.e., `dodge`) with:
    - a facet grid for `tense`
    - plots `lifetime` on the x-axis
    - and fills the bars based on `accept`
    - change the labels accordingly
    - customise as you like

```
df_lifetime |>
  distinct(px,trial,.keep_all=T) |>
  ggplot(aes(x = lifetime, fill = as.factor(accept))) +
  facet_grid(.~tense) +
  labs(title = "Binary responses",
       x = "Liftime",
       fill = "Response") +
  geom_bar(position = "dodge") +
  theme_bw()
```

Grouped bar plots

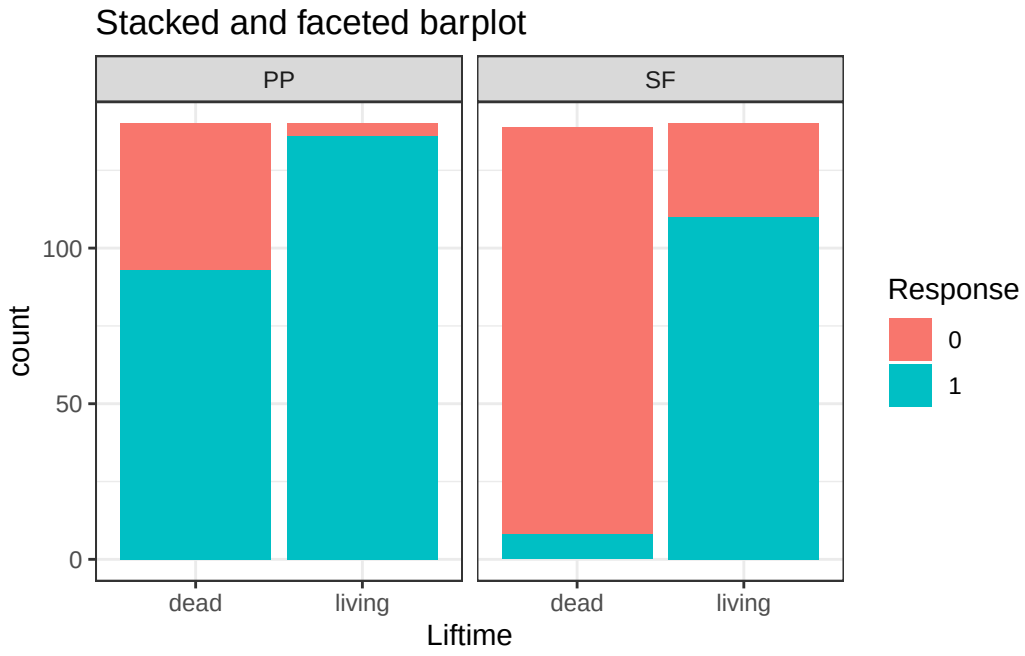
```
df_lifetime |>
  distinct(px,trial,.keep_all=T) |>
  ggplot(aes(x = lifetime, fill = as.factor(accept))) +
  facet_grid(.~tense) +
  labs(title = "Grouped and faceted barplot",
       x = "Liftime",
       fill = "Response") +
  geom_bar(position = "dodge") +
  theme_bw()
```



Stacked bar plots

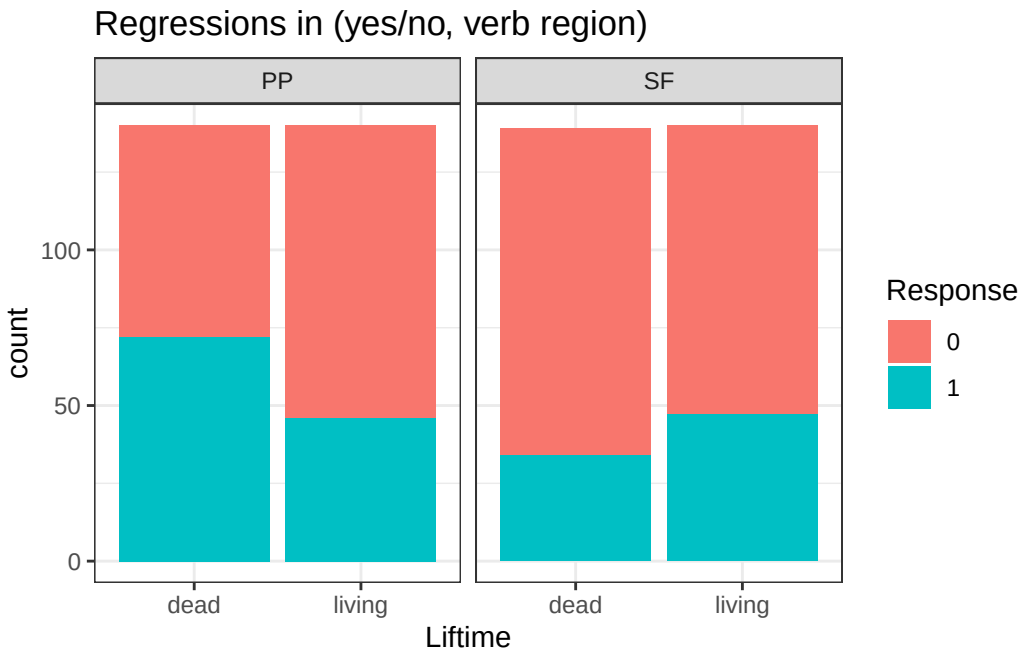
```
df_lifetime |>
  distinct(px,trial,.keep_all=T) |>
  ggplot(aes(x = lifetime, fill = as.factor(accept))) +
  facet_grid(.~tense) +
  labs(title = "Stacked and faceted barplot",
       x = "Liftime",
```

```
        fill = "Response") +
    geom_bar(position = "stack") +
    theme_bw()
```



Exercise

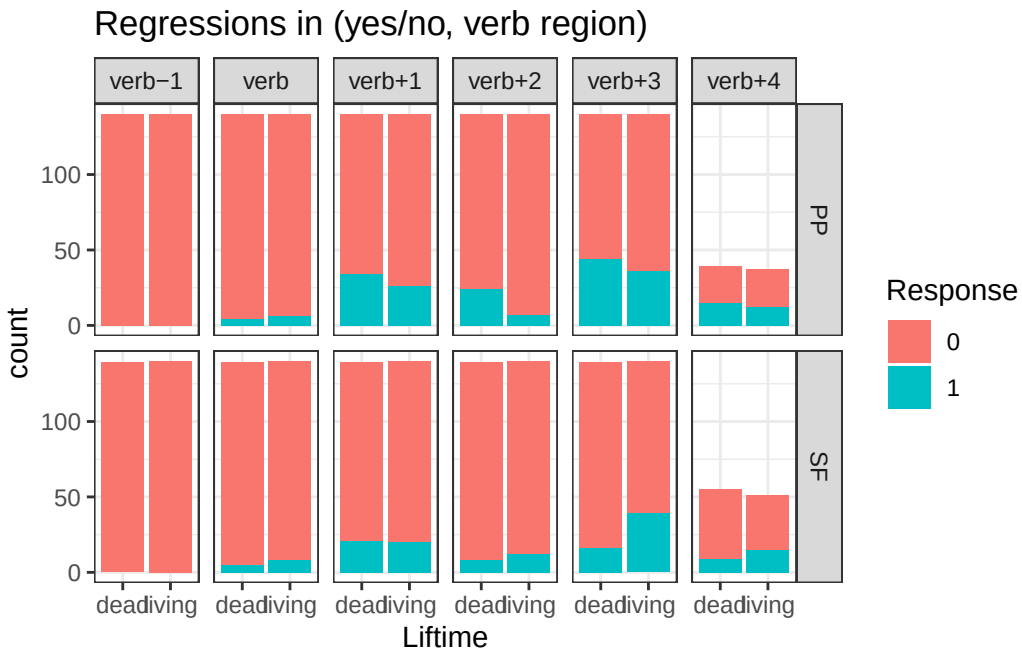
1. Choose the barplot you like best for binary data
2. Reproduce that barplot, but with `reg_in` at the `verb1` region



Extra exercise

1. Create another bar plot, but for `reg_out` for all sentence regions
2. Use `facet_grid()`

- to have facets by region (columns) and by tense (in 2 rows)



Resources

Nordmann et al. (2022)

Nordmann & DeBruine (2022)

Wickham et al. (n.d.), Chapter 2

References

Nordmann, E., & DeBruine, L. (2022). *Applied data skills* (Version 1.0). Zenodo. https://doi.org/10.5281/zenodo.6365078

Nordmann, E., McAleer, P., Toivo, W., Paterson, H., & DeBruine, L. M. (2022). Data Visualization Using R for Researchers Who Do Not Use R. *Advances in Methods and Practices in Psychological Science*, *5*(2), 251524592210746. https://doi.org/10.1177/25152459221074654

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (n.d.). *R for Data Science* (2nd ed.). https://r4ds.hadley.nz/