Data visualization

Graphs for a single categorical variable

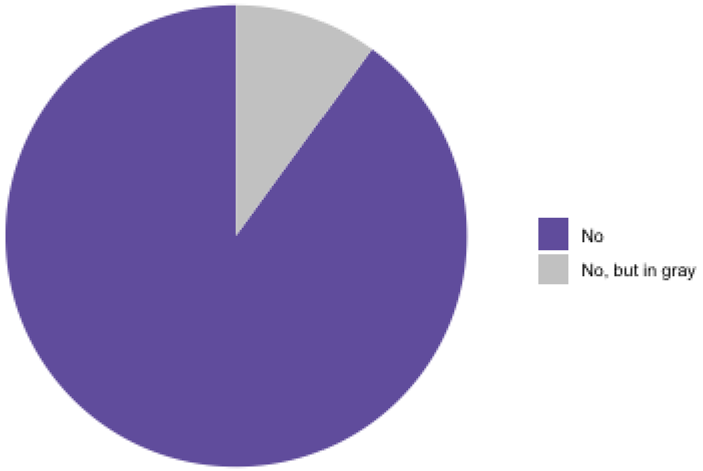
Jenine Harris
Brown School

Types of graphs for a single categorical variable

- Before selecting a graph type, it is useful to think about the goal of the graph.
- Making graphs to check whether something is normally distributed before calculating a mean is very different from making graphs to communicate information to an audience.
- A categorical variable has categories that are either ordinal with a logical order or nominal with no logical order.
- Categorical variables are the factor data type in R.
- Some of the more commonly used graphs for a single categorical variable are:
    - pie chart
    - waffle plot
    - bar graph
    - point chart
- Pie charts and waffle plots are similar, they are both used for showing parts of a whole.
- Bar graphs and point charts tend to be used to compare groups.

Pie charts

- While pie charts are often seen in newspapers and other popular media, they are considered by most analysts as an unclear way to display data.
    - Pie charts are difficult to read since the relative size of pie pieces is often hard to determine
    - Pie charts take up a lot of space to convey little information
    - People often use fancy formatting like 3D, which takes up more space and makes understanding relative size of pie pieces even more difficult
- Should I make a pie chart?



Bar graphs

- The nationally representative **National Health and Nutrition Examination Survey**, or NHANES asked about gun use in the *audiology* section concerned with how loud noise may influence hearing loss.
- The most recent year of NHANES data available with a gun use question was 2011-2012.
- One of the cool things about NHANES data is that an R package called `RNHANES` allows direct access to NHANES data from R, which is great for reproduciblity.
- Used `read.csv()` to import the cleaned data and check that the import worked by using `head()`.

```r
# import the data
nhanes.2012 <- read.csv(file = "~/Box/teaching/Teaching/Fall2020/data/nh

# check the import
head(x = nhanes.2012)
```

Check the imported data

```
##    SEQN     cycle SDDSRVYR RIDSTATR RIAGENDR RIDAGEYR RIDAGEMN RIDRETH1
## 1 62161 2011-2012        7        2        1       22       NA        3
## 2 62162 2011-2012        7        2        2        3       NA        1
## 3 62163 2011-2012        7        2        1       14       NA        5
## 4 62164 2011-2012        7        2        2       44       NA        3
## 5 62165 2011-2012        7        2        2       14       NA        4
## 6 62166 2011-2012        7        2        1        9       NA        3
##   RIDRETH3 RIDEXMON RIDEXAGY RIDEXAGM DMQMILIZ DMQADFC DMDBORN4 DMDCITZN
## 1        3        2       NA       NA        2      NA        1        1
## 2        1        1        3       41       NA      NA        1        1
## 3        6        2       14      177       NA      NA        1        1
## 4        3        1       NA       NA        1       2        1        1
## 5        4        2       14      179       NA      NA        1        1
## 6        3        2       10      120       NA      NA        1        1
##   DMDYRSUS DMDEDUC3 DMDEDUC2 DMDMARTL RIDEXPRG SIALANG SIAPROXY SIAINTRP
## 1       NA       NA        3        5       NA       1        1        2
## 2       NA       NA       NA       NA       NA       1        1        2
## 3       NA        8       NA       NA       NA       1        1        2
## 4       NA       NA        4        1        2       1        2        2
## 5       NA        7       NA       NA       NA       1        1        2
## 6       NA        3       NA       NA       NA       1        1        2
##   FIALANG FIAPROXY FIAINTRP MIALANG MIAPROXY MIAINTRP AIALANGA   WTINT2YR
## 1       1        2        2       1        2        2        1 102641.406
## 2       1        2        2      NA       NA       NA       NA  15457.737
## 3       1        2        2       1        2        2        1   7397.685
## 4       1        2        2      NA       NA       NA       NA 127351.373
## 5       1        2        2       1        2        2        1  12209.745
```

Examining the gun use question

- The question `AUQ300` in the 2011-2012 data asked participants, "Have you ever used firearms for any reason?"
- Use `summary()` to check the `AUQ300` variable.

```
# check the data
summary(object = nhanes.2012$AUQ300)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   1.000   1.000   2.000   1.656   2.000   7.000    4689
```

Checking the variable coding

- The gun use data imports as a numeric variable type.

- The audiology data codebook shows five possible values for `AUQ300`:

  - 1 = Yes
  - 2 = No
  - 7 = Refused
  - 9 = Don't know
  - . = Missing

R code to clean the gun use variable

```r
# open tidyverse
library(package = "tidyverse")

# recode gun use variable
nhanes.2012.clean <- nhanes.2012 %>%
  mutate(AUQ300 = recode_factor(.x = AUQ300,
                                `1` = 'Yes',
                                `2` = 'No',
                                `7` = 'Refused',
                                `9` = 'Don\'t know'))

# check the recoding
summary(object = nhanes.2012.clean$AUQ300)
```

```
##     Yes      No Refused    NA's
##    1613    3061       1    4689
```

Coding infrequent values to missing

- There was a single *Refused* response to the gun use question and no *Don't know* responses.
    - These categories are not likely to be useful for visualizing or analyzing this variable, recode them to be `NA`
- It would be easier to work with the gun use variable if the variable name were something more intuitive that is easy to remember.
- Use `rename()` to rename the variable.

```
# recode gun use variable
nhanes.2012.clean <- nhanes.2012 %>%
  mutate(AUQ300 = na_if(x = AUQ300, y = 7)) %>%
  mutate(AUQ300 = recode_factor(.x = AUQ300,
                                `1` = 'Yes',
                                `2` = 'No')) %>%
  rename(gun.use = AUQ300)

# check recoding
summary(object = nhanes.2012.clean$gun.use)
```

```
##  Yes   No NA's
## 1613 3061 4690
```

Using ggplot()

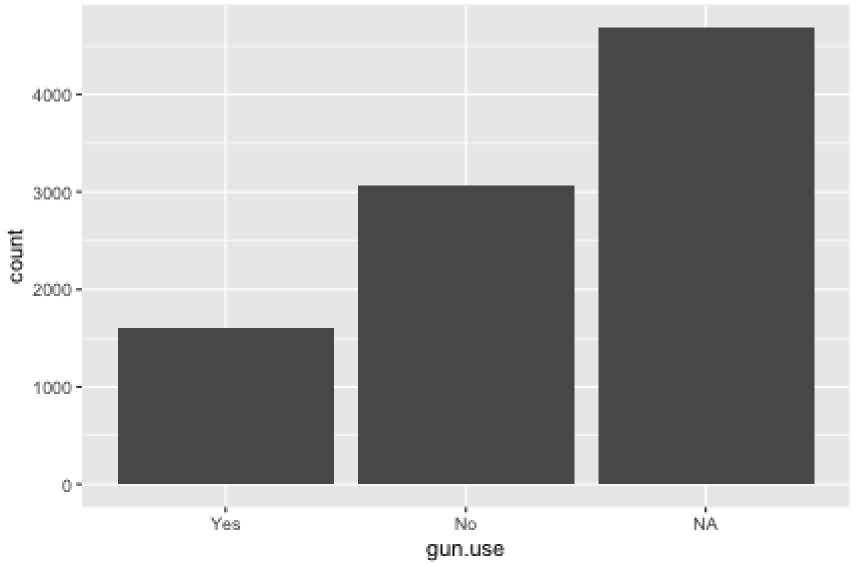
- Use the `ggplot()` function from the ggplot2 package in the tidyverse to create a bar plot of gun use.
- The gg stands for the *grammar of graphics*.
- Graphs built with `ggplot()` are built in layers.
- The first layer starts with `ggplot()` and `aes()` or *aesthetics*, which contains the basic information about which variables are included in the graph and whether each variable should be represented on the x-axis, the y-axis, as a color, as a line type, or something else.

The next layer typically gives the graph type, or graph **geometry**, in the grammar of graphics language, and starts with `geom_` followed by one of the available types.

- In this case, use `geom_bar()` as the geometry for this graph.
- Remember that `geom_bar()` is a layer of the plot and so is added with a `+` instead of a `%>%`.

Creating the first plot

```
# plot gun use in US 2011-2012
nhanes.2012.clean %>%
  ggplot(aes(x = gun.use)) +
  geom_bar()
```

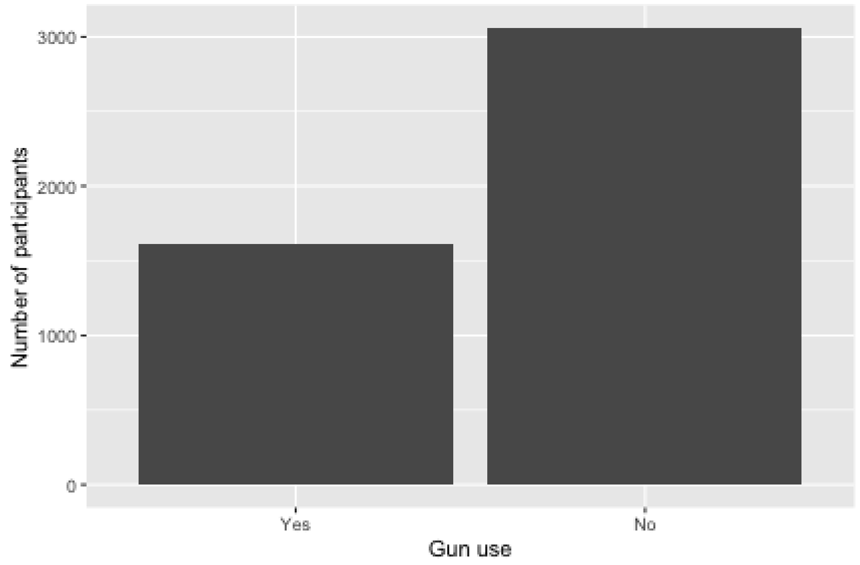


Examine the graph and make a plan

- A few things could change about this initial graph:
  - Remove the `NA` bar from the graph
  - Change the labels on the axes to provide more information
  - Use a theme that does not use so much ink
  - Make each bar a different color
  - Show percentages instead of counts on the y-axis

Removing NA and adding axis labels

```
# omit NA category from gun.use plot and add axis labels
nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use)) +
  geom_bar() +
  labs(x = "Gun use", y = "Number of participants")
```



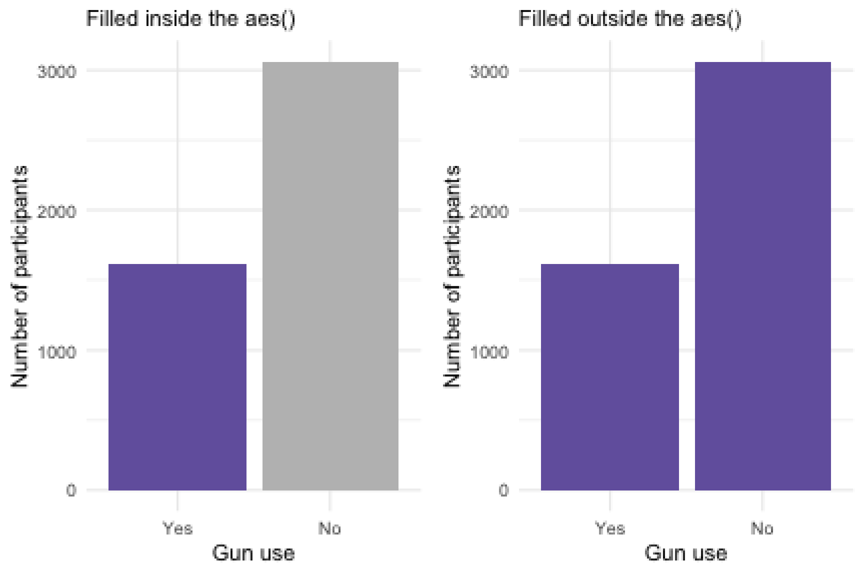
Adding color

- Changing the way the graph looks *based on the data* should happen within the `aes()`.
- So, changin the colors of the bars to be different *depending on gun use category* would happen in the `aes()`.
- Add `fill =` within `aes()` like this: `geom_bar(aes(fill = gun.use))`.
- To make the bars a color without linking it to the categories of `gun.use`, put the `fill =` *outside* the `aes()` parentheses like this: `geom_bar(fill = "purple")`.
- Try writing the code both ways and using the `grid.arrange()` function from the gridExtra package to show the plots side-by-side.

Two ways of adding color

```r
# install gridExtra if it is not already installed
# fill bars inside aes
fill.aes <- nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use)) +
  geom_bar(aes(fill = gun.use)) +
  labs(x = "Gun use", y = "Number of participants",
       subtitle = "Filled inside the aes()") +
  scale_fill_manual(values = c("#7463AC", "gray"), guide = FALSE) +
  theme_minimal()

# fill bars outside aes
fill.outside <- nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use)) +
  geom_bar(fill = "#7463AC") +
  labs(x = "Gun use", y = "Number of participants",
       subtitle = "Filled outside the aes()") +
  theme_minimal()

# arrange the two plots side-by-side
gridExtra::grid.arrange(fill.aes, fill.outside, ncol = 2)
```

Examine the difference



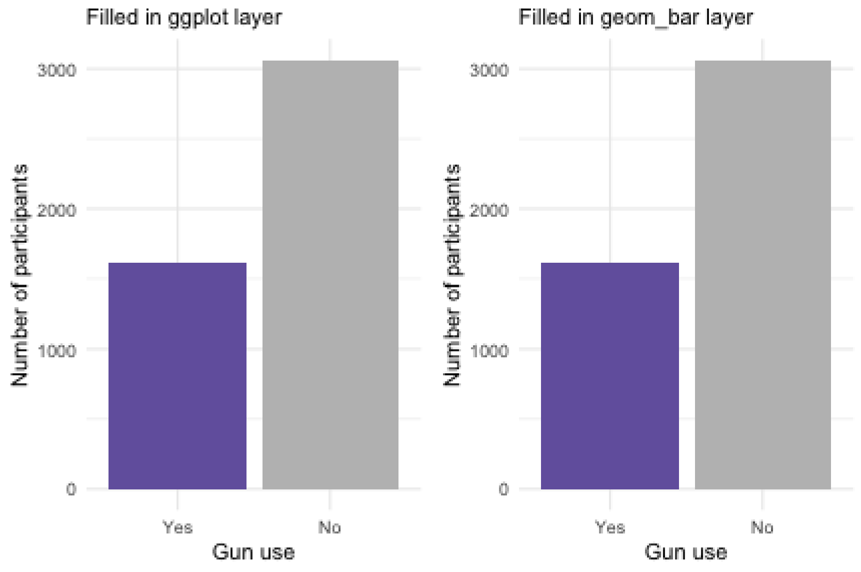
More about the aes()

- Notice that there are `aes()` for both the `ggplot()` layer and the `geom_bar()` layer.
- Some aesthetics can be set in either place, like the color of the bars, for example.
- Change to the code to try setting the color in the `ggplot()` layer `aes()` and in the `geom_bar()` layer `aes()`.

Setting aes() in ggplot vs. geom_bar

```r
# fill inside aes for ggplot layer
fill.aes <- nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use, fill = gun.use)) +
  geom_bar() +
  labs(x = "Gun use", y = "Number of participants",
       subtitle = "Filled in ggplot layer") +
  scale_fill_manual(values = c("#7463AC", "gray"), guide = FALSE) +
  theme_minimal()

# fill inside aes for geom_bar layer
fill.outside <- nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use)) +
  geom_bar(aes(fill = gun.use)) +
  labs(x = "Gun use", y = "Number of participants",
       subtitle = "Filled in geom_bar layer") +
  scale_fill_manual(values = c("#7463AC", "gray"), guide = FALSE) +
  theme_minimal()

# arrange the two plots side-by-side
gridExtra::grid.arrange(fill.aes, fill.outside, ncol = 2)
```

Setting aes() in ggplot vs. geom_bar

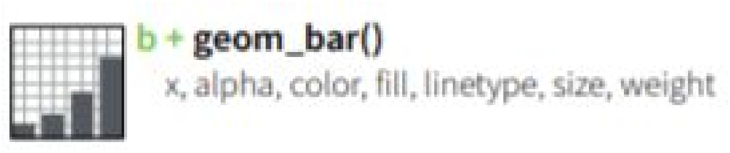


Asthetics and graph geometry

- Some aesthetics are specific to the type of graph geometry.
- For example, there is an aesthetic called `linetype =` that can make lines appear in different ways like dotted.
- This is not an available aesthetic for graphs that have no lines in them, but is available for graphs that have lines in them, so it might be better in the `geom_` layer instead of in the `ggplot()` layer.
- The Data Visualization Cheat Sheet from R Studio shows the aesthetics available for the different geometries.

Data viz cheat sheet

- Below the name of each type of `geom_` is a list and these were the aesthetics available for that `geom_`.
- The `geom_bar()` entry lists the following available aesthetics: x, alpha, color, fill, linetype, size, and weight.



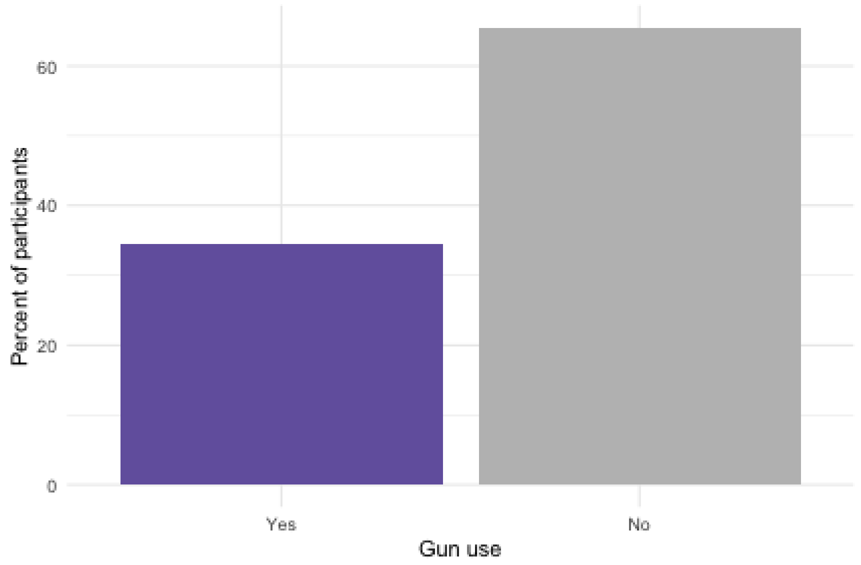
b + geom_bar()
x, alpha, color, fill, linetype, size, weight

Changing axis to percentage

- To get the y-axis to show percentage rather than count, the y-axis uses *special variables* with double periods around them.
- Special variables are statistics computed from a data set; the special variable, *count*, counts the number of observations.
- Add the special variables to the aesthetics using `..count..` to represent the frequency of a category and `sum(..count..)` to represent the sum of all the frequencies; multiply by 100 to get a percent

```
# formatted bar graph of gun use
nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use,
             y = 100*(..count..)/sum(..count..))) +
  geom_bar(aes(fill = gun.use)) +
  labs(x = "Gun use", y = "Percent of participants") +
  scale_fill_manual(values = c("#7463AC", "gray"), guide = FALSE) +
  theme_minimal()
```

Changing axis to percentage



Setting axis limits

- The y-axis of the graph only goes to 60 percent.
- People sometimes limit the range of the y-axis in order to make a difference between groups or a change over time look bigger (or smaller) than it actually is.
- Make the y-axis go to 100 percent by creating a `ylim()` layer in the `ggplot()`.
    - The `ylim()` layer takes the lowest value for the y-axis and the highest value for the y-axis, separated by a comma.
    - For a y-axis that goes from 0 to 100 it looks like this: `ylim(0, 100)`.

```
# formatted bar graph of gun use
nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use,
             y = 100*(..count..)/sum(..count..))) +
  geom_bar(aes(fill = gun.use)) +
  labs(x = "Gun use", y = "Percent of participants") +
  scale_fill_manual(values = c("#7463AC", "gray"), guide = FALSE) +
  theme_minimal() +
  ylim(0, 100)
```

Examining the axis limits

```
# formatted bar graph of gun use
nhanes.2012.clean %>%
  drop_na(gun.use) %>%
  ggplot(aes(x = gun.use,
             y = 100*(..count..)/sum(..count..))) +
  geom_bar(aes(fill = gun.use)) +
  labs(x = "Gun use", y = "Percent of participants") +
  scale_fill_manual(values = c("#7463AC", "gray"), guide = FALSE) +
  theme_minimal() +
  ylim(0, 100)
```

Waffle charts

- Waffle charts are similar to pie charts in showing the parts of a whole.
- However, the structure of a waffle chart visually shows the relative contributions of categories to the whole waffle more clearly.
- The `AUQ310` variable is the response to the question "How many total rounds have you ever fired?" for survey participants who reported that they had used a gun.
- The audiology data codebook shows eight categories for `AUQ310`:
    - 1 = 1 to less than 100 rounds
    - 2 = 100 to less than 1000 rounds
    - 3 = 1000 to less than 10,000 rounds
    - 4 = 10,000 to less than 50,000 rounds
    - 5 = 50,000 rounds or more
    - 7 = Refused
    - 9 = Don't know
    - . = Missing

Change data type, add labels, rename the variable

- Change `AUQ310` to a factor, and rename `AUQ310` to `fired`, which was easier to remember and type.

```
# recode gun use variable
nhanes.2012.clean <- nhanes.2012 %>%
  mutate(AUQ300 = na_if(x = AUQ300, y = 7)) %>%
  mutate(AUQ300 = recode_factor(.x = AUQ300,
                                `1` = 'Yes',
                                `2` = 'No')) %>%
  rename(gun.use = AUQ300) %>%
  mutate(AUQ310 = recode_factor(.x = AUQ310,
                                `1` = "1 to less than 100",
                                `2` = "100 to less than 1000",
                                `3` = "1000 to less than 10k",
                                `4` = "10k to less than 50k",
                                `5` = "50k or more",
                                `7` = "Refused",
                                `9` = "Don't know")) %>%
  rename(fired = AUQ310)

#check recoding
summary(object = nhanes.2012.clean$fired)
```

Use the waffle package

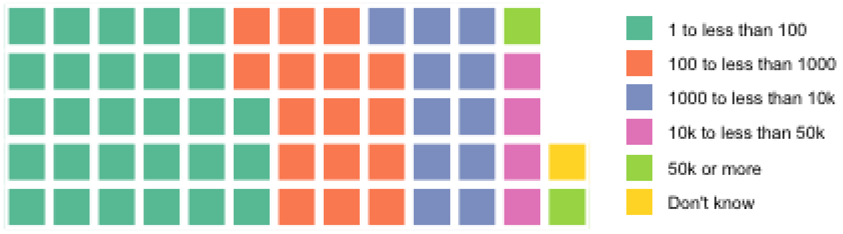
- There is no built-in `geom_waffle()` option for `ggplot()`, so would use the waffle package instead.
- The first argument for the `waffle()` is a table or vector of **summary statistics** used to make the waffle squares.
- The data used by `waffle()` is not the **individual level** data with one observation per row, it is a frequency table or a vector of frequencies that shows how many observations are in each category.
- The `table()` function works well for use with `waffle()`.

The squares in a waffle

- By default, `waffle()` makes one square per observation.
- There are more than 1,000 observations in the `nhanes.2012` data frame, which would be a lot of squares!
- Instead, use code to make each square represent 25 observations.
- Finally, the last argument for `waffle()` is the number of rows of squares, start with 5 rows and see what happens.

```r
# open the waffle library
library(package = "waffle")

# make a table of rounds fired data
rounds <- table(nhanes.2012.clean$fired)

# each square is 25 people
# 5 rows of squares
waffle(parts = rounds / 25, rows = 5)
```

Reviewing the waffle plot



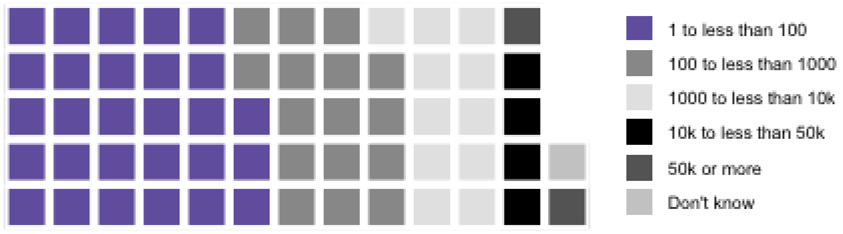
- It is clear from this waffle that the *1 to less than 100* category is the biggest.

Changing colors in the waffle plot

- Color can be used to make a point about the size of a certain category.
- For example, if a goal were to examine people who own firearms but are less experienced in using firearms, they could use a bright color to highlight the group that had fired fewer rounds.
- Color is added by using a `colors =` option and listing the colors in a vector.
- Make sure the colors are assigned to the right parts of the waffle by entering the category labels for each color.

```
# change the colors
waffle(parts = rounds / 25, rows = 5,
       colors = c("1 to less than 100" = "#7463AC",
                  "100 to less than 1000" = "gray60",
                  "1000 to less than 10k" = "gray90",
                  "10k to less than 50k" = "black",
                  "50k or more" = "gray40",
                  "Don't know" = "gray80"))
```

Reviewing the waffle plot



- The bright color for the **1 to less than 100** category made this category stand out.

- The two recommended graphs for displaying a single **categorical** or **factor** type variable are *(1) bar graph*, and *(2) waffle graph*.

    - The bar graph is useful for showing relative group sizes.
    - The waffle graph is an alternative to a pie chart and is best when demonstrating parts of a whole.
    - Pie charts are available in R but are not recommended because they tend to be less clear for comparing group sizes.

Check your understanding

Create a bar graph and a waffle chart for the gender variable (`RIAGENDR`) from the NHANES 2012 data set. Examine the codebook for coding hints and clean up the data first.

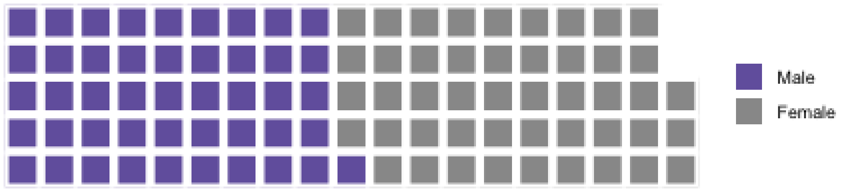
Answer

```r
# clean gender variable
nhanes.2012.clean <- nhanes.2012 %>%
  mutate(RIAGENDR = recode_factor(.x = RIAGENDR,
                                  `1` = 'Male',
                                  `2` = 'Female')) %>%
  rename(gender = RIAGENDR)

#check recoding
summary(object = nhanes.2012.clean$gender)
```

```
##   Male Female
##   4663   4701
```

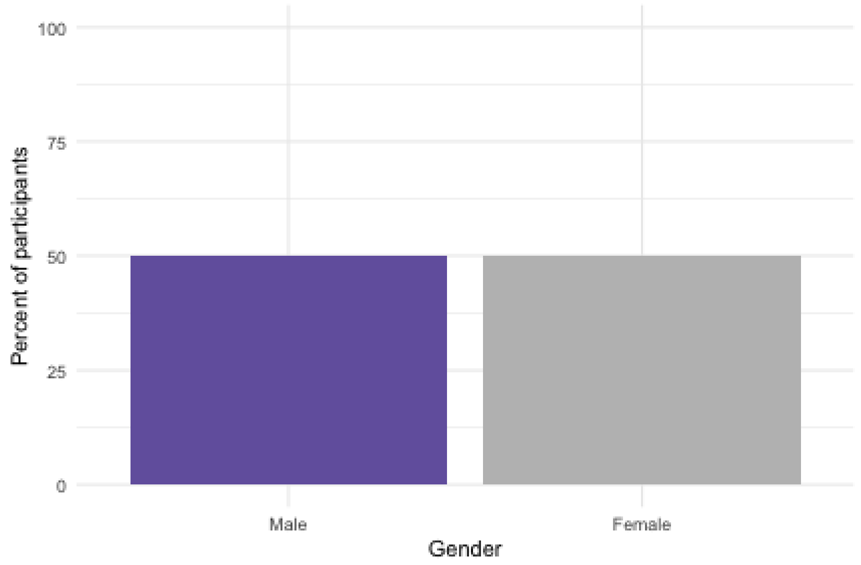
Answer waffle plot

```r
# make a table of gender
genderTable <- table(nhanes.2012.clean$gender)

# change the colors
waffle(parts = genderTable / 100, rows = 5,
       colors = c("Male" = "#7463AC",
                  "Female" = "gray60"))
```



Answer bar plot

```r
# formatted bar graph of gender
nhanes.2012.clean %>%
  drop_na(gender) %>%
  ggplot(aes(x = gender,
             y = 100*(..count..)/sum(..count..))) +
  geom_bar(aes(fill = gender)) +
  labs(x = "Gender", y = "Percent of participants") +
  scale_fill_manual(values = c("#7463AC", "gray"), guide = FALSE) +
  theme_minimal() +
  ylim(0, 100)
```


Percent of participants
Male
Female
Gender