Math 300 NTI Lesson 3

Linegraphs, Histograms and Facets

Professor Bradley Warner

June, 2022

Contents

Objectives

1. Understand and wrangle data in preparation for plotting.
2. Explain when it is appropriate to use a linegraphs, histogram, and facets.
3. Create and interpret a linegraph using the `ggplot()` function.
4. Create and interpret a histogram using the `ggplot()` function.
5. Use facets to improve the visual presentation of data and then interpret plots that use facets.

Reading

Chapter 2.4 - 2.6

Lesson

Remember that you will be running this more like a lab than a lecture. You want them using `R` and answering questions.

Work through the learning checks LC2.9 - LC2.21. Because you have so many learning checks, you might not want to use a pair and share learning strategy for each one. You can pick and choose.

- You want to emphasize that we are starting to look at different plots. These plots change based on the nature of our data. For the linegraph, the variable we choose to put on the x-axis has a sequential nature, usually time. For the histogram we want to understand the distribution of a single quantitative variable. Facets allow us to bring in another variable, usually categorical.
- We will be using `filter()` again to create a subset of data but we are putting in an **and** condition. This is the setup for LC2.9, so walk them through it.

Setup

```
library(nycflights13)
library(ggplot2)
library(dplyr)
```

We need to create the `early_january_weather` *data object. You may want to walk them through this code. Again, this could be pulled via the* `moderndive` *package, but building it on our own is helpful and informative.*

Use `?weather` *to understand the variables.*

```
glimpse(weather)
```

```
## Rows: 26,115
## Columns: 15
## $ origin     <chr> "EWR", "EWR", "EWR", "EWR", "EWR", "EWR", "EWR", "EWR", "EW~
## $ year       <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013,~
## $ month      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ day        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ hour       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, ~
## $ temp       <dbl> 39.02, 39.02, 39.02, 39.92, 39.02, 37.94, 39.02, 39.92, 39.~
## $ dewp       <dbl> 26.06, 26.96, 28.04, 28.04, 28.04, 28.04, 28.04, 28.04, 28.~
## $ humid      <dbl> 59.37, 61.63, 64.43, 62.21, 64.43, 67.21, 64.43, 62.21, 62.~
## $ wind_dir   <dbl> 270, 250, 240, 250, 260, 240, 240, 250, 260, 260, 260, 330,~
## $ wind_speed <dbl> 10.35702, 8.05546, 11.50780, 12.65858, 12.65858, 11.50780, ~
## $ wind_gust  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 20.~
## $ precip     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ pressure   <dbl> 1012.0, 1012.3, 1012.5, 1012.2, 1011.9, 1012.4, 1012.2, 101~
## $ visib      <dbl> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,~
## $ time_hour  <dttm> 2013-01-01 01:00:00, 2013-01-01 02:00:00, 2013-01-01 03:00~
```

```
# This code finds the flights from EWR between January 1 up
# to and including January 15
early_january_weather <- weather %>%
  filter(origin == "EWR" & month == 1 & day <= 15)
```

LC 2.9 (Objective 1)

(LC2.9) Take a look at both the `weather` and `early_january_weather` data frames by running `View(weather)` and `View(early_january_weather)` in the console. In what respect do these data frames differ?

Solution: The rows of `early_january_weather` are a subset of `weather`.

```
str(weather)
```

```
## tibble [26,115 x 15] (S3: tbl_df/tbl/data.frame)
##  $ origin    : chr [1:26115] "EWR" "EWR" "EWR" "EWR" ...
##  $ year      : int [1:26115] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month     : int [1:26115] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : int [1:26115] 1 1 1 1 1 1 1 1 1 1 ...
##  $ hour      : int [1:26115] 1 2 3 4 5 6 7 8 9 10 ...
##  $ temp      : num [1:26115] 39 39 39 39.9 39 ...
##  $ dewp      : num [1:26115] 26.1 27 28 28 28 ...
##  $ humid     : num [1:26115] 59.4 61.6 64.4 62.2 64.4 ...
##  $ wind_dir  : num [1:26115] 270 250 240 250 260 240 240 250 260 260 ...
##  $ wind_speed: num [1:26115] 10.36 8.06 11.51 12.66 12.66 ...
##  $ wind_gust : num [1:26115] NA NA NA NA NA NA NA NA NA NA ...
##  $ precip    : num [1:26115] 0 0 0 0 0 0 0 0 0 0 ...
##  $ pressure  : num [1:26115] 1012 1012 1012 1012 1012 ...
##  $ visib     : num [1:26115] 10 10 10 10 10 10 10 10 10 10 ...
##  $ time_hour : POSIXct[1:26115], format: "2013-01-01 01:00:00" "2013-01-01 02:00:00" ...
```

```
str(early_january_weather)
```

```
## tibble [358 x 15] (S3: tbl_df/tbl/data.frame)
##  $ origin    : chr [1:358] "EWR" "EWR" "EWR" "EWR" ...
##  $ year      : int [1:358] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month     : int [1:358] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : int [1:358] 1 1 1 1 1 1 1 1 1 1 ...
##  $ hour      : int [1:358] 1 2 3 4 5 6 7 8 9 10 ...
##  $ temp      : num [1:358] 39 39 39 39.9 39 ...
##  $ dewp      : num [1:358] 26.1 27 28 28 28 ...
##  $ humid     : num [1:358] 59.4 61.6 64.4 62.2 64.4 ...
##  $ wind_dir  : num [1:358] 270 250 240 250 260 240 240 250 260 260 ...
##  $ wind_speed: num [1:358] 10.36 8.06 11.51 12.66 12.66 ...
##  $ wind_gust : num [1:358] NA NA NA NA NA NA NA NA NA NA ...
##  $ precip    : num [1:358] 0 0 0 0 0 0 0 0 0 0 ...
##  $ pressure  : num [1:358] 1012 1012 1012 1012 1012 ...
##  $ visib     : num [1:358] 10 10 10 10 10 10 10 10 10 10 ...
##  $ time_hour : POSIXct[1:358], format: "2013-01-01 01:00:00" "2013-01-01 02:00:00" ...
```

LC 2.10 (Objective 1)

(LC2.10) `View()` the `flights` data frame again. Why does the `time_hour` variable uniquely identify the hour of the measurement whereas the `hour` variable does not?

Solution: Because to uniquely identify an hour, we need the `year`/`month`/`day`/`hour` sequence, whereas there are only 24 possible `hour`s.

```
str(flights)
```

```
## tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
##  $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
##  $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
##  $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

LC 2.11 (Objective 2)

(LC2.11) Why should linegraphs be avoided when there is not a clear ordering of the horizontal axis?

Solution: Because lines suggest connectedness and ordering.

LC 2.12 (Objective 2)

(LC2.12) Why are linegraphs frequently used when time is the explanatory variable?

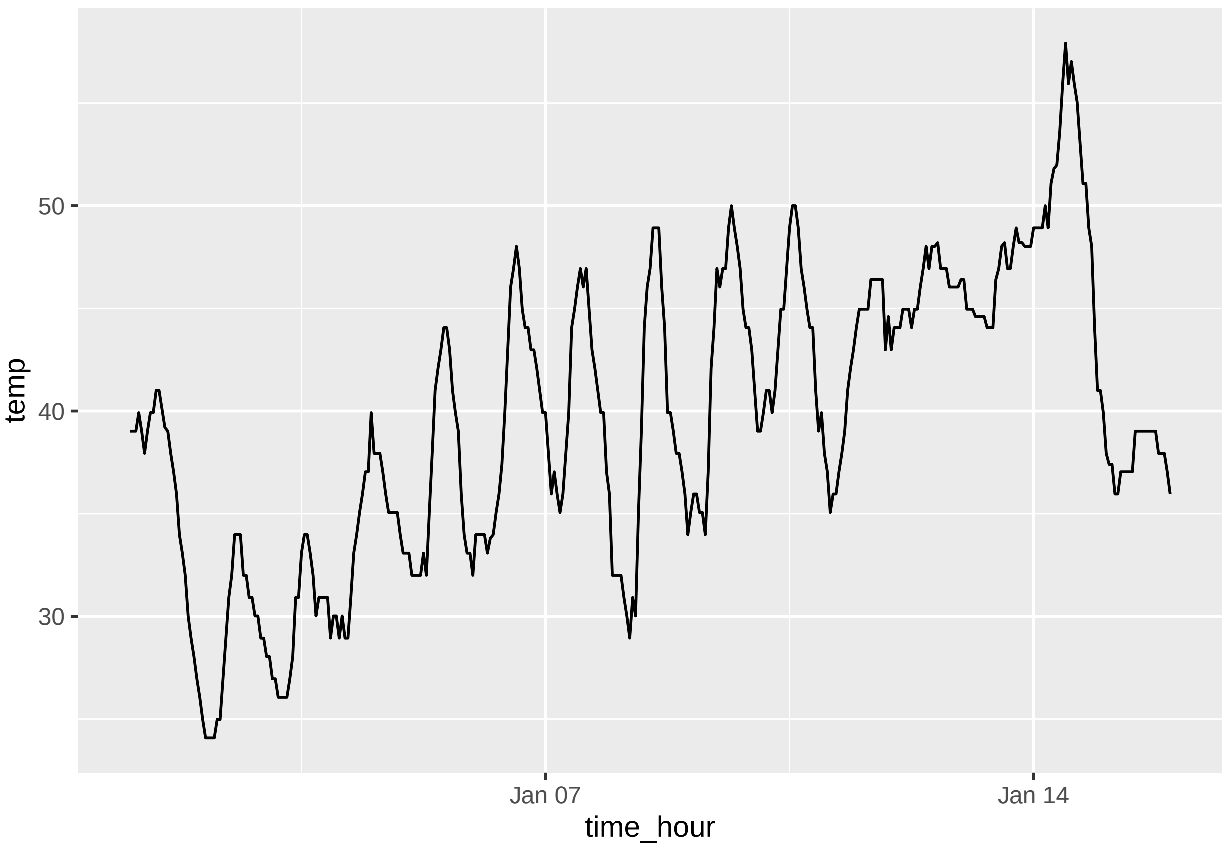
Solution: Because time is sequential: subsequent observations are closely related to each other. Also, we often see only one observation recorded per time period.

LC 2.13 (Objective 3)

(LC2.13) Plot a time series of a variable other than `temp` for Newark Airport in the first 15 days of January 2013.

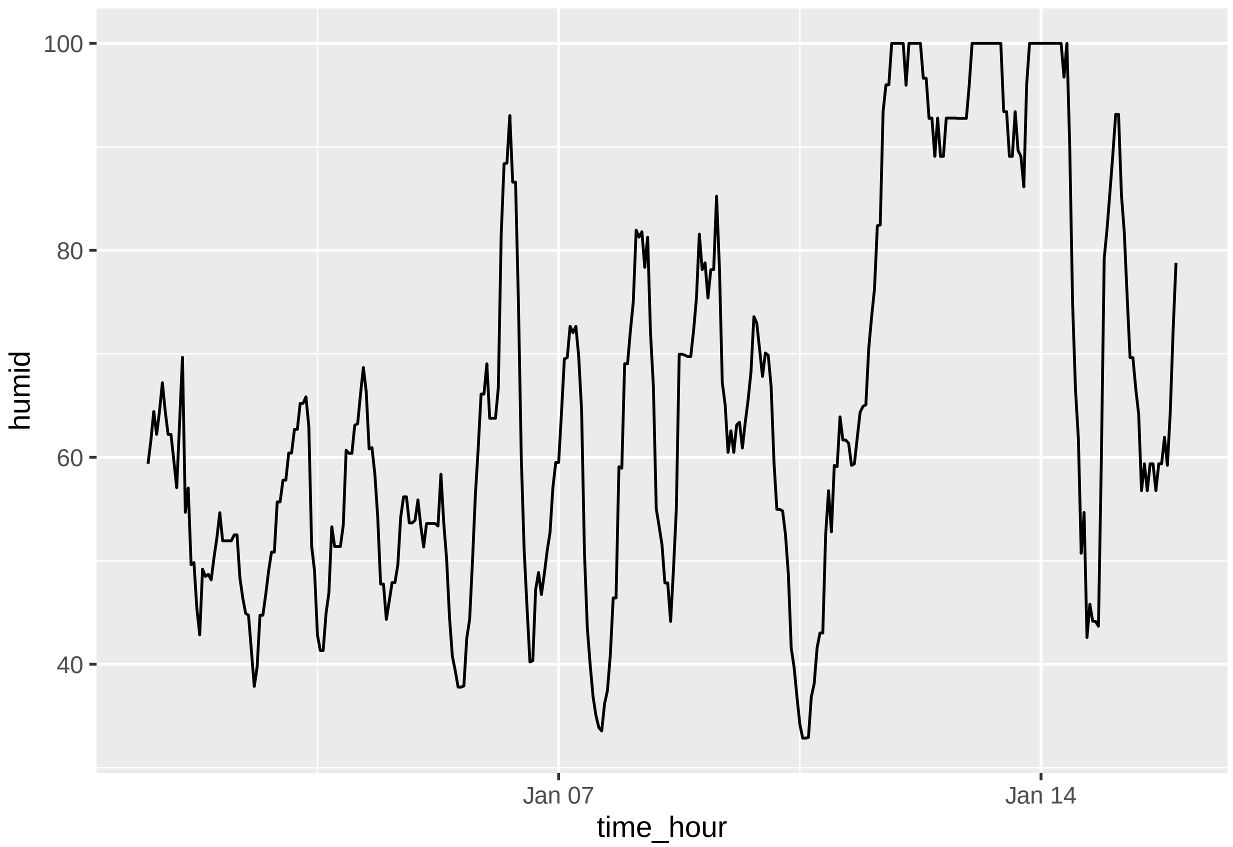
First let's create the plot in the book. Walk through the code.

```
ggplot(data = early_january_weather,
       mapping = aes(x = time_hour, y = temp)) +
  geom_line()
```



Solution: Humidity is a good one to look at, since this very closely related to the cycles of a day.

```
ggplot(data = early_january_weather, mapping = aes(x = time_hour, y = humid)) +
  geom_line()
```



Notice the cyclic nature of humid over the course of a day.

LC 2.14 (Objective 4)

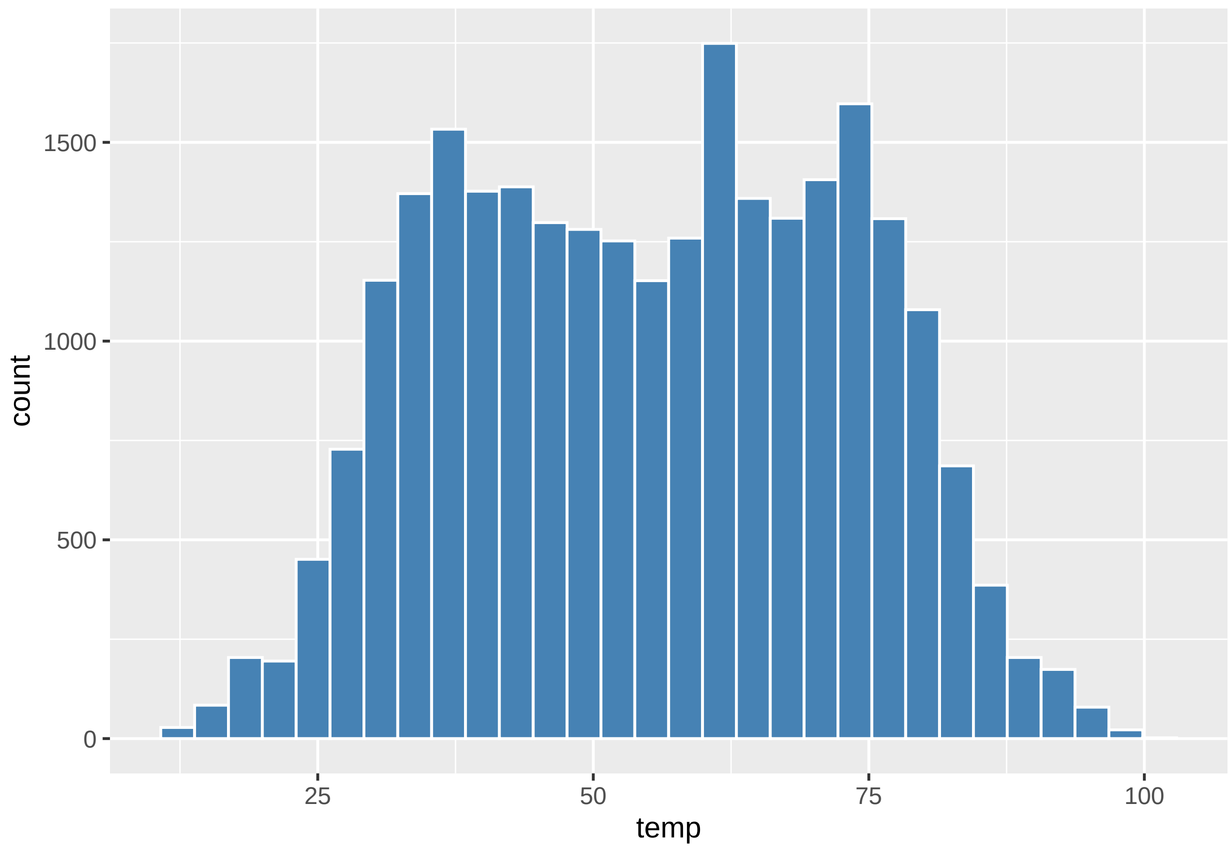
To understand a histogram, discuss the histogram from the book.

- Explain the warning and notification.
- Explain the options in the `geom_histogram()` function.
- Explain that this might be multimodal data.

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(color = "white", fill = "steelblue")
```

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

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



- A list of the available colors

```
head(colors())
```

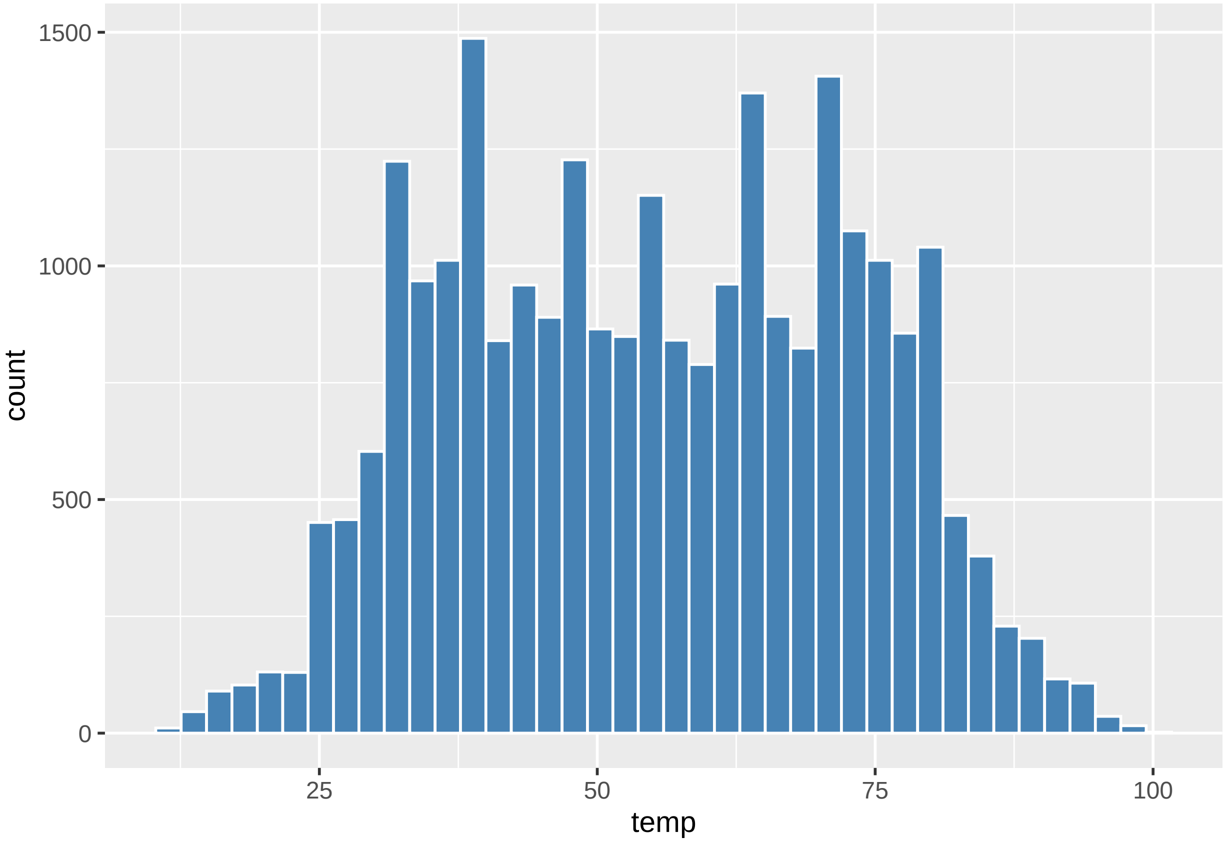
```
## [1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
## [5] "antiquewhite2" "antiquewhite3"
```

(LC2.14) What does changing the number of bins from 30 to 40 tell us about the distribution of temperatures?

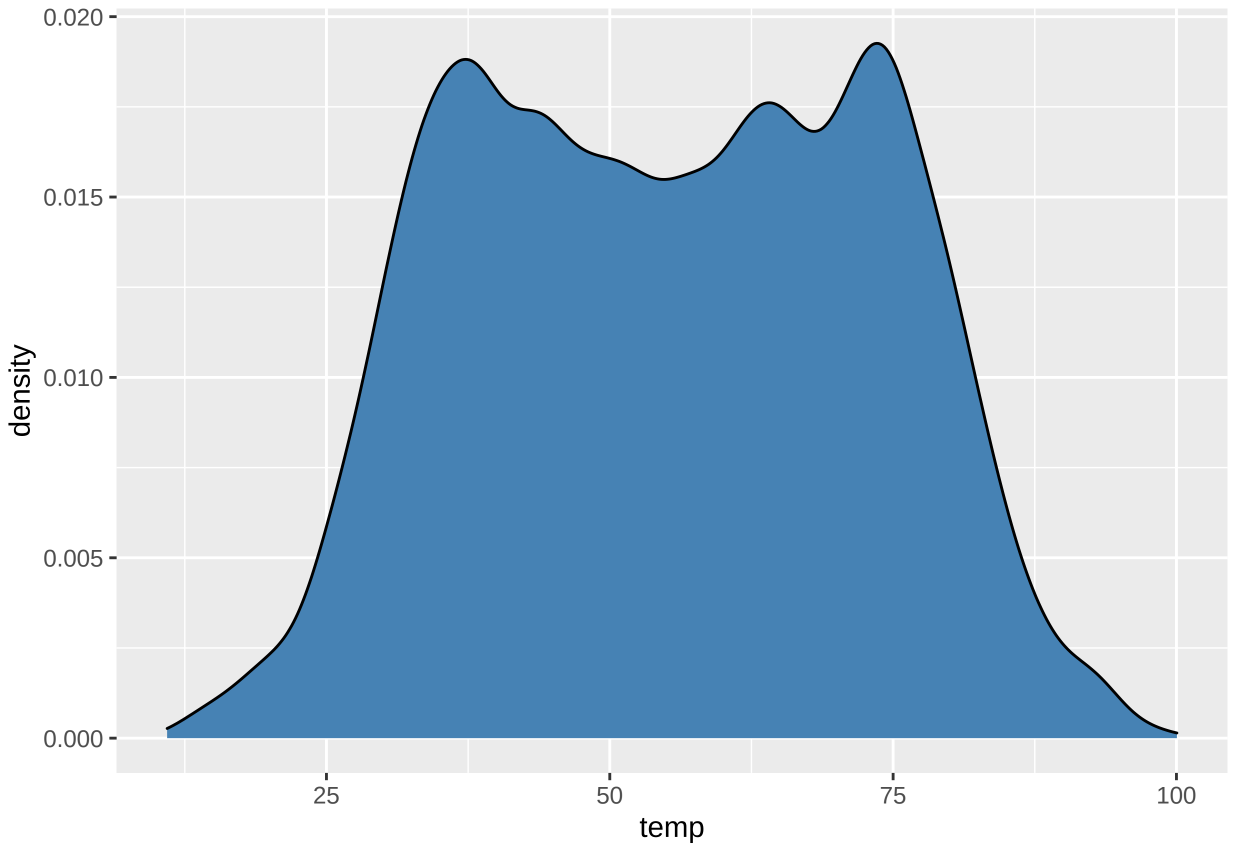
Solution: The distribution doesn't change much. But by refining the bin width, we see that the temperature data has a high degree of accuracy. What do I mean by accuracy? Looking at the `temp` variable by `View(weather)`, we see that the precision of each temperature recording is 2 decimal places. Increasing the number of bins gives the impression that are more modes in the data because the number of data points in each bin get smaller. Interpretation of histograms depends on the number of bins and the location of the bins. This is why some people prefer density plots. We add one here for reference using `geom_density()` rather than `geom_histogram()`.

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(bins=40,color = "white", fill = "steelblue")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_density(fill = "steelblue")
```



LC 2.15 (Objective 4)

(LC2.15) Would you classify the distribution of temperatures as symmetric or skewed?

Solution: It is rather symmetric, i.e. there are no **long tails** on only one side of the distribution

LC 2.16 (Objective 4)

(LC2.16) What would you guess is the "center" value in this distribution? Why did you make that choice?

Solution: The center is around 55.2603921°F. By running the `summary()` command, we see that the mean and median are very similar. In fact, when the distribution is symmetric the mean equals the median. It is not true that when the mean equals the median, the distribution is symmetric, so be careful. A simple example is the discrete data below which is skewed but has same mean and median.

```
temp <- c(3,3,5,5,9)
```

LC 2.17 (Objective 4)

(LC2.17) Is this data spread out greatly from the center or is it close? Why?

Solution: This can only be answered relatively speaking! It depends on the context of the problem. Let's pick things to be relative to Seattle, WA temperatures:

Keep in mind that this is a plot of two linegraphs, not histograms. We are merely using these linegraphs to get an idea of the *spread* of temperature in Seattle, which we can compare to our idea of spread of temperature

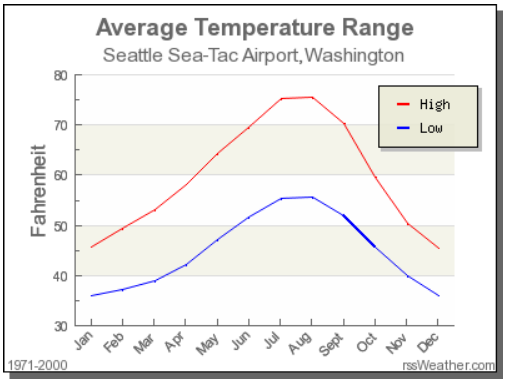


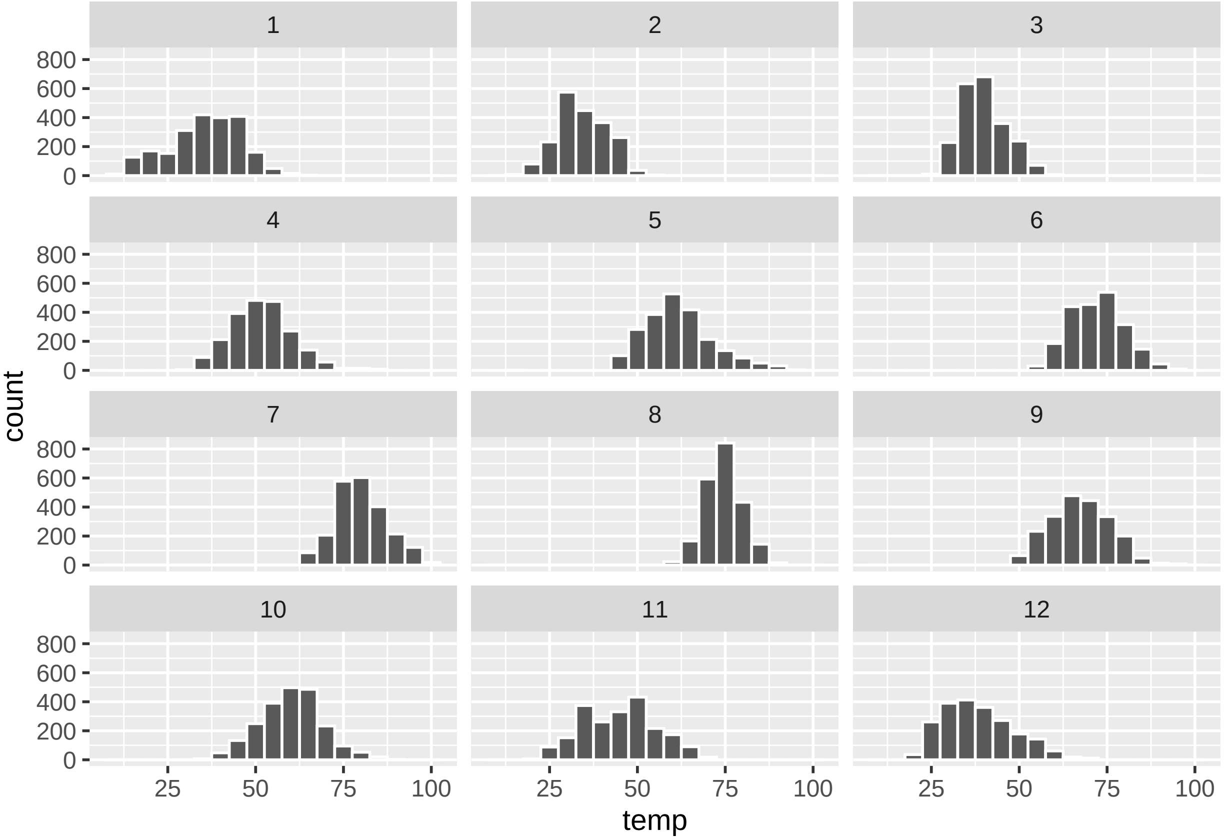
Figure 1: Annual temperatures at SEATAC Airport.

in New York. While, it appears that Seattle weather has a similar center of 55°F, its temperatures are almost entirely between 35°F and 75°F for a range of about 40°F. Seattle temperatures are much less spread out than New York i.e. much more consistent over the year. New York on the other hand has much colder days in the winter and much hotter days in the summer. Expressed differently, the middle 50% of values, as delineated by the **interquartile range** is 30°F.

LC 2.18 (Objective 5)

```
ggplot(data = weather, mapping = aes(x = temp)) +
  geom_histogram(binwidth = 5, color = "white") +
  facet_wrap(~ month, nrow = 4)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



(LC2.18) What other things do you notice about the faceted plot above? How does a faceted plot help us see relationships between two variables?

Solution:

- Certain months have much more consistent weather (August in particular), while others have crazy variability like January and October, representing changes in the seasons.
- Because we see `temp` recordings split by `month`, we are considering the relationship between these two variables. For example, for summer months, temperatures tend to be higher.

LC 2.19 (Objective 5)

(LC2.19) What do the numbers 1-12 correspond to in the plot above? What about 25, 50, 75, 100?

Solution:

- They correspond to the month of the flight. While month is technically a number between 1-12, we're viewing it as a categorical variable here. Specifically, this is an **ordinal categorical** variable since there is an ordering to the categories.
- 25, 50, 75, 100 are temperatures

LC 2.20 (Objective 2, 5)

(LC2.20) For which types of datasets would these types of faceted plots not work well in comparing relationships between variables? Give an example describing the nature of these variables and other important characteristics.

Solution:

- It would not work if we had a very large number of facets. For example, if we faceted by individual days rather than months, as we would have 365 facets to look at. When considering all days in 2013, it could be argued that we shouldn't care about day-to-day fluctuation in weather so much, but rather month-to-month fluctuations, allowing us to focus on seasonal trends.

LC 2.21 (Objective 5)

(LC2.21) Does the `temp` variable in the `weather` dataset have a lot of variability? Why do you say that?

Solution: Again, like in LC (LC2.17), this is a relative question. We would say yes, because in New York City, you have 4 clear seasons with different weather. Whereas in Seattle WA and Portland OR, you have two seasons: summer and rain!

Documenting software

- File creation date: 2022-06-14
- R version 4.1.3 (2022-03-10)
- `ggplot2` package version: 3.3.6
- `dplyr` package version: 1.0.9
- `nycflights13` package version: 1.0.2