Statistics Using R

Exploratory Data Analysis

Introduction

When your dataset is represented as a table or a database, it's difficult to observe much about it beyond its size and the types of variables it contains. In this course, you'll learn how to use graphical and numerical techniques to begin uncovering the structure of your data. Which variables suggest interesting relationships? Which observations are unusual?

Contingency table review

In this analysis I will be working with the `comics` dataset. This is a collection of characteristics on all of the superheroes created by Marvel and DC comics in the last 80 years.

Let's start by creating a contingency table, which is a useful way to represent the total counts of observations that fall into each combination of the levels of categorical variables.

```
> comics
# A tibble: 23,272 × 11
                                     name        id   align         eye        hair
                                   <fctr>    <fctr>  <fctr>      <fctr>      <fctr>
1             Spider-Man (Peter Parker)   Secret    Good Hazel Eyes Brown Hair
2        Captain America (Steven Rogers)   Public    Good  Blue Eyes White Hair
3  Wolverine (James \\"Logan\\" Howlett)   Public Neutral  Blue Eyes Black Hair
4    Iron Man (Anthony \\"Tony\\" Stark)   Public    Good  Blue Eyes Black Hair
5                   Thor (Thor Odinson) No Dual    Good  Blue Eyes Blond Hair
6              Benjamin Grimm (Earth-616)   Public    Good  Blue Eyes    No Hair
7               Reed Richards (Earth-616)   Public    Good Brown Eyes Brown Hair
8              Hulk (Robert Bruce Banner)   Public    Good Brown Eyes Brown Hair
9               Scott Summers (Earth-616)   Public Neutral Brown Eyes Brown Hair
10             Jonathan Storm (Earth-616)   Public    Good  Blue Eyes Blond Hair
# ... with 23,262 more rows, and 6 more variables: gender <fctr>, gsm <fctr>,
#   alive <fctr>, appearances <int>, first_appear <fctr>, publisher <fctr>
> # Check levels of align
> levels(comics$align)
[1] "Bad"                "Good"               "Neutral"
[4] "Reformed Criminals"
>
> # Check the levels of gender
> levels(comics$gender)
[1] "Female" "Male"   "Other"
>
```

```
> # Create a 2-way contingency table
> table(comics$align,comics$gender)

                     Female Male Other
  Bad                  1573 7561    32
  Good                 2490 4809    17
  Neutral               836 1799    17
  Reformed Criminals      1    2     0
```

Dropping levels

The contingency table from the last exercise revealed that there are some levels that have very low counts. To simplify the analysis, it often helps to drop such levels.

In R, this requires two steps: first filtering out any rows with the levels that have very low counts, then removing these levels from the factor variable with `droplevels()`. This is because the `droplevels()` function would keep levels that have just 1 or 2 counts; it only drops levels that don't exist in a dataset.

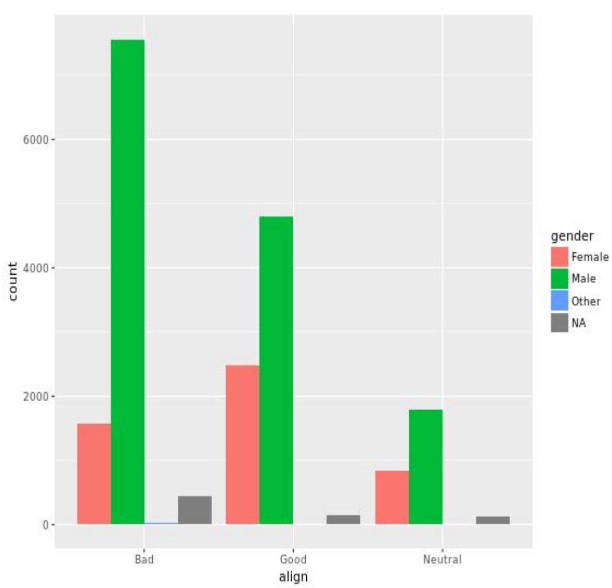
```
# Load dplyr
library(dplyr)

# Print tab
tab

# Remove align level
comics <- comics %>%
  filter(align != "Reformed Criminals") %>%
  droplevels()
```

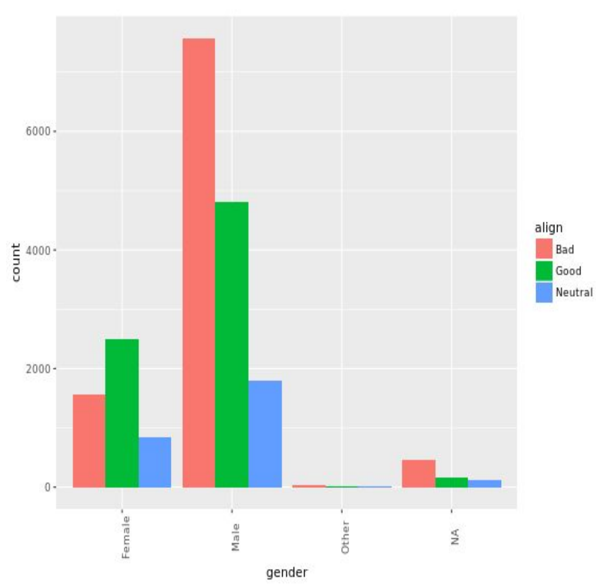
Side-by-side barcharts

While a contingency table represents the counts numerically, it's often more useful to represent them graphically.

Here you'll construct two side-by-side barcharts of the `comics` data. This shows that there can often be two or more options for presenting the same data. Passing the argument `position = "dodge"` to `geom_bar()` says that you want a side-by-side (i.e. not stacked) barchart.

```
# Load ggplot2
library(ggplot2)

# Create side-by-side barchart of gender by alignment
ggplot(comics, aes(x = align, fill = gender)) +
  geom_bar(position = "dodge")

# Create side-by-side barchart of alignment by gender
ggplot(comics, aes(x = gender, fill = align)) +
  geom_bar(position = "dodge") +
  theme(axis.text.x = element_text(angle = 90))
```





Bar chart interpretation

Which of the following interpretations of the bar charts to you right is **not** valid?

INSTRUCTIONS 50XP

Possible Answers

- Among characters with `"Neutral"` alignment, males are the most common. press 1
- In general, there is an association between gender and alignment. press 2
- (selected) Across all genders, `"Bad"` is the most common alignment. press 3
- There are more male characters than female characters in this dataset. press 4

Counts vs. proportions

Conditional proportions

```
> prop.table(tab_cnt, 1)
```
Condition on the rows (i.e. rows sum to 1)

```
          Bad  Good Neutral
No Dual 0.314 0.428   0.258
Public  0.358 0.483   0.159
Secret  0.567 0.312   0.121
Unknown 0.778 0.000   0.222
```

```
> prop.table(tab_cnt, 2)
```
Condition on the columns (i.e. columns sum to 1)

```
             Bad     Good  Neutral
No Dual 0.066331 0.106907 0.168394
Public  0.303946 0.484137 0.416667
Secret  0.628743 0.408956 0.414076
Unknown 0.000980 0.000000 0.000864
```

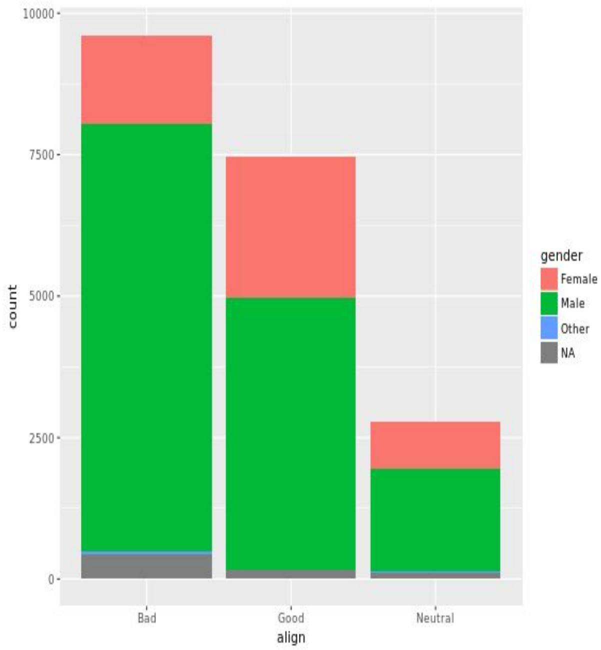
What proportion of all female characters are good?

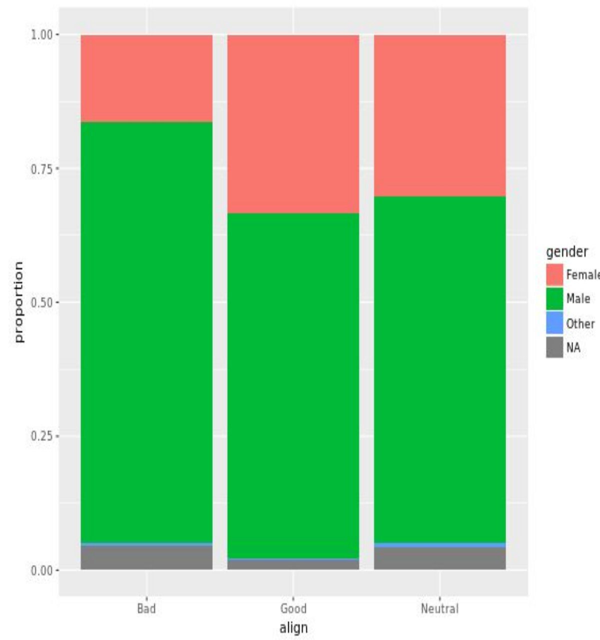
```
> tab <- table(comics$align, comics$gender)
> options(scipen = 999, digits =3)
> prop.table(tab)

           Female     Male    Other
  Bad     0.082210 0.395160 0.001672
  Good    0.130135 0.251333 0.000888
  Neutral 0.043692 0.094021 0.000888
> prop.table(tab,2)

         Female  Male Other
  Bad     0.321 0.534 0.485
  Good    0.508 0.339 0.258
  Neutral 0.171 0.127 0.258
>
```

Bar charts can tell dramatically different stories depending on whether they represent counts or proportions and, if proportions, what the proportions are conditioned on. To demonstrate this difference, you'll construct two barcharts in this exercise: one of counts and one of proportions.

SCRIPT.R

```
# Plot of gender by align
ggplot(comics, aes(x = align, fill = gender)) +
  geom_bar()

# Plot proportion of gender, conditional on align
ggplot(comics, aes(x = align, fill = gender)) +
  geom_bar(position = "fill") +
  ylab("proportion")
```





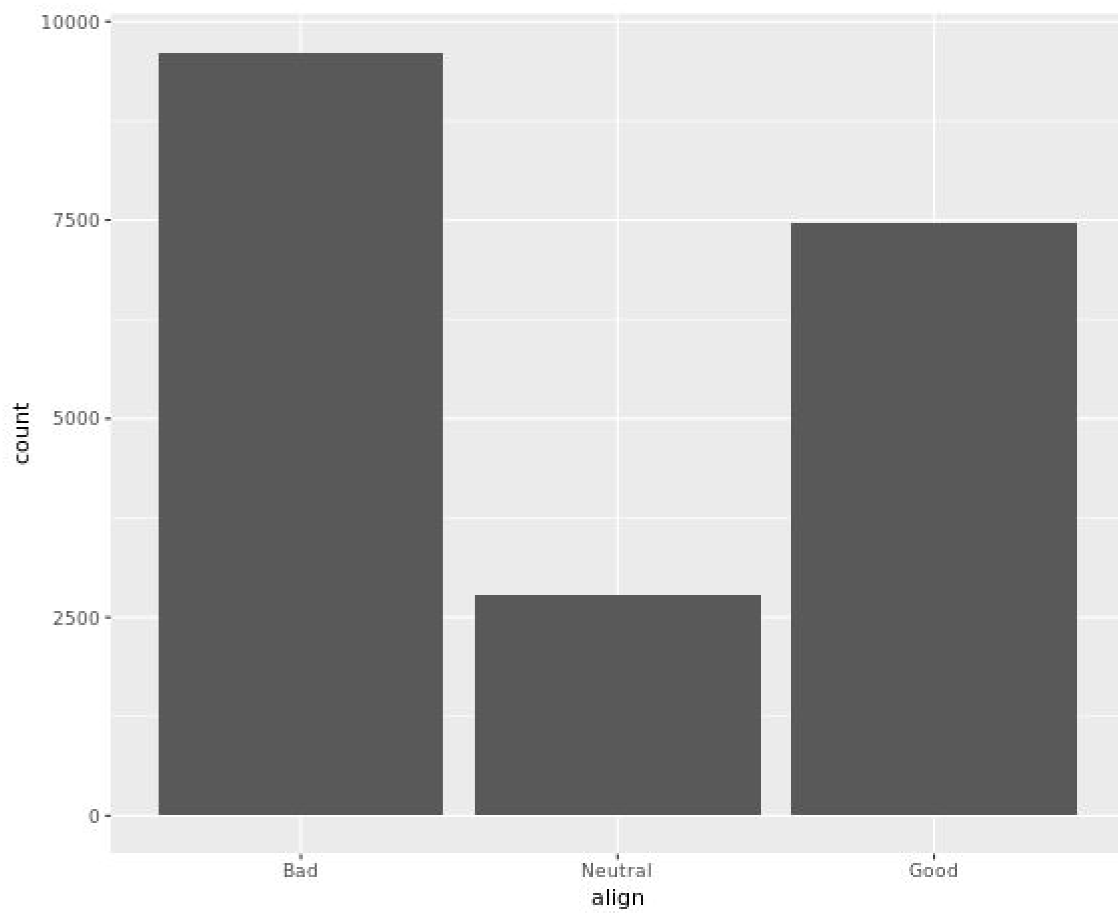
By adding position = "fill" to geom_bar(), you are saying you want the bars to fill the entire height of the plotting window, thus displaying proportions and not raw counts.

Marginal barchart

If you are interested in the distribution of alignment of *all* superheroes, it makes sense to construct a barchart for just that single variable.

You can improve the interpretability of the plot, though, by implementing some sensible ordering. Superheroes that are `"Neutral"` show an alignment between `"Good"` and `"Bad"`, so it makes sense to put that bar in the middle.

```r
# Change the order of the levels in align
comics$align <- factor(comics$align,
                       levels = c("Bad", "Neutral", "Good"))

# Create plot of align
ggplot(comics, aes(x = align)) +
  geom_bar()
```



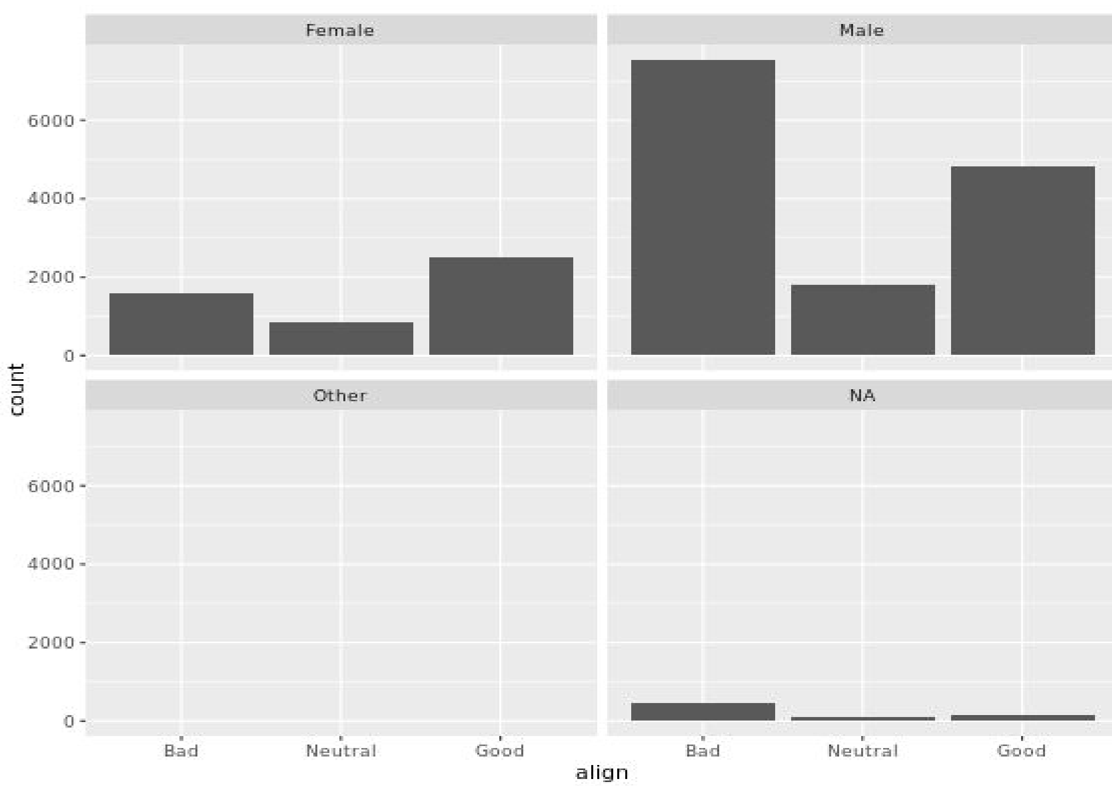
** Reordered the levels of `align` using the [`factor()`]() function so that printing them reads `"Bad"`, `"Neutral"`, then `"Good"`.

Conditional barchart

Now, if you want to break down the distribution of alignment based on gender, you're looking for conditional distributions.

You could make these by creating multiple filtered datasets (one for each gender) or by faceting the plot of alignment based on gender.

```
# Plot of alignment broken down by gender
ggplot(comics, aes(x = align)) +
  geom_bar() +
  facet_wrap(~ gender)
```

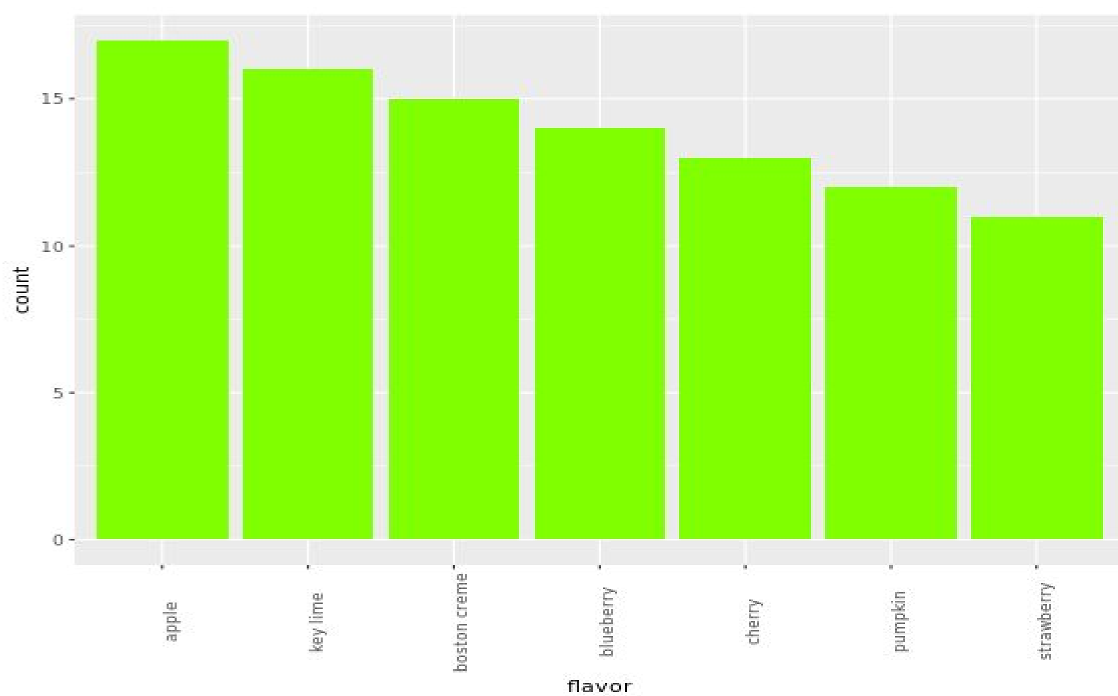


Improve piechart

The piechart is a very common way to represent the distribution of a single categorical variable, but they can be more difficult to interpret than barcharts.

This is a piechart of a dataset called `pies` that contains the favorite pie flavors of 98 people. Improve the representation of these data by constructing a *barchart* that is ordered in descending order of count.

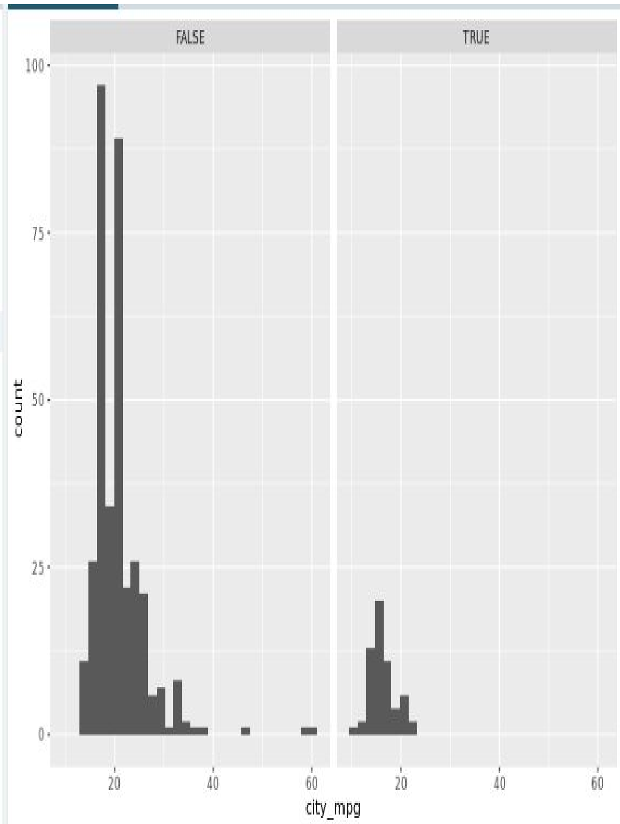
```
# Put levels of flavor in decending order
lev <- c("apple", "key lime", "boston creme",
"blueberry", "cherry", "pumpkin", "strawberry")
pies$flavor <- factor(pies$flavor, levels = lev)

# Create barchart of flavor
ggplot(pies, aes(x = flavor)) +
  geom_bar(fill = "chartreuse") +
  theme(axis.text.x = element_text(angle = 90))
```



Exploring numerical data

Faceted histogram

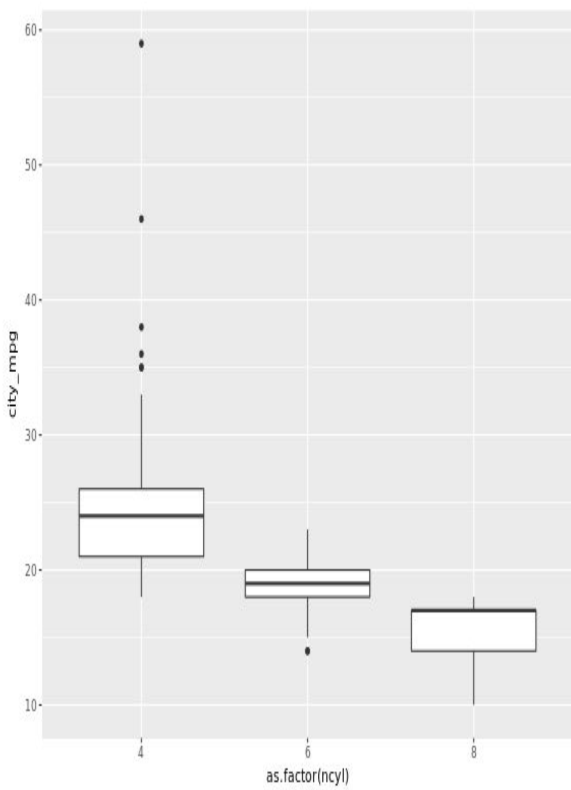
```
# Load package
library(ggplot2)

# Learn data structure
str(cars)

# Create faceted histogram
ggplot(cars, aes(x = city_mpg)) +
  geom_histogram() +
  facet_wrap(~ cars$suv)
```

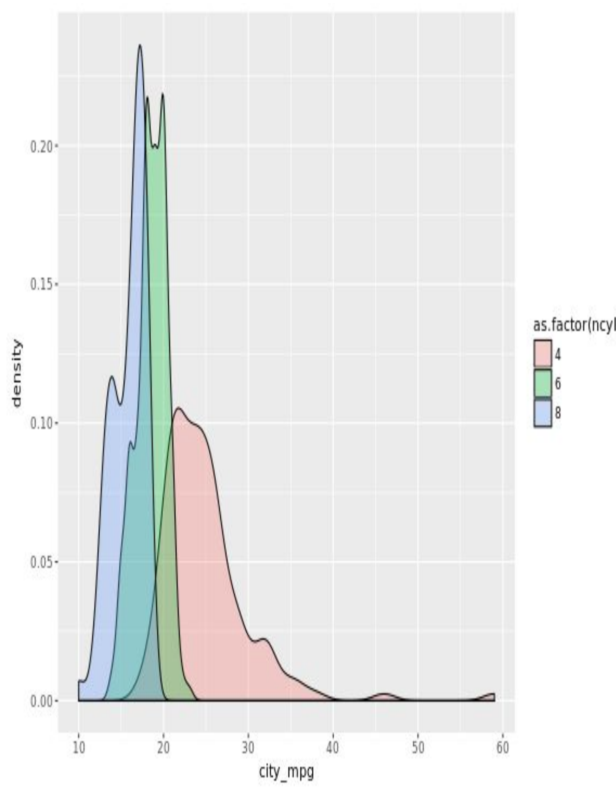


Boxplots and density plots

The mileage of a car tends to be associated with the size of its engine (as measured by the number of cylinders). To explore the relationship between these two variables, you could stick to using histograms, but in this exercise you'll try your hand at two alternatives: the box plot and the density plot.

```
# Filter cars with 4, 6, 8 cylinders
common_cyl <- filter(cars, ncyl %in% c(4,6,8))

# Create box plots of city mpg by ncyl
ggplot(common_cyl, aes(x = as.factor(ncyl), y =
city_mpg)) +
  geom_boxplot()

# Create overlaid density plots for same data
ggplot(common_cyl, aes(x = city_mpg, fill = as.factor
(ncyl))) +
  geom_density(alpha = .3)
```





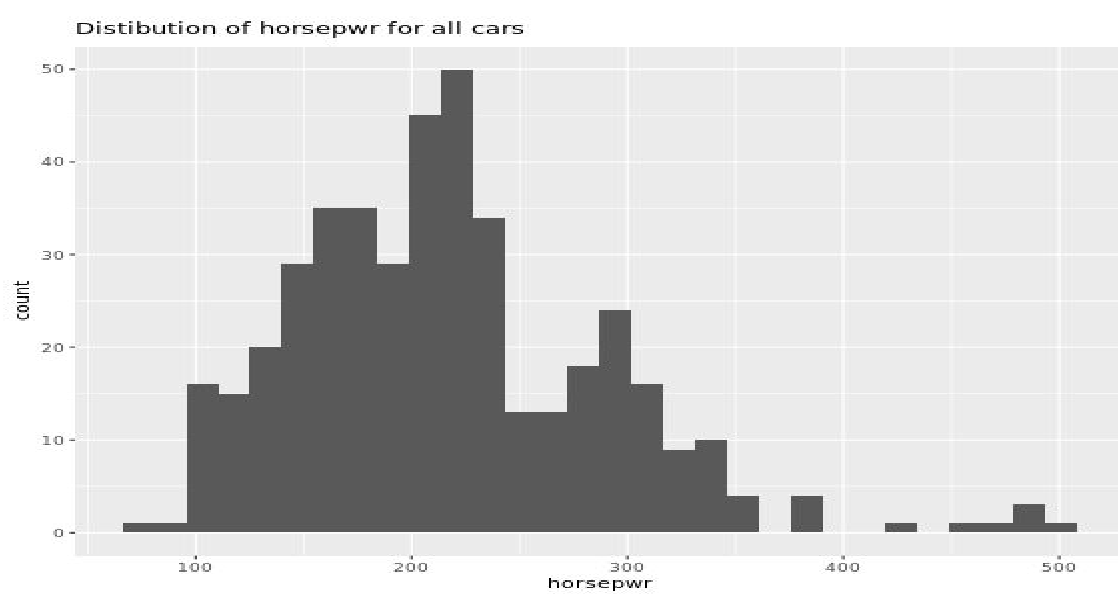
The variability in mileage of 8 cylinder cars seem much smaller than that of 4 cylinder cars.

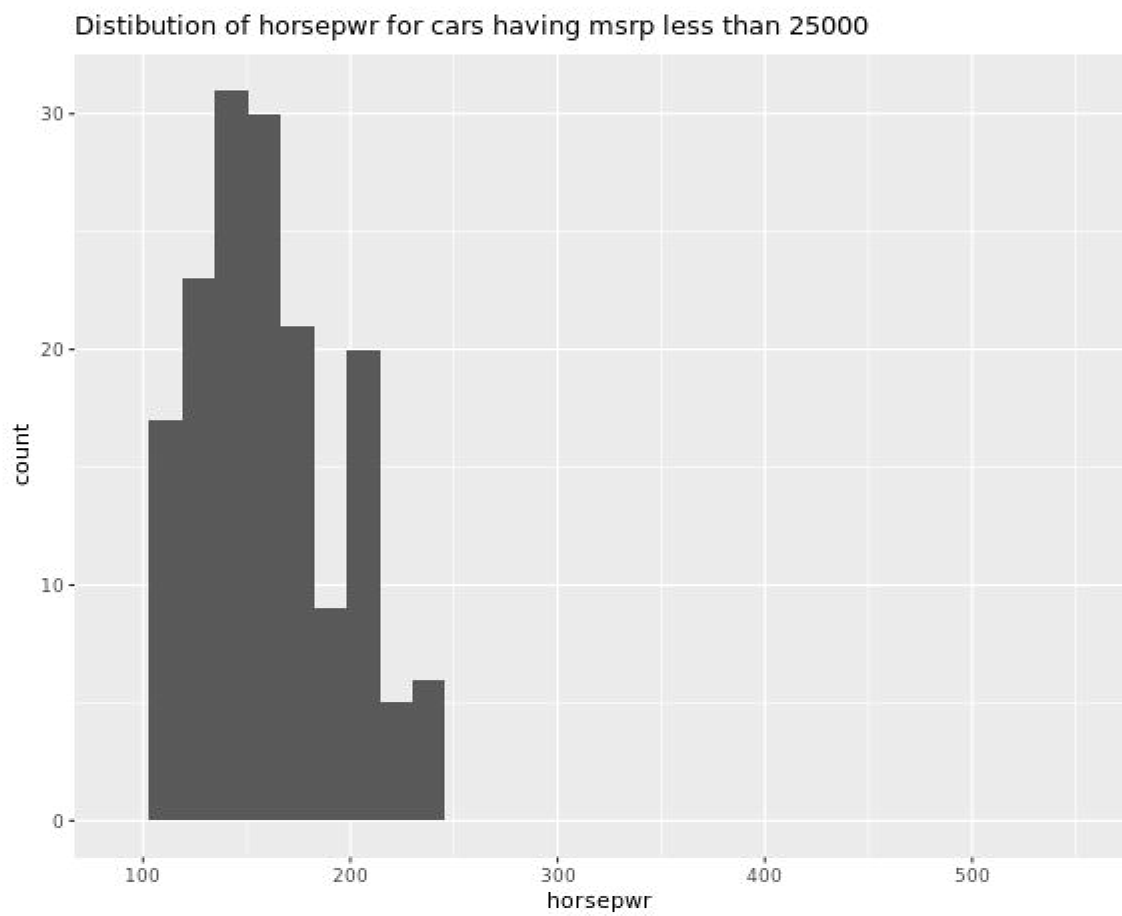
Distribution of one variable

Marginal and conditional histograms

Now, turn your attention to a new variable: `horsepwr`. The goal is to get a sense of the marginal distribution of this variable and then compare it to the distribution of horsepower conditional on the price of the car being less than $25,000.

```
# Create hist of horsepwr
cars %>%
  ggplot(aes(horsepwr)) +
  geom_histogram() +
  ggtitle("Distibution of horsepwr for all cars")

# Create hist of horsepwr for affordable cars
cars %>%
  filter(msrp < 25000) %>%
  ggplot(aes(horsepwr)) +
  geom_histogram() +
  xlim(c(90, 550)) +
  ggtitle("Distibution of horsepwr for cars having msrp less than 25000")
```



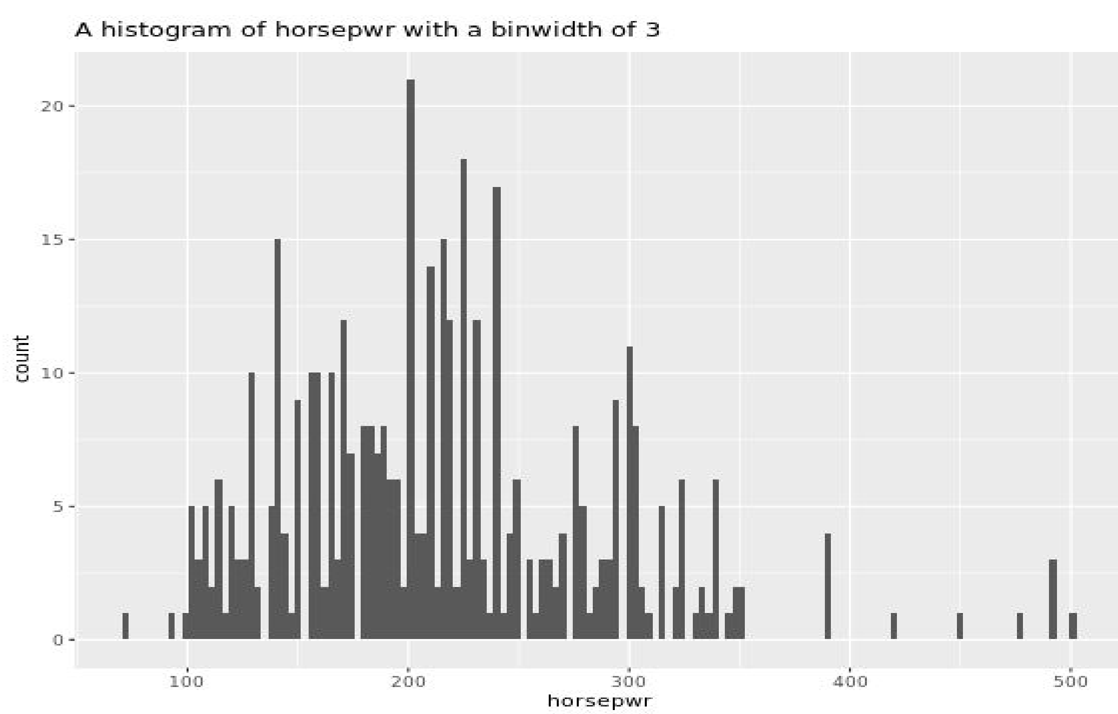


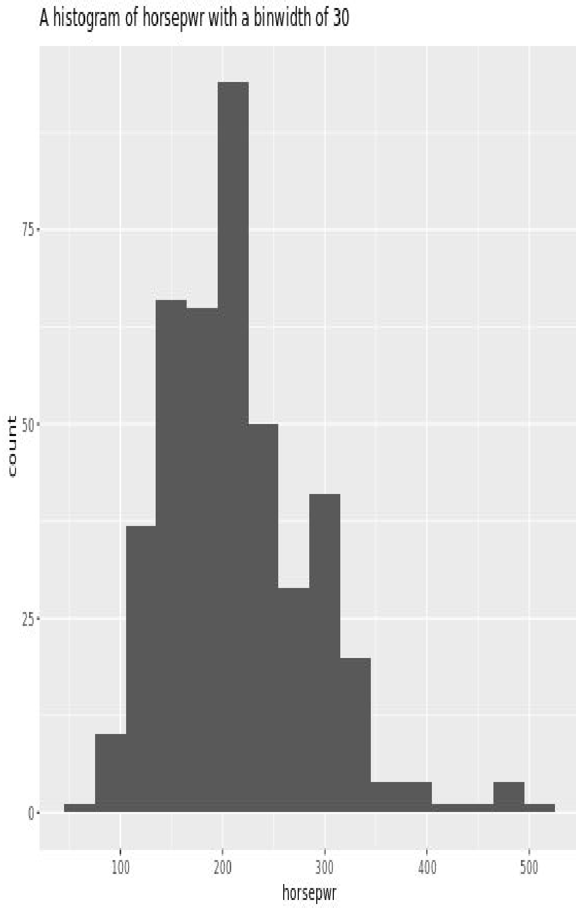
The highest horsepower car in the less expensive range has just under 250 horsepower.

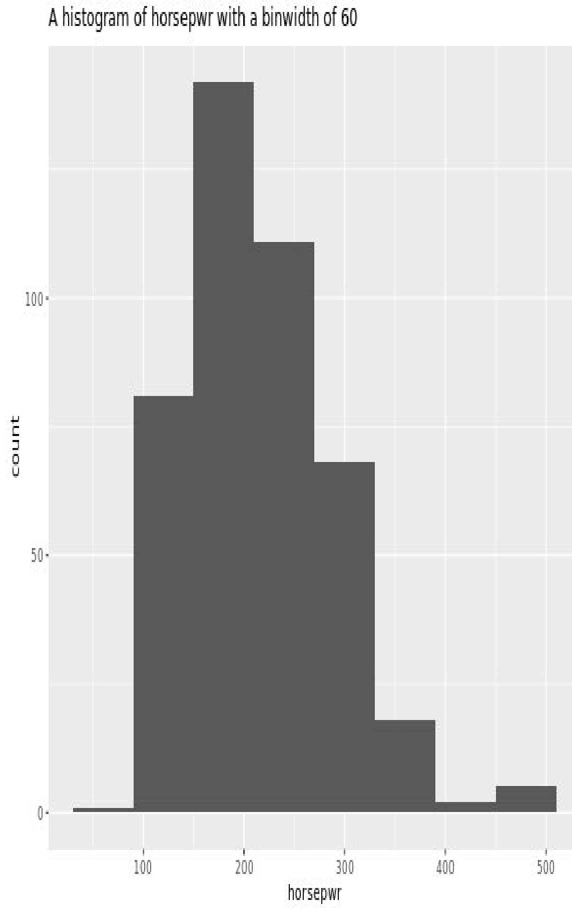
Three binwidths

Before you take these plots for granted, it's a good idea to see how things change when you alter the binwidth. The binwidth determines how smooth your distribution will appear: the smaller the binwidth, the more jagged your distribution becomes. It's good practice to consider several binwidths in order to detect different types of structure in your data.

```
# Create hist of horsepwr with binwidth of 3
cars %>%
  ggplot(aes(horsepwr)) +
  geom_histogram(binwidth = 3) +
  ggtitle("A histogram of horsepwr with a binwidth of 3")

# Create hist of horsepwr with binwidth of 30
cars %>% ggplot(aes(horsepwr)) + geom_histogram(binwidth = 30) +
ggtitle("A histogram of horsepwr with a binwidth of 30")

# Create hist of horsepwr with binwidth of 60
cars %>% ggplot(aes(horsepwr)) + geom_histogram(binwidth = 60) +
ggtitle("A histogram of horsepwr with a binwidth of 60")
```



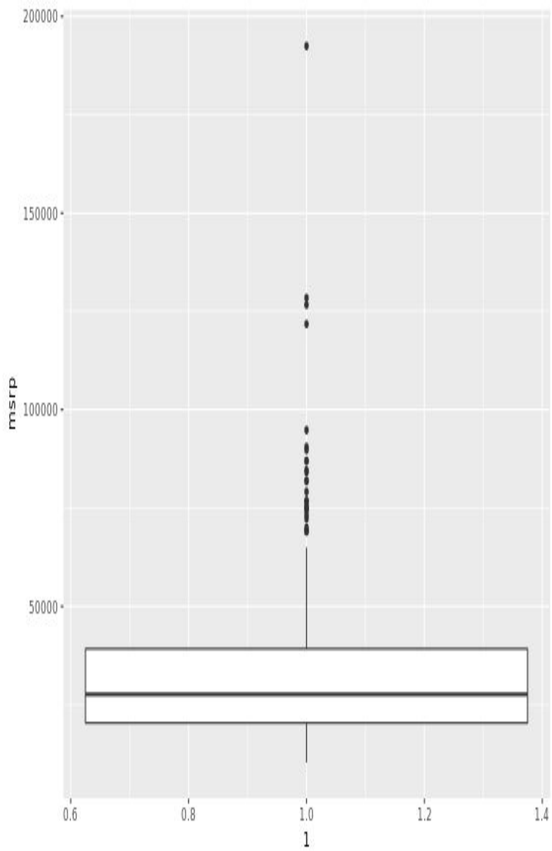

A histogram of horsepwr with a binwidth of 30
count
horsepwr
0
25
50
75
100
200
300
400
500

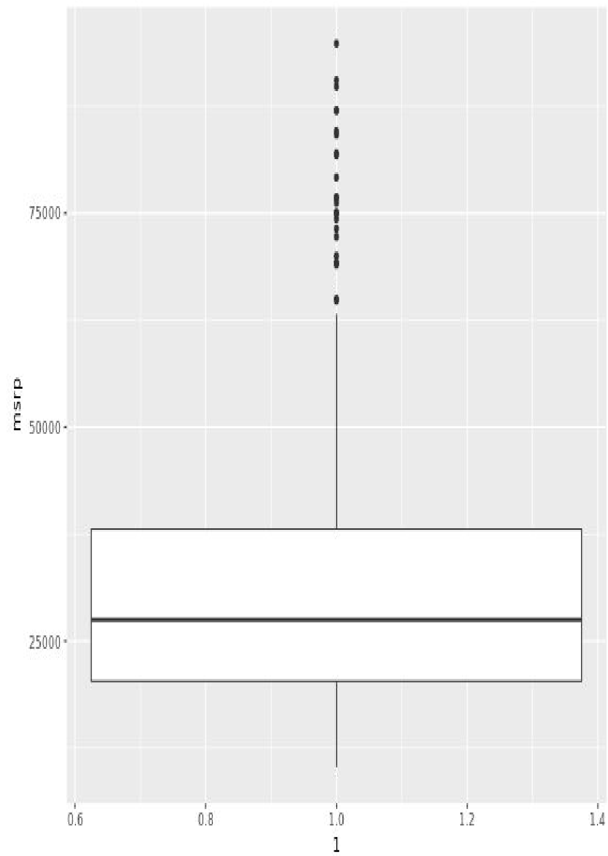

A histogram of horsepwr with a binwidth of 60
count
horsepwr
0
50
100
100
200
300
400
500

Box plots for outliers

In addition to indicating the center and spread of a distribution, a box plot provides a graphical means to detect outliers. You can apply this method to the `msrp` column (manufacturer's suggested retail price) to detect if there are unusually expensive or cheap cars.

```
# Construct box plot of msrp
cars %>%
  ggplot(aes(x = 1, y = msrp)) +
  geom_boxplot()

# Exclude outliers from data
cars_no_out <- cars %>%
  filter(msrp < 100000)

# Construct box plot of msrp using the reduced dataset
cars_no_out %>%
  ggplot(aes(x = 1, y = msrp)) +
  geom_boxplot()
```





Plot selection

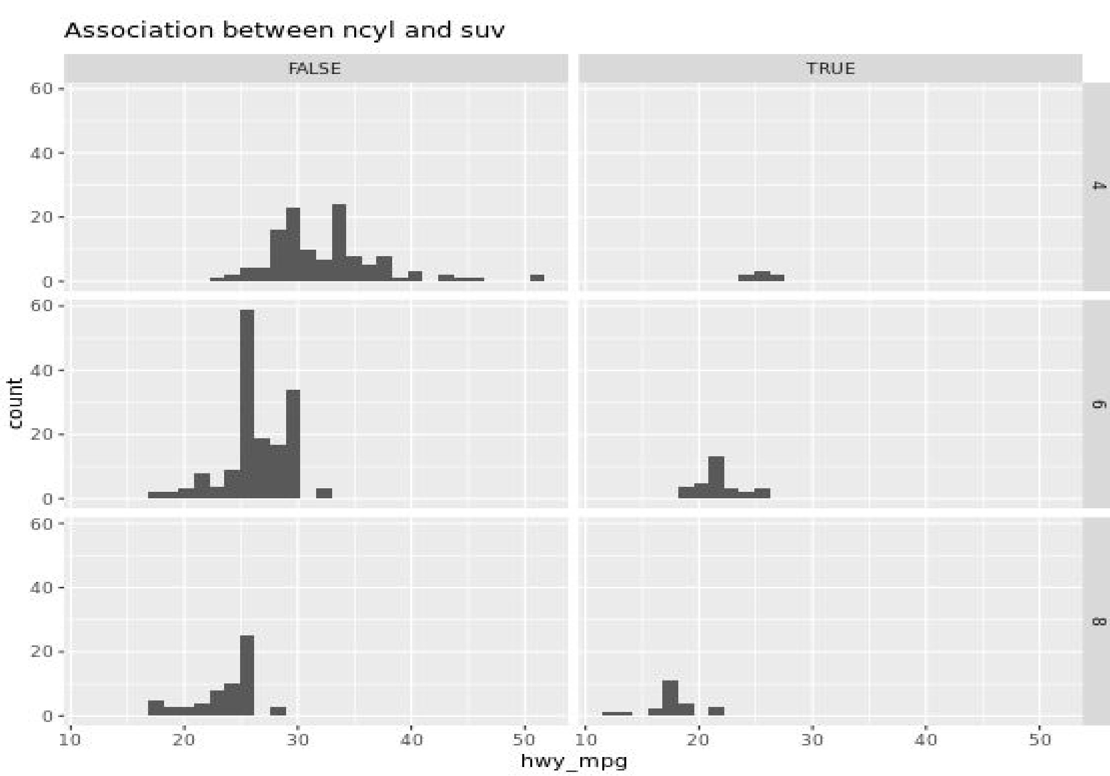
Consider two other columns in the `cars` dataset: `city_mpg` and `width`. Which is the most appropriate plot for displaying the important features of their distributions? Remember, both **density plots** and **box plots** display the **central tendency** and **spread** of the data, but the **box plot is more robust to outliers.**

Visualization in higher dimensions

<u>3 variable plot</u>

Faceting is a valuable technique for looking at several conditional distributions at the same time. If the faceted distributions are laid out in a grid, you can consider the association between a variable and two others, one on the rows of the grid and the other on the columns.

```
# Facet hists using hwy mileage and ncyl
common_cyl %>%
  ggplot(aes(x = hwy_mpg)) +
  geom_histogram() +
  facet_grid(ncyl ~ suv) +
  ggtitle("Association between ncyl and suv")
```



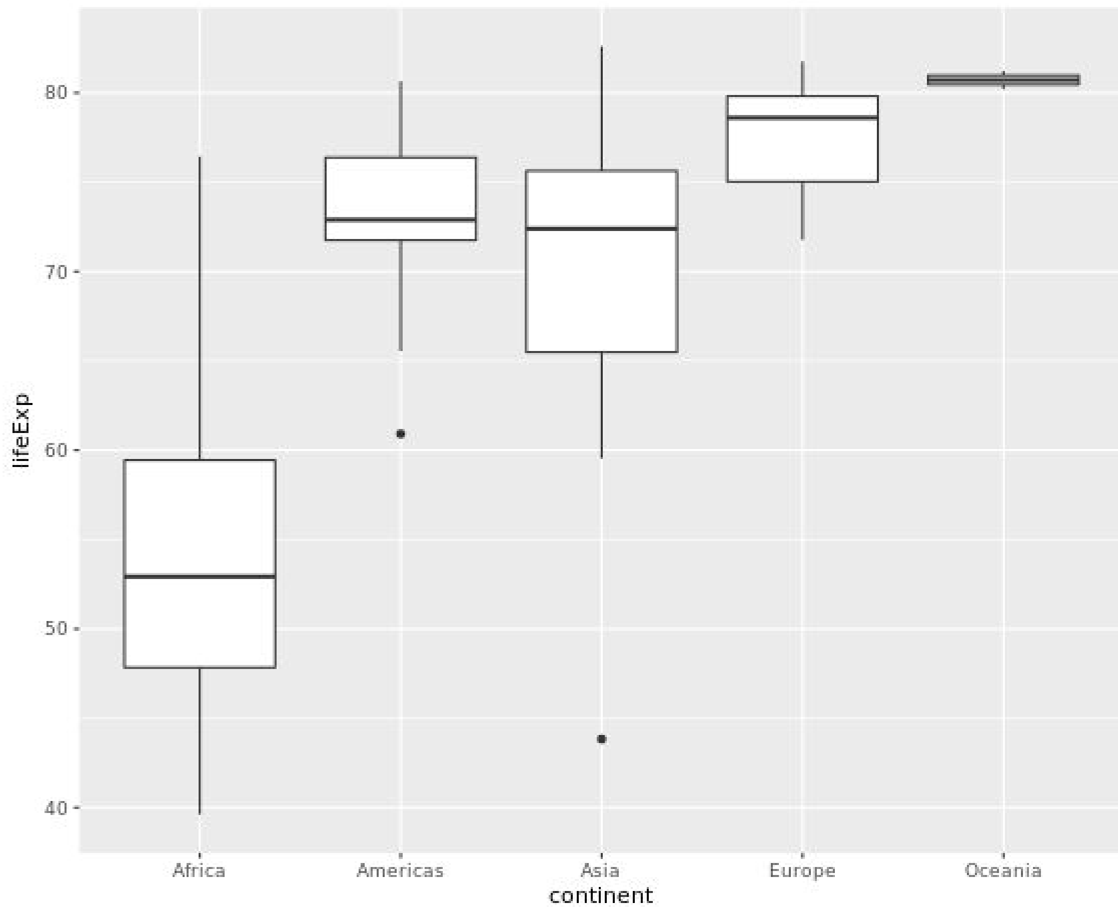
Measures of center

Calculate center measures

Throughout this chapter, you will use data from `gapminder`, which tracks demographic data in countries of the world over time. To learn more about it, you can bring up the help file with `?gapminder`.

For this exercise, focus on how the life expectancy differs from continent to continent. This requires that you conduct your analysis not at the country level, but aggregated up to the continent level. This is made possible by the one-two punch of `group_by()` and `summarize()`, a very powerful syntax for carrying out the same analysis on different subsets of the full dataset.

```
# Create dataset of 2007 data
gap2007 <- filter(gapminder, year == 2007)

# Compute groupwise mean and median lifeExp
gap2007 %>%
  group_by(continent) %>%
  summarize(mean(lifeExp),
            median(lifeExp))

# Generate box plots of lifeExp for each continent
gap2007 %>%
  ggplot(aes(x = continent, y = lifeExp)) +
  geom_boxplot()
```



Measures of variability

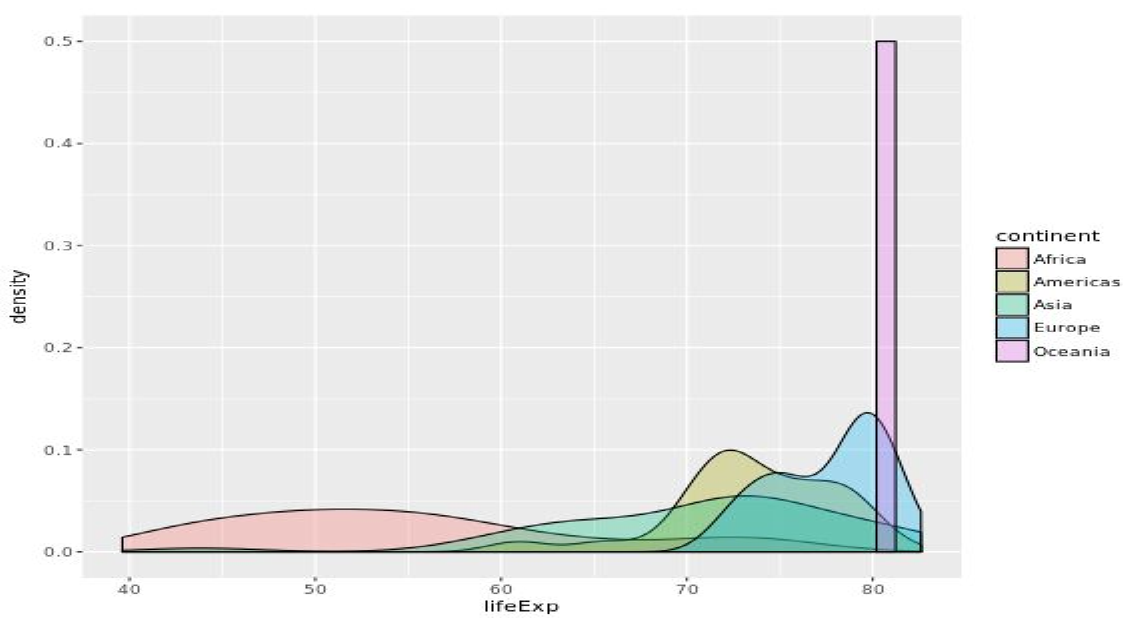
- **Standard Deviation** - gets affected by skewed values specially outliers.
- **Variance** - gets affected by skewed values specially outliers.
- **Range** - gets affected by skewed values specially outliers.
- **IQR** - Does not get affected by outliers.

Calculate spread measures

Let's extend the powerful `group_by()` and `summarize()` syntax to measures of spread. If you're unsure whether you're working with symmetric or skewed distributions, it's a good idea to consider a robust measure like IQR in addition to the usual measures of variance or standard deviation.

```
# Compute groupwise measures of spread
gap2007 %>%
  group_by(continent) %>%
  summarize(sd(lifeExp),
            IQR(lifeExp),
            n())

# Generate overlaid density plots
gap2007 %>%
  ggplot(aes(x = lifeExp, fill = continent)) +
  geom_density(alpha = 0.3)
```

```
# A tibble: 5 × 4
  continent `sd(lifeExp)` `IQR(lifeExp)` `n()`
     <fctr>         <dbl>          <dbl> <int>
1    Africa     9.6307807       11.61025    52
2  Americas     4.4409476        4.63200    25
3      Asia     7.9637245       10.15200    33
4    Europe     2.9798127        4.78250    30
5   Oceania     0.7290271        0.51550     2
```

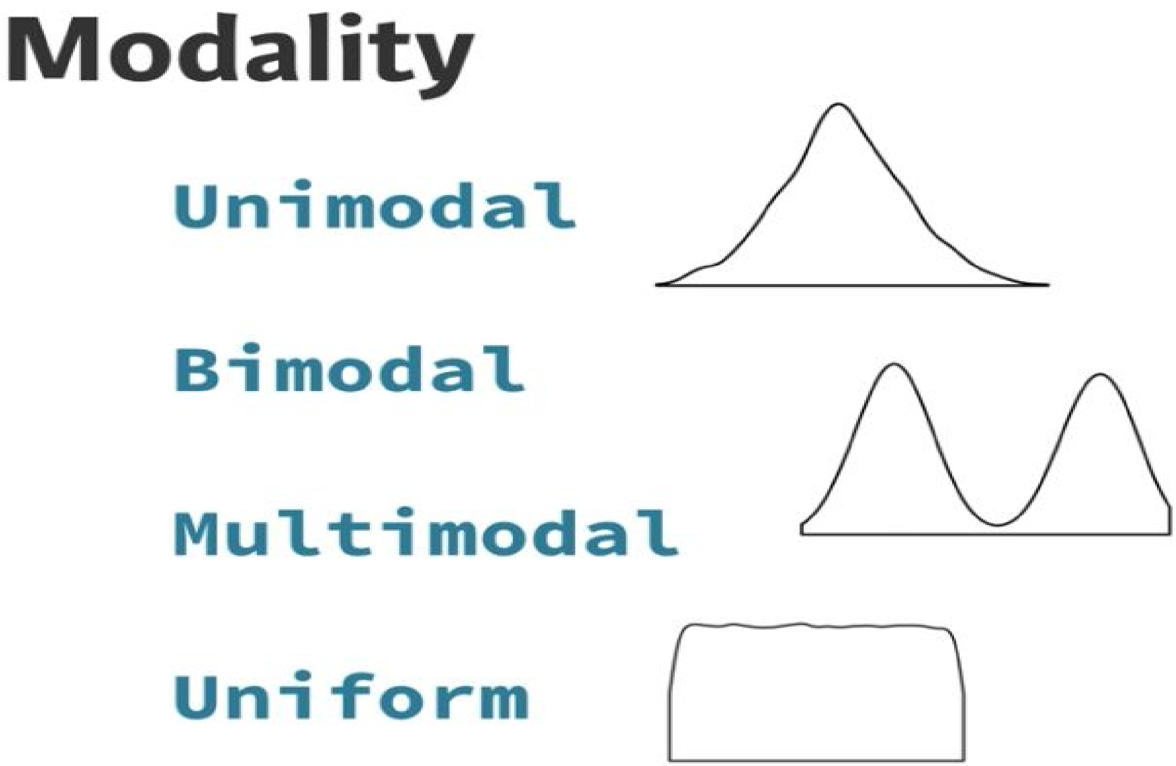


Like mean and standard deviation, median and IQR measure the central tendency and spread, respectively, but are robust to outliers and non-normal data.

Shape and transformations

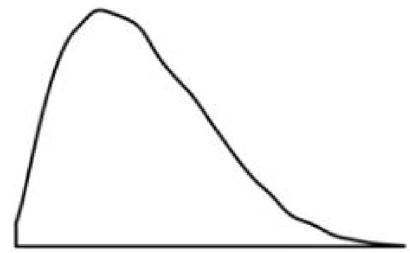
There are generally four characteristics of distributions that are of interest.

1. The **Center**
2. The **spread** or variability of the distribution
3. The **shape** of the distribution which can be described in terms of modality and the skewness.
   The modality of distribution is the number of prominent humps that shows up in the distribution.
4. **Outliers**

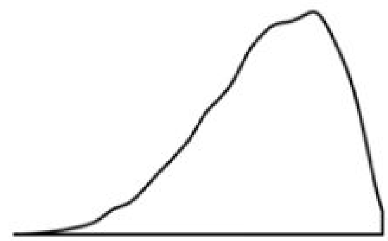


Skew

Right-skewed



Left-skewed

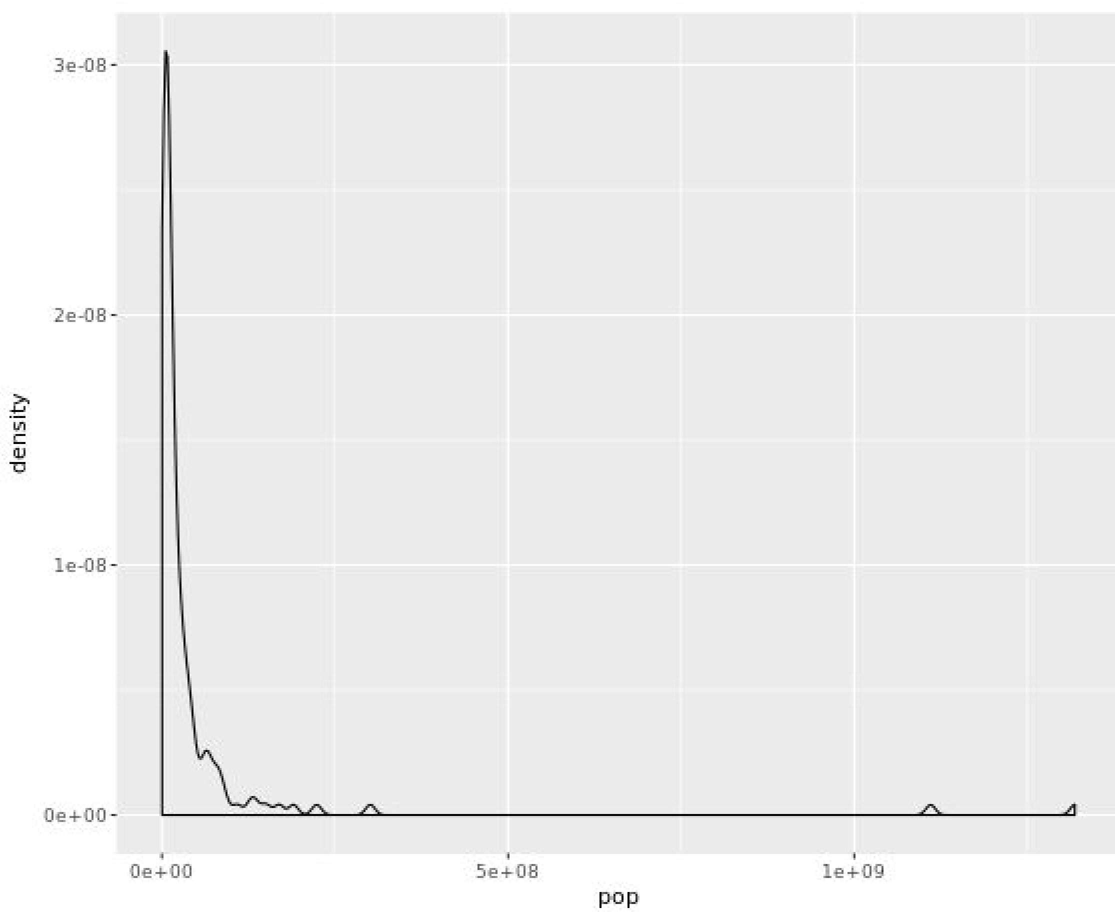


Transformations

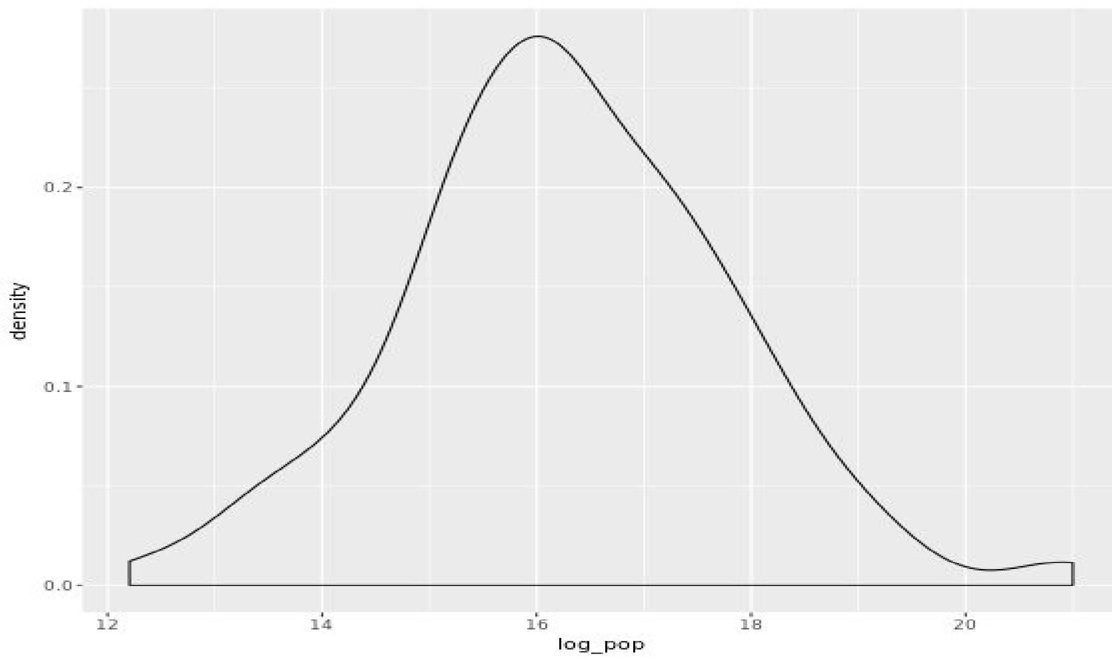
Highly skewed distributions can make it very difficult to learn anything from a visualization. Transformations can be helpful in revealing the more subtle structure.

Here you'll focus on the population variable, which exhibits strong right skew, and transform it with the natural logarithm function (`log()` in R).

```
# Create density plot of old variable
gap2007 %>%
  ggplot(aes(x = pop)) +
  geom_density(alpha = 0.3)
```



```
# Transform the skewed pop variable
gap2007 <- gap2007 %>%
  mutate(log_pop = log(pop))

# Create density plot of new variable
gap2007 %>%
  ggplot(aes(x = log_pop)) +
  geom_density(alpha = 0.3)
```



Outliers

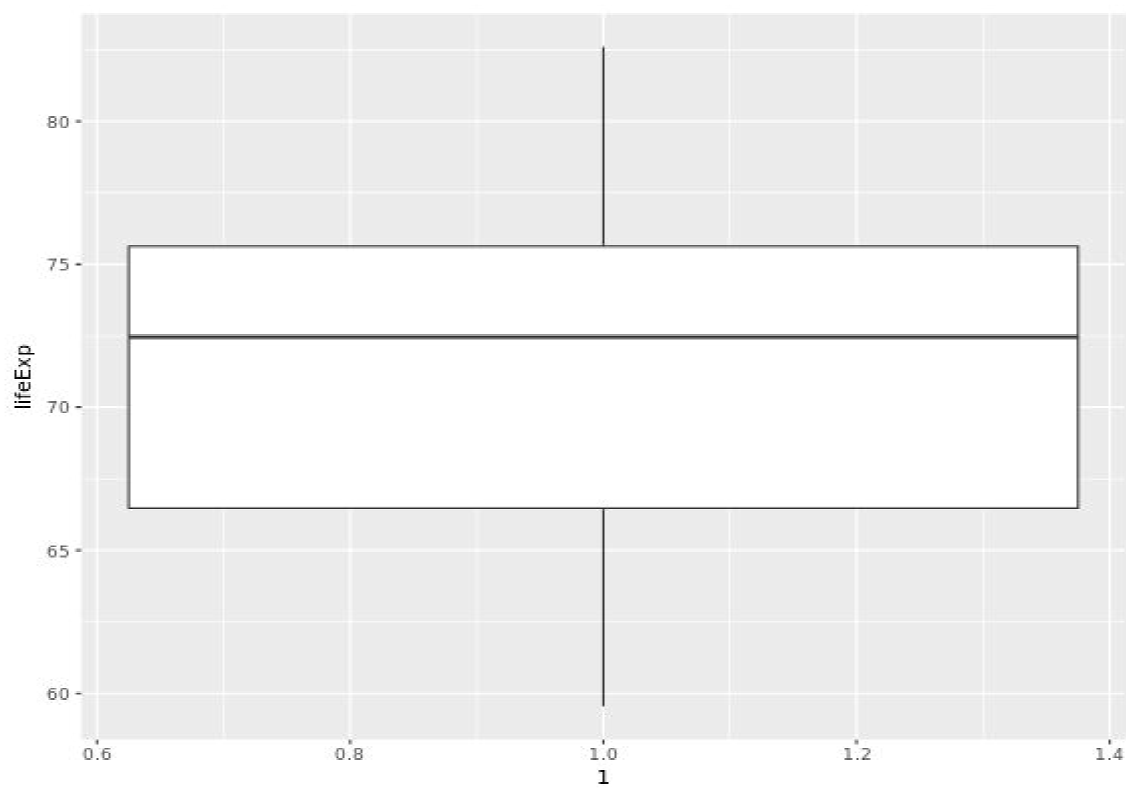
Four important characteristics of a distribution: -

1. Center
2. Variability
3. Shape
4. Outliers

Identify outliers

Consider the distribution, shown here, of the life expectancies of the countries in Asia. The box plot identifies one clear outlier: a country with a notably low life expectancy. Do you have a guess as to which country this might be? Test your guess in the console using either `min()` or `filter()`, then proceed to building a plot with that country removed.

```
# Filter for Asia, add column indicating outliers
gap_asia <- gap2007 %>%
  filter(continent == 'Asia') %>%
  mutate(is_outlier = lifeExp < 50)

# Remove outliers, create box plot of lifeExp
gap_asia %>%
  filter( !is_outlier) %>%
  ggplot(aes(x = 1, y = lifeExp)) +
  geom_boxplot()
```



Spam and num_char

Is there an association between spam and the length of an email? You could imagine a story either way:

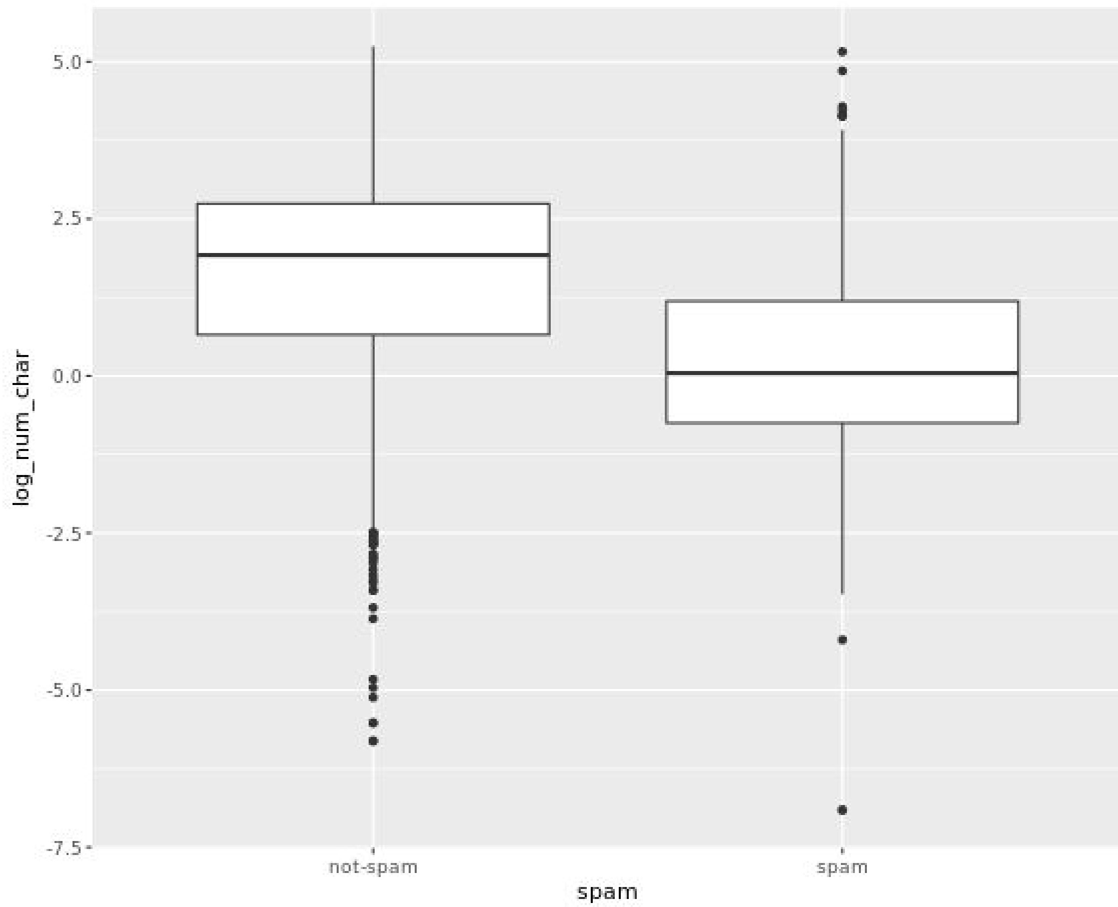
- *Spam is more likely to be a short message tempting me to click on a link*, or
- *My normal email is likely shorter since I exchange brief emails with my friends all the time*.

For this analysis email dataset is used.

These data represent incoming emails for the first three months of 2012 for an email account.

- Exploring the association between spam and the length of an email.

```
# Load packages
library(ggplot2)
library(dplyr)
library(openintro)



# Compute summary statistics
email %>%
  group_by(spam) %>%
  summarize(median(num_char),IQR(num_char))

# Create plot
email %>%
  mutate(log_num_char = log(num_char)) %>%
  ggplot(aes(x = spam, y = log_num_char)) +
geom_boxplot()
```

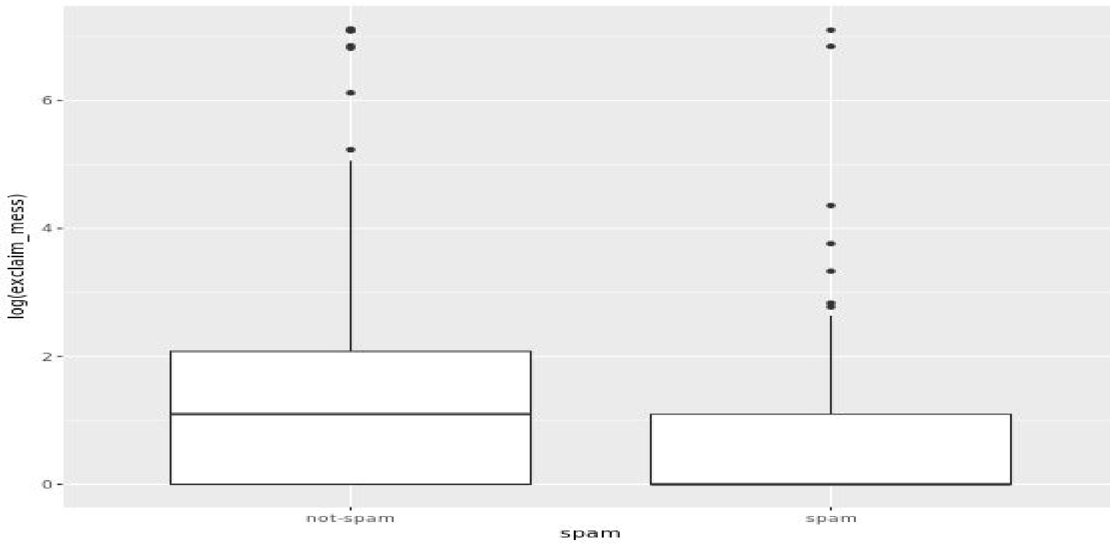


Plot Interpretation - The median length of not-spam emails is greater than that of spam emails.

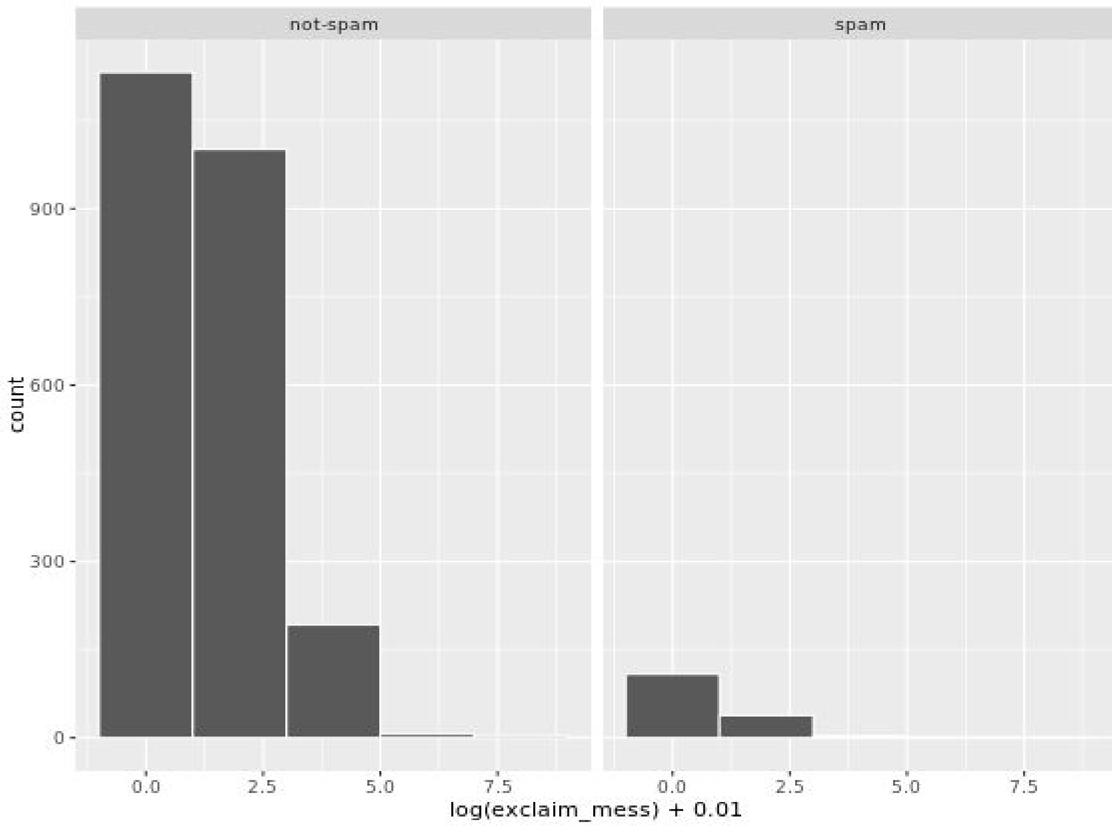
```
# Compute center and spread for exclaim_mess by spam
email %>% group_by(spam) %>% summarise(median
(exclaim_mess), IQR(exclaim_mess))
```

```
# A tibble: 2 × 3
      spam `median(exclaim_mess)` `IQR(exclaim_mess)`
    <fctr>                  <dbl>                <dbl>
1 not-spam                      1                    5
2     spam                      0                    1
```

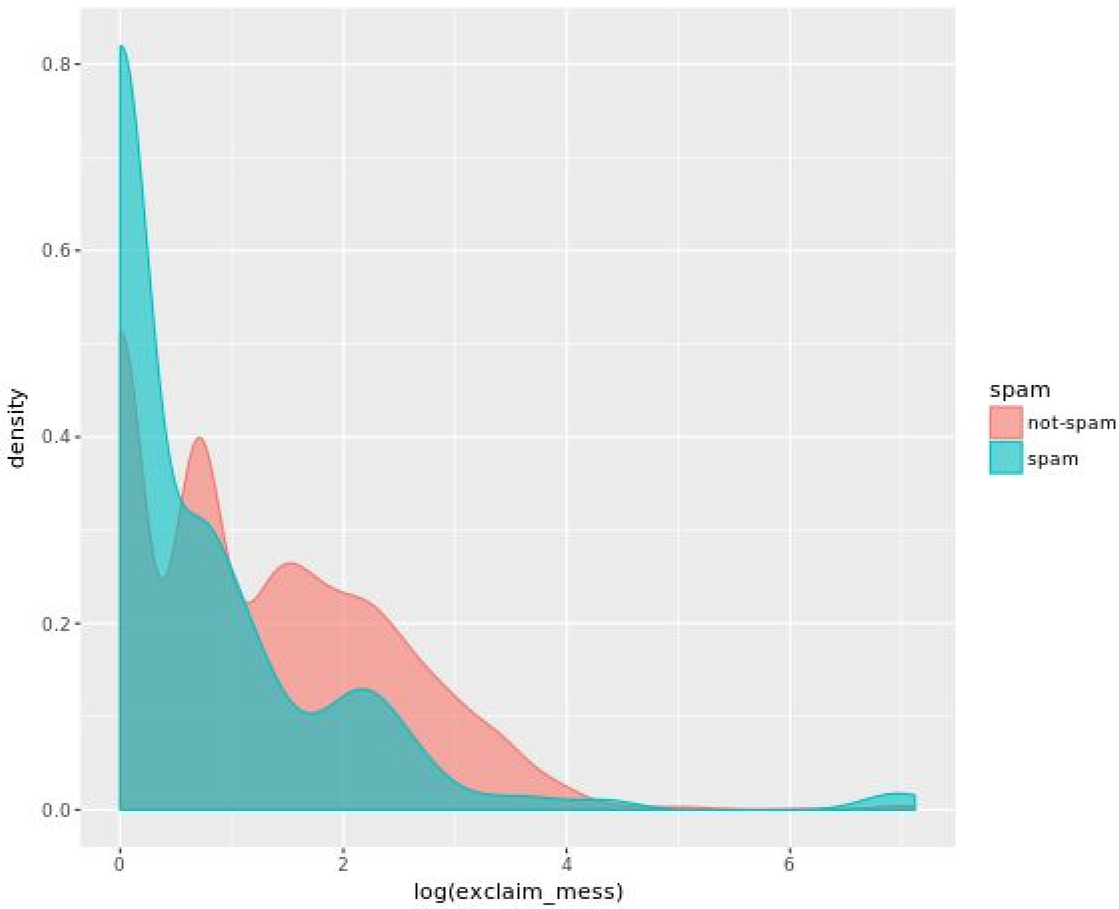
```
# Create plot for spam and exclaim_mess
ggplot(email, aes(x = spam, y= log(exclaim_mess))) +
geom_boxplot()
```



```
ggplot(email, aes(log(exclaim_mess)+0.01)) +
geom_histogram(binwidth=2,colour="white") + facet_grid
(~ spam)

```



```
11  ggplot(email, aes(log(exclaim_mess), color=spam, fill
    =spam)) + geom_density(alpha=0.6)
```



```
# Create plot of proportion of spam by image
email %>%
  mutate(has_image = image > 0) %>%
  ggplot(aes(x = has_image, fill = spam)) +
  geom_bar(position = "fill")
```

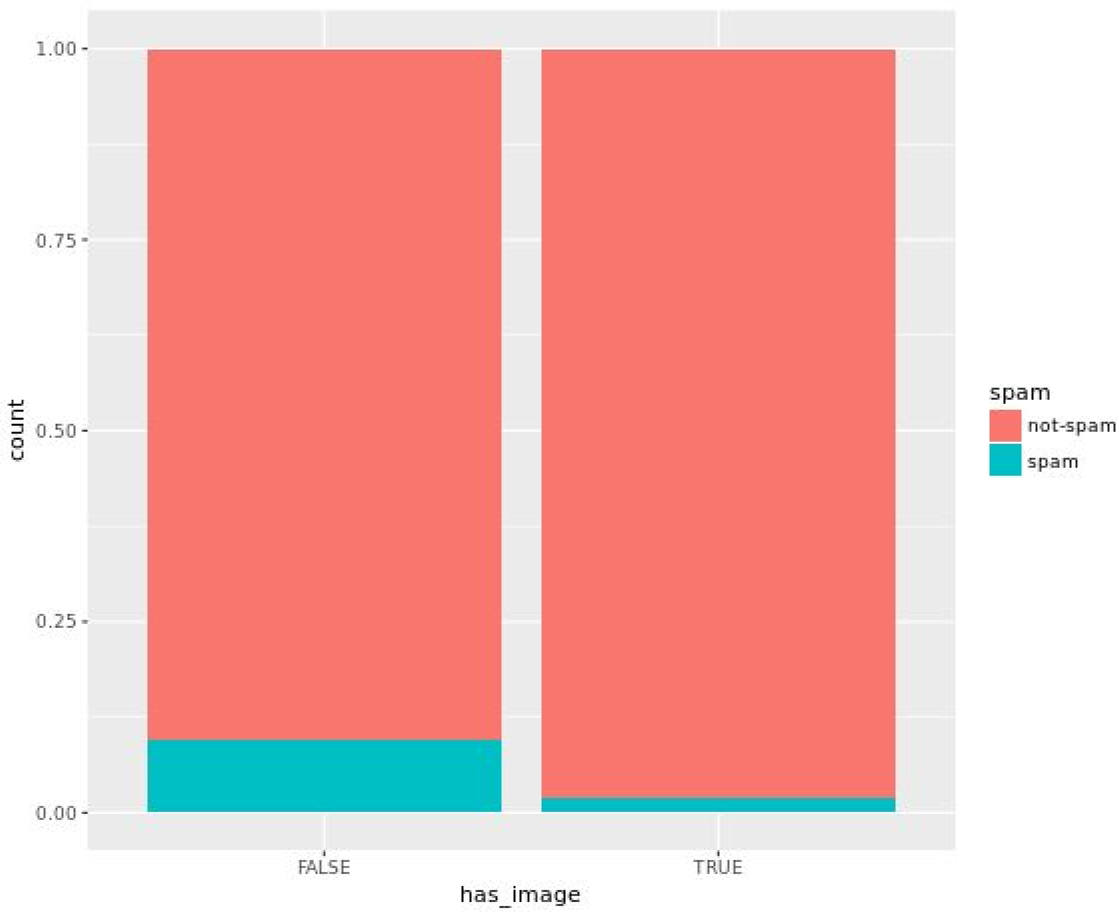


Image and spam interpretation

Which of the following interpretations of the plot is valid?

- **An email without an image is more likely to be not-spam than spam.**

Data Integrity

In the process of exploring a dataset, you'll sometimes come across something that will lead you to question how the data were compiled. For example, the variable `num_char` contains the number of characters in the email, in thousands, so it could take decimal values, but it certainly shouldn't take negative values.

You can formulate a test to ensure this variable is behaving as we expect:

```
email$num_char < 0
```

If you run this code at the console, you'll get a long vector of logical values indicating for each case in the dataset whether that condition is `TRUE`. Here, the first 1000 values all appear to be `FALSE`. To verify that *all* of the cases indeed have non-negative values for `num_char`, we can take the *sum* of this vector:

```
sum(email$num_char < 0)
```

This is a handy shortcut. When you do arithmetic on logical values, R treats `TRUE` as `1` and `FALSE` as `0`. Since the sum over the whole vector is zero, you learn that every case in the dataset took a value of `FALSE` in the test. That is, the `num_char` column is behaving as we expect and taking only non-negative values.

```
> # Test if images count as attachments
> sum(email$image > email$attach)
[1] 0
```

Since image is never greater than attach, we can infer that images are counted as attachments.

Answering questions with chains

When you have a specific question about a dataset, you can find your way to an answer by carefully constructing the appropriate chain of R code. For example, consider the following question:

"Within non-spam emails, is the typical length of emails shorter for those that were sent to multiple people?"

This can be answered with the following chain:

```
email %>%
   filter(spam == "not-spam") %>%
   group_by(to_multiple) %>%
   summarize(median(num_char))
```

The code makes it clear that you are using `num_char` to measure the length of an email and `median()` as the measure of what is typical. If you run this code, you'll learn that the answer to the question is "yes": the typical length of non-spam sent to multiple people is a bit lower than those sent to only one person.

This chain concluded with summary statistics, but others might end in a plot; it all depends on the question that you're trying to answer.

Build a chain to answer each of the following questions, both about the variable `dollar`.

- For emails containing the word "dollar", does the typical spam email contain a greater number of occurrences of the word than the typical non-spam email? Create a summary statistic that answers this question.

email %>%

filter(dollar > 0) %>%

group_by(spam) %>%

summarize(median(dollar))

```
# A tibble: 2 × 2
      spam `median(dollar)`
    <fctr>            <dbl>
1 not-spam                4
2     spam                2
```

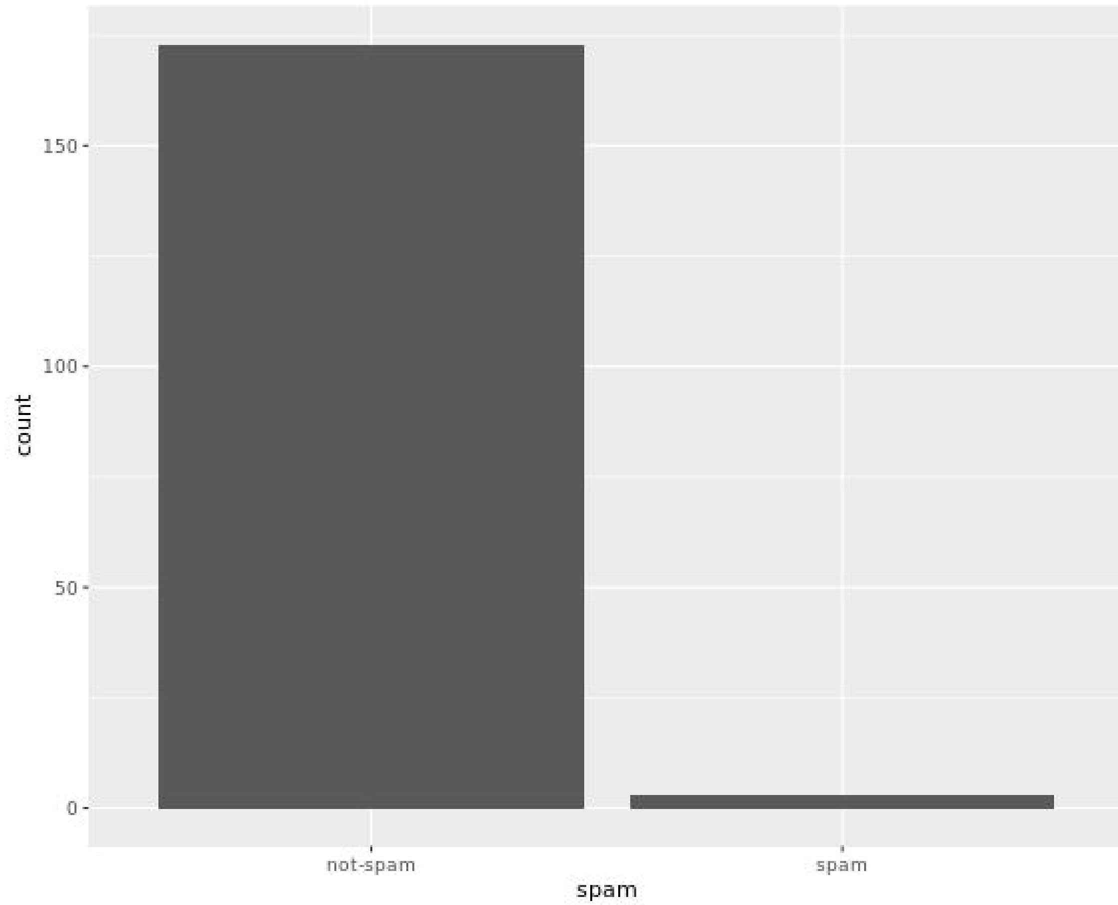
- If you encounter an email with greater than 10 occurrences of the word "dollar", is it more likely to be spam or not-spam? Create a barchart that answers this question.

email %>%

filter(dollar > 10) %>%

ggplot(aes(x = spam)) +

geom_bar()



What's in a number?

Turn your attention to the variable called `number`. Read more about it by pulling up the help file with `?email`.

To explore the association between this variable and `spam`, select and construct an informative plot. For illustrating relationships between categorical variables, you've seen

- Faceted barcharts
- Side-by-side barcharts
- Stacked and normalized barcharts.

```r
# Reorder levels
email$number <- factor(email$number, levels = c("none",
"small", "big"))

# Construct plot of number
ggplot(email, aes(x = number)) +
  geom_bar() +
  facet_wrap(~ spam)
```