Introduction to Data Visualization

LTC Melanie Vinton

2017-11-09

Contents

Chapter 1

Visualization Module Intro

First, restart your R session.

Motivation

The purpose of visualization is improve or reinforce human cognition of abstract data.

This module is an introduction to visualization in R using the **ggplot2** package. Many of the most impressive visualizations seen today are made in R with packages like ggplot2.

Here's some examples: http://www.r-graph-gallery.com/portfolio/ggplot2-package

Some elements of this module use the following reference: http://jcyhong.github.io/ggplot_demo.html

Chapter 2

ggplot2 Package

The **ggplot2** package is based on the grammar of graphics, which breaks graphics into parts which are controlled separately and combined.

- Plots are constructed in layers, which can be more intuitive.
- This package is among the most useful and widely used for visualization in R, created by Hadley Wickham.
- There are plotting functions available in base R but **ggplot2** provides a powerful, intuitive framework for creating and customizing visualizations.

2.1 Layering

Every graph has the same basic components:

1. Data

- usually a data frame

2. Aesthetic mapping

- how variables in the data frame map to visual objects on the graph
- objects (or aesthetics) to map to include x-axis, y-axis, size, shape, color, transparency
- specified with the `aes()` function

3. Geoms that represent data points visually.

- specifies the type of plot
- geom = geometric object
- examples (see the cheatsheet!): `geom_line()`, `geom_point()`, `geom_bar()`, `geom_histogram()`

2.2 Packages

First, lets install the packages we will use in this module.

```
library(ggplot2)
library(RColorBrewer)
library(plyr)
library(dplyr)
library(lubridate)
library(scales)
library(plotly)
```

2.3 References

Some useful references include:

- http://www.cookbook-r.com
- http://ggplot2.tidyverse.org/reference/
- https://www.rstudio.com/resources/cheatsheets/
- "R Graphics Cookbook" by Winston Chang (PDF can be found online at http://ase.tufts.edu/bugs/guide/assets/R%20Graphics%20Cookbook.pdf)
- http://www.google.com

Chapter 3

Scatterplots

A scatterplot displays two variables from a data set. It is often used to explore the relationship between those variables. You can also add variables to the visualization with different colors, shapes, and sizes of the dots.

3.1 Cars Data

We will start with a data set of 32 cars and 11 performance and design features.

- This data was taken from a 1974 issue of Motor Trend magazine and is built into base R as a dataframe called "mtcars".
- It is often used for educational or demonstration purposes.
- We will read in the data from a CSV because that is the way you would handle data in the real world.

```
cars <- read.csv("mtcars.csv", stringsAsFactors = FALSE)
```

```
head(cars)
```

```
##                 car  mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1         Mazda RX4 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## 2     Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## 3        Datsun 710 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## 4    Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## 5 Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## 6           Valiant 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

To get the variable descriptions, call the "mtcars" dataset in help.

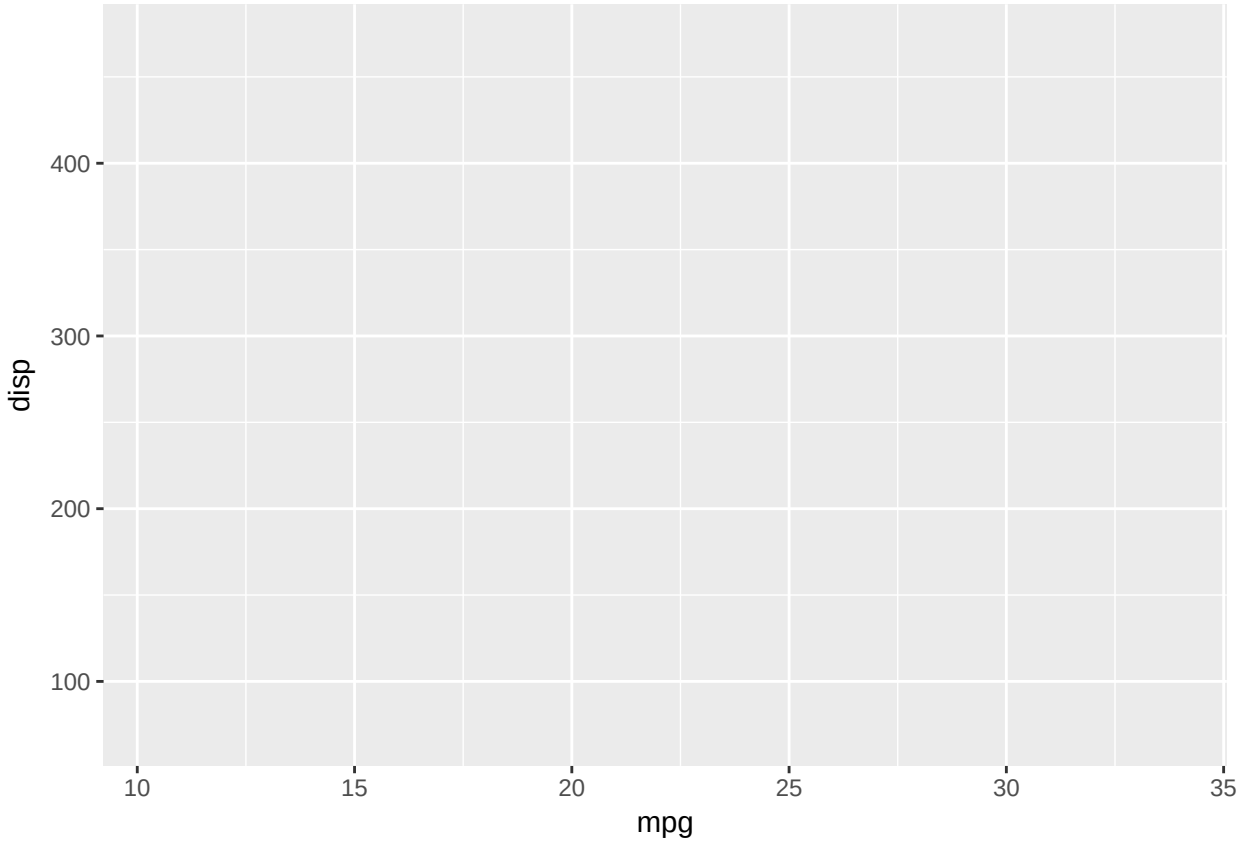
```
?mtcars
```

3.2 Scatterplot

Let's create a scatterplot of miles per gallon (mpg) versus engine size (disp).

The first step is to initialize our plot using the `ggplot()` function. This creates a coordinate system that we will then add layers to. We identify the data set to use with the "data" parameter and map the x and y aesthetics with `aes()`.

```
ggplot(data = cars, aes(x = mpg, y = disp))
```

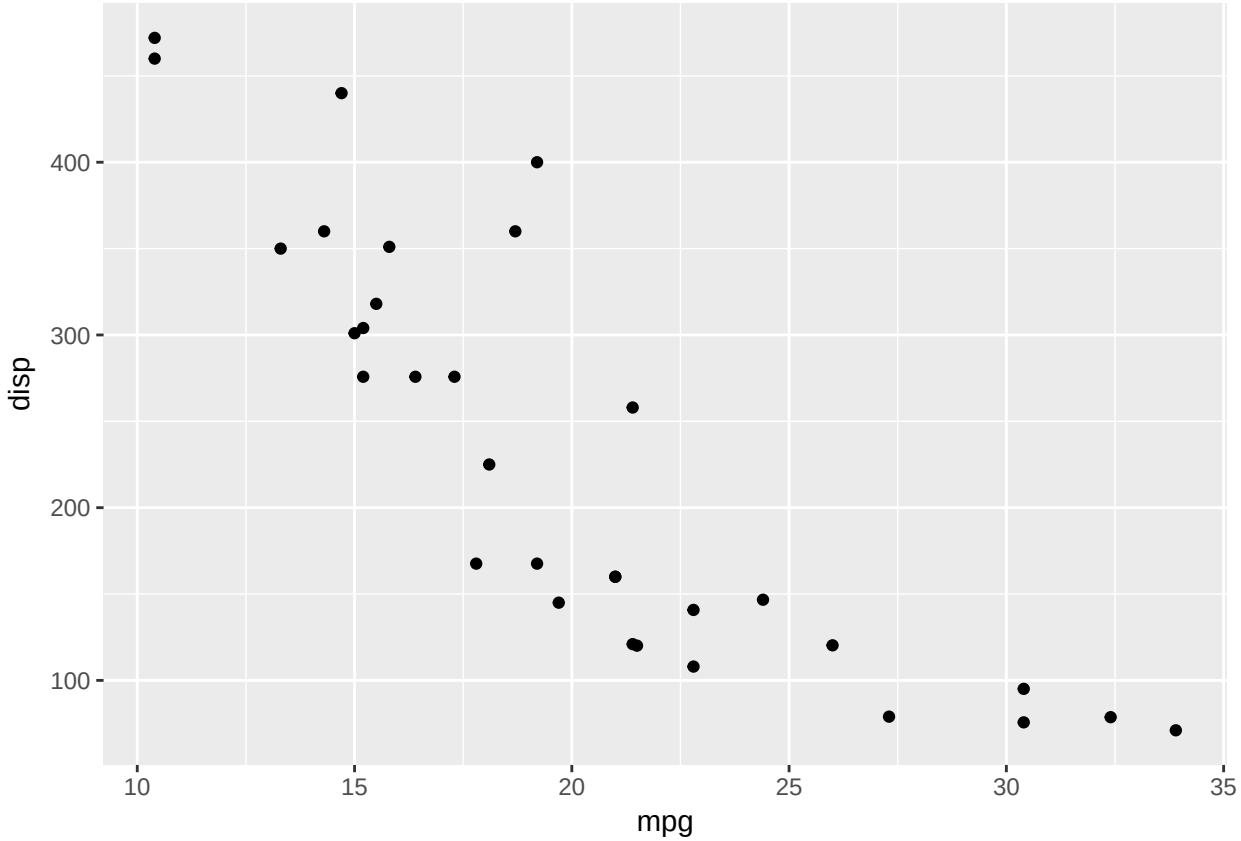


Notice that R automatically adds the variable names to the axes and has a default look for the axis lines and background. We will go over how to customize those later.

Now we will specify the geometric object we want to see on the plot, in this case points, using the `geom_point` function.

Use a "+" sign to add the next layer. Be sure it is at the *end* of the previous line!

```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point()
```



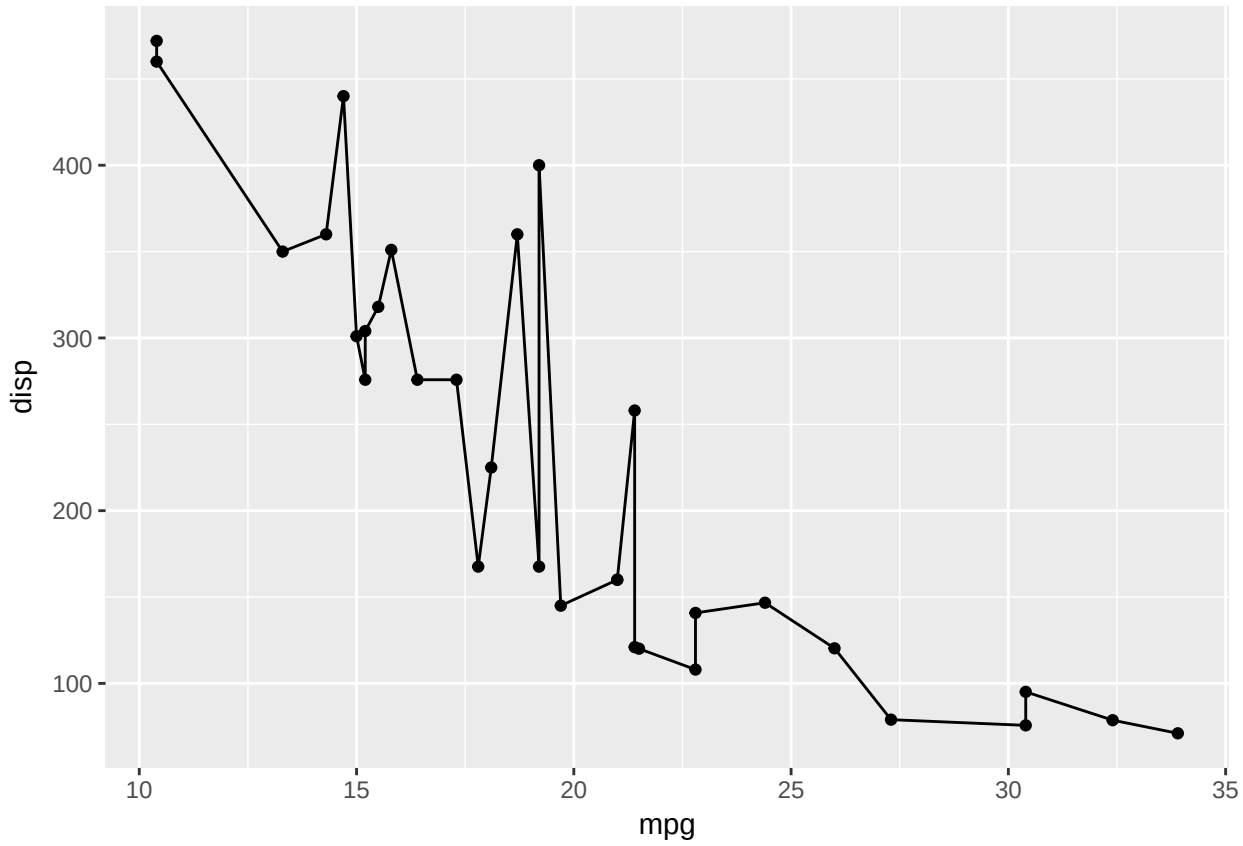
It appears there is a negative relationship between miles per gallon and engine size.

3.3 Basic Exercise

Now lets add another layer - a line - using the `geom_line()` function.

3.4 Basic Exercise Solution

```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point() +
  geom_line()
```


400
300
200
100
disp
10
15
20
25
30
35
mpg

Chapter 4

Customize

There are nearly infinite possibilities for customizing charts with **ggplot2**. We will focus on colors, labels, themes, and scales.

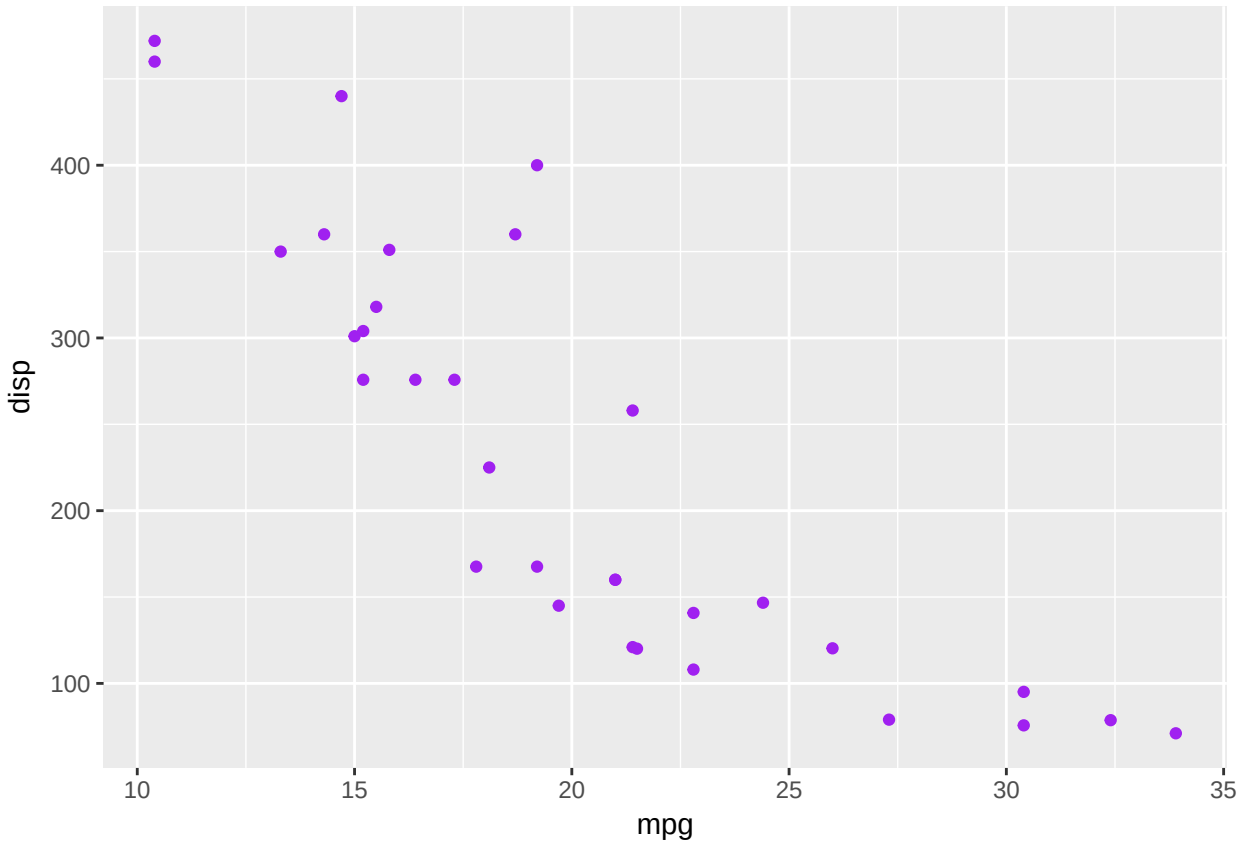
4.1 Colors

Using color is a great way to customize a visualization and make it more interesting and easier to understand.

- R has over 600 colors built in which you can refer to by name.
- The `colors()` function provides a list of the names.
- You can also select colors by hexadecimal codes, such as "#990000" for deep red.
- R Cookbook Graphs section has a very useful section on colors and palettes.
- Another good reference for color palettes is http://colorbrewer2.org
- The *RColorBrewer* package provides a set of useful color palettes that can be used to customize the color ranges on a graph. Use *display.brewer.all()* to see the set of palettes.
- Note: Sometimes you will see the word "color" spelled "colour" - they are interchangeable in R.

Let's change the color of the points to purple, by setting it directly in the `aes()` function of the geom.

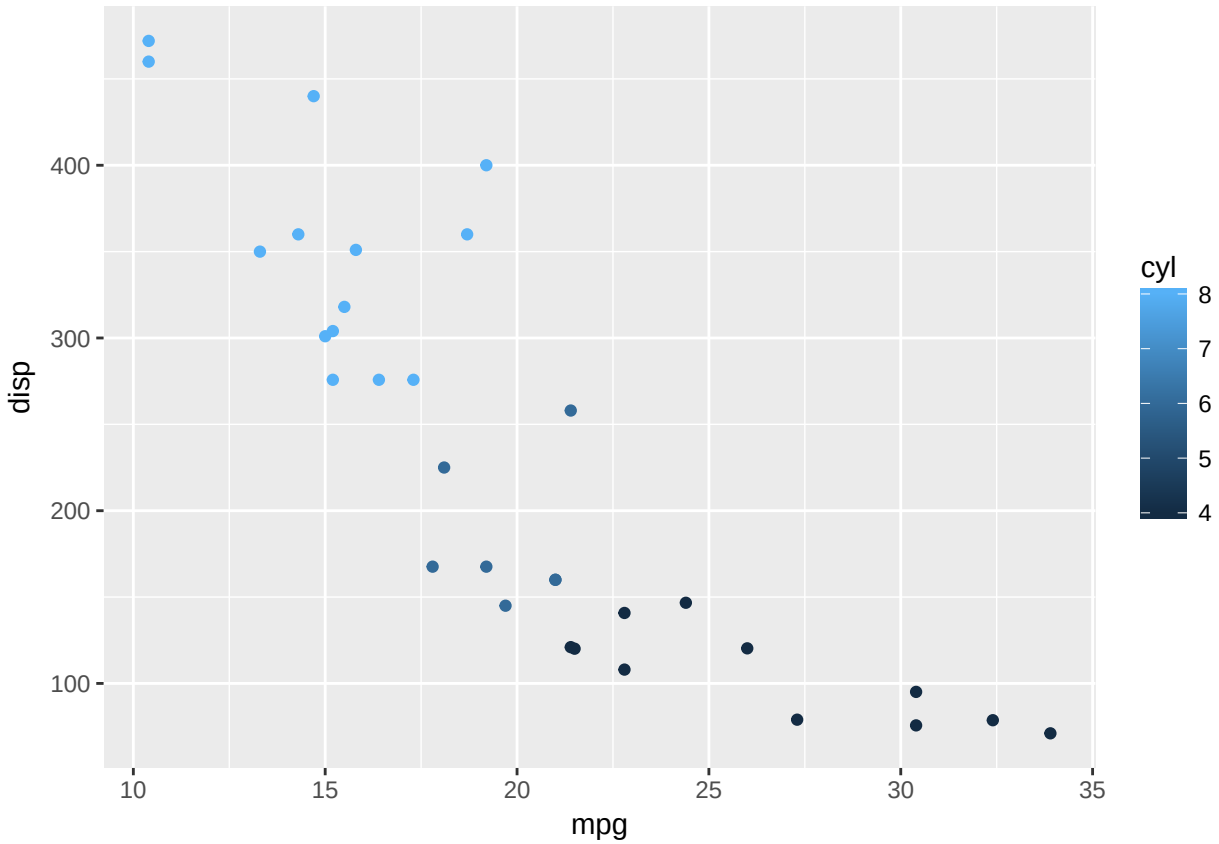
```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(color = "purple")
```



You may also want to map the color of the points to another variable in your dataset. This is done using the `aes()` function and setting color equal to the variable name.

Let's map the color to the number of cylinders.

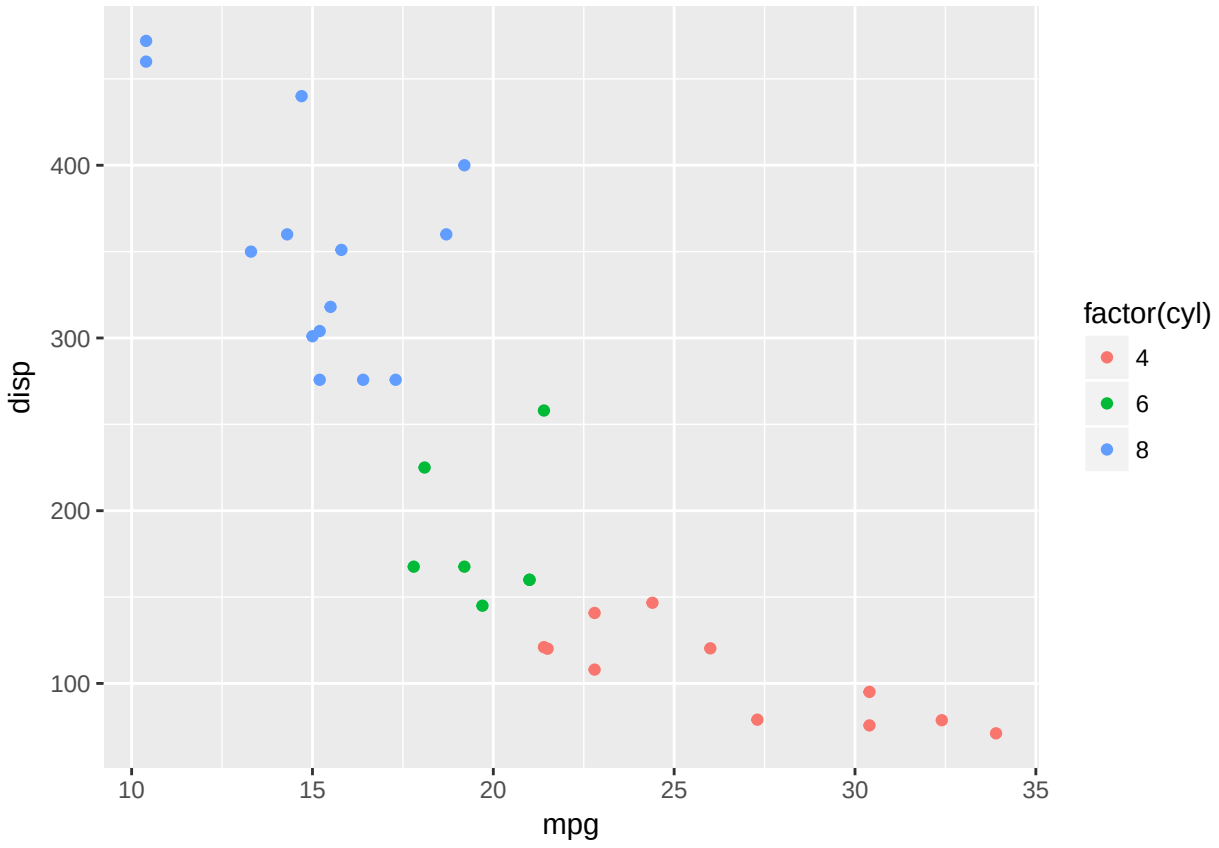
```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(aes(color = cyl))
```



Notice that a legend for the color is automatically added to the graph, and a palette of blues is used by default. However, the cylinders variable is treated as continuous but it is really just three values: 4, 6, and 8.

To treat the cylinders variable as categorical, you can convert it to a factor inside the `ggplot()` function. Notice what happens to the legend.

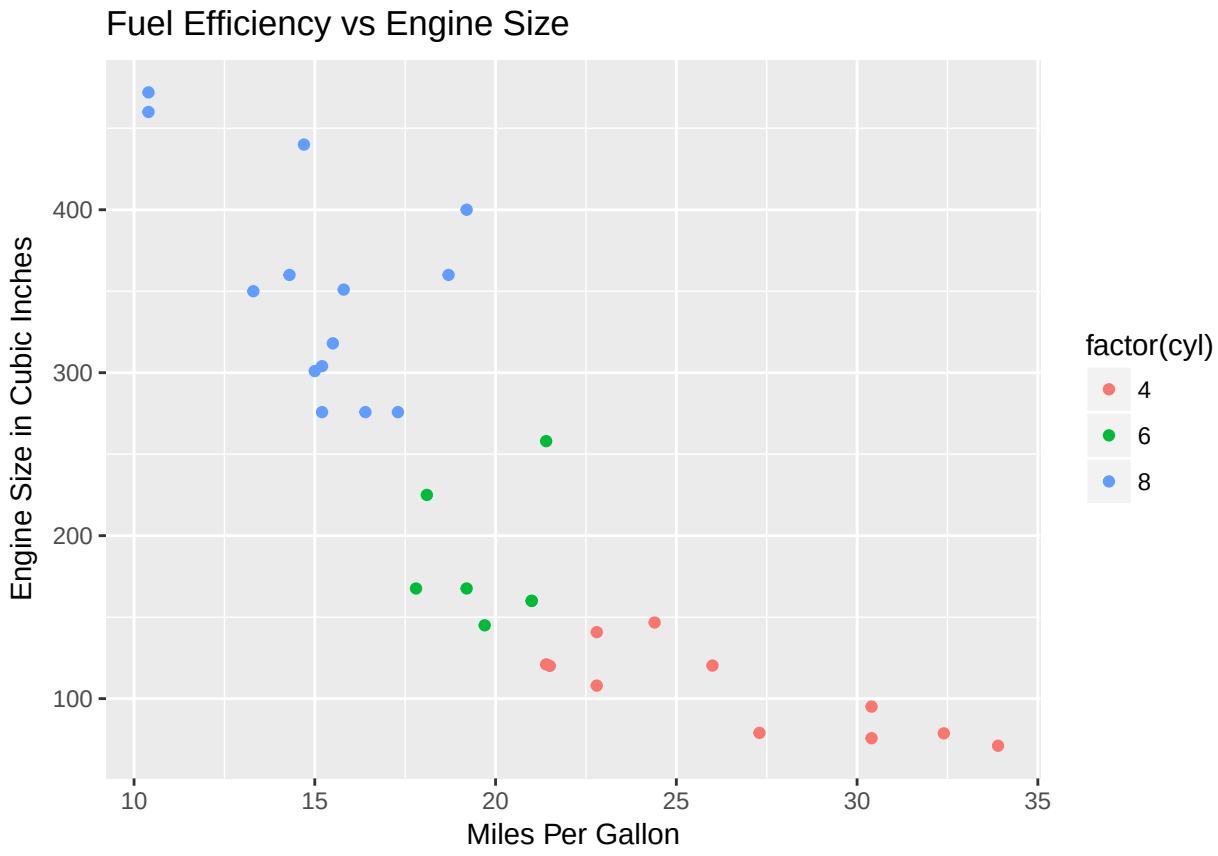
```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(aes(color = factor(cyl)))
```



4.2 Labels

You may want to customize the labels on the axes, using the `xlab()` and `ylab()` functions. You can also add a graph title to the `ggtitle()` function. Again, the "+" sign is used to add these layers.

```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(aes(color = factor(cyl))) +
  xlab("Miles Per Gallon") +
  ylab("Engine Size in Cubic Inches") +
  ggtitle("Fuel Efficiency vs Engine Size")
```



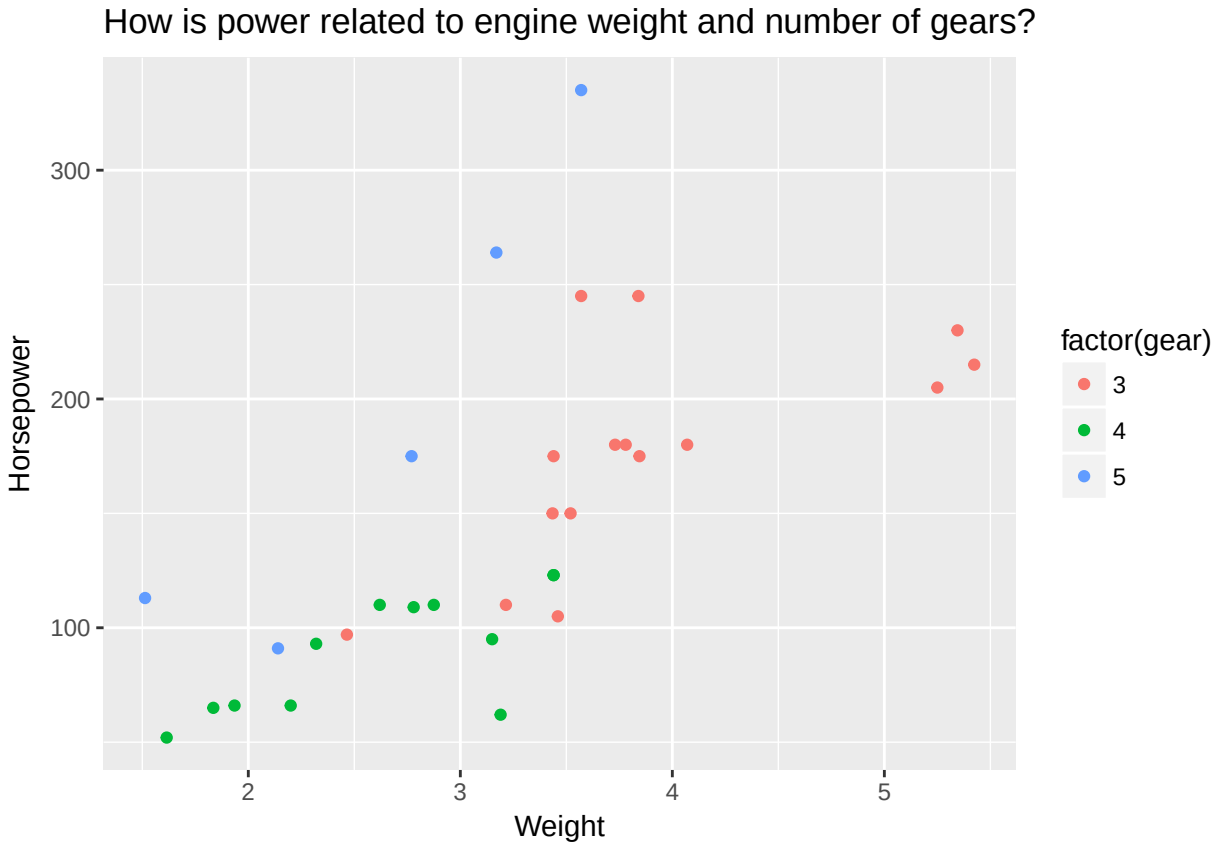
4.3 Customize Exercise

Using the cars data:

1. Plot wt versus hp with a scatterplot.
2. Color the points based on gear.
3. Change both axis labels - spell out wt and hp.
4. Add a chart title.

4.4 Customize Exercise Solution

```
ggplot(data = cars, aes(x = wt, y = hp)) +
  geom_point(aes(color = factor(gear))) +
  xlab("Weight") +
  ylab("Horsepower") +
  ggtitle("How is power related to engine weight and number of gears?")
```



4.5 Themes

There are many more options for customizing a graph. Adjusting the look involves of two different types of elements, *themes* and *scales*.

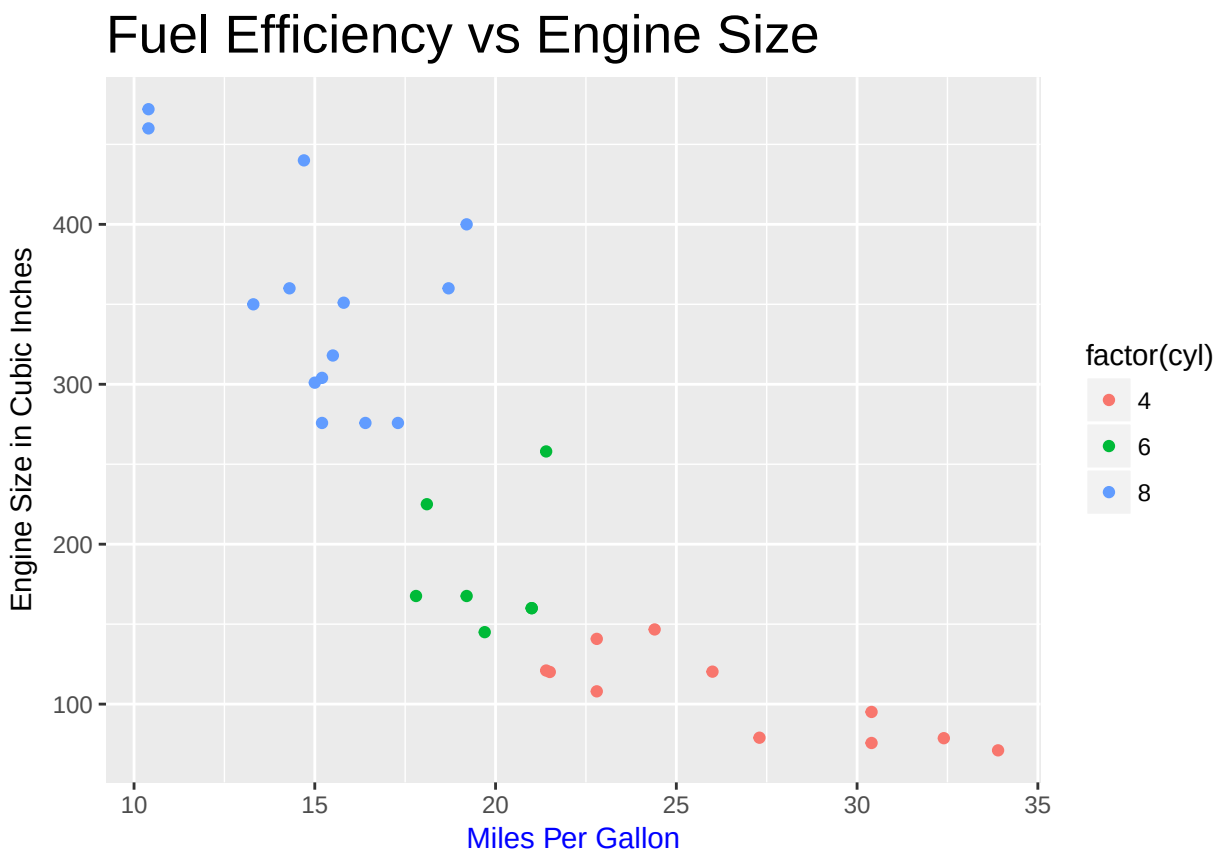
Theme elements are non-data components of the plot such as the title, legend, and axes.

- To change the appearance of a theme element, use the `theme()` function on the component and identify the corresponding element object, such as "element_text".
- Common element objects are text or lines and not all element objects are associated with every component.
- The R Graphics Cookbook covers this in section 9.4 very clearly.

Let's change the text of the axis labels:

- Font color of x-axis label to blue.
- Font size of graph title to 20.

```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(aes(color = factor(cyl))) +
  xlab("Miles Per Gallon") +
  ylab("Engine Size in Cubic Inches") +
  ggtitle("Fuel Efficiency vs Engine Size") +
  theme(axis.title.x = element_text(color = 'blue')) +
  theme(plot.title = element_text(size = 20))
```



A few other useful theme functions:

- Remove an axis label: `theme(axis.title.x = element_blank())`
- Remove grid lines: `theme(panel.grid.major = element_blank())`
- Rotate x-axis text labels by 90 degrees: `theme(axis.text.x = element_text(angle = 90))`
- Move the position of a legend: `theme(legend.position = "bottom")`

4.6 Scales

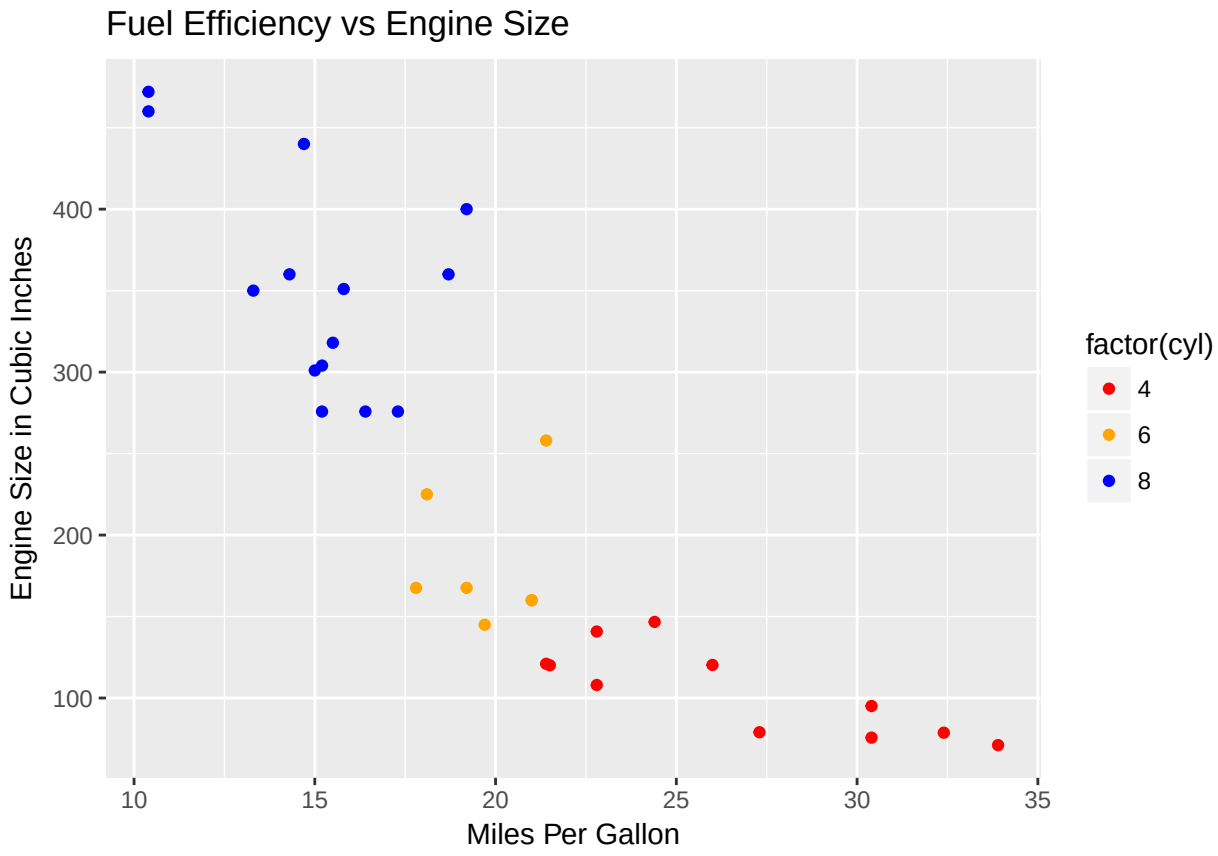
Scale elements map data values to visual values.

- To change the default mapping, a new scale layer is added.
- A `scale` function identifies the aesthetic you want to adjust, a pre-packaged scale to use, and the specific arguments for what you are trying to change.
- Different types of data use different types of aesthetics and scales - it is important to match them up correctly.
- The Data Visualization Cheatsheet are a good reference for different types of scales.

We'll use `scale_color_manual()` and define the colors we want use.

- "scale" indicates that we want to customize a data element.
- "color" indicates which data element.
- "manual" is used to specify the color mapping we want to use.

```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(aes(color = factor(cyl))) +
  xlab("Miles Per Gallon") +
  ylab("Engine Size in Cubic Inches") +
  ggtitle("Fuel Efficiency vs Engine Size") +
  scale_color_manual(values = c("red","orange","blue"))
```



4.7 Single Scale Rule

Once you use a "scale" function on an aesthetic (in this case, color), you need to include all the scale parameters in that function. If you add another scale layer for the same aesthetic, it will replace the previous layers.

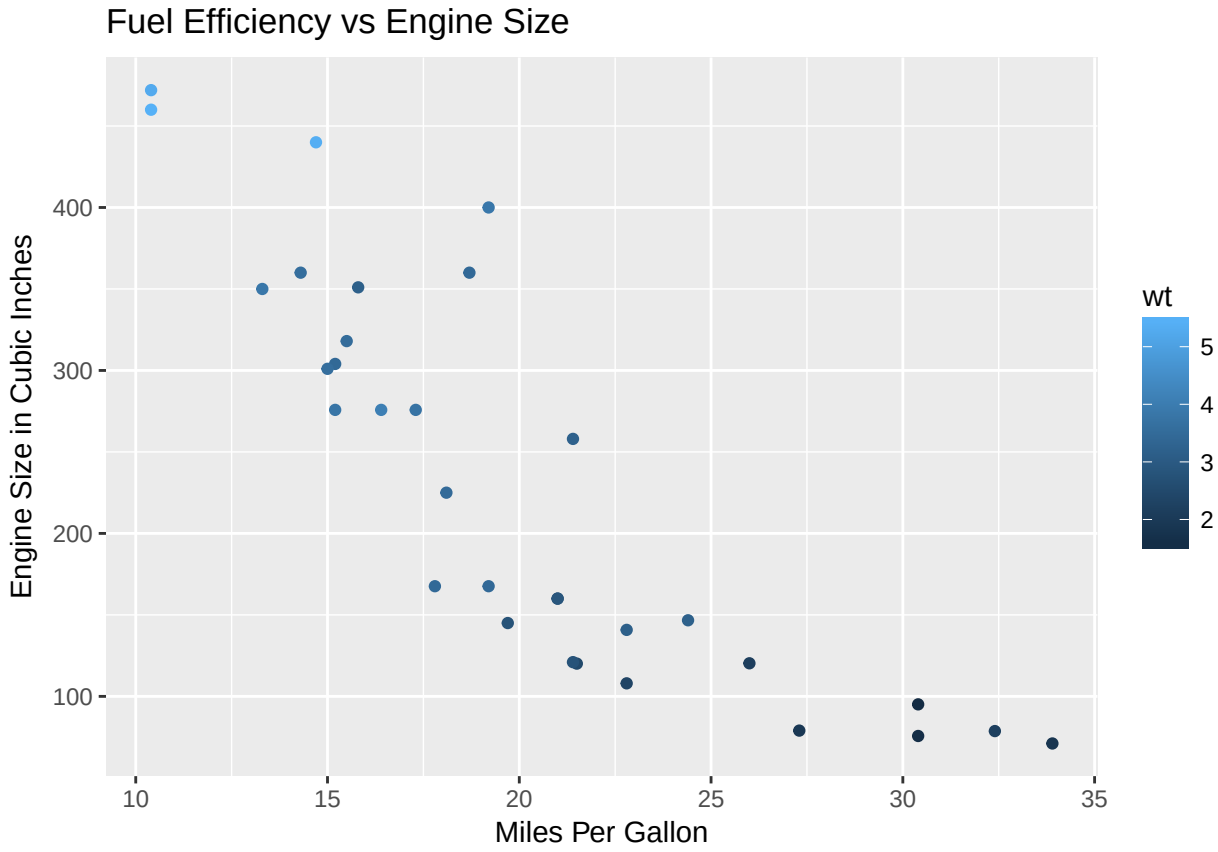
Here's an example:

- Change the mapping for the color aesthetic from *cyl* to the *wt* variable, which is continuous.
- You could use the `scale_color_continuous()` or `scale_color_gradient()` to effect the color aesthetic.
- Let's change the color scheme with `scale_color_gradient()` and the breaks on the legend with `scale_color_continuous()`.

```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(aes(color = wt)) +
  xlab("Miles Per Gallon") +
  ylab("Engine Size in Cubic Inches") +
```

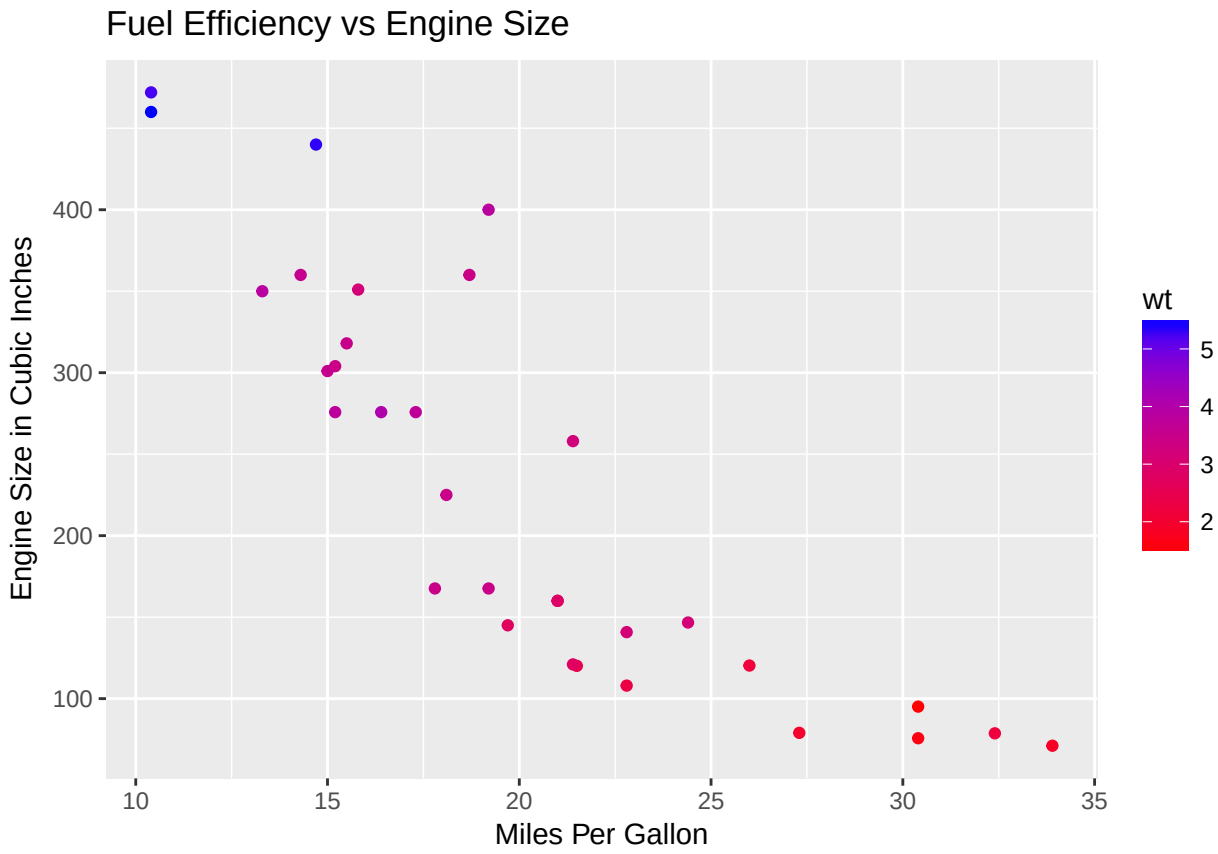
```
  ggtitle("Fuel Efficiency vs Engine Size") +
  scale_color_gradient(low = "red", high = "blue") +
  scale_color_continuous(breaks = c(2,3,4,5))
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```



But notice the error message and that the colors don't change. Instead, combine the parameters into just one "scale" function.

```
ggplot(data = cars, aes(x = mpg, y = disp)) +
  geom_point(aes(color = wt)) +
  xlab("Miles Per Gallon") +
  ylab("Engine Size in Cubic Inches") +
  ggtitle("Fuel Efficiency vs Engine Size") +
  scale_color_gradient(low = "red", high = "blue", breaks = c(2,3,4,5))
```



4.8 ggthemes

ggthemes is a package that provides some customized geoms, scales, and themes.

- https://github.com/jrnold/ggthemes

4.9 Saving Outputs

To save your visualizations outside of RStudio, simply click the Export button on the Plots tab.
Options:

- Save as an Image.
- Save as a PDF.
- Copy to Clipboard.

Chapter 5

Bar Charts

Bar charts are generally used to display numeric values on the y-axis for different categories or discrete values on the x-axis. Bar charts are not well suited for continuous data on the x-axis because it will try to make a bar for every possible value of the continuous range.

5.1 Thor Data

For this section we will work with the THOR data from Vietnam, 1965.

```
nam65 <- read.csv("thor_vietnam_65_clean.csv", stringsAsFactors = FALSE)
dim(nam65)
```

```
## [1] 63557    21
```

This data set includes a subset of the variables from the raw THOR data and cleaned up events with missing or inaccurate geospatial coordinates.

5.2 geom_bar

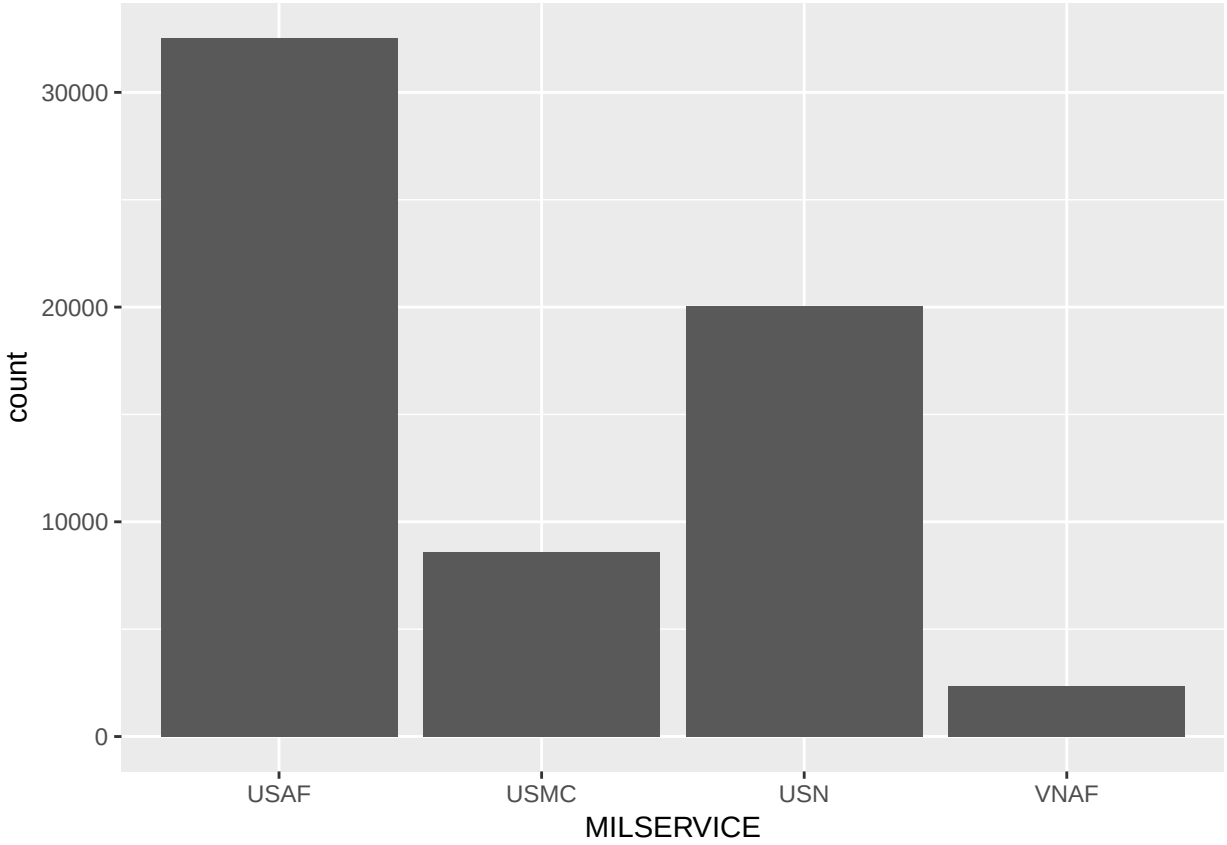
The heights of the bar on the y-axis can represent counts or values from your data set. The code for the graph changes depending on which is the case.

- The geom for bar charts is: `geom_bar()`

Let's make a bar chart that shows the number of missions performed by each military service.

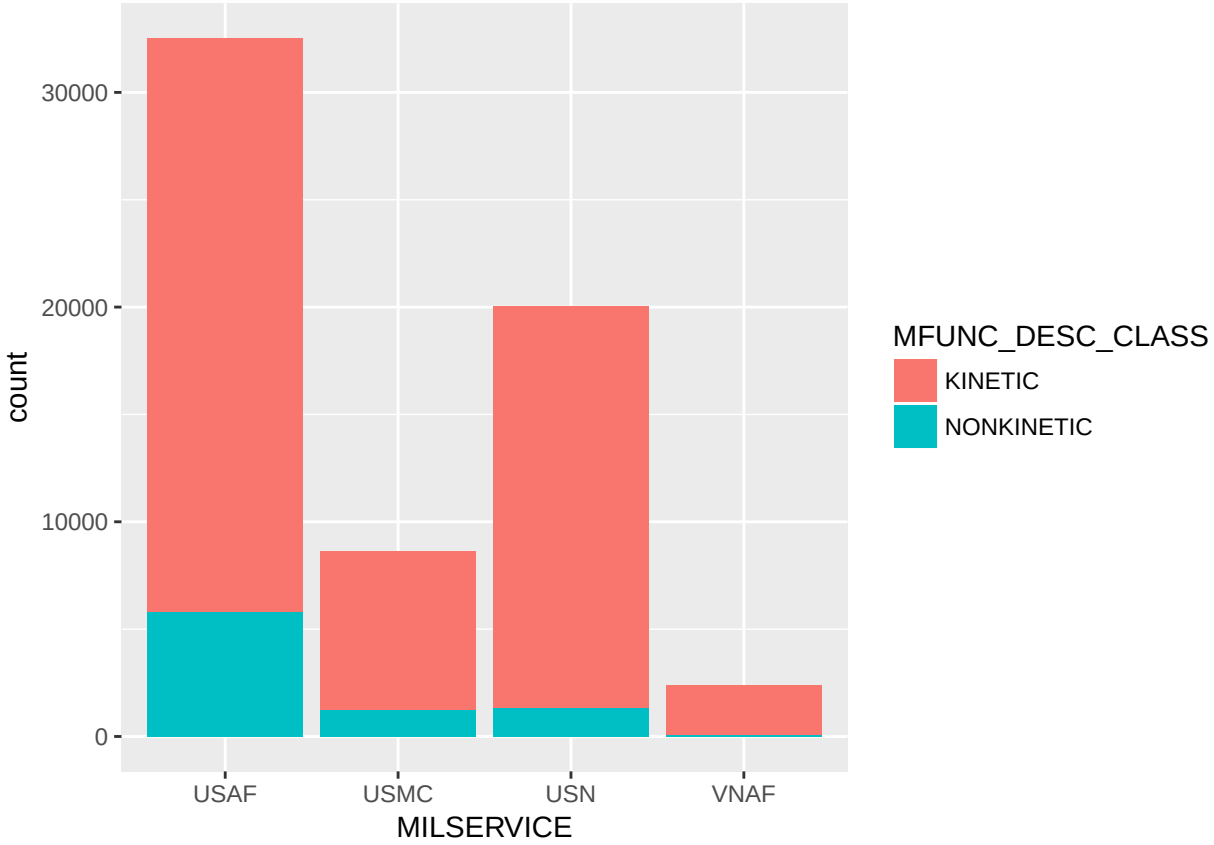
- Use just one variable from our data set because we are interested in the counts for that variable.

```
ggplot(data = nam65, aes(x = MILSERVICE))+
  geom_bar()
```



Now we will add another variable called "MFUNC_DESC_CLASS", which has two factor levels, "KINETIC" and "NON-KINETIC". We will use the "fill" aesthetic to map the new variable to the graph.

```
ggplot(data = nam65, aes(x = MILSERVICE, fill = MFUNC_DESC_CLASS))+
  geom_bar()
```



Notice this creates a stacked bar. If you want to have bars side by side for that variable, set the "position" parameter to "dodge" in the geom like this: `geom_bar(position = "dodge")`.

If we want to change the default colors, we use a "scale" function.

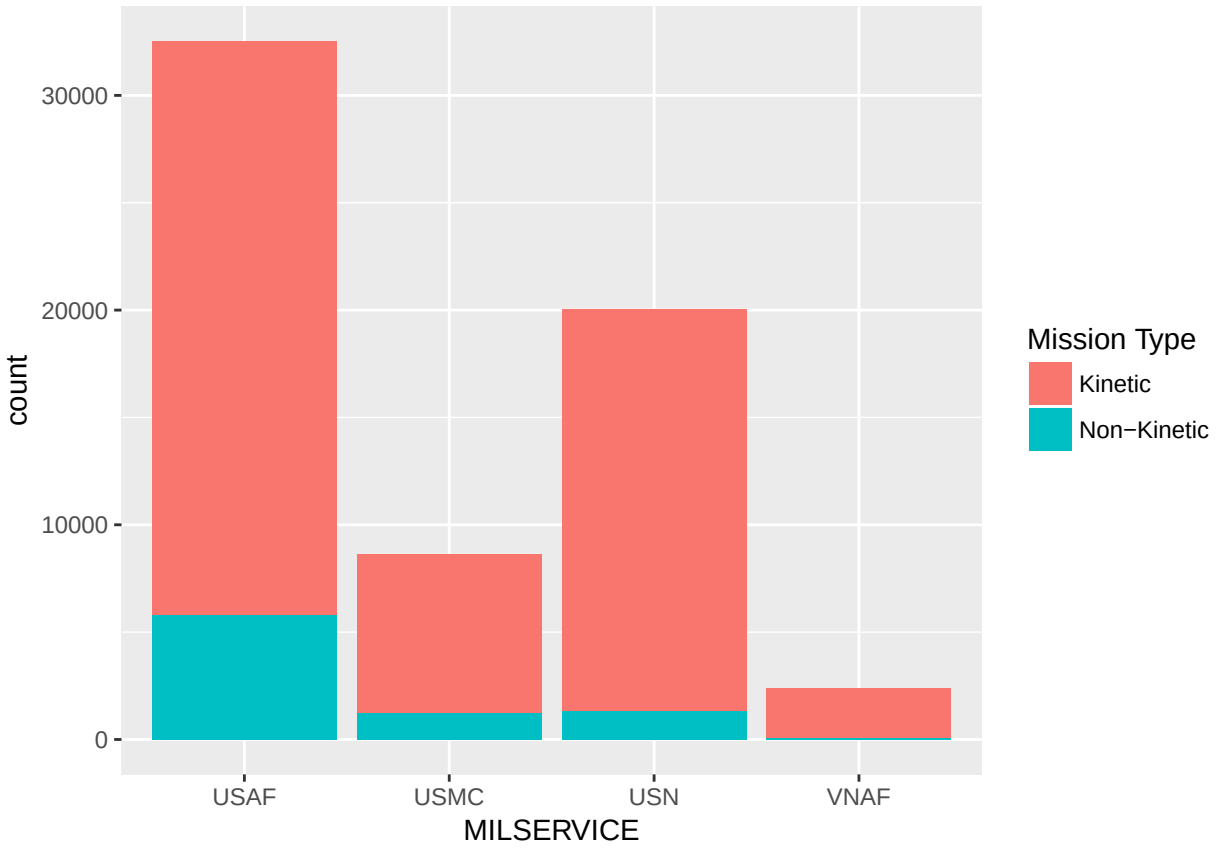
- Set the colors manually with: `scale_fill_manual(values = c("red","purple"))`
- Use a palette from *RColorBrewer* with: `scale_fill_brewer(palette = "Spectral")`

5.3 Legends

We want to make some changes to the legend on our chart.

- Change the legend title with `labs(fill = "Mission Type")`
- Change the text of the labels with `scale_fill_discrete(labels = c("Kinetic","Non-Kinetic"))`

```
ggplot(data = nam65, aes(x = MILSERVICE, fill = MFUNC_DESC_CLASS))+
  geom_bar() +
  labs(fill = "Mission Type") +
  scale_fill_discrete(labels = c("Kinetic","Non-Kinetic"))
```



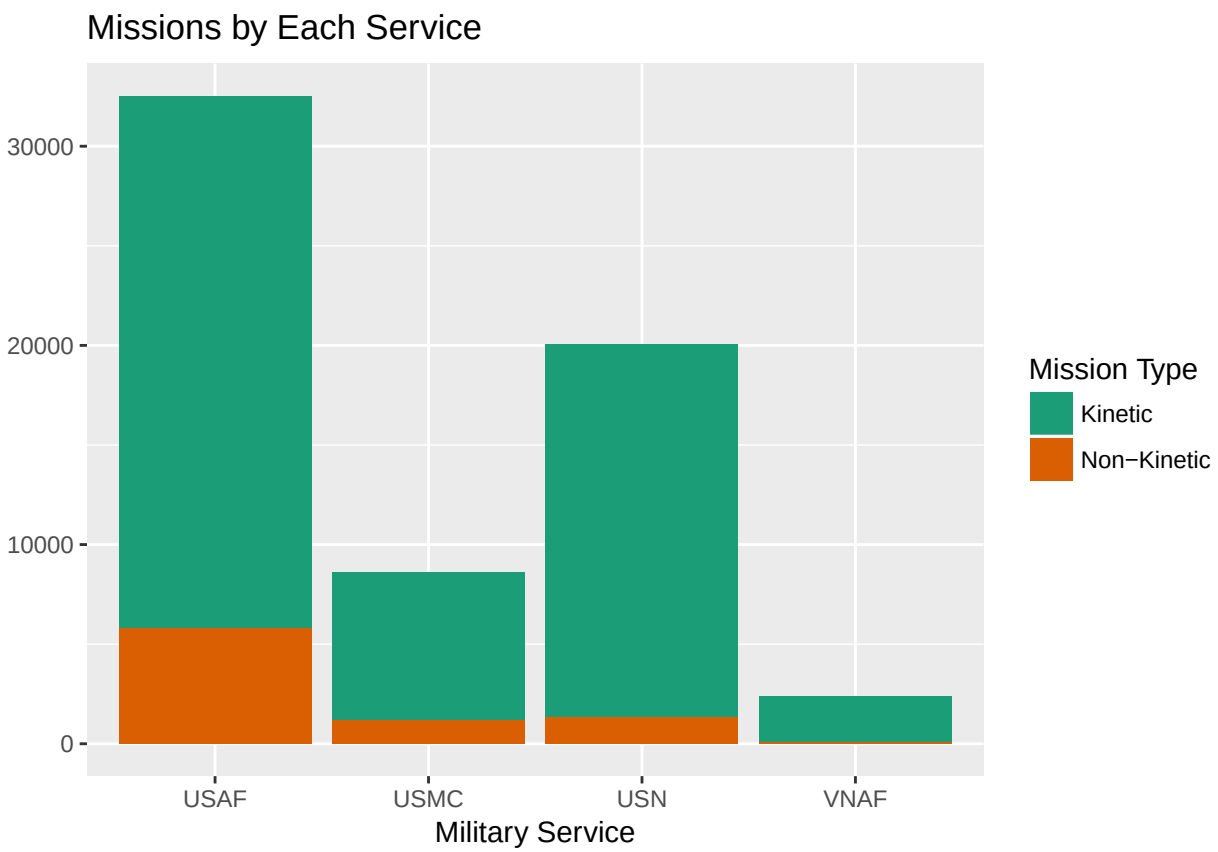
5.4 Bar Exercise

Update our bar chart:

1. Add a title
2. Remove the y-axis label ("count")
3. Change the x-axis label
4. Change the colors for the fill aesthetic (don't forget the single scale rule!)

5.5 Bar Exercise Solution

```
ggplot(data = nam65, aes(x = MILSERVICE, fill = MFUNC_DESC_CLASS))+
  geom_bar() +
  labs(fill = "Mission Type") +
  #scale_fill_discrete(labels = c("Kinetic","Non-Kinetic")) +      #single scale rule!
  ggtitle("Missions by Each Service") +
  theme(axis.title.y = element_blank()) +
  xlab("Military Service") +
  scale_fill_brewer(palette = "Dark2",labels = c("Kinetic","Non-Kinetic"))   #single scale rule!
```


Missions by Each Service
30000
20000
10000
0
USAF
USMC
USN
VNAF
Military Service
Mission Type
Kinetic
Non−Kinetic

Chapter 6

Barcharts with Values

Remember that there's a difference in how to make a bar chart with values from a variable in the data on the y-axis vice counts for a single variable.

6.1 Aircraft Data

For this example, we will summarize the average number of aircraft per mission across the different target types.

```
aircraft <- summarize(group_by(nam65, TGTTYPE), avgNumAcft = mean(NUMOFACFT))
head(aircraft)
```

```
## # A tibble: 6 x 2
##              TGTTYPE avgNumAcft
##                <chr>      <dbl>
## 1                      3.163743
## 2 AIRCRAFT (UNIDENT)   2.400000
## 3           AIRFIELD   1.742788
## 4      ANTI-AIRCRAFT   2.504082
## 5      "AREA\\DEPOT"   2.811982
## 6             BRIDGE   2.559007
```

Notice that the first row is missing the target type information. Let's fill in the blank space.

```
aircraft[1,1] <- "UNIDENTIFIED"
head(aircraft)
```

```
## # A tibble: 6 x 2
##              TGTTYPE avgNumAcft
##                <chr>      <dbl>
## 1       UNIDENTIFIED   3.163743
## 2 AIRCRAFT (UNIDENT)   2.400000
## 3           AIRFIELD   1.742788
## 4      ANTI-AIRCRAFT   2.504082
## 5      "AREA\\DEPOT"   2.811982
## 6             BRIDGE   2.559007
```

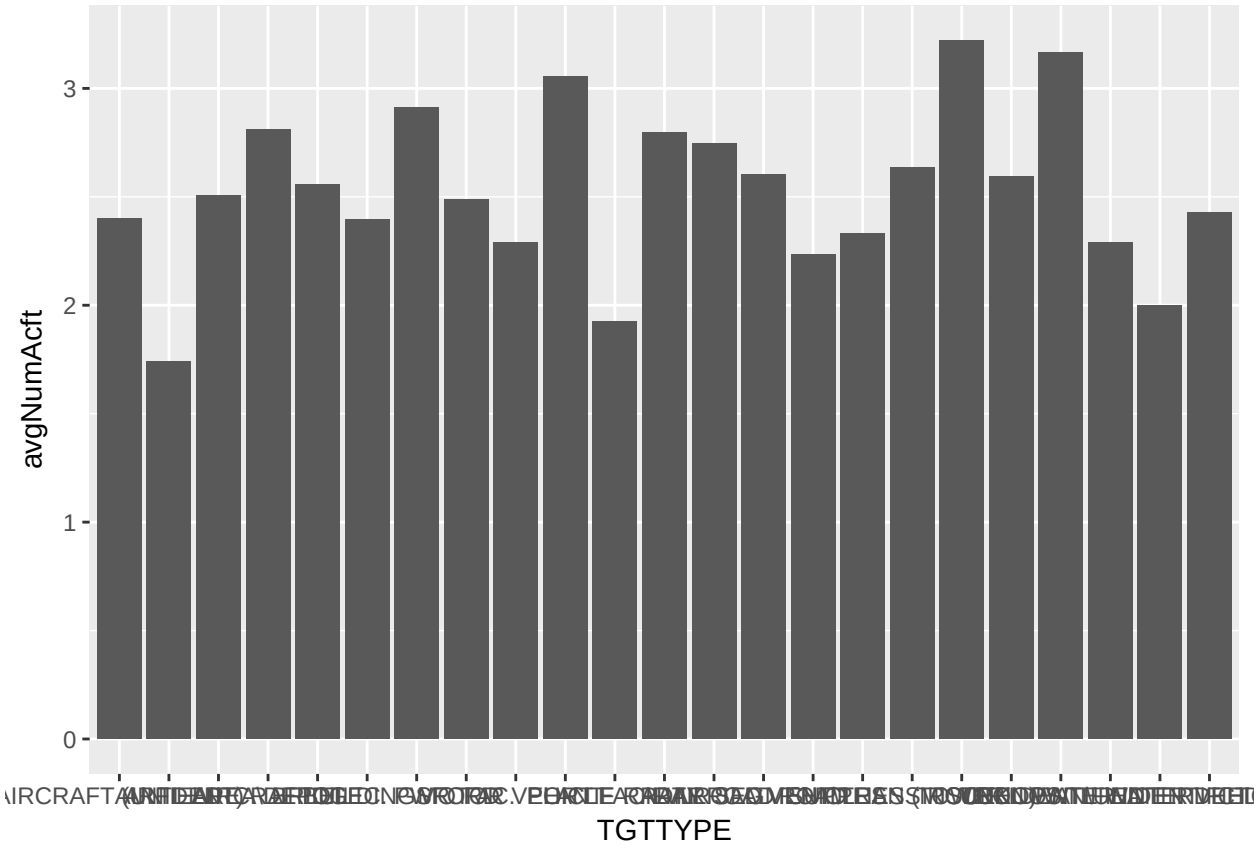
6.2 Stat Parameter

To make a barchart of this data, you need to set the "stat" parameter in the `geom_bar()` function to "identity", instead of the default "bin".

- The "stat" refers to the statistical transformation that ggplot does on the raw data, which by default in bar charts is set to bin and count the number of points for each bin.
- To use the variable data on the y-axis instead of the count, you need to change the "stat" parameter.

For this example, we want to see the average number of aircraft (y-axis) per target type (x-axis).

```
ggplot(data = aircraft, aes(x = TGTTYPE, y = avgNumAcft))+
  geom_bar(stat= "identity")
```



6.3 hjust and vjust

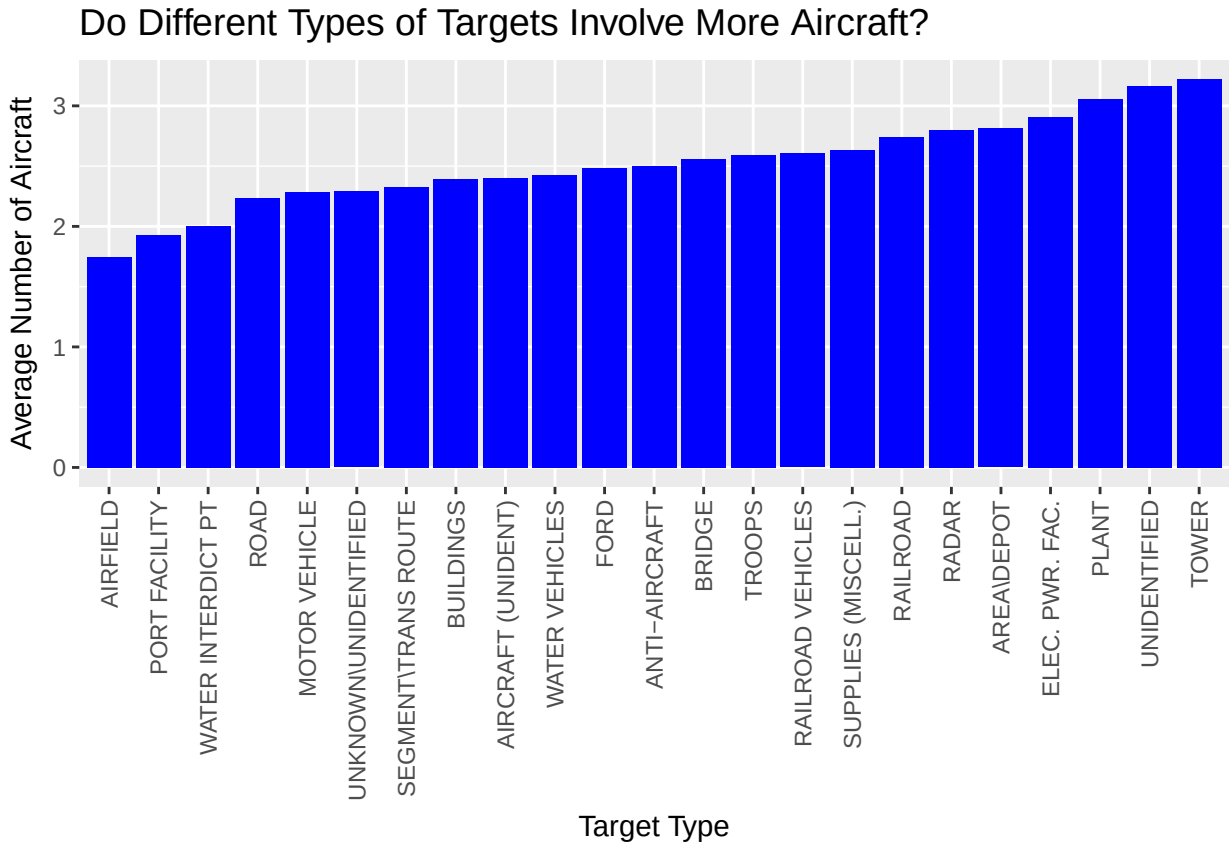
The "hjust" and "vjust" parameters in the `theme()` function for an axis change the horizontal and vertical justification of the text.

- These work in conjunction with the "angle" parameter to make the text more readable.
- A good reference for this is https://www.r-bloggers.com/hjust-and-vjust/
- Note that each parameter can only take certain values and the combination results in different positions.
- The theme function looks like: `theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = .5))`

6.4 Sorting Bars

We can sort the bars by height using the `reorder()` function in the `aes()` for the x-axis.

```
ggplot(data = aircraft, aes(x = reorder(TGTTYPE, avgNumAcft), y = avgNumAcft))+
  geom_bar(stat= "identity", fill = "blue") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = .5)) +
  ggtitle("Do Different Types of Targets Involve More Aircraft?") +
  theme(plot.title = element_text(size = 14)) +
  xlab("Target Type") +
  ylab("Average Number of Aircraft")
```



You can also change the sort to from most to least like this: `aes(x = reorder(TGTTYPE, desc(avgNumAcft)), y = avgNumAcft)`

6.5 Bar with Values Exercise

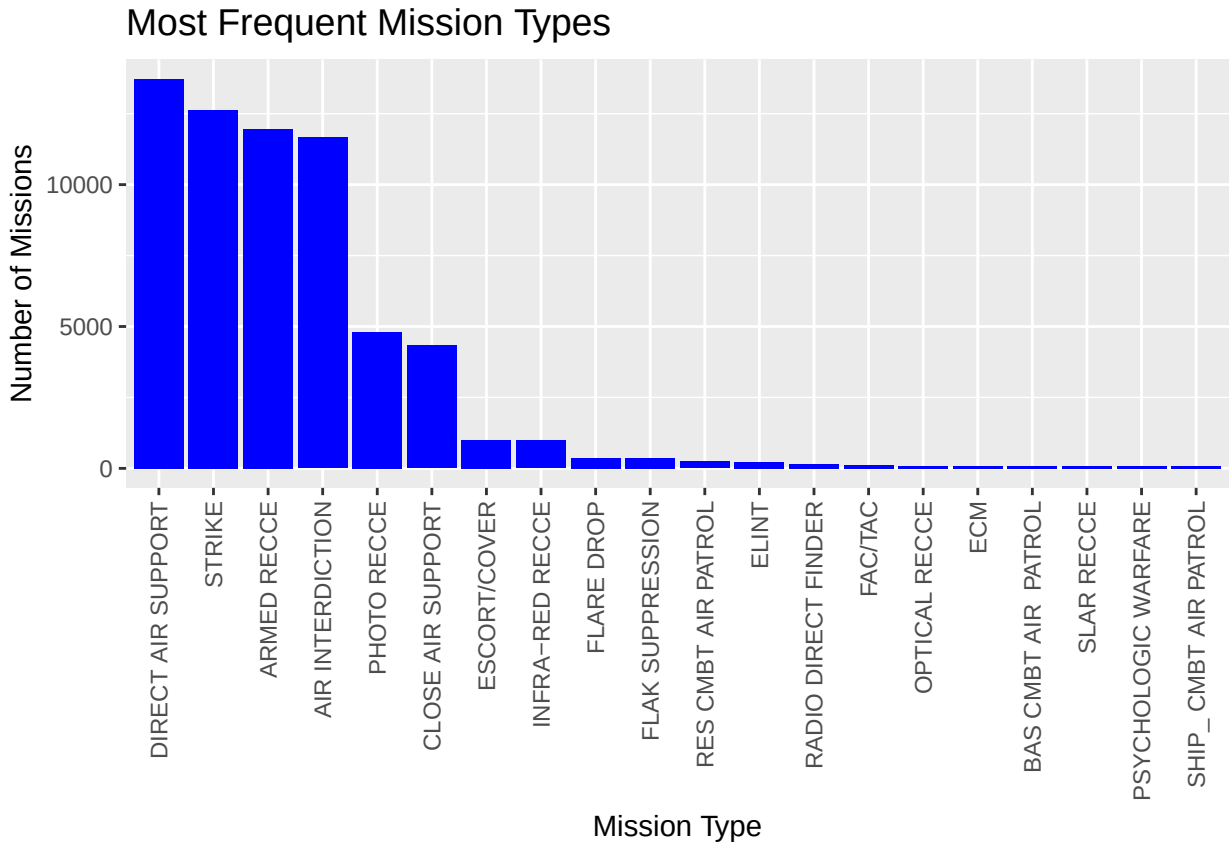
Update the average number of aircraft per target type chart with:

1. Create a dataframe with the number of missions per mission type (hint: MFUNC_DESC variable)
2. Make a bar chart that displays the 20 most frequent mission types, in descending order.
3. Rotate the labels on the x-axis to make it readable.
4. Add a chart title and make it a larger font.
5. Change the axis titles.
6. Change the color of the bars to blue.

6.6 Bar with Values Exercise Solution

```
bar_y <- summarize(group_by(nam65, MFUNC_DESC), num_missions = n())
bar_y <- arrange(bar_y, desc(num_missions))
bar_y <- bar_y[1:20,]

ggplot(data = bar_y, aes(x = reorder(MFUNC_DESC, desc(num_missions)), y = num_missions))+
  geom_bar(stat = "identity", fill = "blue")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = .5)) +
  ggtitle("Most Frequent Mission Types") +
  theme(plot.title = element_text(size = 14)) +
  xlab("Mission Type") +
  ylab("Number of Missions")
```



Chapter 7

Line Charts

A time series is a series of data points indexed in time order, most commonly in an equally spaced sequence. While this makes the data discrete, you often see a time series visualized with a line. We will create our line charts once again with the **ggplot2** package.

7.1 Missions Per Day

Let's make a time series chart of the missions, showing the number of missions each day.

First we need to format the "MSNDATE" variable using the **lubridate** package. This will fix the inconsistent date formats.

```
nam65$MSNDATE<-lubridate::ymd(nam65$MSNDATE)
```

Notice that the data type for "MSNDATE" changed to Date.

Next, we need to count the number of missions per day.

```
msn <- summarise(group_by(nam65, MSNDATE), missions = n())
dim(msn)
```

```
## [1] 92  2
```

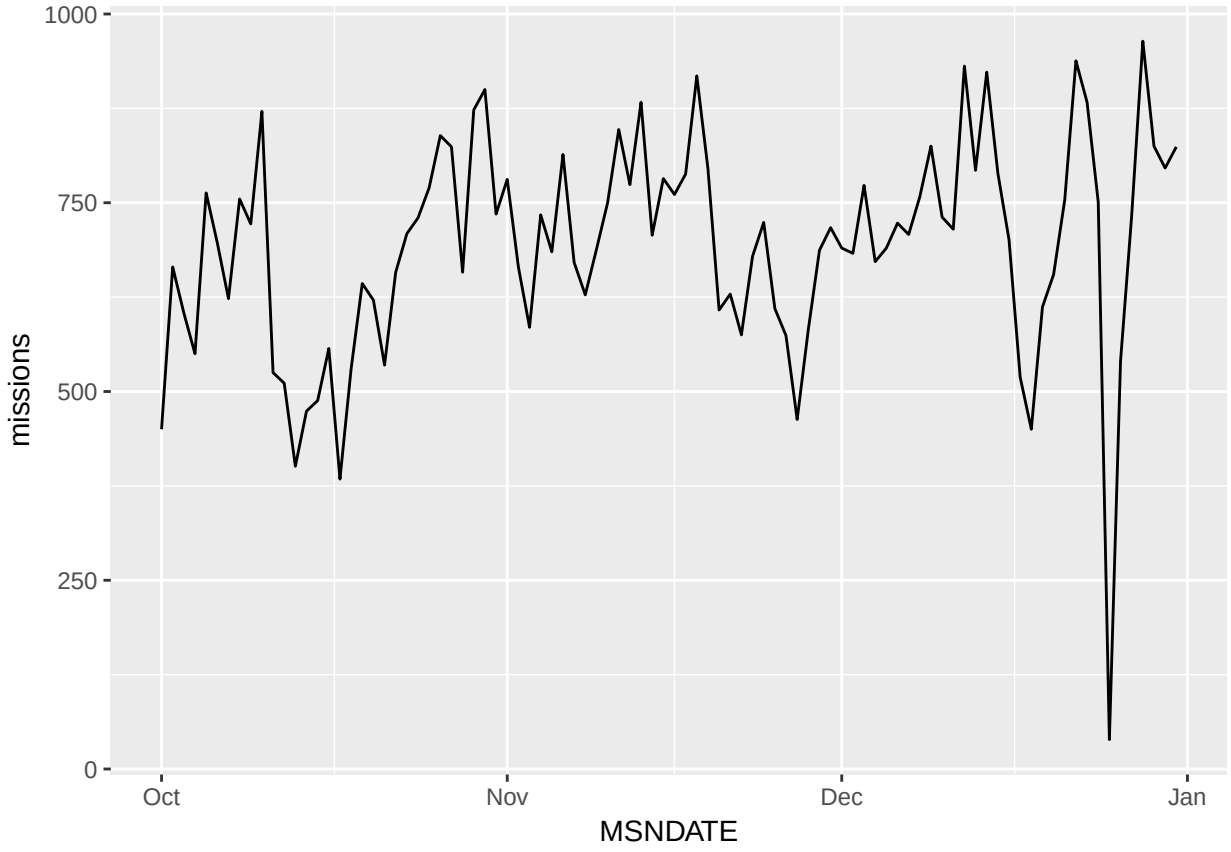
Notice that there are 92 days from 1Oct65 to 31Dec65, and there are 92 rows in our new dataframe, indicating that there are no missing dates.

To do time series, you need to make sure there are no missing dates. You may need to fill in the data set with zeros for dates that are missing or aggregate to a lower level of precision.

7.2 geom_line

Now we will create our plot, using `geom_line()`.

```
ggplot(msn, aes(x = MSNDATE, y = missions))+
  geom_line()
```



7.3 Date Scales

Notice that ggplot only used the months as labels on the x-axis. To create weekly breaks across the time series, use the `seq()` and `as.Date()` functions.

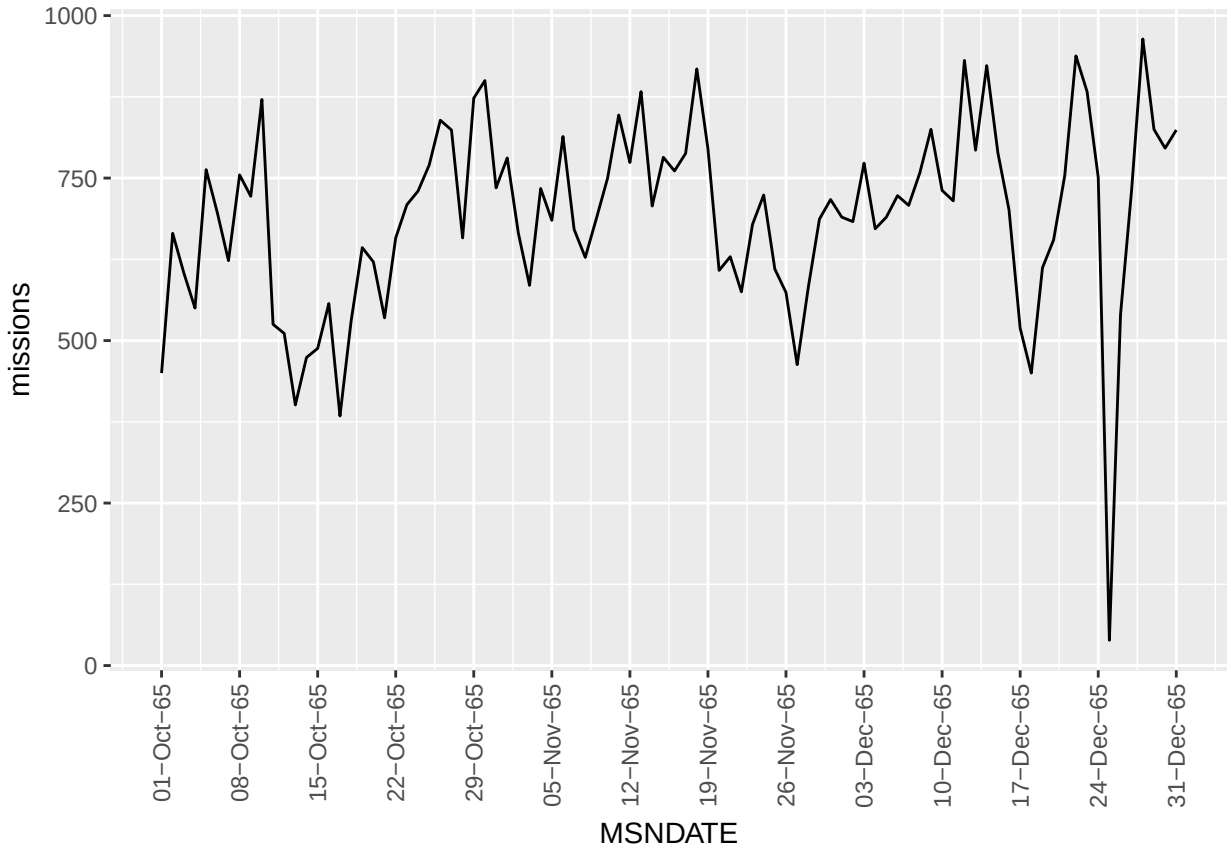
```
datebreaks<-seq(as.Date("1965-10-01"),as.Date("1965-12-31"), by="week")
head(datebreaks)
```

```
## [1] "1965-10-01" "1965-10-08" "1965-10-15" "1965-10-22" "1965-10-29"
## [6] "1965-11-05"
```

To change the labels on the x-axis:

- Use a `scale_x_date()` function to adjust the labels on the x-axis with the "datebreaks" vector.
- Use the `date_format()` function from the **scales** package to change the format of the dates to "day-month-year".

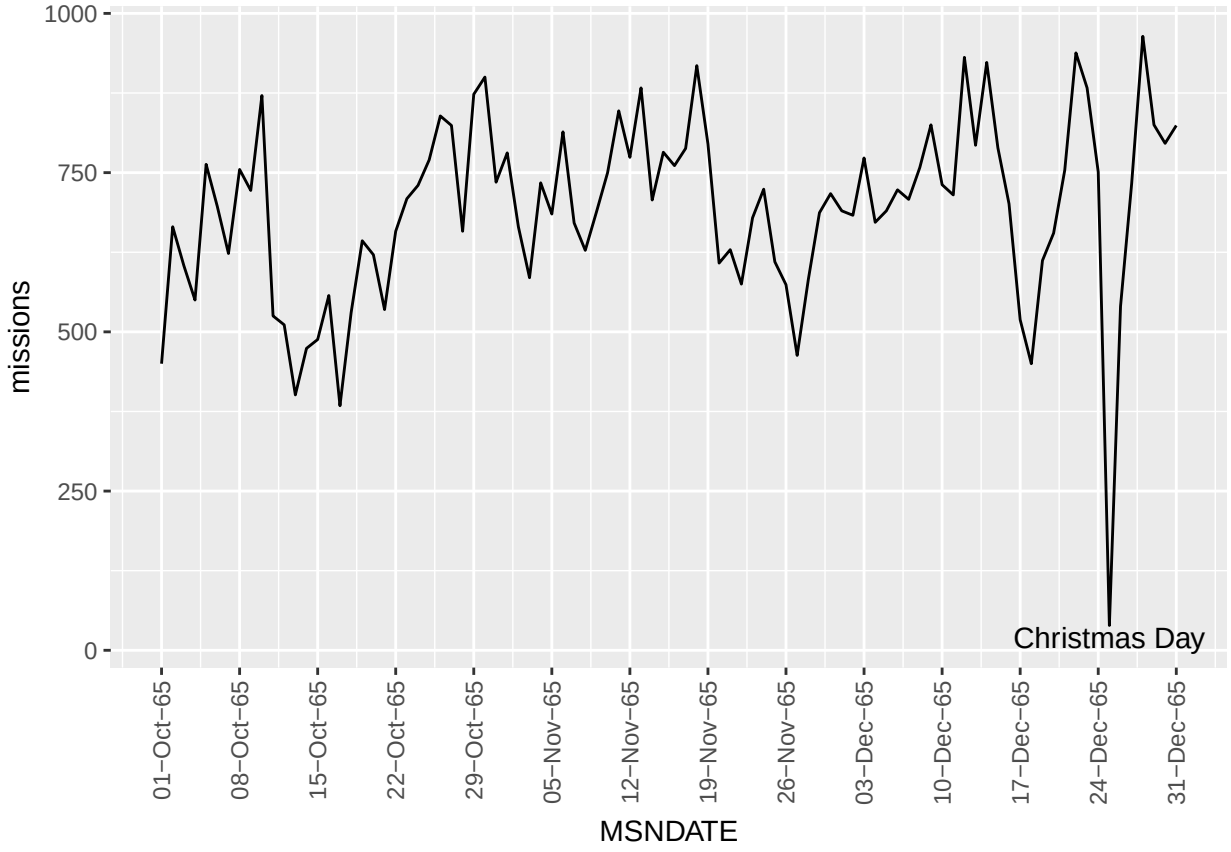
```
ggplot(msn,aes(x=MSNDATE, y= missions))+
  geom_line()+
  scale_x_date(breaks=datebreaks, labels = date_format("%d-%b-%y")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust = .5))
```



7.4 Annotation

The deep dip in our time series chart occurs on Christmas Day. (hint: use the `arrange()` function to determine that) We can label that information on the chart using the `annotate()` function in the ggplot.

```
ggplot(msn,aes(x=MSNDATE, y= missions))+
  geom_line()+
  scale_x_date(breaks=datebreaks, labels = date_format("%d-%b-%y")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust = .5)) +
  annotate("text", x = as.Date('1965-12-25'), y = 20, label = "Christmas Day")
```



7.5 Line Width, Color, and Points

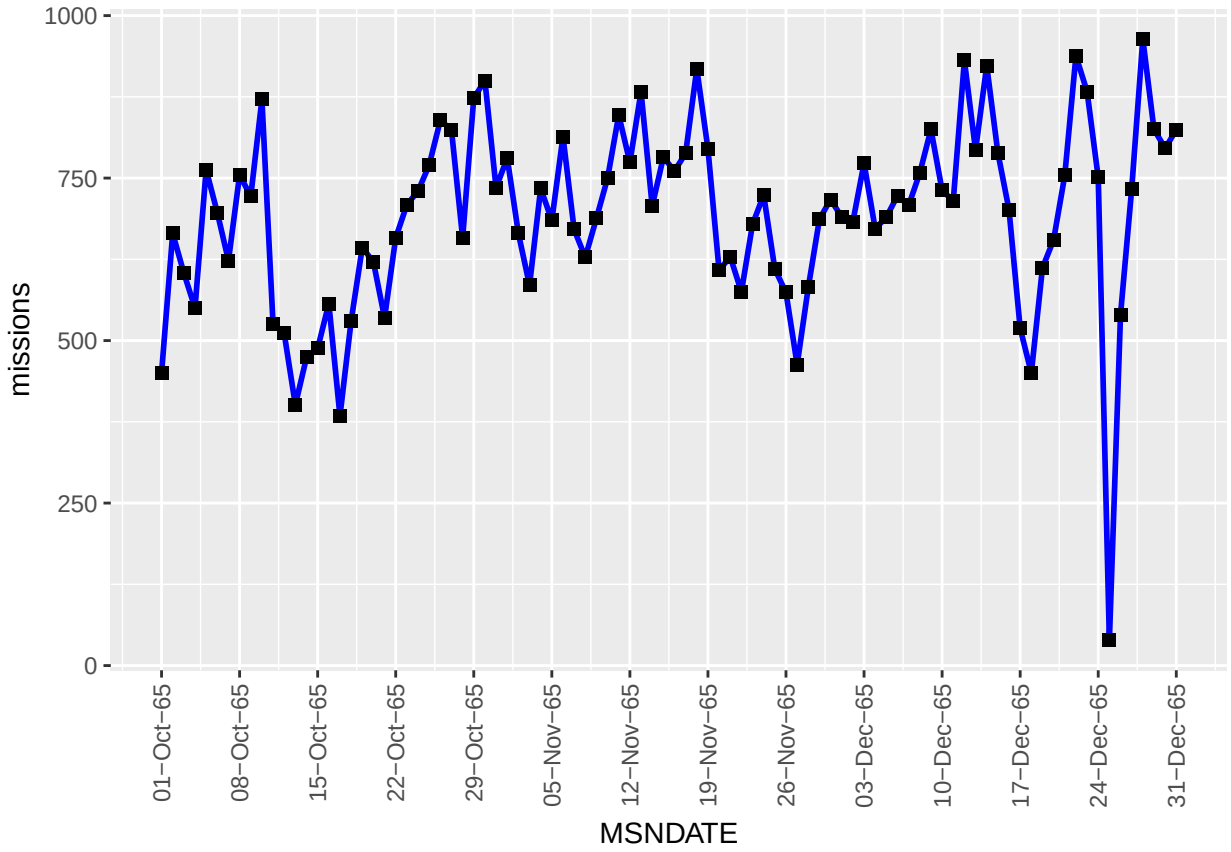
We can make some adjustments to the look of a line chart.

A good reference for points and lines: http://www.cookbook-r.com/Graphs/Shapes_and_line_types/

Let's make the following changes:

1. Increase the width and color of the line with `geom_line(size = 1, color = "blue")`.
2. Add points and change the shape and size of the point with `geom_point(shape = 15, size = 2)`.

```
ggplot(msn,aes(x=MSNDATE, y= missions)) +
  geom_line(size = 1, color = "blue") +
  geom_point(shape = 15, size = 2) +
  scale_x_date(breaks=datebreaks, labels = date_format("%d-%b-%y")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust = .5))
```



7.6 Time Series Exercise

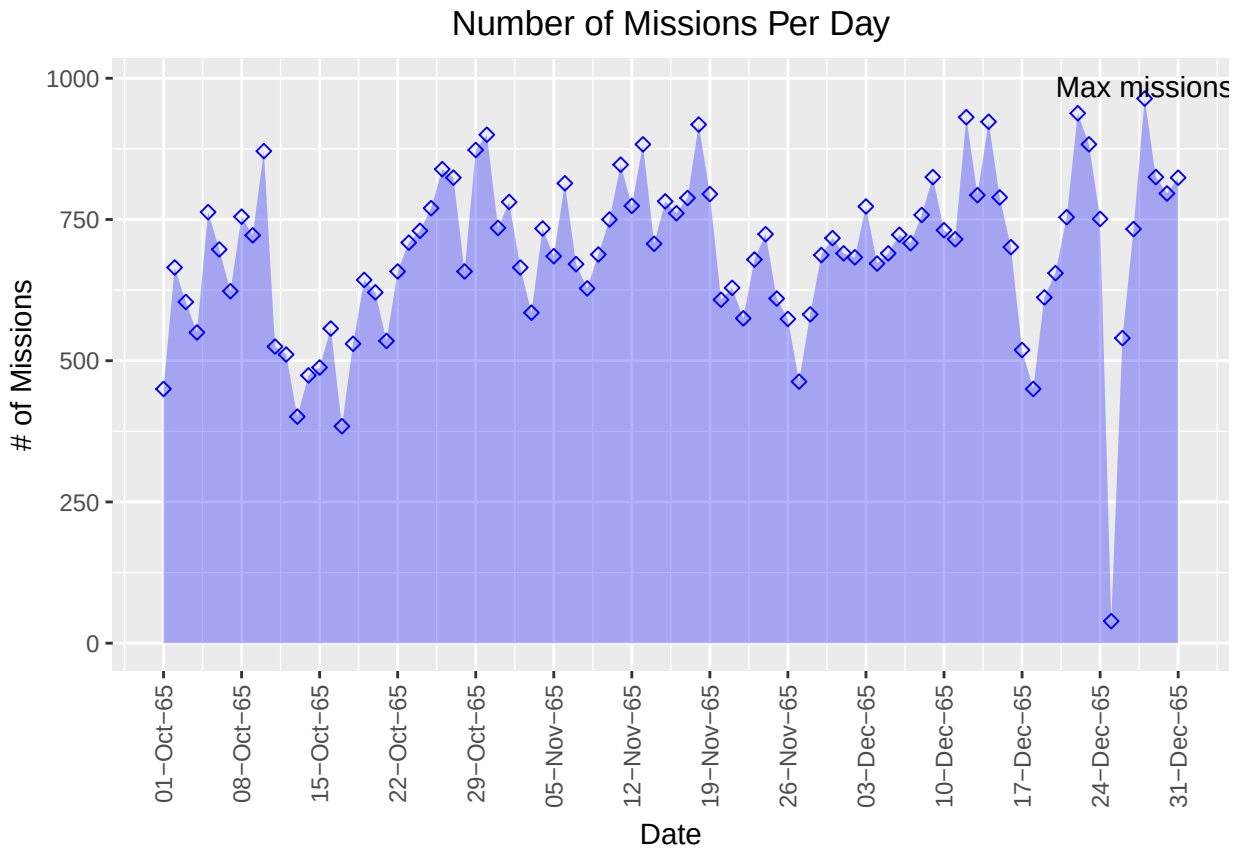
With the mission time series chart:

1. Use the area geom instead of line geom.
2. Change the points to blue and the shape to a diamond.
3. Add a title to the graph and center it.
4. Change the x-axis label to “Date”
5. Change the y-axis label to “# of Missions”
6. Use “alpha” parameter to adjust the transparency of the area geom.
7. Change the fill color of the area geom to blue.
8. Make an annotation of “Max Missions” for the day with the most missions above the peak on the chart.

7.7 Time Series Exercise Solution

```
ggplot(msn,aes(x=MSNDATE, y= missions))+
  geom_area(fill='blue',alpha=.3)+
  geom_point(color='blue', shape = 5)+
  scale_x_date(breaks=datebreaks, labels = date_format("%d-%b-%y")) +
```

```
  theme(axis.text.x=element_text(angle=90, hjust = 1, vjust = .5)) +
  ggtitle("Number of Missions Per Day") +
  theme(plot.title = element_text(hjust = .5)) +
  xlab("Date") +
  ylab('# of Missions') +
  annotate("text", x = as.Date("1965-12-28"), y = 985, label = "Max missions")
```



7.8 Multi-Line Charts

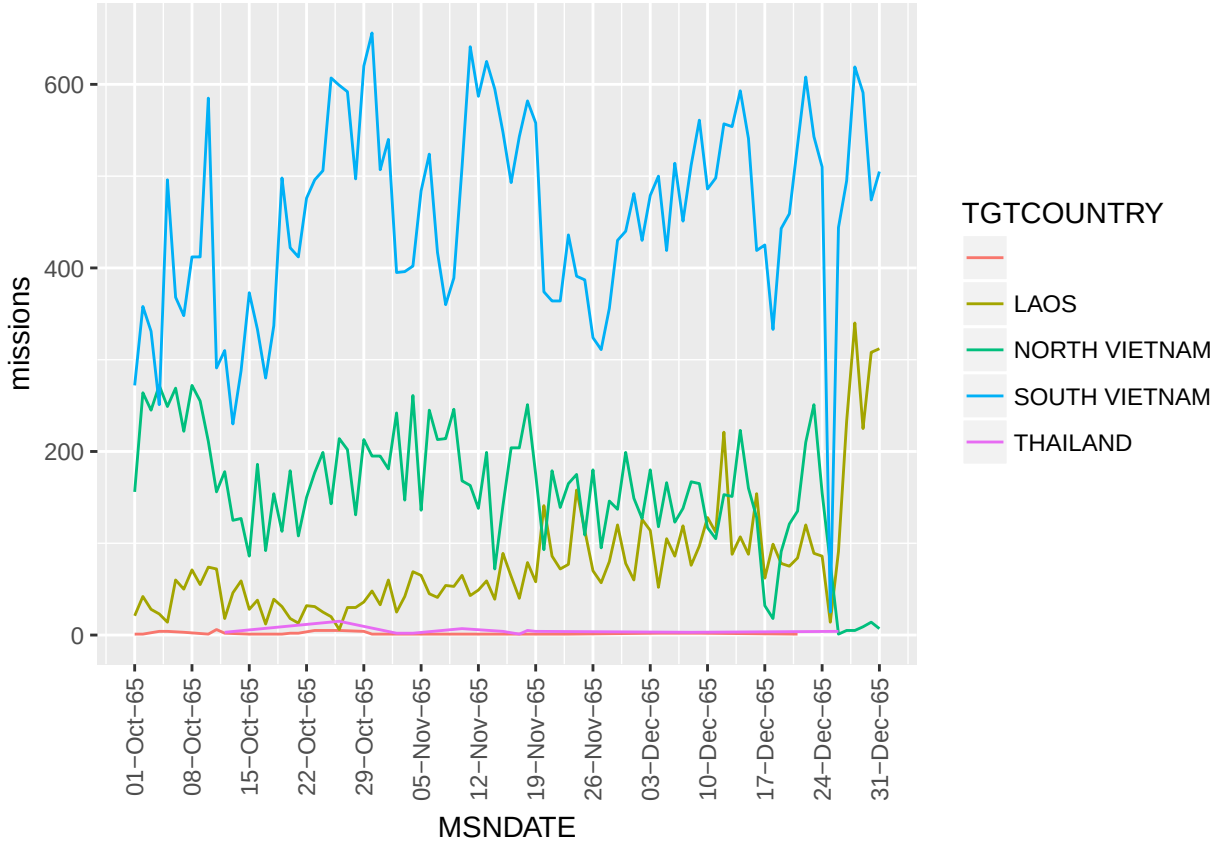
First, lets summarize by "MSNDATE" and "TGTCOUNTRY".

```
msn2 <- summarise(group_by(nam65, MSNDATE, TGTCOUNTRY), missions = n())
head(msn2)
```

```
## # A tibble: 6 x 3
## # Groups:   MSNDATE [2]
##      MSNDATE    TGTCOUNTRY missions
##       <date>         <chr>    <int>
## 1 1965-10-01                      1
## 2 1965-10-01          LAOS       21
## 3 1965-10-01 NORTH VIETNAM      156
## 4 1965-10-01 SOUTH VIETNAM      272
## 5 1965-10-02                      1
## 6 1965-10-02          LAOS       42
```

Chart a line for each country, using the `color` aesthetic to map to the "TGTCOUNTRY" variable.

```
ggplot(msn2,aes(x=MSNDATE, y= missions, color = TGTCOUNTRY)) +
  geom_line() +
  scale_x_date(breaks=datebreaks, labels = date_format("%d-%b-%y")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust = .5))
```

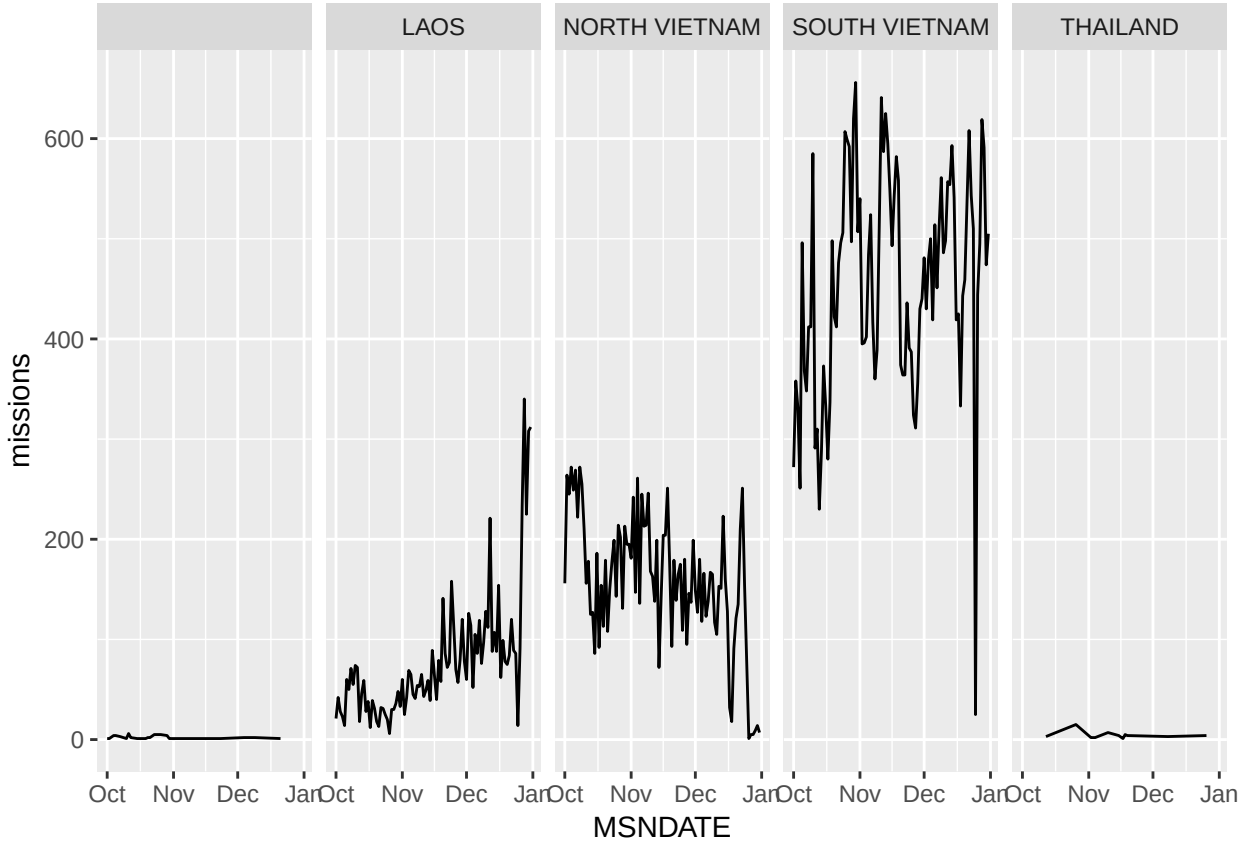


7.9 Facet Charts

A useful technique in data visualization is to render sub-plots of a set of variables, as opposed to having all the variables on one plot. This is called faceting. In this case, we will split out our multi-line chart into a sub-plot for each target country.

- Use `facet_grid()` or `facet_wrap()` to plot subsets on separate panels.
- To display sub-panels vertically, use `facet_grid(TGTCOUNTRY ~ .)`
- To display sub-panels horizontally, use `facet_grid(. ~ TGTCOUNTRY)`
- To display a sequence that wraps based on a number of rows and columns, use `facet_wrap(~TGTCOUNTRY, ncol = 2)`

```
ggplot(msn2,aes(x=MSNDATE, y= missions)) +
  geom_line() +
  facet_grid(. ~ TGTCOUNTRY)
```



7.10 Facet Exercise

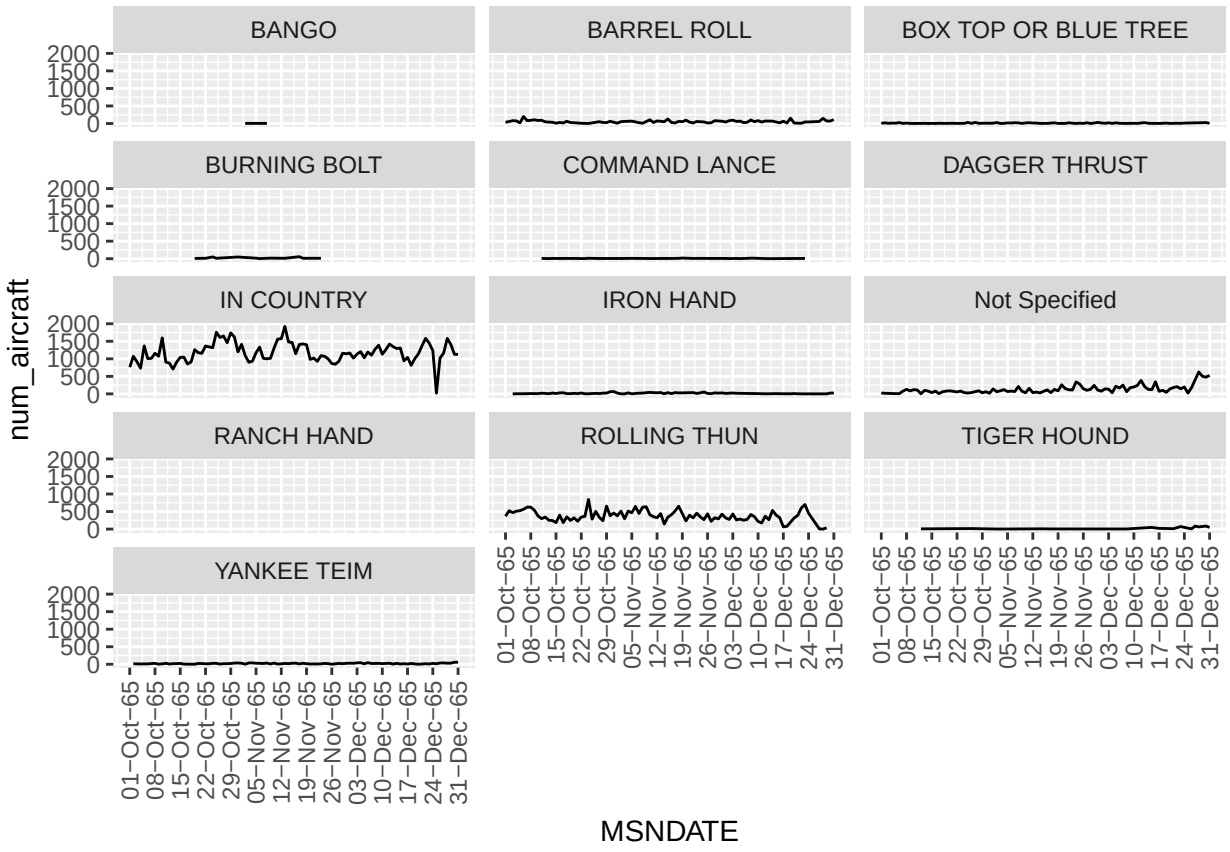
Using the vietnam dataset:

1. Create a new dataframe for number of aircraft used per operation by day. Hint: `summarize(group_by())`
2. Replace the blanks in the OPERATIONSUPPORTED variable with "Not Specified".
3. Make a facet time series chart with number of aircraft on the y-axis and facets by operation supported.

7.11 Facet Exercise Solution

```
pe_facet <- summarize(group_by(nam65, MSNDATE, OPERATIONSUPPORTED), num_aircraft = sum(NUMOFACFT))
pe_facet$OPERATIONSUPPORTED[pe_facet$OPERATIONSUPPORTED==""] <- "Not Specified"
datebreaks<-seq(as.Date("1965-10-01"),as.Date("1965-12-31"), by="week")

ggplot(data = pe_facet, aes(x = MSNDATE, y=num_aircraft))+
  geom_line()+
  scale_x_date(breaks=datebreaks, labels = date_format("%d-%b-%y")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust = .5))+
  facet_wrap(~OPERATIONSUPPORTED, ncol = 3)
```


BANGO
BARREL ROLL
BOX TOP OR BLUE TREE
BURNING BOLT
COMMAND LANCE
DAGGER THRUST
IN COUNTRY
IRON HAND
Not Specified
RANCH HAND
ROLLING THUN
TIGER HOUND
YANKEE TEIM
num_aircraft
2000
1500
1000
500
0
01–Oct–65
08–Oct–65
15–Oct–65
22–Oct–65
29–Oct–65
05–Nov–65
12–Nov–65
19–Nov–65
26–Nov–65
03–Dec–65
10–Dec–65
17–Dec–65
24–Dec–65
31–Dec–65
MSNDATE

Chapter 8

Histograms

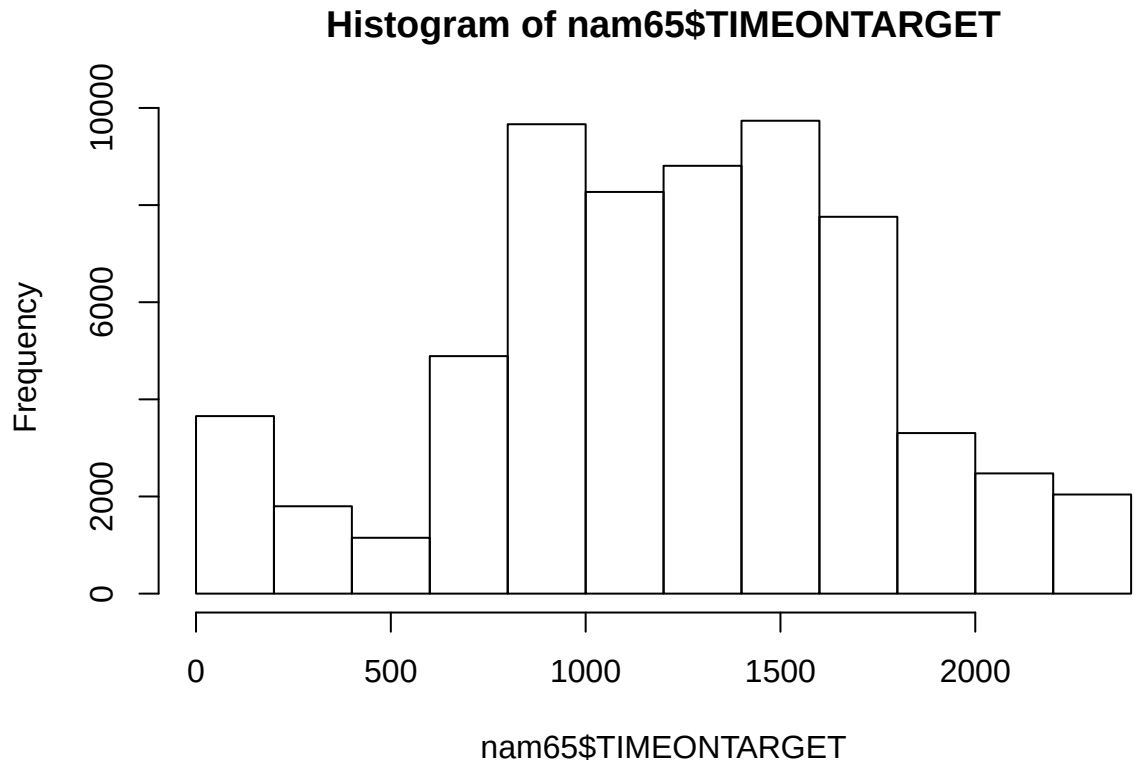
A histogram is used to map a continuous variable to the x-axis and use bins to depict the distribution of that variable.

8.1 hist()

Let's take a look at the distribution of missions by time of day based on the "TIMEONTARGET" variable.

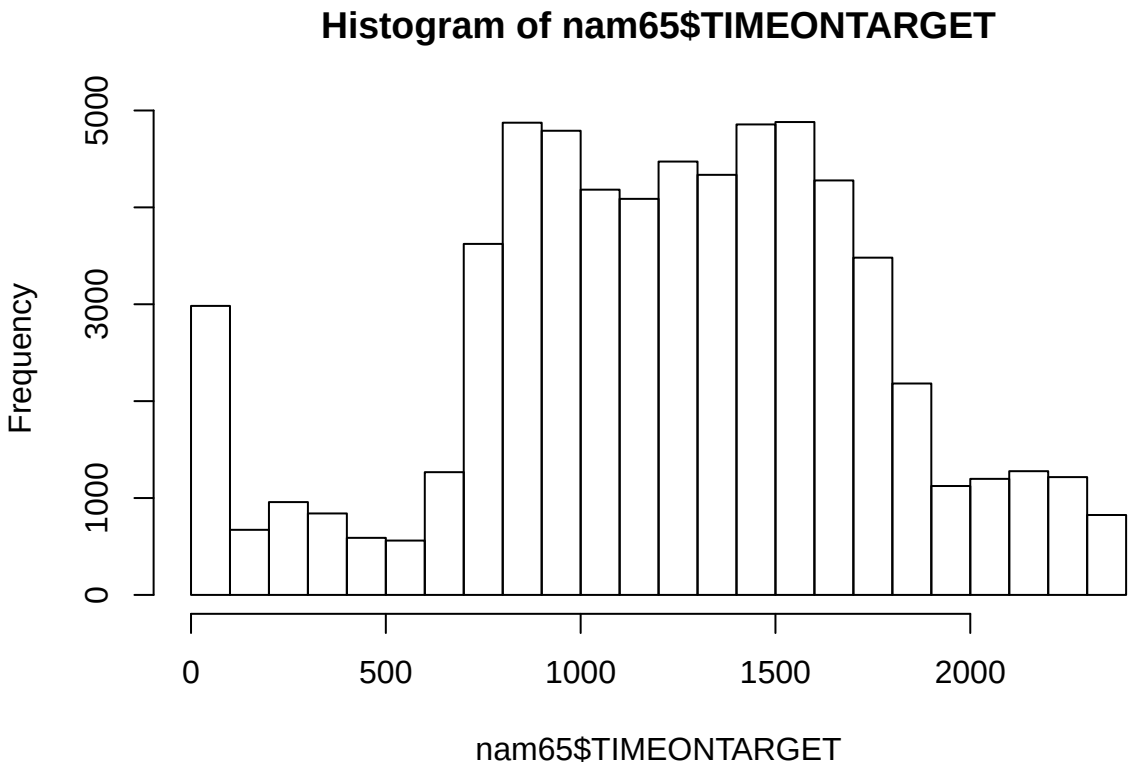
Base R has a simple histogram function, `hist()`.

```
hist(nam65$TIMEONTARGET)
```



We can adjust the number of bins with the "breaks" parameter.

```
hist(nam65$TIMEONTARGET, breaks = 24)
```



8.2 geom_histogram()

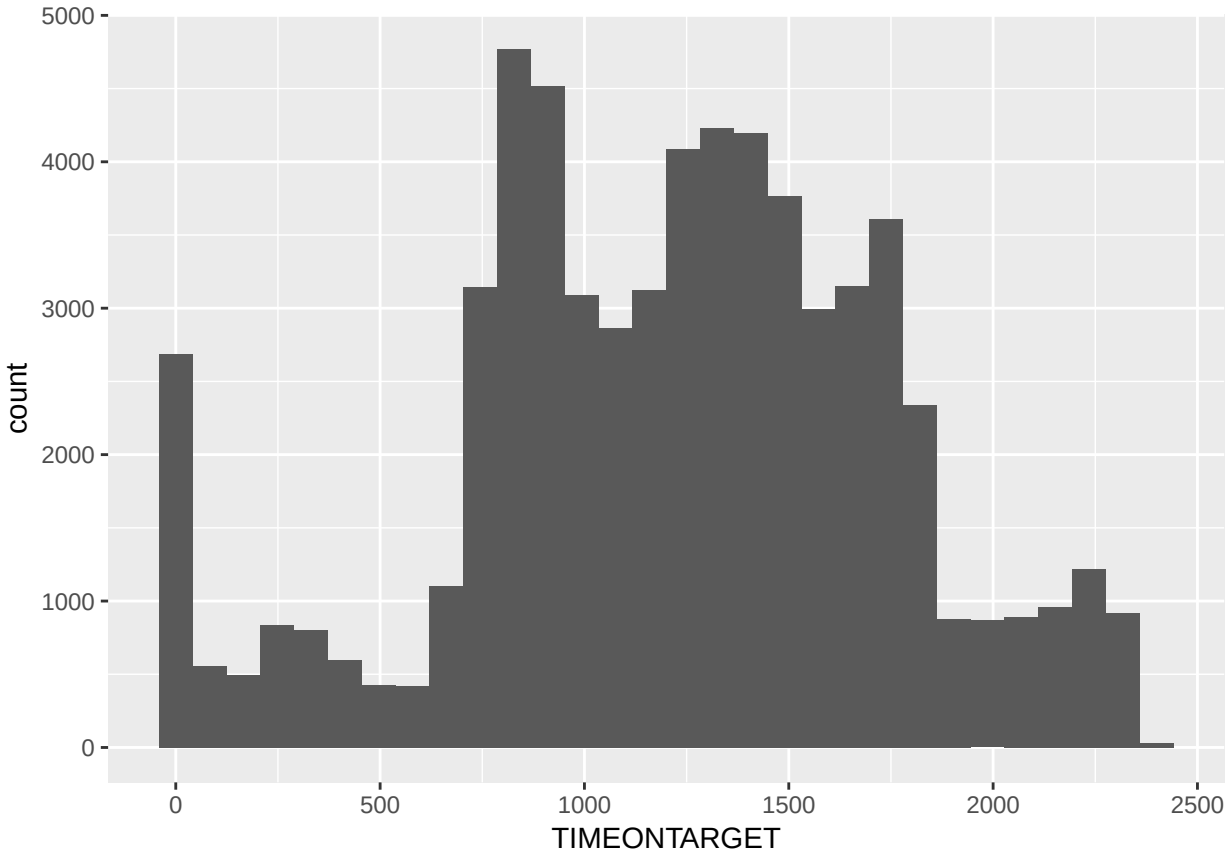
But if we want to apply the customization and other techniques we've learned to this plot, we need to use **ggplot2**.

In **ggplot2**, the geom for histograms is: `geom_histogram()`.

Here's a basic histogram with **ggplot2**:

```
ggplot(data = nam65, aes(x = TIMEONTARGET)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

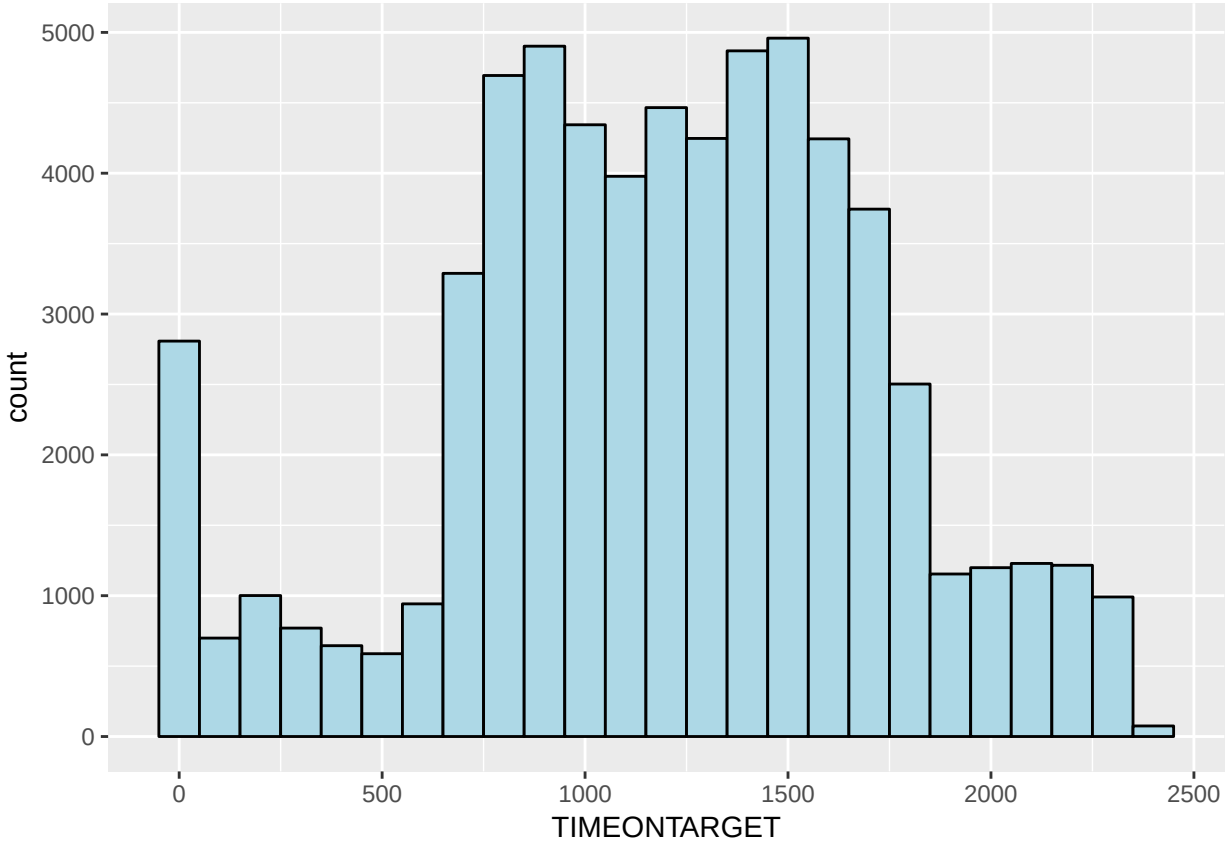


8.3 Bins

The default in **ggplot** is 30 bins. But we can set the number of bins in two ways.
First, we can change the width of the bin.

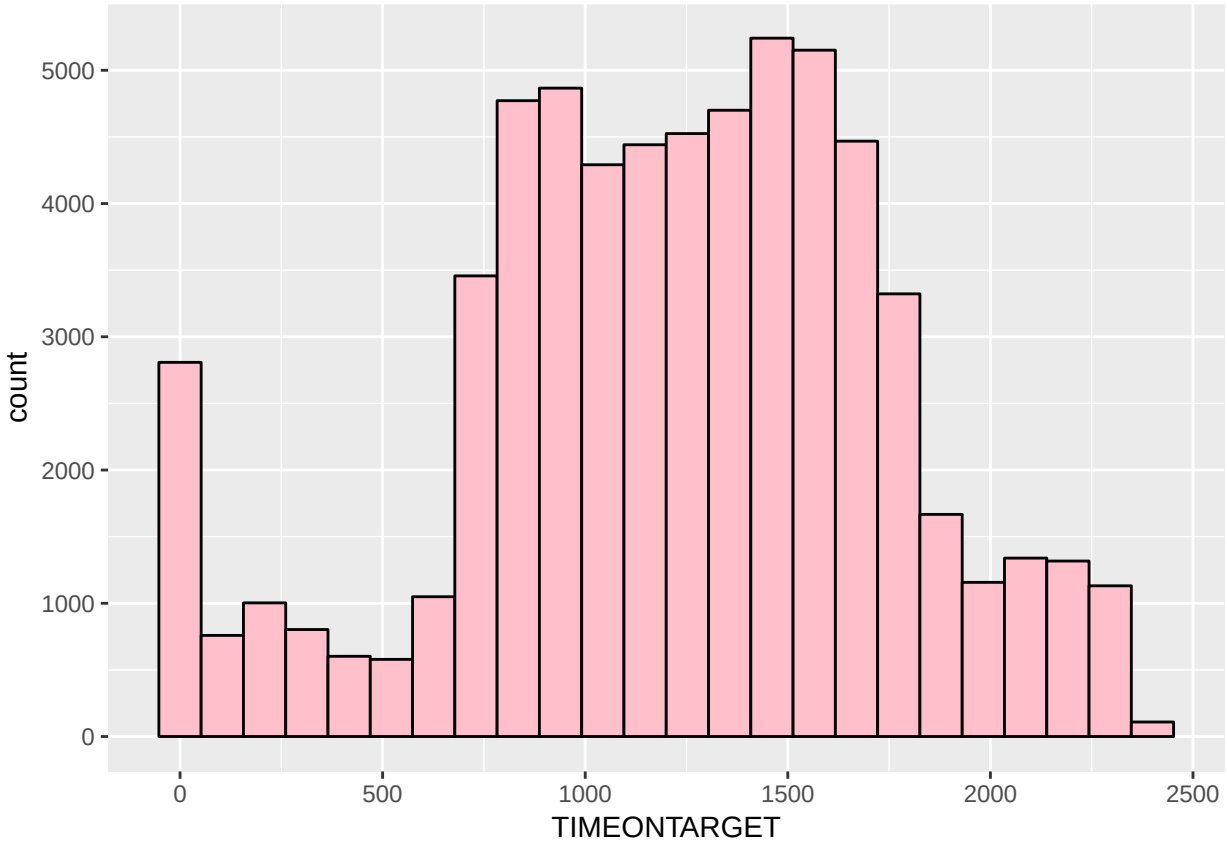
- Use "100" as the binwidth to approximate an hour.
- We will also adjust the fill and border colors to make the graph easier to read.

```
ggplot(data = nam65, aes(x = TIMEONTARGET)) +
  geom_histogram(binwidth = 100, fill = "lightblue", color = "black")
```



Another option is to set the number of bins with the "bins" parameter.

```
ggplot(data = nam65, aes(x = TIMEONTARGET)) +
  geom_histogram(bins = 24, fill = "pink", color = "black")
```



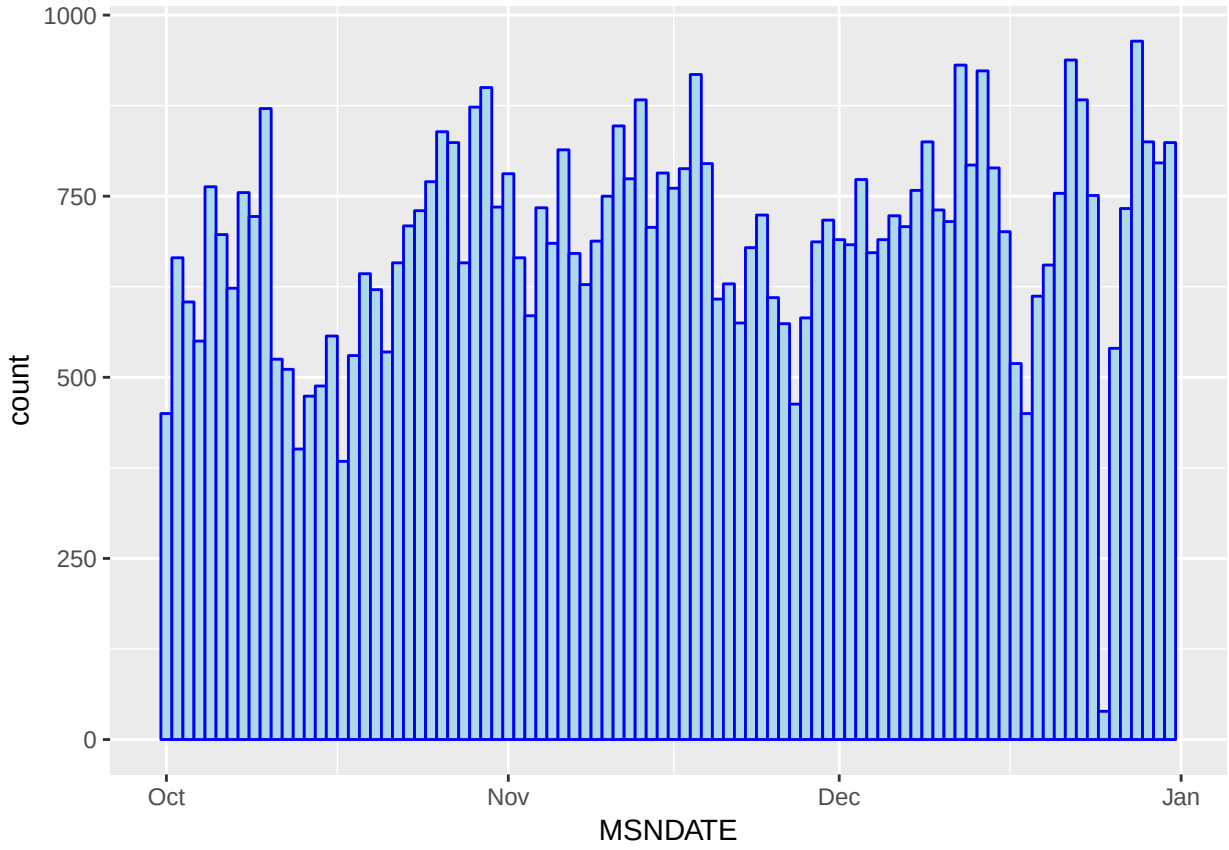
8.4 Histogram Exercise

1. Build a histogram for the distribution of missions per day. Hint: set the bins equal to the number of days in the dataset.

2. Make the color of the bars light blue and the outline of the bars navy blue.

8.5 Histogram Exercise Solution

```
ggplot(data = nam65, aes(x = MSNDATE))+
  geom_histogram(bins = 92, fill = "lightblue", color = "blue")
```


1000
750
500
250
0
count
Oct
Nov
Dec
Jan
MSNDATE

Chapter 9

Interactive Graphics

You can create interactive graphics with the **plotly** package.

- Plotly works as a wrapper around charts made in base R or ggplot.
- Create an object with your ggplot code, then use `ggplotly()` to convert the plot.

```
g <- ggplot(msn2,aes(x=MSNDATE, y= missions, color = TGTCOUNTRY)) +
  geom_line()+
  scale_x_date(breaks=datebreaks, labels = date_format("%d-%b-%y")) +
  theme(axis.text.x=element_text(angle=90, hjust=1, vjust = .5))

ggplotly(g)
```