Histograms, Boxplots, and Stacked bars

INFO 526 Data Analysis and Visualization

Adriana Picoral

Experimental data

For this part of this tutorial we will be working with open data from Heyman, T., Goossens, K., Hutchison, K. A., & Storms, G. (2017). Does a working memory load really influence semantic priming? A self-replication attempt. Collabra: Psychology, 3(1).. All files are on the OSF platform.

Read data in

There are two zipped data files on the OSF the OSF platform. I unzipped the files and put them both in a folder called data in my project. I will first create a list of all the files in these two folders using the function `list.files()`. I want to keep the full names (with complete path) for these so we can read them later.

```
library(tidyverse)
experiment_files <- c(list.files(path = "data/Experiment 1/",
                                 full.names = TRUE),
                      list.files(path = "data/Experiment 2/",
                                 full.names = TRUE))
```

Now we can read all the files (which are tab separated) and specify that our id for each file is called "path" – we will then separate the path into its different components wit `separate()` and remove extra information using `select()`. We will also ensure age is a number using `mutate()` combined with `parse_number`.

```
experiment_data <- read_tsv(experiment_files,
                            id = "path") %>%
  separate(path,
           into = c("X1", "X2", "experiment", "participant", "age",
                    "gender", "month", "day", "hour", "minute",
                    "second", "extension")) %>%
  select(-X1, -X2, -extension) %>%
  mutate(age = parse_number(age))
```

Summary stats of participants

It is always a good idea to check your expectations of what is in the data with what actually is in the data. In this case we will compare summary statistics of participants per experiment to what the authors state in their paper.

They report number of participants per (binary) gender per experiment. Note that we have multiple observations per participant, so we need to use `distinct()` first to ensure we are not counting participants more than once.

```r
# get gender distribution by experiment
experiment_data %>%
  distinct(experiment, participant, age, gender) %>%
  count(experiment, gender)
```

```
## # A tibble: 4 x 3
##   experiment gender     n
##   <chr>      <chr>  <int>
## 1 1          f         47
## 2 1          m         31
## 3 2          f        137
## 4 2          m         23
```

They also report on the mean age of participants by experiment. We also need to run `distinct()` before our `group_by()` and `summarize()`.

```r
# get mean age by experiment
experiment_data %>%
  distinct(experiment, participant, age) %>%
  group_by(experiment) %>%
  summarize(mean_age = mean(age))
```

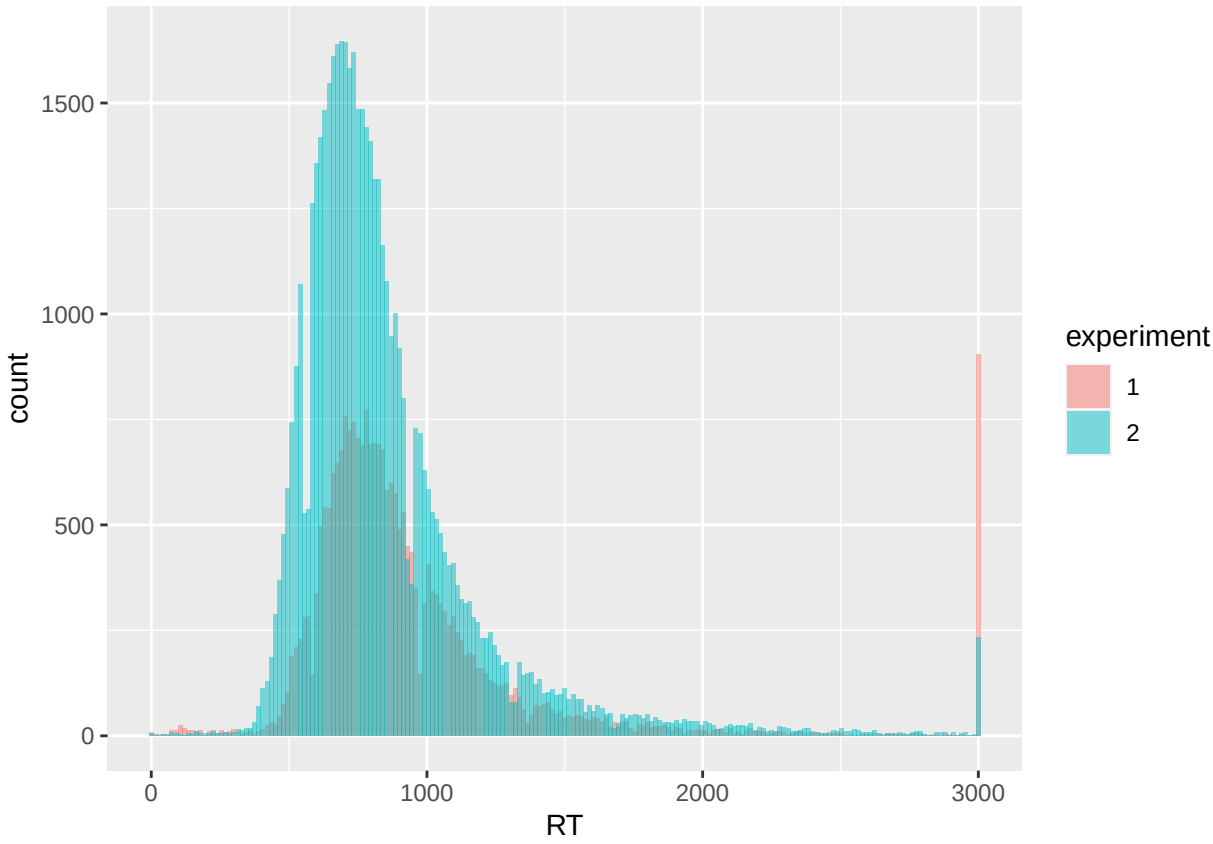
```
## # A tibble: 2 x 2
##   experiment mean_age
##   <chr>         <dbl>
## 1 1              19.9
## 2 2              18.7
```

Our numbers from the data match what is reported in the paper, so we can proceed.

Histogram

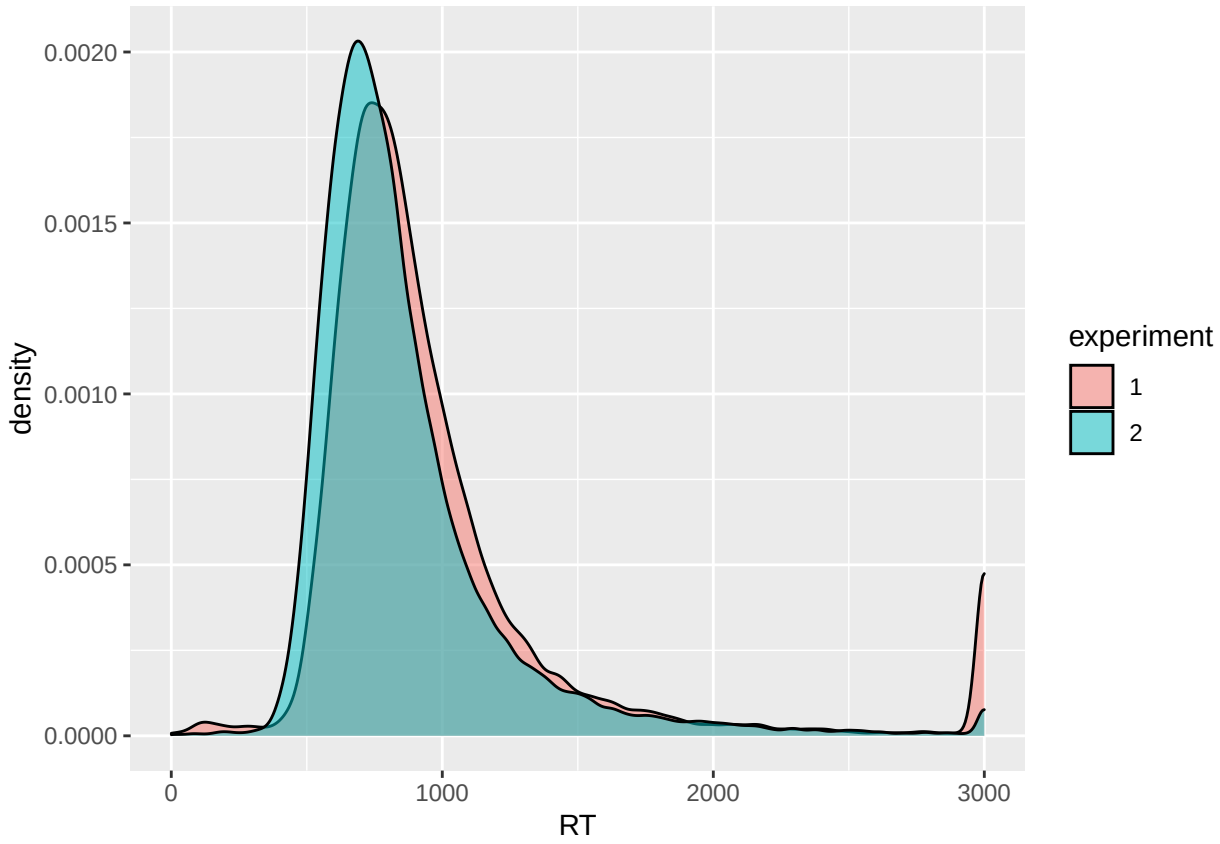
It is common practice to do exploratory analyses on our data before running any statistical analysis and modeling. For the experimental data we are using, the response variable is reaction time (RT), which is a numeric continuous variable. We can look at the distribution of numeric continuous variables using histograms, which splits the numbers into bins and then counts how many data points fall within each bin. One of the axes (usually y) is always going to be number of observations, the other will be our continuous numeric variable. We can also map a categorical variable to `fill`. By default the bars in the histogram in `geom_histogram` will be stacked, we can change that by setting `position to "identity"` and setting an `alpha` that is low enough for us to see the overlap across groups and high enough for us to be able to read the plot. You can set total number of `bins` or the window for each bin using `binwidth`. These parameters will depend on your data range.

```r
experiment_data %>%
  ggplot(aes(x = RT,
             fill = experiment)) +
  geom_histogram(alpha = .5,
                 position = "identity",
                 binwidth = 15)
```



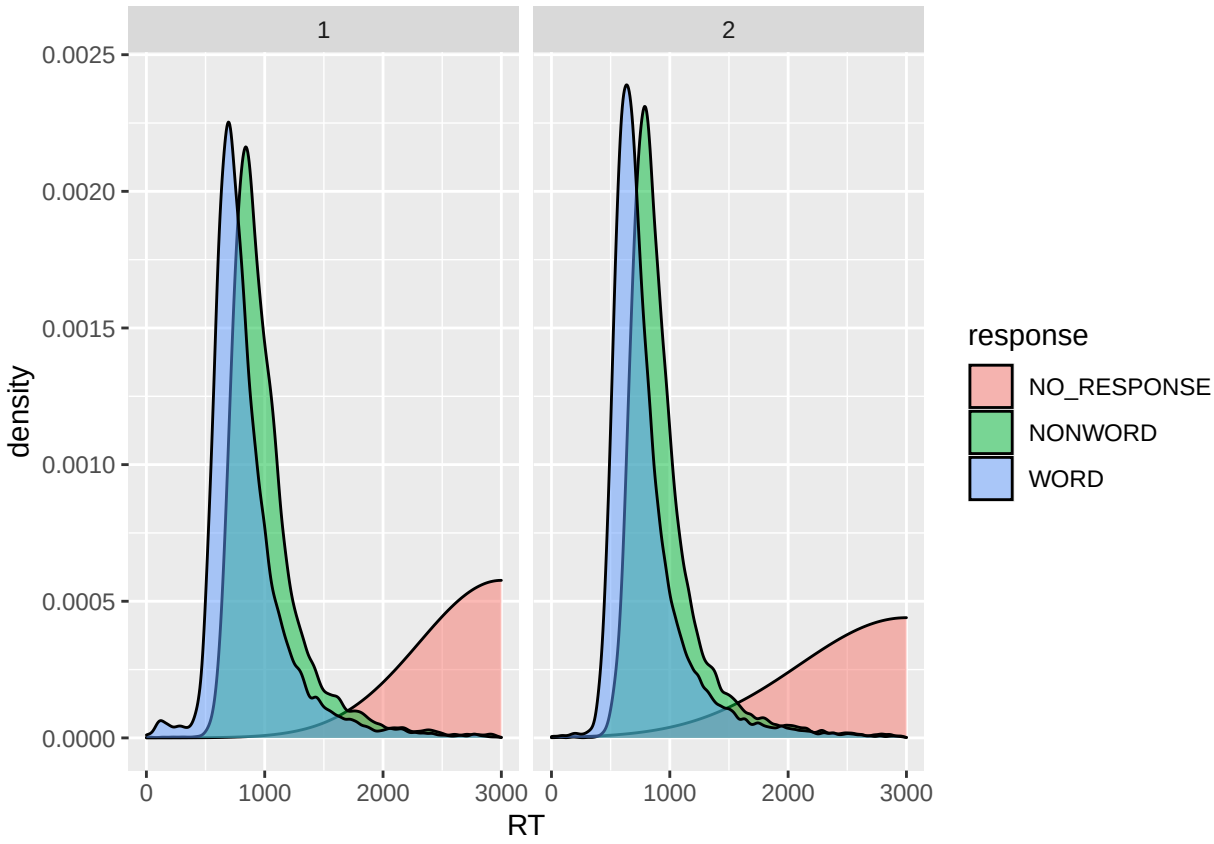
An alternative to histograms, which display counts, is a plot of the kernel density estimate of the underlying distribution using `geom_density` instead.

```
experiment_data %>%
  ggplot(aes(x = RT,
             fill = experiment)) +
  geom_density(alpha = .5)
```



Let's try plotting a density plot with two categorical variable, faceting the plot by experiment. In this plot we can notice that the very large reaction times (RT) are mostly from non responses (i.e., when the participant failed to press any buttons during the experiment).

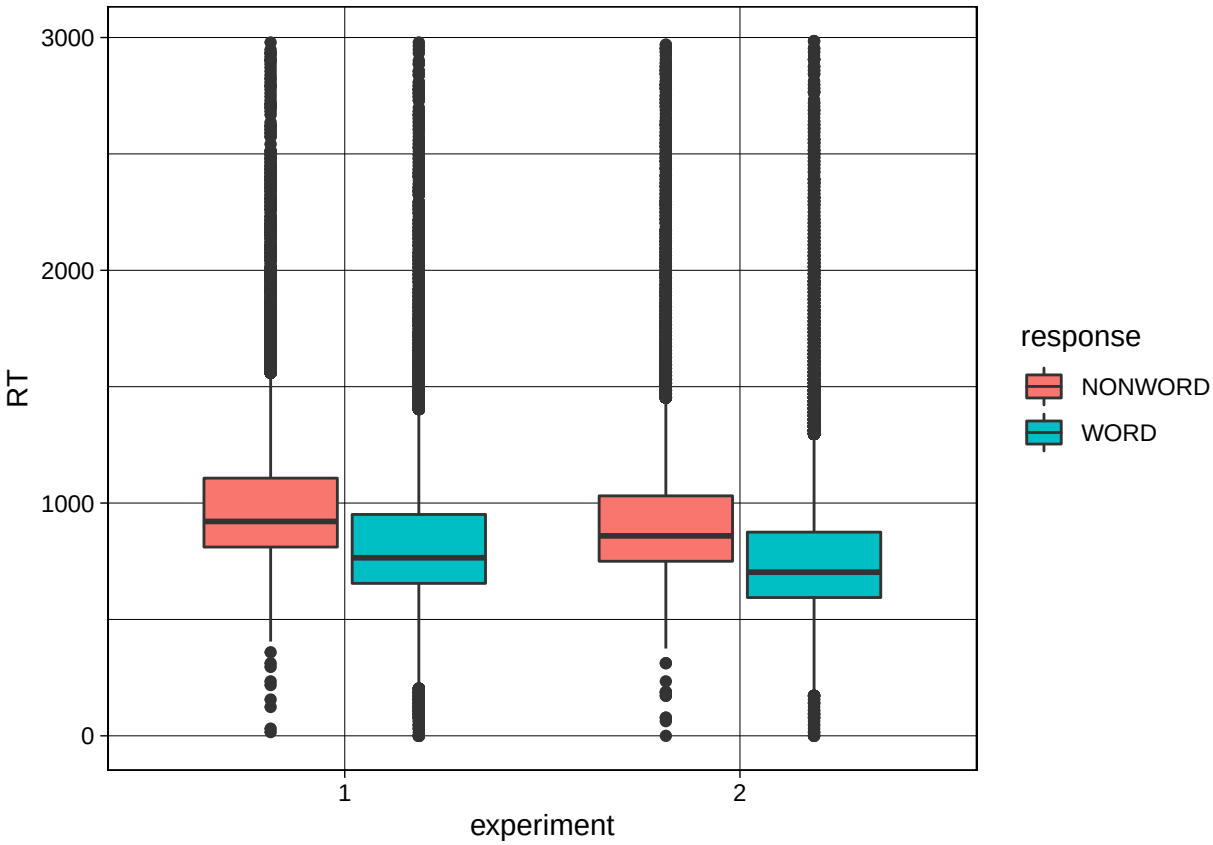
```
experiment_data %>%
  ggplot(aes(x = RT,
             fill = response)) +
  geom_density(alpha = .5) +
  facet_wrap(~experiment)
```



Boxplots

Bases on your density plots, we can move forward with our exploratory analysis filtering out no responses. We will build boxplots next, which displays the distribution of a continuous variable (RT in our case). The middle line in each box is the **median** and the two hinges are the 25th and 75th percentiles. The whiskers is bases on the inter-quartile range (IQR), which is distance between the first and third quartiles. The lower whisker goes to the smallest value at most 1.5 * IQR of the hinge, and the upper goes to the upper 1.5 * IQR value. Data beyond the end of the whiskers are outliers and are plotted individually.

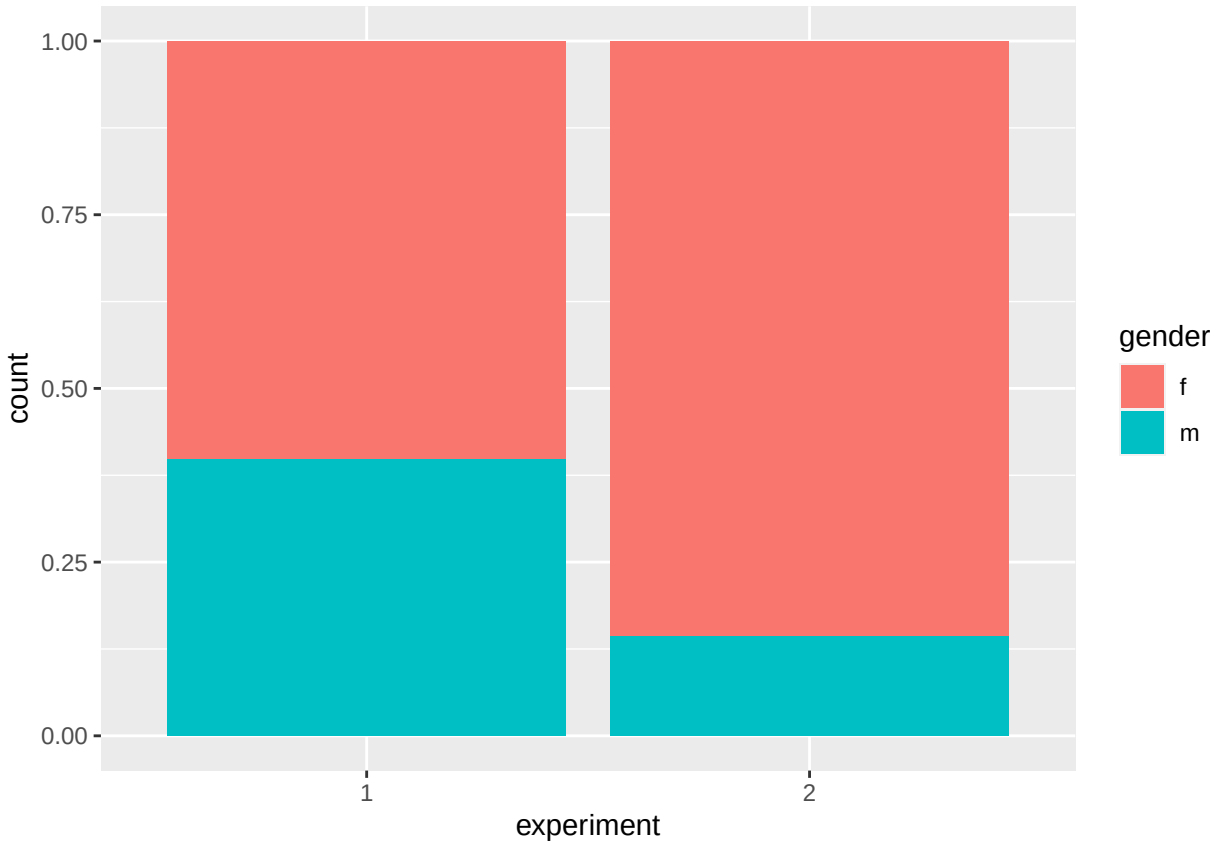
```
experiment_data %>%
  filter(response != "NO_RESPONSE") %>%
  ggplot(aes(x = experiment,
             y = RT,
             fill = response)) +
  geom_boxplot() +
  theme_linedraw()
```



Stacked bars and pie charts

Let's switch our focus to the distribution of categorical variables. For that, we can rely on our `geom_bar()` function setting `position` to "fill" to get proportions.

```
experiment_data %>%
  ggplot(aes(x = experiment,
             fill = gender)) +
  geom_bar(position = "fill")
```

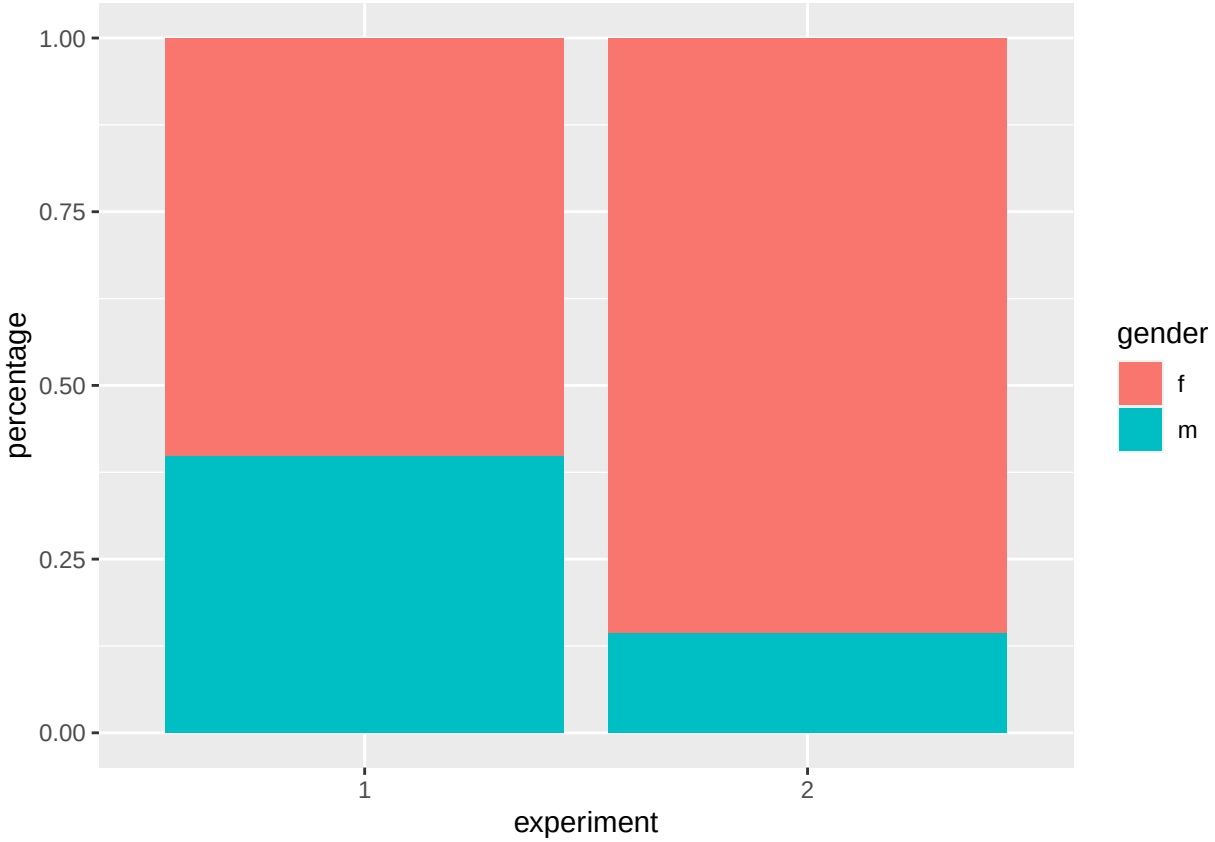


We can of course calculate percentages using `group_by()`, with `summarize()` and `mutate()`.

```
percent_gender_experiment <- experiment_data %>%
  group_by(experiment, gender) %>%
  summarize(count = n()) %>%
  mutate(total = sum(count),
         percentage = count/total)
```

And then recreate the same stacked barplot we did before.

```
percent_gender_experiment %>%
  ggplot(aes(x = experiment,
             y = percentage,
             fill = gender)) +
  geom_col()
```



Let's combine our calculations of counts, totals, and percentages with ggplot to plot the distribution of data points per type of semantic priming across experiments. Here's how the authors explain this variable in their paper:

Critical prime word types:

- FA - Forward associates (i.e., the target is an associate of the prime, but not vice verse; e.g., panda-bear)
- BA - backward associates (i.e., the prime is an associate of the target, but not vice versa; e.g., baby-stork)
- SYM - symmetric associates (i.e., the prime is an associate of the target and vice versa; e.g., cat-dog)

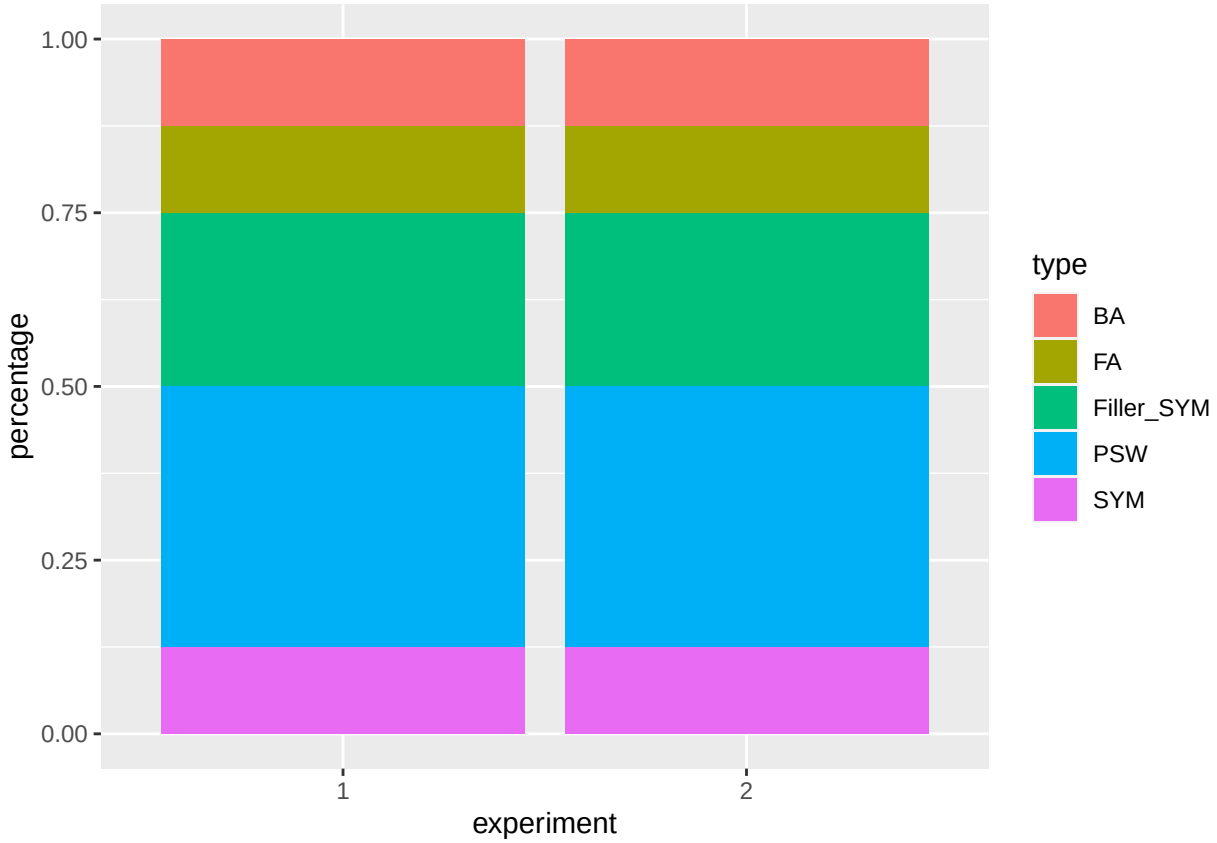
Filler prime word types:

- Filler_SYM - symmetric associates (i.e., the prime is an associate of the target and vice versa; e.g., cat-dog)
- PSW - word-non-word pairs

Stacked bars allows us to check if the distribution of observations for each of these categories are equal across the two experiments.

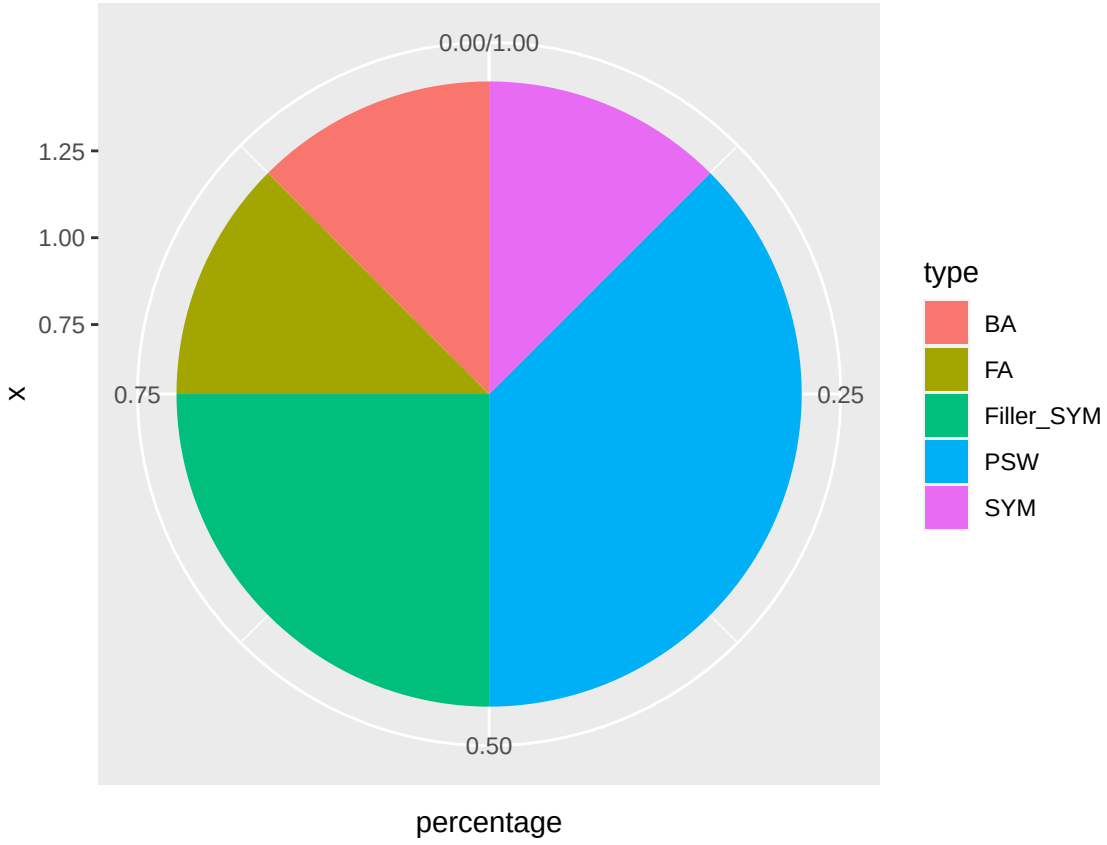
```
experiment_data %>%
  group_by(experiment, type) %>%
  summarize(count = n()) %>%
  mutate(total = sum(count),
         percentage = count/total) %>%
  ggplot(aes(x = experiment,
```

```
            y = percentage,
            fill = type)) +
  geom_col()
```



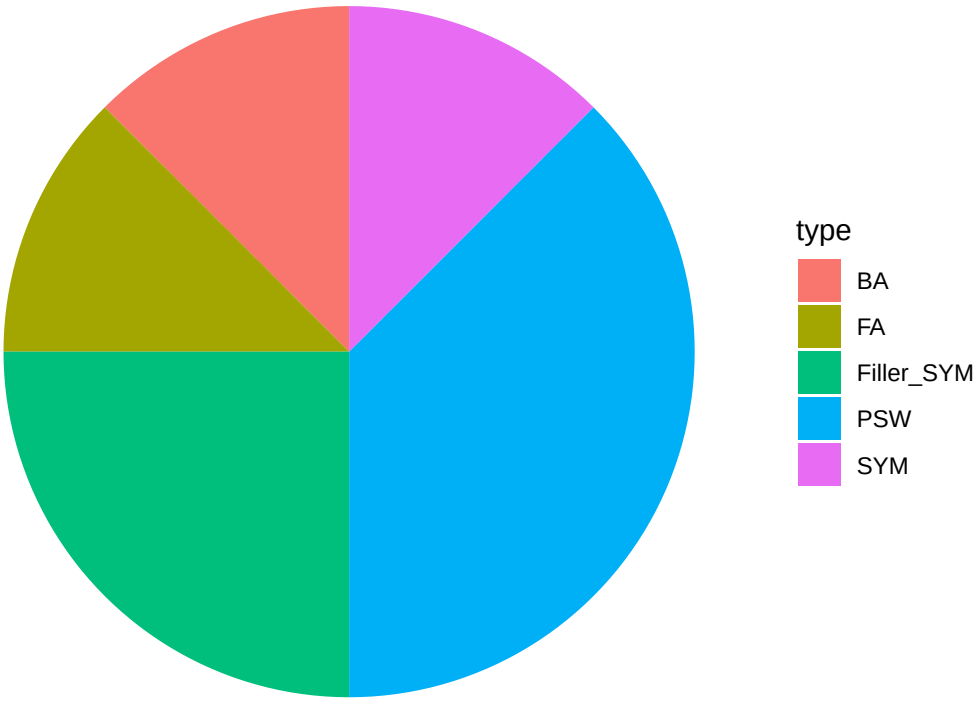
Pie charts are not known by their clarity, and many data analysts avoid pie charts all together. In fact, there's no straightforward way to plot a pie chart with ggplot, we need to so some wrangling of the mapping to get it done. Note how difficult it is to compare the angle of different slices in the chart.

```
experiment_data %>%
  group_by(type) %>%
  summarize(count = n()) %>%
  mutate(total = sum(count),
         percentage = count/total) %>%
  ggplot(aes(x = 1,
             y = percentage,
             fill = type)) +
  geom_col() +
  coord_polar("y", start = 0)
```



Let's clean things up with `theme_void()`.

```
experiment_data %>%
  group_by(type) %>%
  summarize(count = n()) %>%
  mutate(total = sum(count),
         percentage = count/total) %>%
  ggplot(aes(x = 1,
             y = percentage,
             fill = type)) +
  geom_col() +
  coord_polar("y", start = 0) +
  theme_void()
```



Heat maps

Heat maps are used to display the magnitude change of a numeric variable across two categorical variables. Let's calculate mean reaction time (RT) across prime word type and on whether the prime relates to the target word, or not (the authors state that unrelated prime-target pairs were created by recombining primes and targets within every group). We will also ensure that light colors represent low numbers and dark colors high numbers.

For specifying colors you can use the color's hex code or color names.

```
experiment_data %>%
  #filter(type != "PSW") %>%
  group_by(type, target_prime_relatedness) %>%
  summarize(mean_RT = mean(RT)) %>%
  ggplot(aes(y = type,
             x = target_prime_relatedness,
             fill = mean_RT)) +
  geom_tile() +
  scale_fill_continuous(low = "gray86",
                        high = "gray23")
```

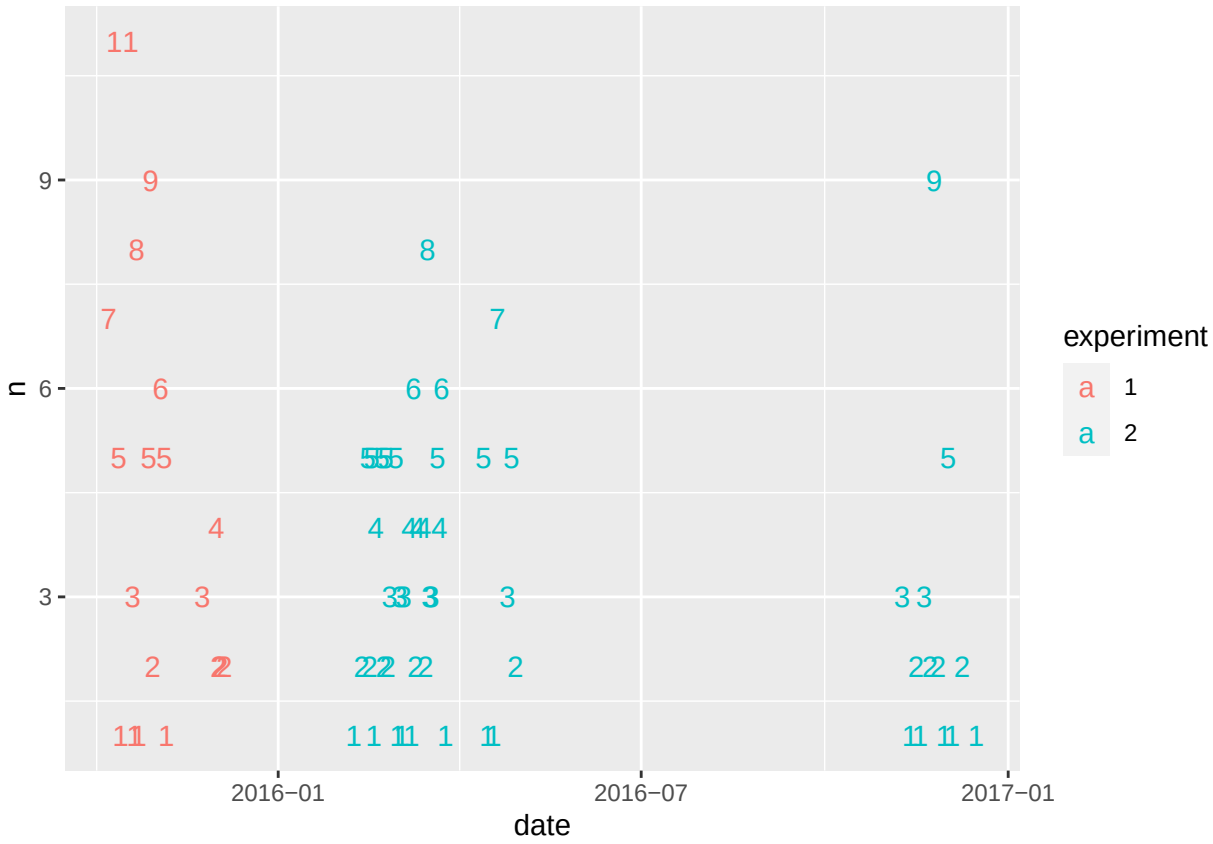
Dealing with dates

The last thing we will be talking about related to this data is how to deal with dates, which is a hard problem in general. We will create a year variable, which does not exist in the data, assuming experiment 1 was performed in 2015 and experiment 2 in 2016. We will then combine year, month, and day (using the ISO 8601 format).

```
library(lubridate)
experiment_data <- experiment_data %>%
  mutate(year = ifelse(experiment == "1",
                       2015, 2016),
         date = paste0(year, "-", month, "-", day),
         date = as.Date(date))
```

We can now plot wow many participants per day participated in the experiment.

```
experiment_data %>%
  distinct(experiment, participant, date) %>%
  count(experiment, date) %>%
  ggplot(aes(x = date,
             y = n,
             color = experiment,
             label = n)) +
  geom_text()
```


n
9
6
3
2016−01
2016−07
2017−01
date
experiment
a 1
a 2