

# R Training Session

Plot Together: Temperatures in Blacksburg

Erod Baybay

# Overview

- Useful functions for data manipulation and plotting
- The ‘hierarchy of aesthetics’ in ggplot2 (I made that term up)
- The troubleshooting and problem solving workflow for presenting data

# Loading Packages

- **tidyverse** – For data organization, ‘ggplot2’, dplyr, etc.’

```
library(tidyverse) # install.packages("tidyverse")
```

# Loading Packages

- **tidyverse** – For data organization, ‘ggplot2’, dplyr, etc.’

```
library(tidyverse) # install.packages("tidyverse")
```

# Importing Data

- **Weather\_Data\_Blacksburg.csv** – Record temperatures for Blacksburg, Virginia from the National Weather Service

```
weather <- read.csv("https://raw.githubusercontent.com/erodb/datasets/master/weather/Weather_Data_Blacksburg.csv")
```

# Viewing the Data

Organized in rows for each day of the year, and columns with individual temperature statistics about that day

```
> colnames(weather)
 [1] "Day"           "High_Temp_Hottest"   "High_Temp_Hottest_Year" "High_Temp_Coldest"    "High_Temp_Coldest_Year" "Low_Temp_Coldest"      "Low_Temp_Coldest_Year"  "Low_Temp_Hottest"    "Low_Temp_Hottest_Year"
[10] "Normal_High"  "Normal_Low"        "Mean_Temp"          "Heating_Days"       "Cooling_Days"        "Min."              "Min."              "Min."              "Min."
>
> dim(weather)
[1] 365  14
>
> class(weather)
[1] "data.frame"
>
> summary(weather)
   Day   High_Temp_Hottest High_Temp_Hottest_Year High_Temp_Coldest High_Temp_Coldest_Year Low_Temp_Coldest Low_Temp_Coldest_Year Low_Temp_Hottest Low_Temp_Hottest_Year Normal_High Normal_Low Mean_Temp Heating_Days
1-Apr : 1 Min.  :59.00     Min.  :1952            Min.  : 2.00     Min.  :1954            Min.  :-18.00    Min.  :1954            Min.  :37.00     Min.  :1952            Min.  :41.0    Min.  :21.00  Min.  :31.00  Min.  : 0.00
1-Aug : 1 1st Qu.:73.00     1st Qu.:1973            1st Qu.:24.00     1st Qu.:1970            1st Qu.: 5.00    1st Qu.:1970            1st Qu.:51.00    1st Qu.:1977            1st Qu.:49.0    1st Qu.:27.00  1st Qu.:38.00  1st Qu.: 2.00
1-Dec : 1 Median :83.00     Median :1988            Median :42.00     Median :1985            Median :21.00    Median :1977            Median :59.00    Median :1996            Median :65.0    Median :39.00  Median :52.00  Median :13.00
1-Feb : 1 Mean   :81.59     Mean   :1986            Mean   :41.59     Mean   :1985            Mean   :21.18    Mean   :1979            Mean   :57.85    Mean   :1991            Mean   :63.2    Mean   :40.15  Mean   :51.64  Mean   :14.89
1-Jan : 1 3rd Qu.:90.00     3rd Qu.:2005            3rd Qu.:60.00     3rd Qu.:2000            3rd Qu.: 38.00   3rd Qu.:1988            3rd Qu.:66.00    3rd Qu.:2010            3rd Qu.:78.0    3rd Qu.:54.00  3rd Qu.:66.00  3rd Qu.:27.00
1-Jul : 1 Max.   :99.00     Max.   :2017            Max.   :75.00     Max.   :2016            Max.   : 53.00   Max.   :2015            Max.   :72.00    Max.   :2017            Max.   :82.0    Max.   :60.00  Max.   :71.00  Max.   :34.00
(Other):359
   Cooling_Days
Min.  :0.000
1st Qu.:0.000
Median :0.000
Mean   :1.542
3rd Qu.:3.000
Max.   :7.000
```

# Viewing the Data

Organized in rows for each day of the year, and columns with individual temperature statistics about that day

```
> colnames(weather)
 [1] "Day"           "High_Temp_Hottest"   "High_Temp_Hottest_Year" "High_Temp_Coldest"    "High_Temp_Coldest_Year" "Low_Temp_Coldest"      "Low_Temp_Coldest_Year"  "Low_Temp_Hottest"    "Low_Temp_Hottest_Year"
[10] "Normal_High"  "Normal_Low"        "Mean_Temp"          "Heating_Days"       "Cooling_Days"
>
> dim(weather)
[1] 365  14
>
> class(weather)
[1] "data.frame"
>
> summary(weather)
   Day   High_Temp_Hottest High_Temp_Hottest_Year High_Temp_Coldest High_Temp_Coldest_Year Low_Temp_Coldest Low_Temp_Coldest_Year Low_Temp_Hottest Low_Temp_Hottest_Year Normal_High Normal_Low Mean_Temp Heating_Days
1-Apr : 1 Min.  :59.00 Min.  :1952      Min.  : 2.00 Min.  :1954      Min.  :-18.00 Min.  :1954      Min.  :37.00 Min.  :1952      Min.  :41.0  Min.  :21.00 Min.  :31.00 Min.  : 0.00
1-Aug : 1 1st Qu.:73.00 1st Qu.:1973      1st Qu.:24.00 1st Qu.:1970      1st Qu.: 5.00 1st Qu.:1970      1st Qu.:51.00 1st Qu.:1977      1st Qu.:49.0 1st Qu.:27.00 1st Qu.:38.00 1st Qu.: 2.00
1-Dec : 1 Median :83.00 Median :1988      Median :42.00 Median :1985      Median :21.00 Median :1977      Median :59.00 Median :1996      Median :65.0  Median :39.00 Median :52.00 Median :13.00
1-Feb : 1 Mean   :81.59 Mean   :1986      Mean   :41.59 Mean   :1985      Mean   :21.18 Mean   :1979      Mean   :57.85 Mean   :1991      Mean   :63.2  Mean   :40.15 Mean   :51.64 Mean   :14.89
1-Jan : 1 3rd Qu.:90.00 3rd Qu.:2005      3rd Qu.:60.00 3rd Qu.:2000      3rd Qu.:38.00 3rd Qu.:1988      3rd Qu.:66.00 3rd Qu.:2010      3rd Qu.:78.0 3rd Qu.:54.00 3rd Qu.:66.00 3rd Qu.:27.00
1-Jul : 1 Max.   :99.00 Max.   :2017      Max.   :75.00 Max.   :2016      Max.   :53.00 Max.   :2015      Max.   :72.00 Max.   :2017      Max.   :82.0  Max.   :60.00 Max.   :71.00 Max.   :34.00
(Other):359
   Cooling_Days
Min.  :0.000
1st Qu.:0.000
Median :0.000
Mean   :1.542
3rd Qu.:3.000
Max.   :7.000
```

> **colnames(weather)**

Gives column names of a data frame

# Viewing the Data

Organized in rows for each day of the year, and columns with individual temperature statistics about that day

```
> colnames(weather)
 [1] "Day"           "High_Temp_Hottest"   "High_Temp_Hottest_Year" "High_Temp_Coldest"    "High_Temp_Coldest_Year" "Low_Temp_Coldest"      "Low_Temp_Coldest_Year"  "Low_Temp_Hottest"    "Low_Temp_Hottest_Year"
[10] "Normal_High"  "Normal_Low"        "Mean_Temp"          "Heating_Days"       "Cooling_Days"        "Min."              "Min."              "Min."              "Min."
>
> dim(weather)
[1] 365  14
>
> class(weather)
[1] "data.frame"
>
> summary(weather)
   Day   High_Temp_Hottest High_Temp_Hottest_Year High_Temp_Coldest High_Temp_Coldest_Year Low_Temp_Coldest Low_Temp_Coldest_Year Low_Temp_Hottest Low_Temp_Hottest_Year Normal_High Normal_Low Mean_Temp Heating_Days
1-Apr : 1 Min.  :59.00 Min.  :1952 Min.  : 2.00 Min.  :1954 Min.  :-18.00 Min.  :1954 Min.  :37.00 Min.  :1952 Min.  :41.0  Min.  :21.00 Min.  :31.00 Min.  : 0.00
1-Aug : 1 1st Qu.:73.00 1st Qu.:1973 1st Qu.:24.00 1st Qu.:1970 1st Qu.: 5.00 1st Qu.:1970 1st Qu.:51.00 1st Qu.:1977 1st Qu.:49.0 1st Qu.:27.00 1st Qu.:38.00 1st Qu.: 2.00
1-Dec : 1 Median :83.00 Median :1988 Median :42.00 Median :1985 Median :21.00 Median :1977 Median :59.00 Median :1996 Median :65.0  Median :39.00 Median :52.00 Median :13.00
1-Feb : 1 Mean   :81.59 Mean   :1986 Mean   :41.59 Mean   :1985 Mean   :21.18 Mean   :1979 Mean   :57.85 Mean   :1991 Mean   :63.2  Mean   :40.15 Mean   :51.64 Mean   :14.89
1-Jan : 1 3rd Qu.:90.00 3rd Qu.:2005 3rd Qu.:60.00 3rd Qu.:2000 3rd Qu.: 38.00 3rd Qu.:1988 3rd Qu.:66.00 3rd Qu.:2010 3rd Qu.:78.0 3rd Qu.:54.00 3rd Qu.:66.00 3rd Qu.:27.00
1-Jul : 1 Max.   :99.00 Max.   :2017 Max.   :75.00 Max.   :2016 Max.   :53.00 Max.   :2015 Max.   :72.00 Max.   :2017 Max.   :82.0  Max.   :60.00 Max.   :71.00 Max.   :34.00
(Other):359
   Cooling_Days
Min.  :0.000
1st Qu.:0.000
Median :0.000
Mean   :1.542
3rd Qu.:3.000
Max.   :7.000
```

> **dim(weather)**

Gives dimensions of a data frame

# Viewing the Data

Organized in rows for each day of the year, and columns with individual temperature statistics about that day

```
> colnames(weather)
 [1] "Day"           "High_Temp_Hottest"   "High_Temp_Hottest_Year" "High_Temp_Coldest"    "High_Temp_Coldest_Year" "Low_Temp_Coldest"      "Low_Temp_Coldest_Year"  "Low_Temp_Hottest"    "Low_Temp_Hottest_Year"
[10] "Normal_High"  "Normal_Low"        "Mean_Temp"          "Heating_Days"       "Cooling_Days"        "Min."              "Min."              "Min."              "Min."
>
> dim(weather)
[1] 365  14
>
> class(weather)
[1] "data.frame"
>
> summary(weather)
   Day   High_Temp_Hottest High_Temp_Hottest_Year High_Temp_Coldest High_Temp_Coldest_Year Low_Temp_Coldest Low_Temp_Coldest_Year Low_Temp_Hottest Low_Temp_Hottest_Year Normal_High Normal_Low Mean_Temp Heating_Days
1-Apr : 1 Min.  :59.00 Min.  :1952 Min.  : 2.00 Min.  :1954 Min.  :-18.00 Min.  :1954 Min.  :37.00 Min.  :1952 Min.  :41.0  Min.  :21.00 Min.  :31.00 Min.  : 0.00
1-Aug : 1 1st Qu.:73.00 1st Qu.:1973 1st Qu.:24.00 1st Qu.:1970 1st Qu.: 5.00 1st Qu.:1970 1st Qu.:51.00 1st Qu.:1977 1st Qu.:49.0 1st Qu.:27.00 1st Qu.:38.00 1st Qu.: 2.00
1-Dec : 1 Median :83.00 Median :1988 Median :42.00 Median :1985 Median :21.00 Median :1977 Median :59.00 Median :1996 Median :65.0  Median :39.00 Median :52.00 Median :13.00
1-Feb : 1 Mean   :81.59 Mean   :1986 Mean   :41.59 Mean   :1985 Mean   :21.18 Mean   :1979 Mean   :57.85 Mean   :1991 Mean   :63.2  Mean   :40.15 Mean   :51.64 Mean   :14.89
1-Jan : 1 3rd Qu.:90.00 3rd Qu.:2005 3rd Qu.:60.00 3rd Qu.:2000 3rd Qu.: 38.00 3rd Qu.:1988 3rd Qu.:66.00 3rd Qu.:2010 3rd Qu.:78.0 3rd Qu.:54.00 3rd Qu.:66.00 3rd Qu.:27.00
1-Jul : 1 Max.   :99.00 Max.   :2017 Max.   :75.00 Max.   :2016 Max.   :53.00 Max.   :2015 Max.   :72.00 Max.   :2017 Max.   :82.0  Max.   :60.00 Max.   :71.00 Max.   :34.00
(Other):359
   Cooling_Days
Min.  :0.000
1st Qu.:0.000
Median :0.000
Mean   :1.542
3rd Qu.:3.000
Max.   :7.000
```

**> class(weather)** Gives the object (data type) of an object

# Viewing the Data

Organized in rows for each day of the year, and columns with individual temperature statistics about that day

```
> colnames(weather)
 [1] "Day"           "High_Temp_Hottest"   "High_Temp_Hottest_Year" "High_Temp_Coldest"    "High_Temp_Coldest_Year" "Low_Temp_Coldest"      "Low_Temp_Coldest_Year"  "Low_Temp_Hottest"    "Low_Temp_Hottest_Year"
[10] "Normal_High"  "Normal_Low"        "Mean_Temp"          "Heating_Days"       "Cooling_Days"        "Min."              "Min."              "Min."              "Min."
>
> dim(weather)
[1] 365 14
>
> class(weather)
[1] "data.frame"
>
> summary(weather)
   Day   High_Temp_Hottest High_Temp_Hottest_Year High_Temp_Coldest High_Temp_Coldest_Year Low_Temp_Coldest Low_Temp_Coldest_Year Low_Temp_Hottest Low_Temp_Hottest_Year Normal_High Normal_Low Mean_Temp Heating_Days
1-Apr : 1 Min.  :59.00 Min.  :1952 Min.  : 2.00 Min.  :1954 Min.  :-18.00 Min.  :1954 Min.  :37.00 Min.  :1952 Min.  :41.0  Min.  :21.00 Min.  :31.00 Min.  : 0.00
1-Aug : 1 1st Qu.:73.00 1st Qu.:1973 1st Qu.:24.00 1st Qu.:1970 1st Qu.: 5.00 1st Qu.:1970 1st Qu.:51.00 1st Qu.:1977 1st Qu.:49.0 1st Qu.:27.00 1st Qu.:38.00 1st Qu.: 2.00
1-Dec : 1 Median :83.00 Median :1988 Median :42.00 Median :1985 Median :21.00 Median :1977 Median :59.00 Median :1996 Median :65.0  Median :39.00 Median :52.00 Median :13.00
1-Feb : 1 Mean   :81.59 Mean   :1986 Mean   :41.59 Mean   :1985 Mean   :21.18 Mean   :1979 Mean   :57.85 Mean   :1991 Mean   :63.2  Mean   :40.15 Mean   :51.64 Mean   :14.89
1-Jan : 1 3rd Qu.:90.00 3rd Qu.:2005 3rd Qu.:60.00 3rd Qu.:2000 3rd Qu.: 38.00 3rd Qu.:1988 3rd Qu.:66.00 3rd Qu.:2010 3rd Qu.:78.0 3rd Qu.:54.00 3rd Qu.:66.00 3rd Qu.:27.00
1-Jul : 1 Max.   :99.00 Max.   :2017 Max.   :75.00 Max.   :2016 Max.   :53.00 Max.   :2015 Max.   :72.00 Max.   :2017 Max.   :82.0  Max.   :60.00 Max.   :71.00 Max.   :34.00
(Other):359
   Cooling_Days
Min.  :0.000
1st Qu.:0.000
Median :0.000
Mean   :1.542
3rd Qu.:3.000
Max.   :7.000
```

> **summary(weather)**

Gives a column by column summary of a dataframe

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- na.omit(weather)
```

`na.omit()` returns the input where the incomplete rows are removed. Any rows with NA are removed.

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- na.omit(weather)
```

`na.omit()` returns the input where the incomplete rows are removed. Any rows with NA are removed.

## The Pipe Operator

The pipe operator `%>%` takes things on the left of it, and inputs it into the right.

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- na.omit(weather)
```

`na.omit()` returns the input where the incomplete rows are removed. Any rows with NA are removed.

## The Pipe Operator

The pipe operator `%>%` takes things on the left of it, and inputs it into the right.

```
weather <- weather %>% separate(Day, sep = '- ', c('Day', 'Month')) %>% mutate(Day = as.numeric(Day))
```

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- na.omit(weather)
```

`na.omit()` returns the input where the incomplete rows are removed. Any rows with NA are removed.

## The Pipe Operator

The pipe operator `%>%` takes things on the left of it, and inputs it into the right.

```
weather <- weather %>% separate(Day, sep = '- ', c('Day', 'Month')) %>% mutate(Day = as.numeric(Day))
```

Is the same as:

```
weather <- separate(weather, Day, sep = '- ', c('Day', 'Month'))
weather <- mutate(weather, Day = as.numeric(Day))
```

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- na.omit(weather)
```

`na.omit()` returns the input where the incomplete rows are removed. Any rows with NA are removed.

## The Pipe Operator

The pipe operator `%>%` takes things on the left of it, and inputs it into the right.

```
weather <- weather %>% separate(Day, sep = '-', c('Day', 'Month')) %>% mutate(Day = as.numeric(Day))
```

Is the same as:

```
weather <- separate(weather, Day, sep = '-', c('Day', 'Month'))
weather <- mutate(weather, Day = as.numeric(Day))
```



Notice how we have **weather** as the first parameter in `mutate()` and `separate()`

The first parameter is usually the **input parameter** - The pipe operator handles the input for us

We use the pipe operator if we want to do a series of operations to a single piece of data – like a conveyor belt

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- weather %>% separate(Day, sep = '-', c('Day', 'Month')) %>% mutate(Day = as.numeric(Day))
```

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- weather %>% separate(Day, sep = '-', c('Day', 'Month')) %>% mutate(Day = as.numeric(Day))
```

`separate()` takes a column, and splits it into multiple columns based on a delimiter (in this case, ‘-’)

The diagram illustrates the transformation of a dataset using the `separate()` function. On the left, a table shows a single column named "Day" containing dates like "1-Jan", "2-Jan", etc. An arrow points to the right, indicating the transformation. On the right, the same data is shown in two columns: "Day" and "Month". The "Day" column now contains the numerical day values (1, 2, 3, 4, 5), and the "Month" column contains the month names ("Jan", "Jan", "Jan", "Jan", "Jan").

	Day
1	1-Jan
2	2-Jan
3	3-Jan
4	4-Jan
5	5-Jan

→

	Day	Month
1	1	Jan
2	2	Jan
3	3	Jan
4	4	Jan
5	5	Jan

# Cleaning Up Data

Manipulating data to be easier to work with.

```
weather <- weather %>% separate(Day, sep = '-', c('Day', 'Month')) %>% mutate(Day = as.numeric(Day))
```

**mutate()** can alter, manipulate, and create columns while keeping the input dataset

In this case we do something uneventful

**Alternative:** **transmute()** can alter, manipulate, and create columns while deleting the input dataset

# Exercise 1

Separate and Mutate (and also Pipes)

A tibble (fancier data frame) is made as follows:

```
myData <- tibble('AB' = c('1:5', '2:3', '4:9', '1:0', '5:4'))
```

# Exercise 1

Separate and Mutate (and also Pipes)

A tibble (fancier data frame) is made as follows:

```
myData <- tibble('AB' = c('1:5', '2:3', '4:9', '1:0', '5:4'))
```

How would you perform the transformation below?



	AB	
1	1:5	
2	2:3	
3	4:9	
4	1:0	
5	5:4	

	A	B	A+B
1	1	5	6
2	2	3	5
3	4	9	13
4	1	0	1
5	5	4	9

# Exercise 1

Separate and Mutate (and also Pipes)

A tibble (fancier data frame) is made as follows:

```
myData <- tibble('AB' = c('1:5', '2:3', '4:9', '1:0', '5:4'))
```

How would you perform the transformation below?



	AB	
1	1:5	
2	2:3	
3	4:9	
4	1:0	
5	5:4	

	A	B	A+B
1	1	5	6
2	2	3	5
3	4	9	13
4	1	0	1
5	5	4	9

A

```
myData <- myData %>%  
  mutate('A+B' = as.numeric(A) + as.numeric(B)) %>%  
  separate('AB', sep = ':', c('A', 'B'))
```

B

```
myData <- myData %>%  
  separate('AB', sep = ':', c('A', 'B')) %>%  
  mutate('A+B' = as.numeric(A) + as.numeric(B))
```

C

```
myData <- myData %>% ddapply(separate('AB', sep = ':', c('A', 'B')),  
  mutate('A+B' = as.numeric(A) + as.numeric(B)),  
  mode = 'simultaneous')
```

# Exercise 1

Separate and Mutate (and also Pipes)

A tibble (fancier data frame) is made as follows:

```
myData <- tibble('AB' = c('1:5', '2:3', '4:9', '1:0', '5:4'))
```

How would you perform the transformation below?



	AB	
1	1:5	
2	2:3	
3	4:9	
4	1:0	
5	5:4	

	A	B	A+B
1	1	5	6
2	2	3	5
3	4	9	13
4	1	0	1
5	5	4	9

A

```
myData <- myData %>%  
  mutate('A+B' = as.numeric(A) + as.numeric(B)) %>%  
  separate('AB', sep = ':', c('A', 'B'))
```

Wrong order

B

```
myData <- myData %>%  
  separate('AB', sep = ':', c('A', 'B')) %>%  
  mutate('A+B' = as.numeric(A) + as.numeric(B))
```

C

```
myData <- myData %>% ddapply(separate('AB', sep = ':', c('A', 'B')),  
  .by = 'AB', .f = function(A, B) {as.numeric(A) + as.numeric(B)},  
  mode = 'simultaneous')
```

ddapply() isn't even a function

# Initial Plotting

Visualizing our data

We begin by piping in our dataset (weather)

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

# Initial Plotting

Visualizing our data

We begin by piping in our dataset (weather)

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

`ggplot()` initializes the canvas of our plot

# Initial Plotting

Visualizing our data

We begin by piping in our dataset (weather)

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

`ggplot()` initializes the canvas of our plot

`aes()` assigns aesthetics to our plot based on our input (position, color, size on our plot)      This is called Mapping

# Initial Plotting

Visualizing our data

We begin by piping in our dataset (weather)

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

**geom\_segment()** creates line segments

Draws a line segment from **(x,y)** to **(xend, yend)**

# Initial Plotting

Visualizing our data

We begin by piping in our dataset (weather)

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

`geom_segment()` creates line segments

Draws a line segment from `(x,y)` to `(xend, yend)`

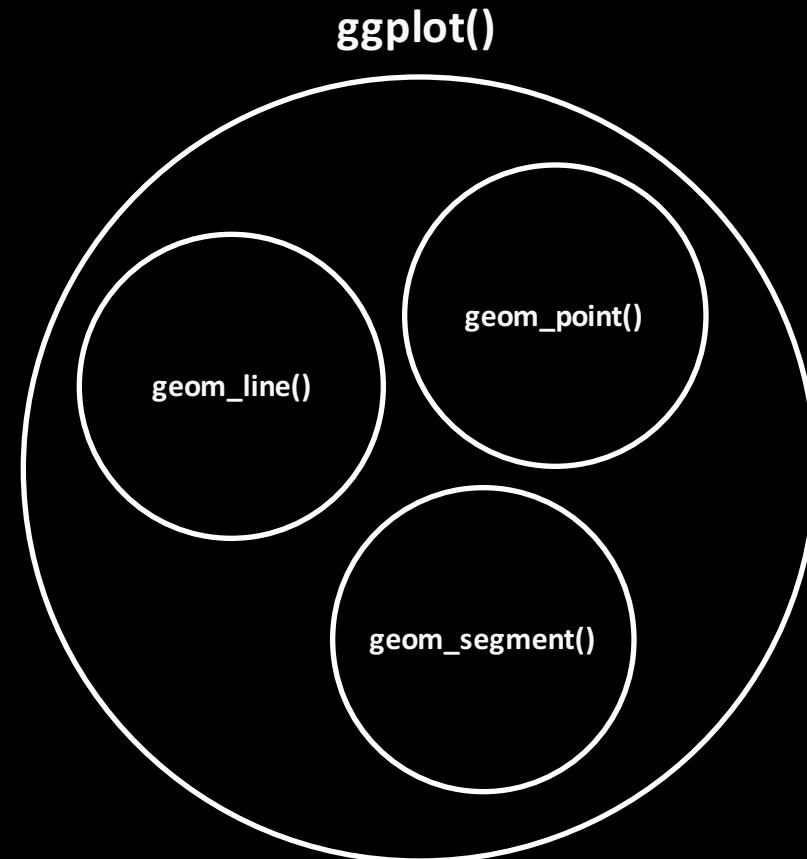


Also note how **Month** is mapped to **color**

Line thickness (size) is fixed to 3

# Initial Plotting

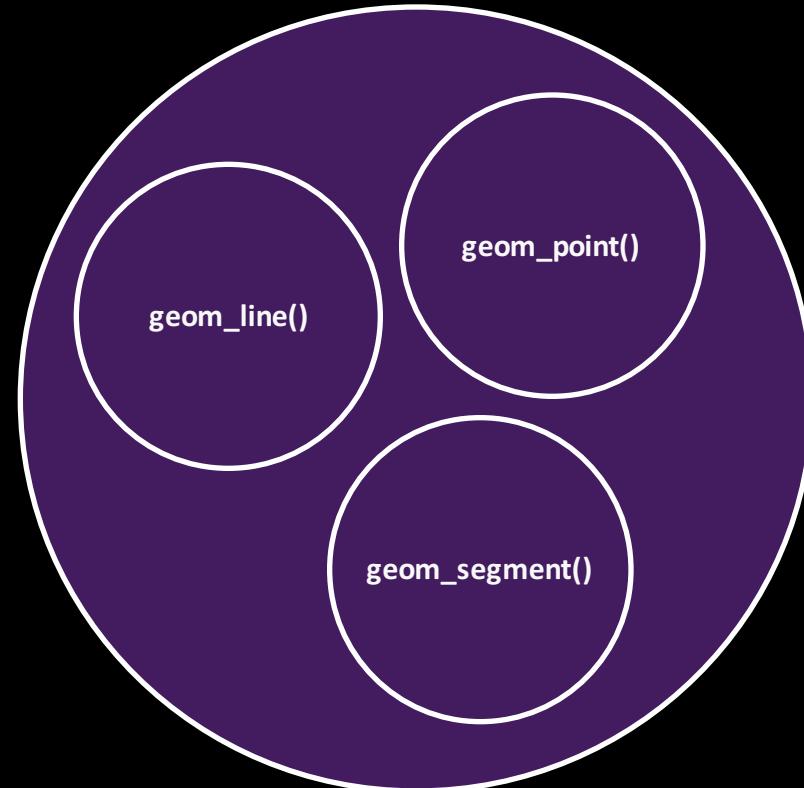
## Hierarchy of Aesthetics



# Initial Plotting

Hierarchy of Aesthetics

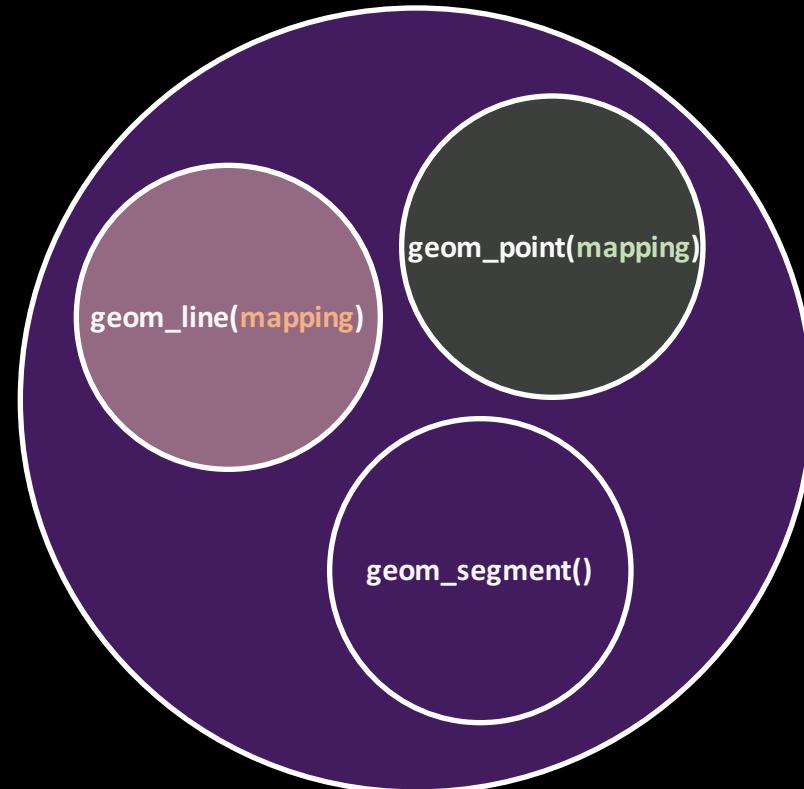
`ggplot(data, mapping)`



# Initial Plotting

## Hierarchy of Aesthetics

`ggplot(data, mapping)`



# Initial Plotting

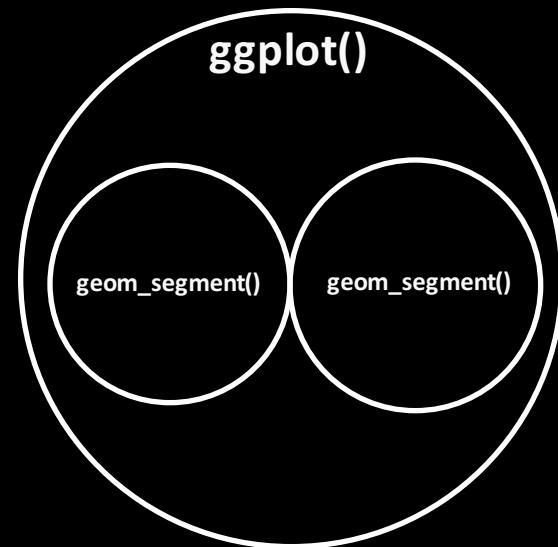
Visualizing our data

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

# Initial Plotting

Visualizing our data

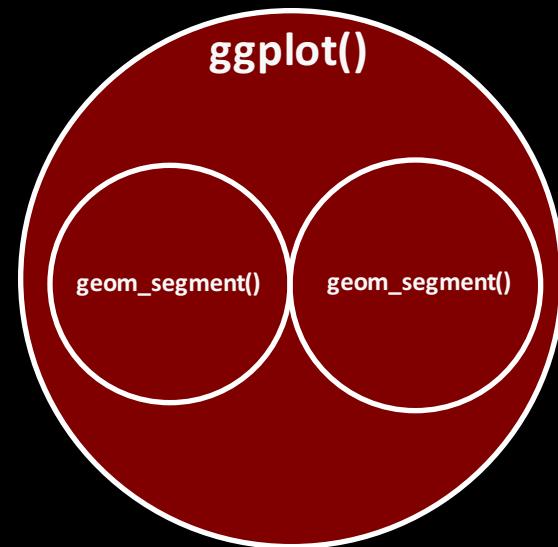
```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```



# Initial Plotting

Visualizing our data

```
weather %>%  
  ggplot(aes(group = Month)) +  This mapping is preserved across all geometries  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

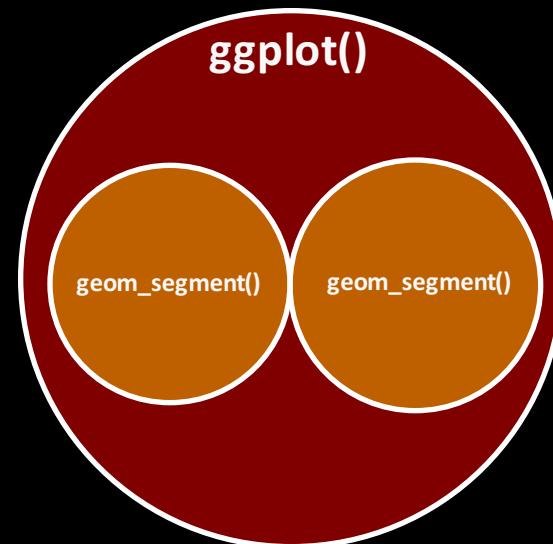


# Initial Plotting

Visualizing our data

```
weather %>%  
  ggplot(aes(group = Month)) +  This mapping is preserved across all geometries  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

These mappings apply only to their respective geometries



# Initial Plotting

Visualizing our data

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

What will we get if we run this?

# Initial Plotting

Visualizing our data

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

**What will we get if we run this?**

Spoiler: It won't be a useful plot

# Exercise 2

## Line Segments and Aesthetics

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

**geom\_segment()** creates line segments

Draws a line segment from **(x,y)** to **(xend, yend)**

Are the lines vertical or horizontal? How many **colors** will there be on our plot?

# Exercise 2

## Line Segments and Aesthetics

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

**geom\_segment()** creates line segments

Draws a line segment from **(x,y)** to **(xend, yend)**

Are the lines vertical or horizontal? How many **colors** will there be on our plot?

A      vertical, 12

C      horizontal, 12

B      vertical, 31

D      horizontal, 31

# Exercise 2

## Line Segments and Aesthetics

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3)
```

`geom_segment()` creates line segments

Draws a line segment from `(x,y)` to `(xend, yend)`

Are the lines vertical or horizontal? How many **colors** will there be on our plot?

`x = Day, xend = Day,`  
`color = Month`

A vertical, 12

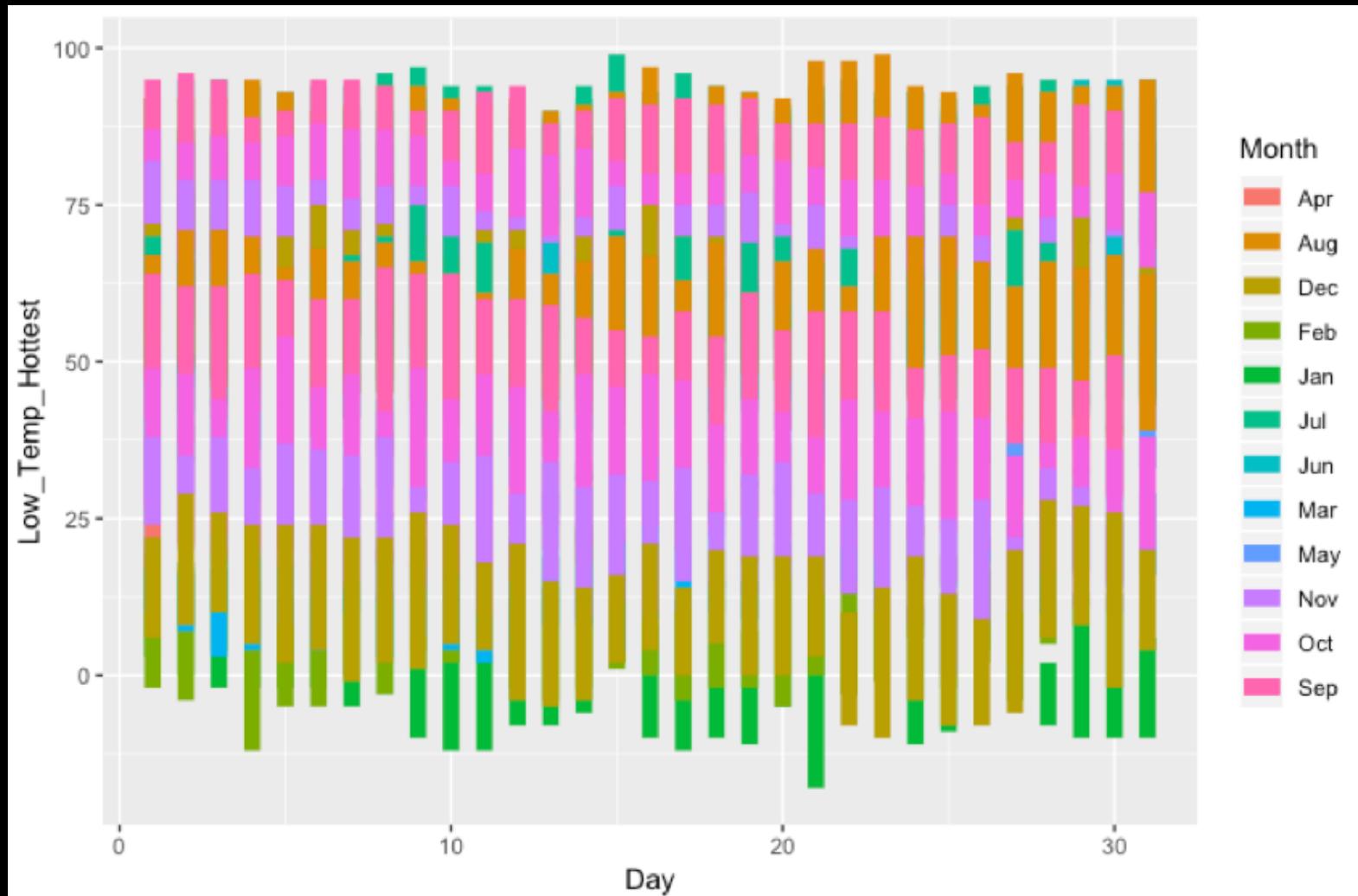
C horizontal, 12

B vertical, 31

D horizontal, 31

## Initial Plotting

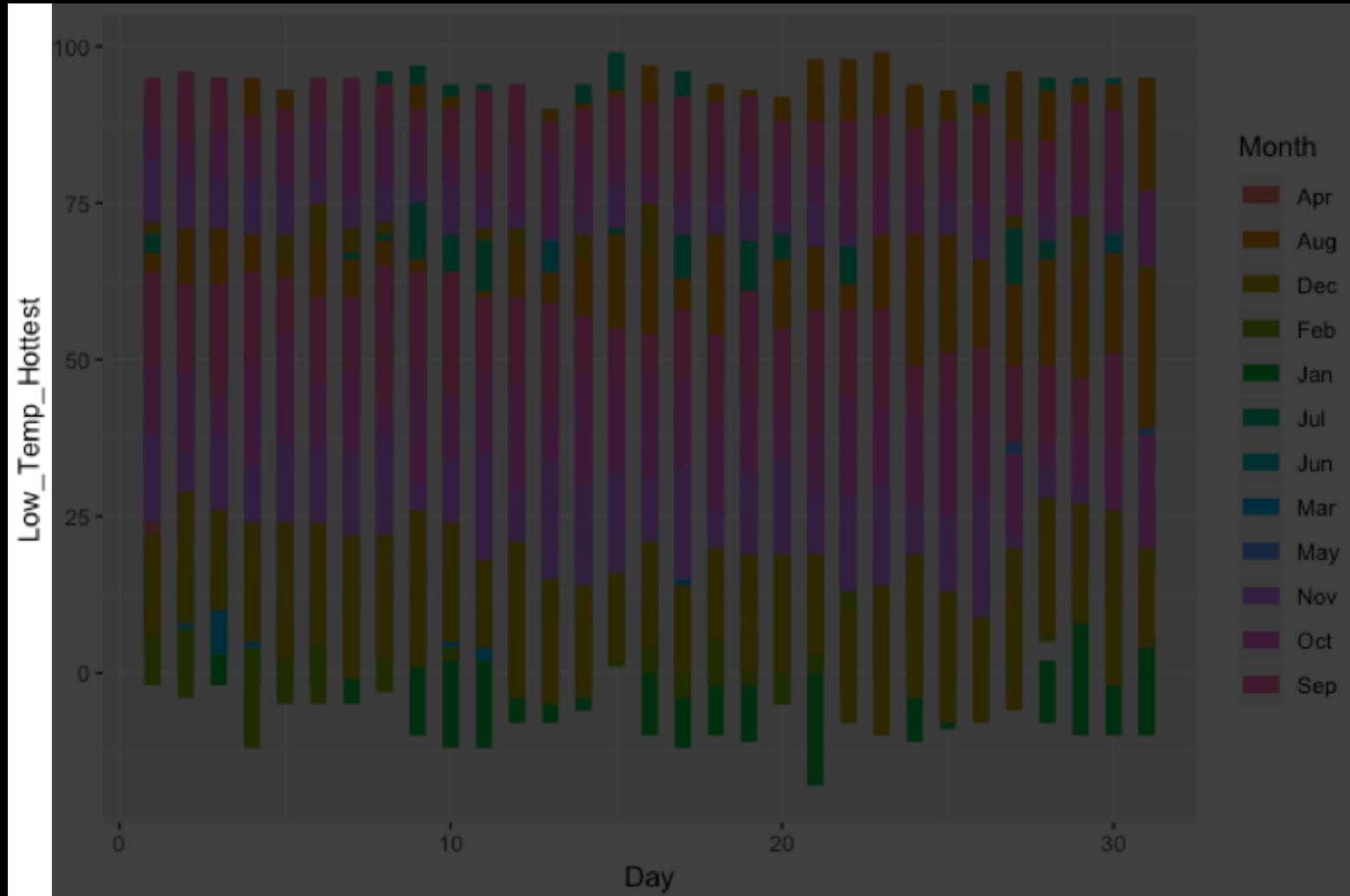
## Visualizing our data



# Initial Plotting

Visualizing our data

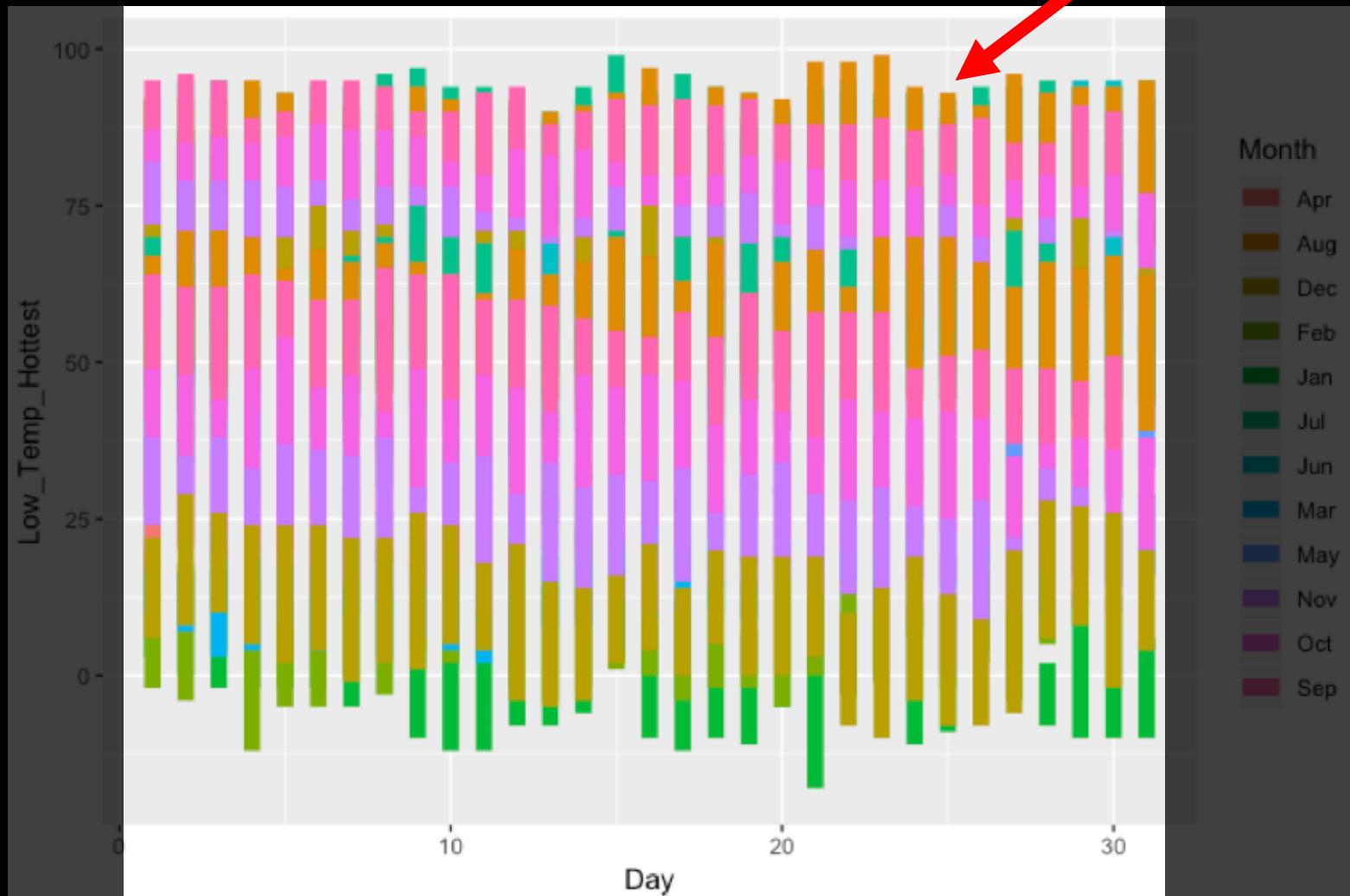
Axis Title  
isn't correct



# Initial Plotting

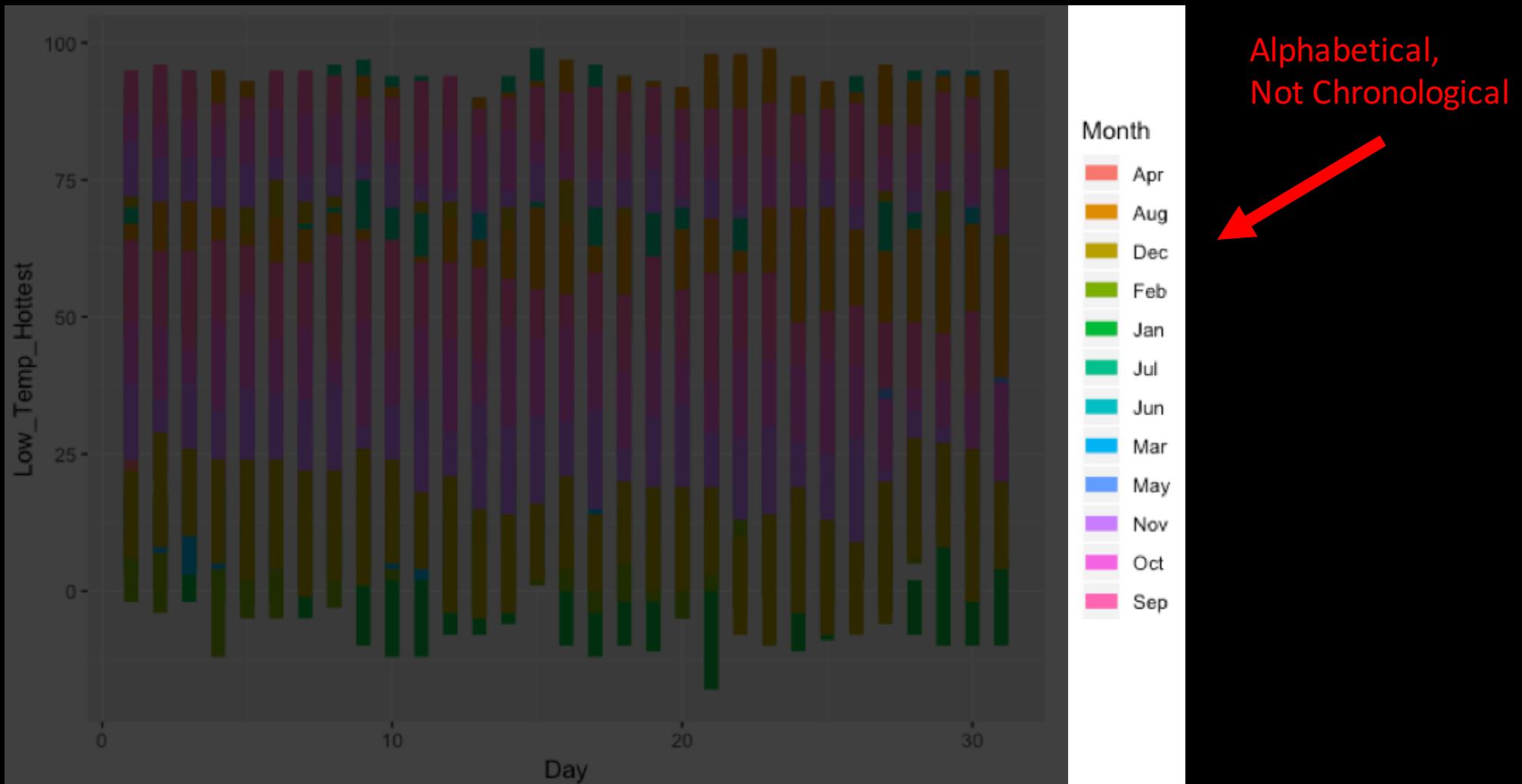
Visualizing our data

Excess overlapping data  
We can't distinguish between highs and lows



# Initial Plotting

## Visualizing our data



# Better Plotting

Making a clearer plot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3) +
  facet_wrap(~Month, nrow = 1, strip.position = "top")
```

# Better Plotting

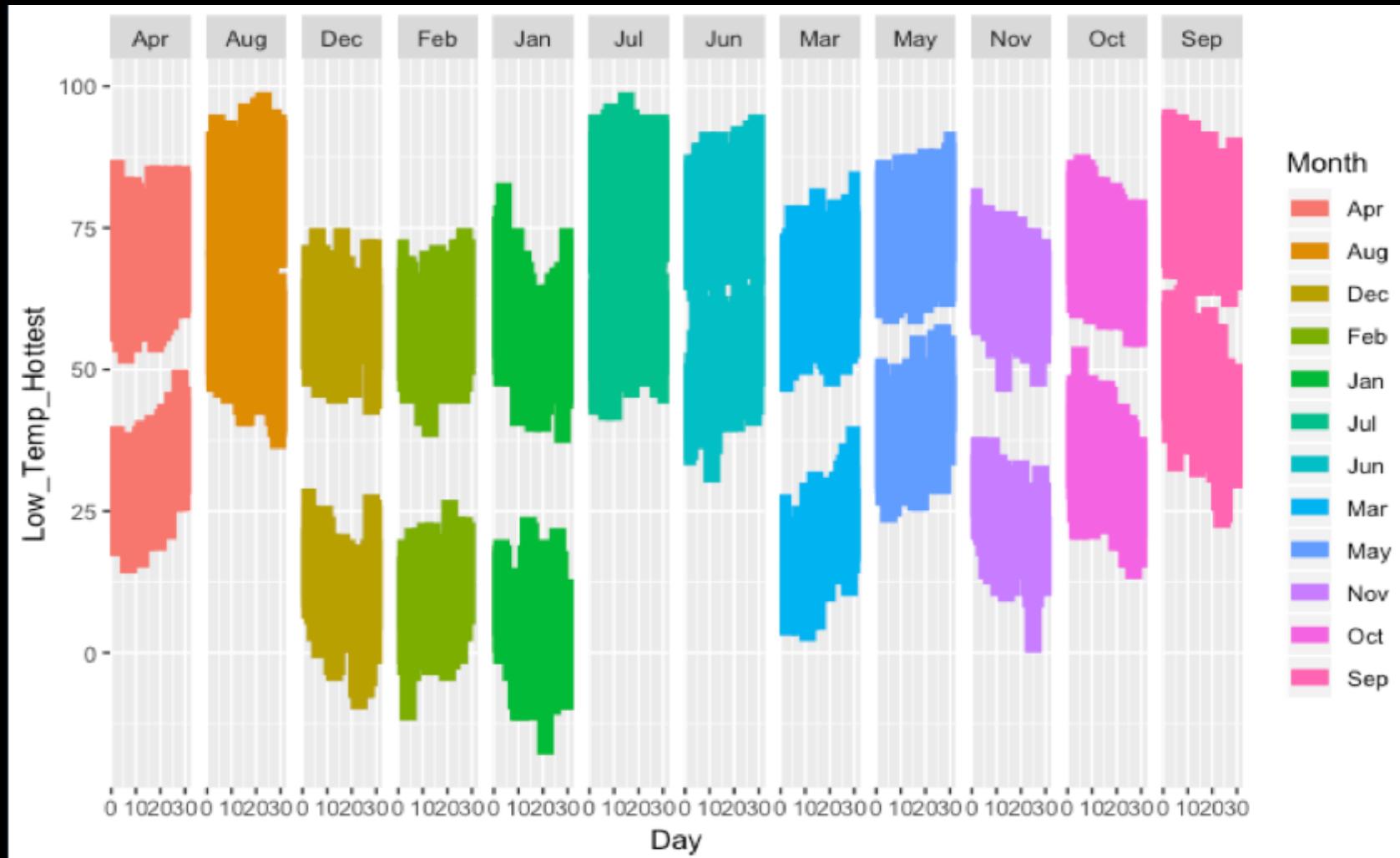
Making a clearer plot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3) +
  facet_wrap(~Month, nrow = 1, strip.position = "top")
```

`facet_wrap()` splits up a single plot into multiple plots, labeled by data type

# Better Plotting

Making a clearer plot



# Better Plotting

Making a clearer plot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```

# Better Plotting

Making a clearer plot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3) +
  facet_wrap(~ factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```

**factor()** converts a vector into a factor

Factors are objects that categorize data into a way that stores levels

# Better Plotting

Making a clearer plot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3) +
  facet_wrap(~ factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```

`unique()` returns all unique names in a factor in the order that they appear

# Better Plotting

Making a clearer plot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = Month), size = 3) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = Month), size = 3) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```

**unique()** returns all unique names in a factor in the order that they appear

This forces **facet\_wrap()** to maintain the original order of the dataset, by altering the levels parameter

# Exercise 3

## Playing with Factors

We create a factor comprised of a series of letters

```
myData <- factor(c("A", "A", "E", "B", "B", "C", "D"))
```

# Exercise 3

## Playing with Factors

We create a factor comprised of a series of letters

```
myData <- factor(c("A", "A", "E", "B", "B", "C", "D"))
```

The **levels()** function returns this:

```
> levels(myData)
[1] "A" "B" "C" "D" "E"
```

What would **unique()** return?

# Exercise 3

## Playing with Factors

We create a factor comprised of a series of letters

```
myData <- factor(c("A", "A", "E", "B", "B", "C", "D"))
```

The **levels()** function returns this:

```
> levels(myData)
[1] "A" "B" "C" "D" "E"
```

What would **unique()** return?

A 

```
> unique(myData)
[1] E D C B A
Levels: A B C D E
```

B 

```
> unique(myData)
[1] A B C D E
Levels: A B C D E
```

C 

```
> unique(myData)
[1] A E B C D
Levels: A B C D E
```

# Exercise 3

## Playing with Factors

We create a factor comprised of a series of letters

```
myData <- factor(c("A", "A", "E", "B", "B", "C", "D"))
```

The **levels()** function returns this:

```
> levels(myData)
[1] "A" "B" "C" "D" "E"
```

What would **unique()** return?

A 

```
> unique(myData)
[1] E D C B A
Levels: A B C D E
```

B 

```
> unique(myData)
[1] A B C D E
Levels: A B C D E
```

C 

```
> unique(myData)
[1] A E B C D
Levels: A B C D E
```

# Exercise 4

Playing with Factors... again

We create a factor comprised of a series of letters

```
myData <- factor(c("A", "A", "E", "B", "B", "C", "D"))
```

The **levels()** function returns this:

```
> levels(myData)
[1] "A" "B" "C" "D" "E"
```

What would the **levels()** function on **factor( myData, levels = unique( myData ) )** return?

# Exercise 4

Playing with Factors... again

We create a factor comprised of a series of letters

```
myData <- factor(c("A", "A", "E", "B", "B", "C", "D"))
```

The **levels()** function returns this:

```
> levels(myData)
[1] "A" "B" "C" "D" "E"
```

What would the **levels()** function on **factor( myData, levels = unique( myData ) )** return?

A 

```
> levels(factor(myData, levels = unique(myData)))
[1] "E" "D" "C" "B" "A"
```

B 

```
> levels(factor(myData, levels = unique(myData)))
[1] "A" "E" "B" "C" "D"
```

C 

```
> levels(factor(myData, levels = unique(myData)))
[1] "A" "B" "C" "D" "E"
```

# Exercise 4

Playing with Factors... again

We create a factor comprised of a series of letters

```
myData <- factor(c("A", "A", "E", "B", "B", "C", "D"))
```

The **levels()** function returns this:

```
> levels(myData)
[1] "A" "B" "C" "D" "E"
```

What would the **levels()** function on **factor( myData, levels = unique( myData ) )** return?

A

```
> levels(factor(myData, levels = unique(myData)))
[1] "E" "D" "C" "B" "A"
```

B

```
> levels(factor(myData, levels = unique(myData)))
[1] "A" "E" "B" "C" "D"
```

C

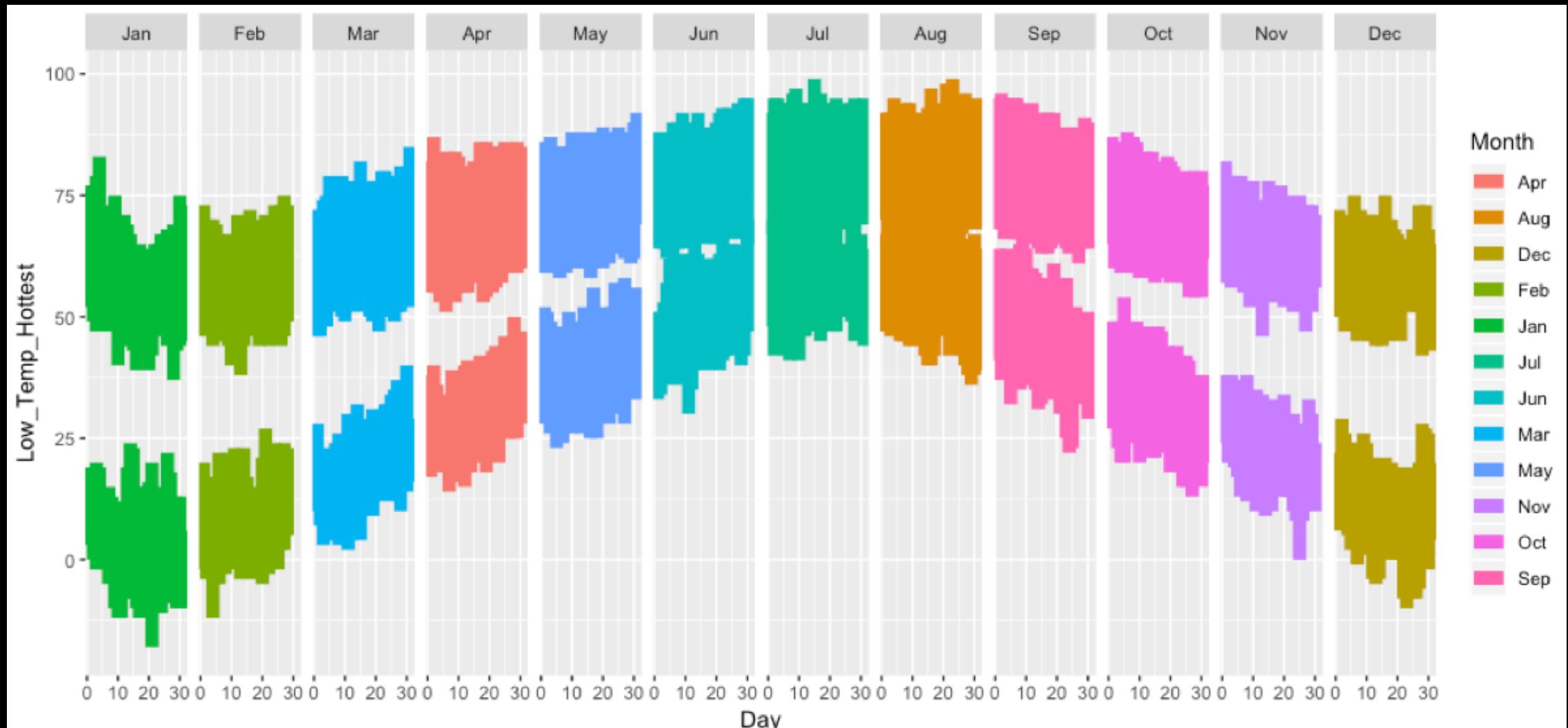
```
> levels(factor(myData, levels = unique(myData)))
[1] "A" "B" "C" "D" "E"
```

We've simply set the  
levels to be the same as  
what **unique(myData)**  
would return

# Better Plotting

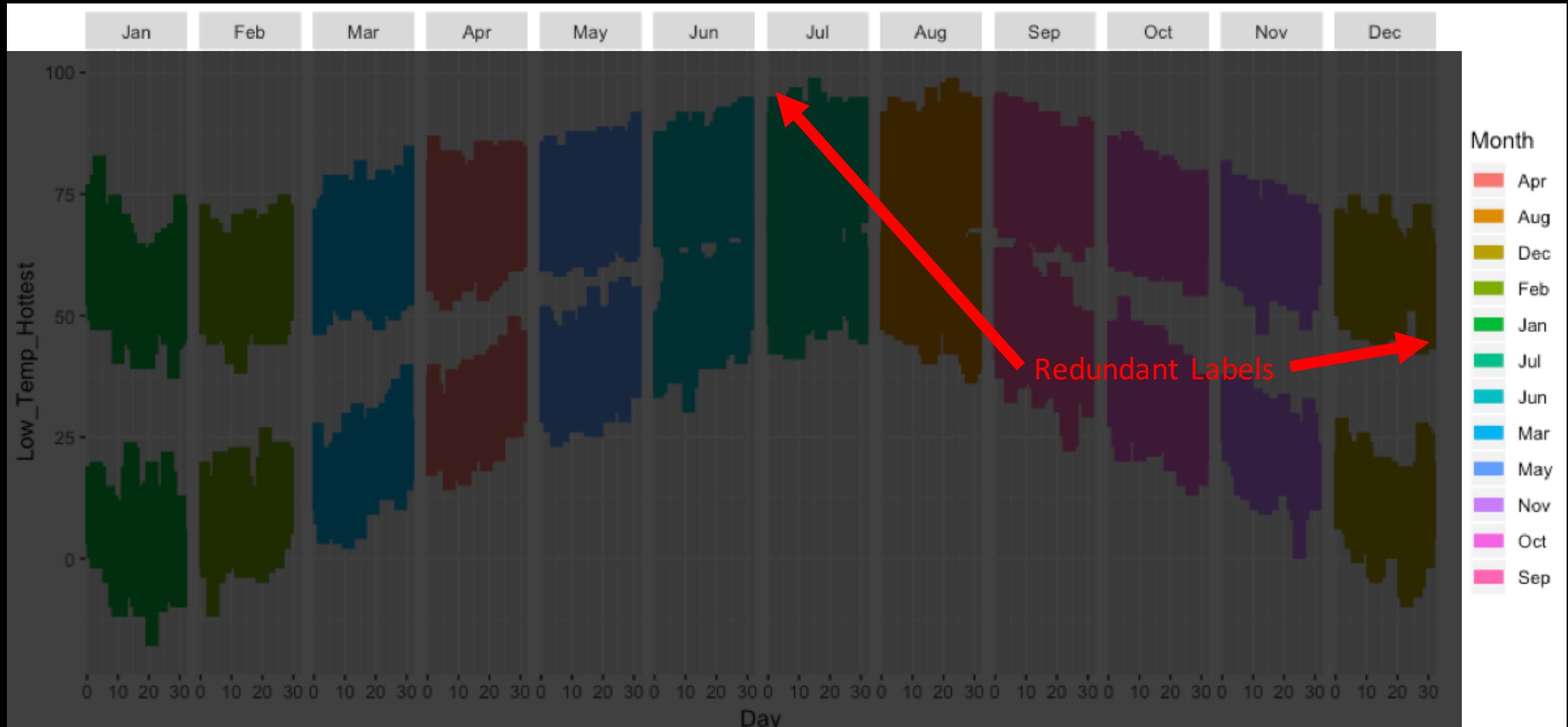
Making a clearer plot

```
facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```



# Better Plotting

Making a clearer plot



# Better Plotting

Let's add more information

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```

# Better Plotting

Let's add more information

Color parameter is no longer inside aes()

Decoupled the color from the data,  
Color is now coupled to the geometry



```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +  
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +  
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```

# Better Plotting

Let's add more information

Color parameter is no longer inside aes()

Decoupled the color from the data,  
Color is now coupled to the geometry

```
weather %>%  
  ggplot(aes(group = Month)) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +  
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +  
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +  
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top")
```

Let's plot the average highs and lows too

# Exercise 5

## Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

# Exercise 5

## Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

We can plot this data with

```
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos, color = category))
```

# Exercise 5

## Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

How many colors will be on this new plot?

A 3

B 5

C 1

We can plot this data with

```
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos, color = category))
```

# Exercise 5

## Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

How many colors will be on this new plot?

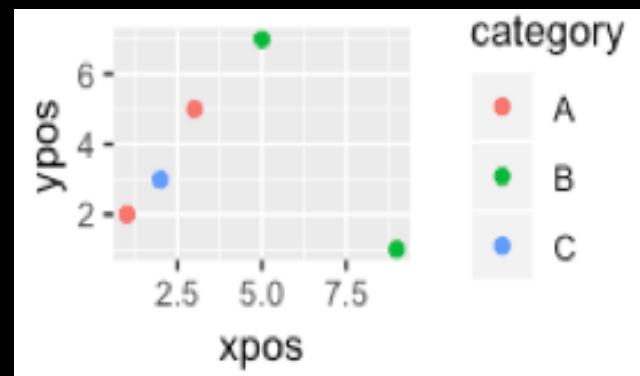
A 3

B 5

C 1

We can plot this data with

```
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos, color = category))
```



# Exercise 6

## More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

We instead plot this data with

```
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos, color = 'category'))
```

# Exercise 6

## More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),  
                 'ypos' = c(2,5,7,1,3),  
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

How many colors will be on this new plot?

A 3

B 5

C 1

We instead plot this data with

```
ggplot(data = myData) +  
  geom_point(aes(x = xpos, y = ypos, color = 'category'))
```

# Exercise 6

## More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

How many colors will be on this new plot?

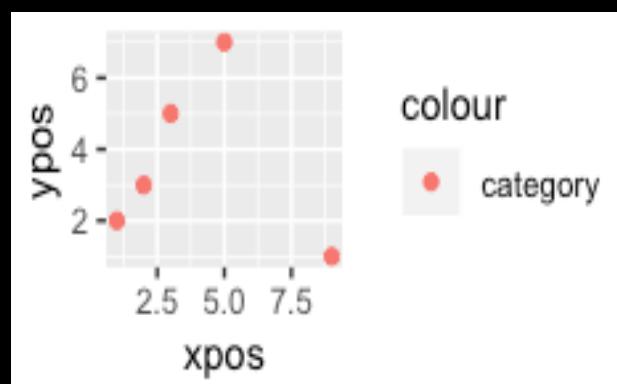
A 3

B 5

C 1

We instead plot this data with

```
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos, color = 'category'))
```



# Exercise 7

Even More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

We instead plot this data with

```
category = 'red'
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos), color = category)
```

# Exercise 7

## Even More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),  
                 'ypos' = c(2,5,7,1,3),  
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

Will there be a legend on this plot?

A Yes

B No

We instead plot this data with

```
category = 'red'  
ggplot(data = myData) +  
  geom_point(aes(x = xpos, y = ypos), color = category)
```

# Exercise 7

## Even More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

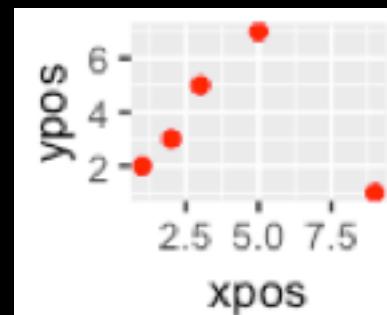
We instead plot this data with

```
category = 'red'
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos), color = category)
```

Will there be a legend on this plot?

A Yes

B No



Legends only appear for mappings inside aes()  
Not parameters outside of aes()

# Exercise 8

## Even More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),
                 'ypos' = c(2,5,7,1,3),
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

We instead plot this data with

```
category = 'red'
ggplot(data = myData) +
  geom_point(aes(x = xpos, y = ypos, size = category), color = category)
```

# Exercise 8

## Even More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),  
                 'ypos' = c(2,5,7,1,3),  
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

Will there be a legend on this plot?

A Yes

B No

We instead plot this data with

```
category = 'red'  
ggplot(data = myData) +  
  geom_point(aes(x = xpos, y = ypos, size = category), color = category)
```

# Exercise 8

## Even More Colors and Aesthetics in ggplot

Given the dataset:

```
myData <- tibble('xpos' = c(1,3,5,9,2),  
                 'ypos' = c(2,5,7,1,3),  
                 'category' = c('A', 'A', 'B', 'B', 'C'))
```

	xpos	ypos	category
1	1	2	A
2	3	5	A
3	5	7	B
4	9	1	B
5	2	3	C

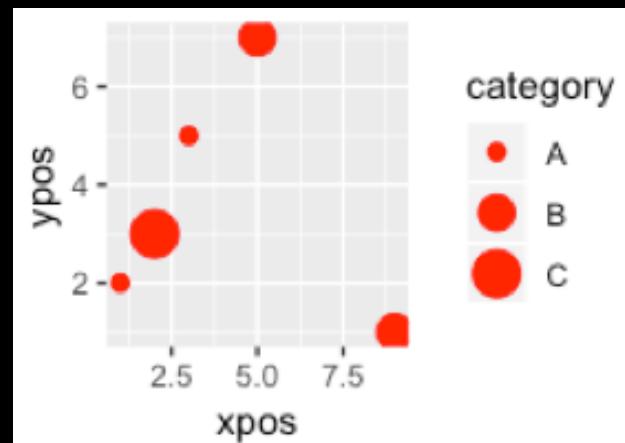
We instead plot this data with

```
category = 'red'  
ggplot(data = myData) +  
  geom_point(aes(x = xpos, y = ypos, size = category), color = category)
```

Will there be a legend on this plot?

A Yes

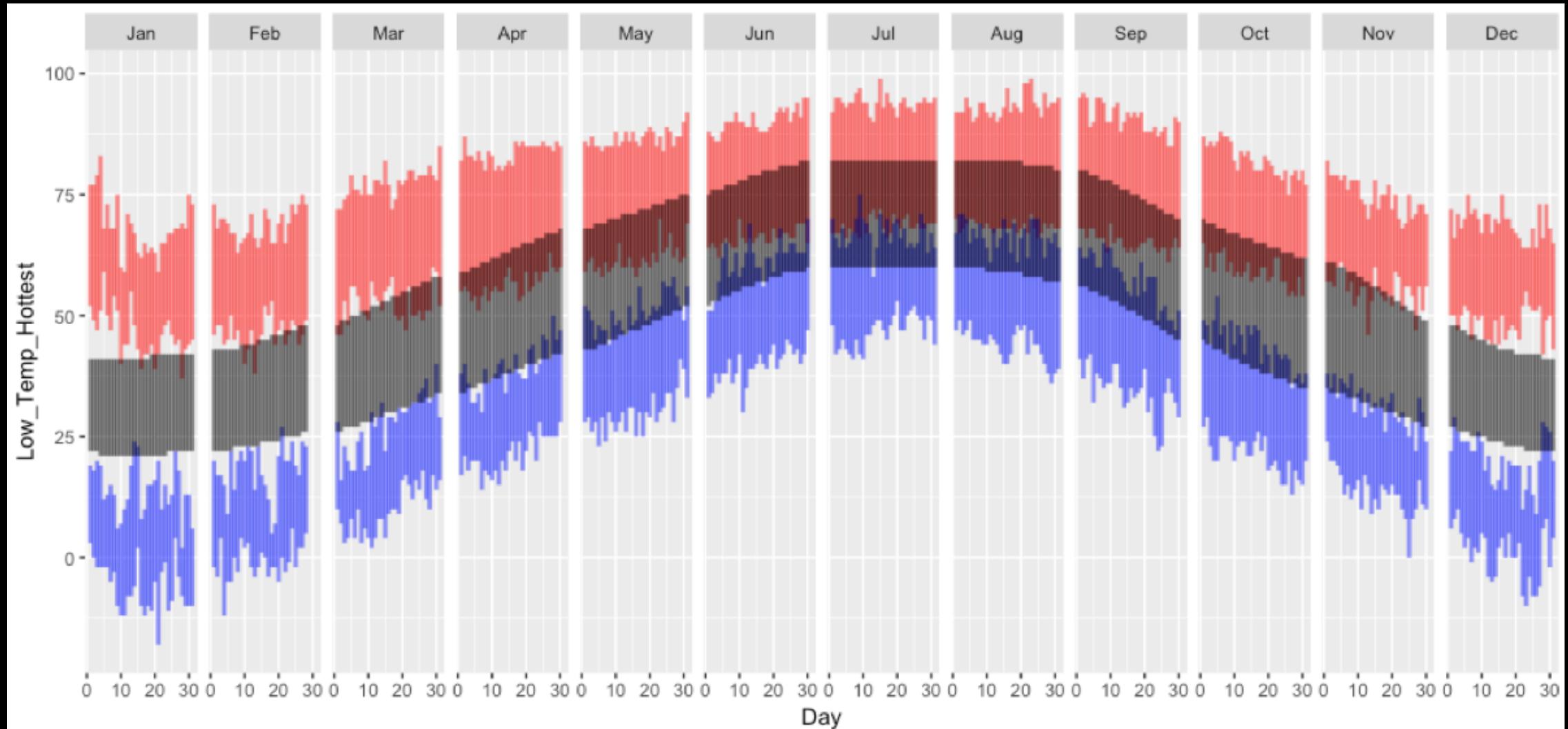
B No



Size is inside the aes() parameter  
Therefor it contributes to the legend

# Better Plotting

Let's add more information



# Cosmetic Adjustment

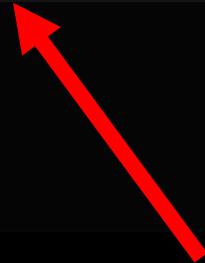
## Finishing Touches

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       y = "Temperature [°F]", x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank())
```

# Cosmetic Adjustment

## Finishing Touches

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32) * (5 / 9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       y = "Temperature [°F]", x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank())
```

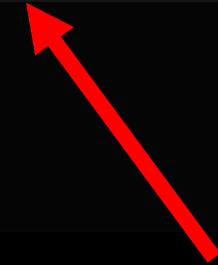


Add a second y-axis scale

# Cosmetic Adjustment

## Finishing Touches

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       y = "Temperature [°F]", x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank())
```



Add a second y-axis scale



\*We still want to keep the Fahrenheit scale so that the plot remains American compatible

# Cosmetic Adjustment

## Finishing Touches

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       y = "Temperature [°F]", x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank())
```

Change labels with labs()

# Cosmetic Adjustment

## Finishing Touches

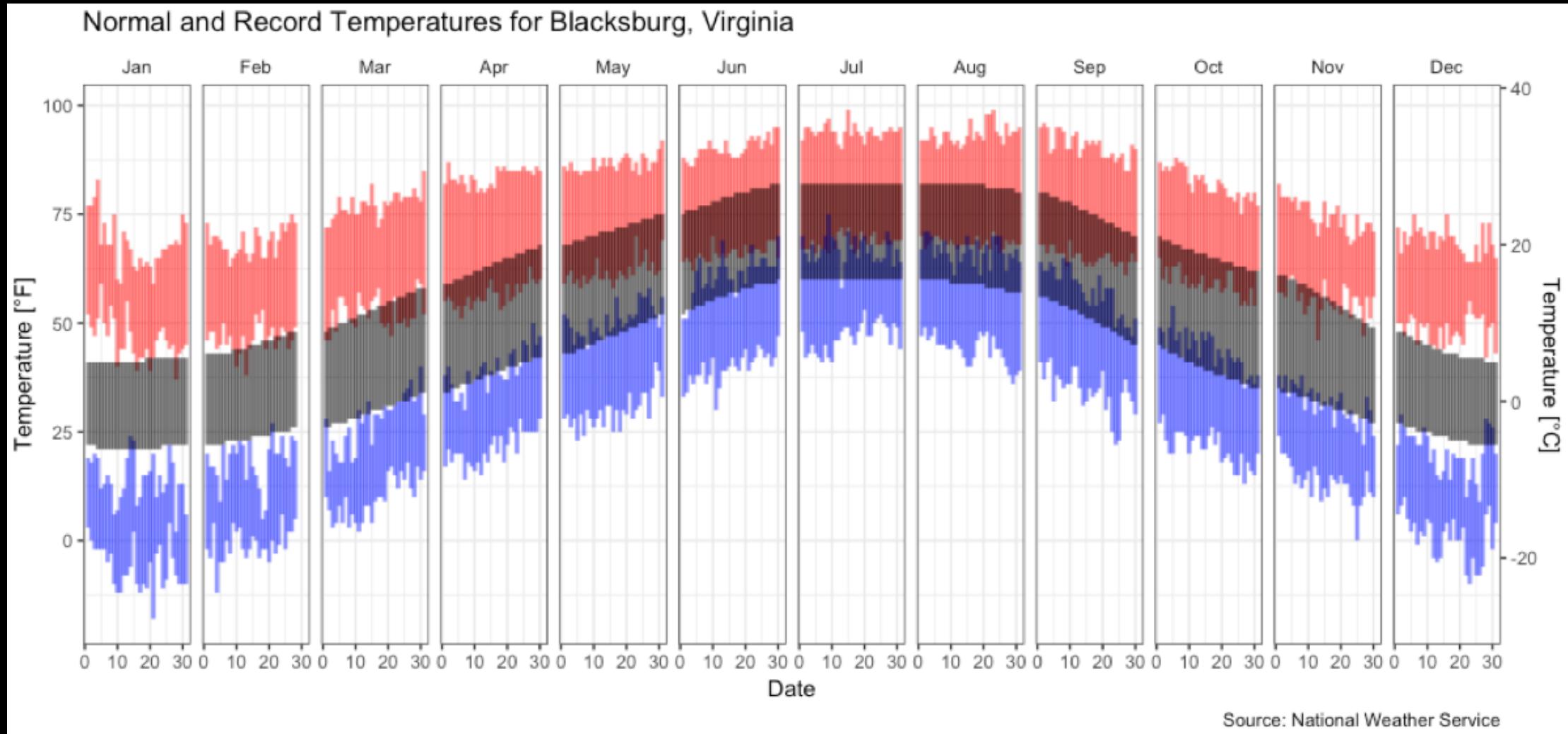
```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest), size = 1, color = 'red', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest), size = 1, color = 'blue', alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High), size = 1, color = 'black', alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       y = "Temperature [°F]", x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank())
```



A cleaner theme

# Cosmetic Adjustment

Finishing Touches



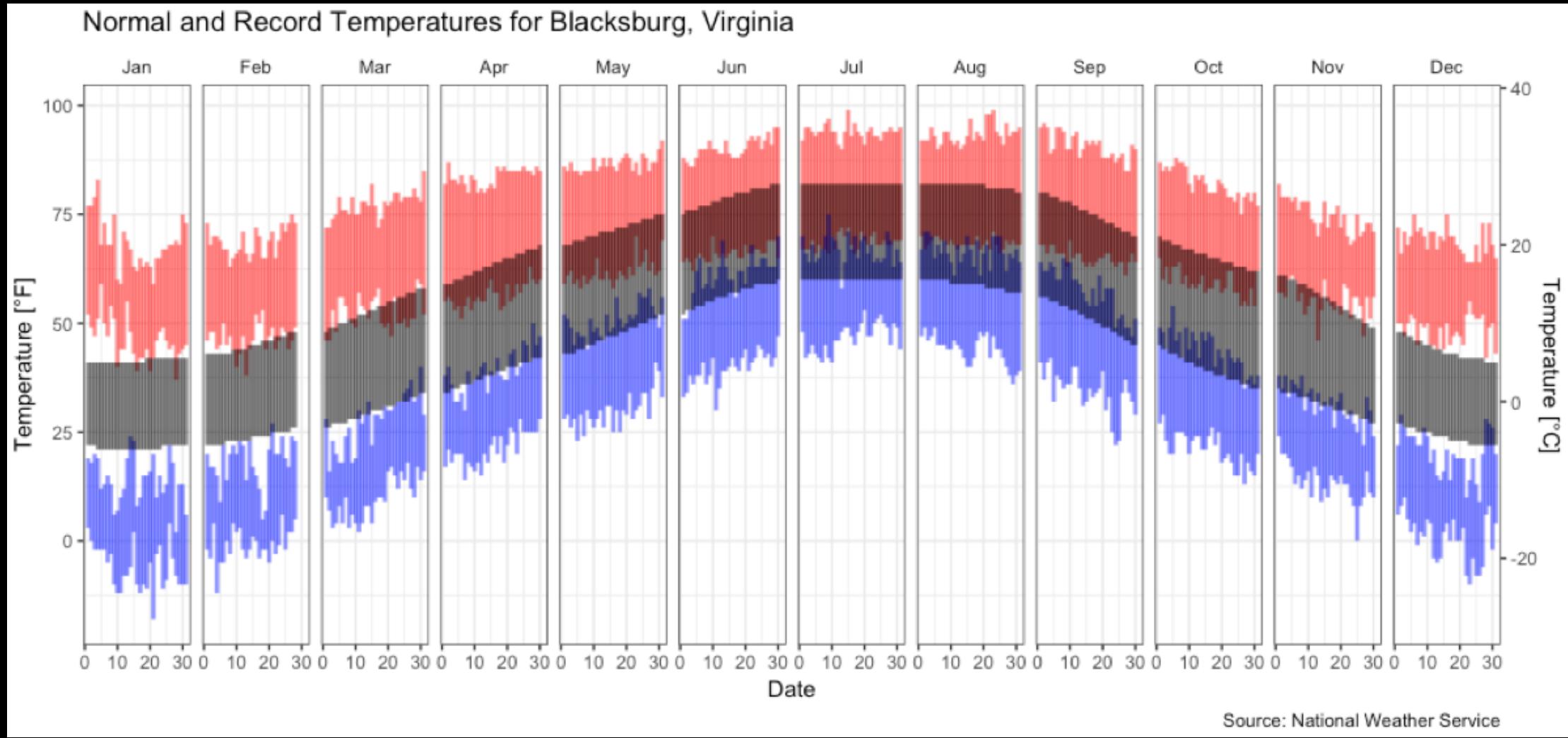
# Even Better Plotting

Adding what we forgot

We forgot some things

# Even Better Plotting

## Adding what we forgot



# Where's the legend?

# Even Better Plotting

Adding what we forgot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = 'Record Hottest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = 'Record Coldest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High, color = 'Normal'), size = 1, alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       color = "Temperature Range",
       y = "Temperature [°F]",
       x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank())
```

# Even Better Plotting

Adding what we forgot

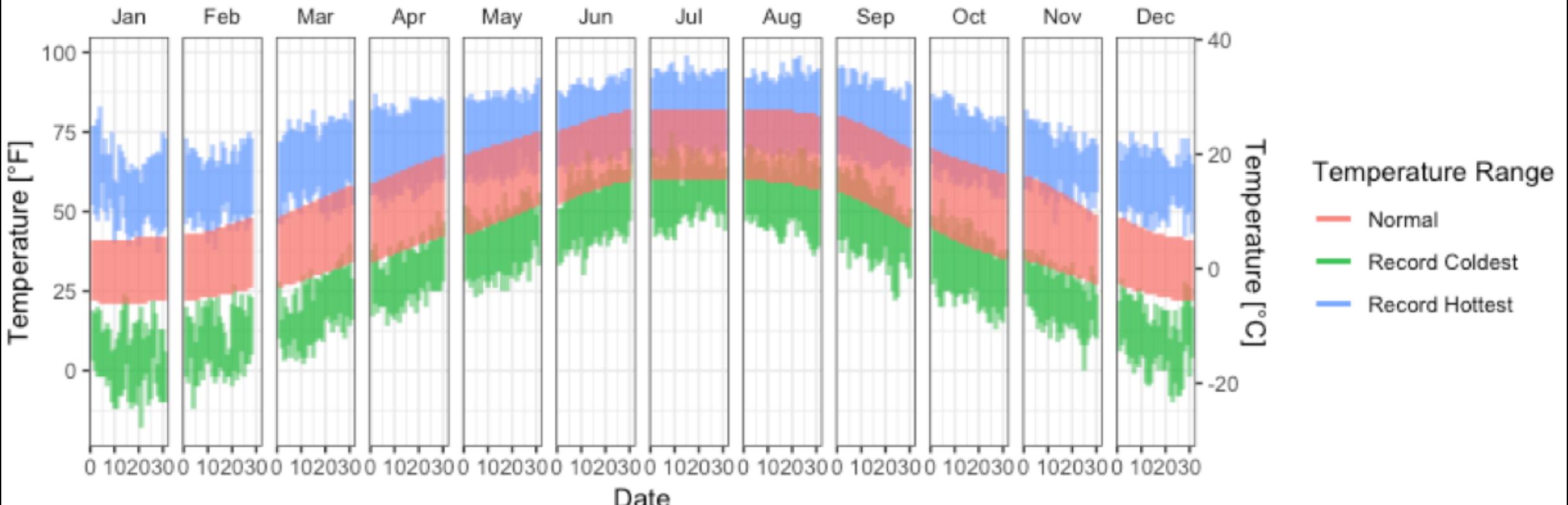
```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = 'Record Hottest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = 'Record Coldest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High, color = 'Normal'), size = 1, alpha = 0.5) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       color = "Temperature Range",
       y = "Temperature [°F]",
       x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank())
```

Recouple the color parameter to aes()

# Even Better Plotting

Adding what we forgot

Normal and Record Temperatures for Blacksburg, Virginia



Source: National Weather Service

# Even Better Plotting

Adding what we forgot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = 'Record Hottest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = 'Record Coldest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High, color = 'Normal'), size = 1, alpha = 0.5) +
  geom_hline(aes(yintercept = 32), alpha = 0.3) +
  scale_color_manual(values = c('black', 'blue', 'red')) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       color = "Temperature Range",
       y = "Temperature [°F]",
       x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank(),
        legend.position = 'bottom',
        legend.text = element_text(size = 8))
```

# Even Better Plotting

Adding what we forgot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = 'Record Hottest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = 'Record Coldest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High, color = 'Normal'), size = 1, alpha = 0.5) +
  geom_hline(aes(yintercept = 32), alpha = 0.3) +
  scale_color_manual(values = c('black', 'blue', 'red')) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32) * (5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       color = "Temperature Range",
       y = "Temperature [°F]",
       x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank(),
        legend.position = 'bottom',
        legend.text = element_text(size = 8))
```

Add a horizontal line for freezing point

# Even Better Plotting

Adding what we forgot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = 'Record Hottest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = 'Record Coldest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High, color = 'Normal'), size = 1, alpha = 0.5) +
  geom_hline(aes(yintercept = 32), alpha = 0.3) +
  scale_color_manual(values = c('black', 'blue','red')) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       color = "Temperature Range",
       y = "Temperature [°F]",
       x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank(),
        legend.position = 'bottom',
        legend.text = element_text(size = 8))
```

Manually adjust the color scale back to our original colors

# Even Better Plotting

Adding what we forgot

```
weather %>%
  ggplot(aes(group = Month)) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Hottest, yend = High_Temp_Hottest, color = 'Record Hottest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Low_Temp_Coldest, yend = High_Temp_Coldest, color = 'Record Coldest'), size = 1, alpha = 0.5) +
  geom_segment(aes(x = Day, xend = Day, y = Normal_Low, yend = Normal_High, color = 'Normal'), size = 1, alpha = 0.5) +
  geom_hline(aes(yintercept = 32), alpha = 0.3) +
  scale_color_manual(values = c('black','blue','red')) +
  facet_wrap(~factor(Month, levels = unique(Month)), nrow = 1, strip.position = "top") +
  scale_y_continuous(sec.axis = sec_axis(~(. - 32)*(5/9), name = "Temperature [°C]")) +
  labs(title = "Normal and Record Temperatures for Blacksburg, Virginia",
       caption = "Source: National Weather Service",
       color = "Temperature Range",
       y = "Temperature [°F]",
       x = "Date") +
  theme_bw() +
  theme(strip.background = element_blank(),
        legend.position = 'bottom',
        legend.text = element_text(size = 8))
```



Reposition the Legend

# Even Better Plotting

## Adding what we forgot

