Packet 2

Todd CadwalladerOlsker

2022-02-09

Reading

Lightly read OpenIntro Statistics, p. 41 – 55.

Descriptive Statistics

In order to describe a data set, we need to *summarize* it. The phrase "Exploratory Data Analysis" is used in *Introduction to Modern Statistics*. We can summarize data by visualizing it, describing it numerically, or (even better) doing some of each.

I'll try to keep these packets brief, they won't contain everything there is to say about the material. You should reference the textbooks and keep your own notes as well.

Medians, Quartiles, and the Five-Number Summary

When we look at **numeric** variables, we can look at *median*-based statistics or *mean*-based statistics. The *median* is the value of the middle data point (or when there are an even number of data points, the value halfway between the two middle data points). In R, we can find the median of a data set (in the example, the `poverty` variable in the `county` dataset) with:

```
median(county$poverty, na.rm = TRUE)
```

In this example, we have the median of the *population*, since we have information about every county. (well, except for the NAs, but it's not like we didn't ask for that information.) This is technically a *parameter*, not a *statistic*. If we only take a small sample of the population, then we are calculating the sample median statistic.

Here, the `na.rm = TRUE` tells R to ignore values that have missing data. (Try the command without the `na.rm = TRUE` and see what happens!) You can also just use `median(county$pop2017,TRUE)` for the same result.

Now, a *measure of central tendency* like the median is not worth very much without a *measure of spread* to go along with it. For the median, one way to get a handle on the spread is to also report the maximum, minimum, and 1st and 3rd quartile values. Together, these are called a *five-number summary*.

Recall that the 1st quartile is the "median" of the minimum value and the median, while the 3rd quartile is the "median" of the maximum value and the median... sort of. In fact, it's actually silghtly more complicated than that. For example, consider the simple example of the data

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$$

The median is 6, but what should the first quartile be? Should we include 6, and get a median of 3.5, or exclude it, at get a median of 3? Try `fivenum(c(1:11))` or `summary(c(1:11))` and see what R does!

Now try both of the following:

```
fivenum(c(1:10))
summary(c(1:10))
```

Why do these give different results? If you want to get *very* deep into the weeds, check out `help("quantile")` and read about the nine different "types" it uses to calculate quantiles. You can choose a type in `summary` with `summary(data,quantile.type=X)` where X is the type described in the help file. The R documentation is not very easy to read, but the Mathworld page for Quantile is a bit better, see `https://mathworld.wolfram.com/Quantile.html`.

For example, try `summary(c(1:10),quantile.type=2)`

To be a bit more technical, the measures of spread are actually the *range*, which is the difference of the maximum and minimum, and the *interquartile range*, or IQR, which is the difference of the 3rd and 1st quartiles. The five number summary contains all this information and more.

```
fivenum(county$poverty)
# The fivenum() function defaults to na.rm = TRUE, so
# we don't need to add it in.

summary(county$poverty)
# This is a little friendlier to the eyes, and also
# includes the mean.
```

We can visualize the five number summary as a plot:

```
county %>%
    na.omit(poverty) %>%
    ggplot(aes(x = poverty)) + stat_boxplot(geom = "errorbar") +
    geom_boxplot()
```

Traditionally, the boxplot includes vertical lines for each of the five numbers in the summary.

A couple of things are going on here: First, notice that we needed to add "error bars" to visualize the whiskers, ggplot does not include them as standard.

Second, notice that we have some *outliers*. By default, the whiskers of the boxplot are at the maximum and minimum, *unless* there are any values more than $1.5 \times$ IQR above Q3, or below Q1. Values outside this region are outliers. This is only one possible guideline for identifying outliers; it may or may not be a good guideline, depending on the data. We can override this if we need to by adding `coef = 10` (or some other large number) to the `stat_boxplot` and `geom_boxplot` command. This overrides the multiple of the IQR needed to call something an outlier.

Third, notice that we have values on a vertical axis. These are meaningless, we can suppress those with the following code:

```
county %>%
    na.omit(poverty) %>%
    ggplot(aes(x = poverty)) + stat_boxplot(geom = "errorbar",
    coef = 10) + geom_boxplot(coef = 10) + labs(title = "Poverty Rates of U.S. Counties",
    y = NULL, x = "Poverty rate", subtitle = "Percentage of residents living in poverty",
    caption = "Source: county dataset") + scale_y_continuous(breaks = NULL,
    labels = NULL)
```

Also notice I added `coef = 10` this time. Use `help(geom_boxplot)` for more.

Boxplots are often used to compare two groups. Try adding a y-axis for a category, like `metro`:

```
county %>%
    na.omit(poverty) %>%
    ggplot(aes(x = poverty, y = metro)) + stat_boxplot(geom = "errorbar") +
    geom_boxplot() + labs(title = "Poverty Rates of U.S. Counties",
    y = "Metropolitan Area", x = "Poverty rate", subtitle = "Percentage of residents living in poverty"
    caption = "Source: county dataset") + scale_x_log10()
```

Before continuing, try to create a boxplot for the pop2017 variable of the `county` dataset. You should see that the boxplot is not very useful as-is. Try

adding `scale_x_log10()` to your plot, and see how it changes! This is a commonly used *transformation* of the data.

Mean, Variance, and Standard Deviation

In many cases, the *mean* is a more useful measurement than the median. In many other cases the median is more useful – it depends on the situation. The mean is the sum of the values of our data points, divided by the number of data points – exactly how you learned to calculate the average since forever. The calculation is the same whether we want to calculate a population mean or sample mean.

The mean is more useful in the sense that we can do a lot more with it, but it is not as *robust* as the median. It's much more sensitive to outliers and skewed data.

Take a look at the `census` data set:

```
# It's a good habit to look at the help file first!
help(census)  # You can also just use ?census as a shortcut
glimpse(census)
mean(census$age)
summary(census$age)
```

The natural measure of spread to pair with the mean is the *variance*, or the *standard deviation*. The definition of variance starts with a natural idea: let's measure the distance between the value of each data point and the mean (this is the *deviation from the mean* or just *deviation* of each data point). Then, we could find the average deviation.

Let's try this out: Let's find the "mean deviation" for the `age variable in thecensus‘` dataset.

In R, we can try the following:

```
mean(census$age)
census_dev <- census %>%
    mutate(Deviation = age - mean(age))
mean(census_dev$Deviation) %>%
    round(14)
```

Note the use of `round` in the last command; here we are rounding to 14 digits. If you just use `mean()` in the last step, you will get a very small number in scientific notation. The actual value of the mean is in fact 0, but you are seeing a very small number due to a *floating point error* in R. These are very common in any programming language, and dealing with them is a necessary evil.

You should find that the mean deviation is 0! The positive deviations are cancelled out by the negative deviations. This will always happen in any data set, since the average value of the data points is 0 units away from the average. Therefore, this is not a very useful measurement of spread.

So what can we do? We'll *square* the deviations, so that they are all positive! Then, we can take the mean squared deviation, AKA the *variance*. Finally, the square root of the variance gives us the *standard deviation*.

You might be wondering why we square the deviations, then take the square root of the mean of those deviations, rather than taking the absolute values. We could do that – in fact, there is a measure called the *mean absolute deviation (MAD)*. It's a perfectly fine statistic, but is not as commonly used, as it doesn't have some of the nice mathematical properties that the standard deviation has.

```
census_dev <- census_dev %>%
    mutate(Sq_Deviation = Deviation^2)
mean(census_dev$Sq_Deviation)
```

```
mean(census_dev$Sq_Deviation) %>%
    sqrt()
```

Of course, there is an R command that gets us the standard deviation directly. Try

```
var(census$age)
sd(census$age)
```

Notice that we get slightly different answers than we expected! More on that in a moment.

To summarize, given a population of n with a numeric value x_i for each member of the population, we can calculate the mean:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

the variance:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}$$

and the standard deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}}$$

However, if we have a *sample* of a population, we can only calculate a sample mean and sample variance (and the corresponding sample standard deviation to go with it.) When we collect a sample, we use the sample mean to *estimate* the population mean. Fortunately, this is very intuitive. Given a sample of n members of the population, each with numeric value x_i, the sample mean

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

is a perfectly good estimate for the population mean.

If we calculate the sample variance in the same way, we get:

$$s_n^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}$$

However, this is *not* the best estimate for the population variance. In fact, this is a *biased* estimate: this estimate is, on average, too small! The corrected, unbiased estimate of the variance is:

$$s^2 = s_{n-1}^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$

Check out `help(var)` and `help(sd)`.

In the `census` data, we only have a sample of the data, so we should be calculating the sample variance and sample standard deviation. Fortunately, this is exactly what `var()` and `sd()` do.

You may be wondering *why* the intuitive idea of using n in the denominator is biased, and why $n-1$ gives us an unbiased estimate. The reason is the following theorem:

Theorem: Given a population with mean μ and variance σ^2, if we were to calculate the sample variance using the formula above

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

for *every possible* sample of n data points: $x_1, x_2, \ldots, x^n$, then the mean of all of these sample variances is the population variance,

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

A proof can be found here (click for link to CSUF Library); I encourage you to read through it. The proof is mostly simple algebra.

To get an idea of what's happening, let's look at a variable where we have a small, known population: say, the height (in inches) of everyone in our class.

```
height <- c(70, 75, 77, 60, 68, 67, 64, 62, 72)
class_data <- data.frame(height)

var(class_data$height)
var(class_data$height) * (length(class_data$height) - 1)/(length(class_data$height))

var.sample <- function(set) {
    n <- length(set)
    sum((set - mean(set))^2)/(n - 1)
}

var.pop <- function(set) {
    n <- length(set)
    sum((set - mean(set))^2)/n
}

var.sample(class_data$height)
var.pop(class_data$height)
```

In the code above, we're looking at the difference between the sample variance and population variance in two ways. Next, let's look at the average sample variance over all possible samples (of a certain size)

```
sample_size <- 3
# We need the `gtools` package for the next command
samples <- permutations(length(class_data$height), sample_size,
    class_data$height, set = F, repeats.allowed = T)
```

```
var_samples <- apply(samples, 1, var.sample)
mean(var_samples)
```

Look at the `samples` data set, you should confirm we have every possible way to sample three of our data points. For each set of three, we are finding the sample standard deviation, then averaging the results for all those samples. You should see that this average is, in fact, the *population* standard deviation.

Let's finish off this packet with some visualizations. A histogram will allow us to summarize data, we can also visualize where the mean and median might be.

```
age_at_mar %>%
    ggplot(aes(x = age)) + geom_histogram(binwidth = 1,
    fill = "forestgreen", color = "black") + geom_vline(aes(xintercept = mean(age)),
    color = "blue", linetype = "dashed", size = 1) + geom_vline(aes(xintercept = median(age)),
    color = "red", linetype = "dotted", size = 1) + labs(title = "Age at first marriage for 5,534 US wo
    subtitle = "Dotted red line indicates median, dashed blue line indicates mean.",
    x = "Age at first marriage", y = "Number of women",
    caption = "age_at_mar dataset")
```

```
county %>%
    ggplot(aes(x = pop2017)) + geom_histogram(binwidth = 0.05,
    fill = "forestgreen", color = "black") + geom_vline(aes(xintercept = mean(pop2017,
    na.rm = T)), color = "blue", linetype = "dashed", size = 1) +
    geom_vline(aes(xintercept = median(pop2017, na.rm = T)),
        color = "red", linetype = "dotted", size = 1) +
    labs(title = "Population of US counties in 2017", subtitle = "Dotted red line indicates median, dash
        x = "Population", y = "Number of counties", caption = "county dataset") +
    scale_x_log10()
```

To turn in:

For this packet, you should again create a .pdf file using RMarkdown.

1. Pick a data set from the `openintro` package. You can use `data(package = "openintro")` to see a list of all available data sets in that package. Use `help()` and `glimpse()` to find a good data set with a substantial number of data points, and at least two numeric variables and at least one categorical variable. Use a different data set then of of the ones we have used in class so far. Write a couple of sentences telling me about the data set you chose, and the variables you will be using for your plots.
2. Create a boxplot with a numerical variable on the x-axis, and a categorical variable on the y-axis. You should see several boxplots, allowing you to compare categories. If you get too many boxplots, you may need to pick a different categorical variable (something with fewer possible values). Give an eyeball estimate of the differences between the groups.

3. Create a histogram with a different numerical variable on the x-axis, and the count on the y-axis (this is the default for a histogram). Also create dotted/dashed lines for the mean and median. What does this plot tell you?
4. Read (`http://www.sthda.com/english/wiki/ggplot2-histogram-plot-quick-start-guide-r-software-and-data-v`
`change-histogram-plot-colors-by-groups`), especially the section titled "Change histogram plot colors by groups." Now, apply this idea to your data set: create the same histogram as in the previous part, but separate out two (or more) groups by color. Again, what does this plot tell you?