Chapter 6 Descriptive Statistics

Adapted by Abigail Noyce from *Learning Statistics with R*, Danielle Navarro.[7]

Any time that you get a new data set to look at, one of the first tasks that you have to do is find ways of summarising the data in a compact, easily understood fashion. This is what ***descriptive statistics*** (as opposed to inferential statistics) is all about. In fact, to many people the term "statistics" is synonymous with descriptive statistics.

For practice, we're going to look at data from an online version of the Stroop Effect. You can participate in the experiment here: https://www.psytoolkit.org/experiment-library/experiment_stroop.html , and read a Wikipedia summary of the effect, and its uses in psychology research, here: https://en.wikipedia.org/wiki/Stroop_effect. Let's load the `experiment_data.csv` file and take a quick look.

```
datafile <- here("datasets/stroop/experiment-data.csv")
stroop_data <- read.csv(datafile)

glimpse(stroop_data)
```

```
## Rows: 540
## Columns: 3
## $ participant_id <int> 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 1…
## $ condition      <chr> "congruent", "incongruent", "congruent", "incongruent",…
## $ reaction_time  <dbl> 847.0311, 910.3084, 748.1366, 967.4626, 786.2370, 975.7…
```

This output doesn't make it easy to get a sense of what the data are actually saying. Just "looking at the data" isn't a terribly effective way of understanding data. In order to get some idea about what's going on, we need to calculate some descriptive statistics and draw some nice pictures.

The focus of this chapter is on the descriptive stats, but nonetheless, let's start by looking at a histogram, since it should help you get a sense of what the data we're trying to describe actually look like.

```r
ggplot(data = stroop_data, aes(x = reaction_time)) +
  geom_histogram() +
  theme_minimal()
```

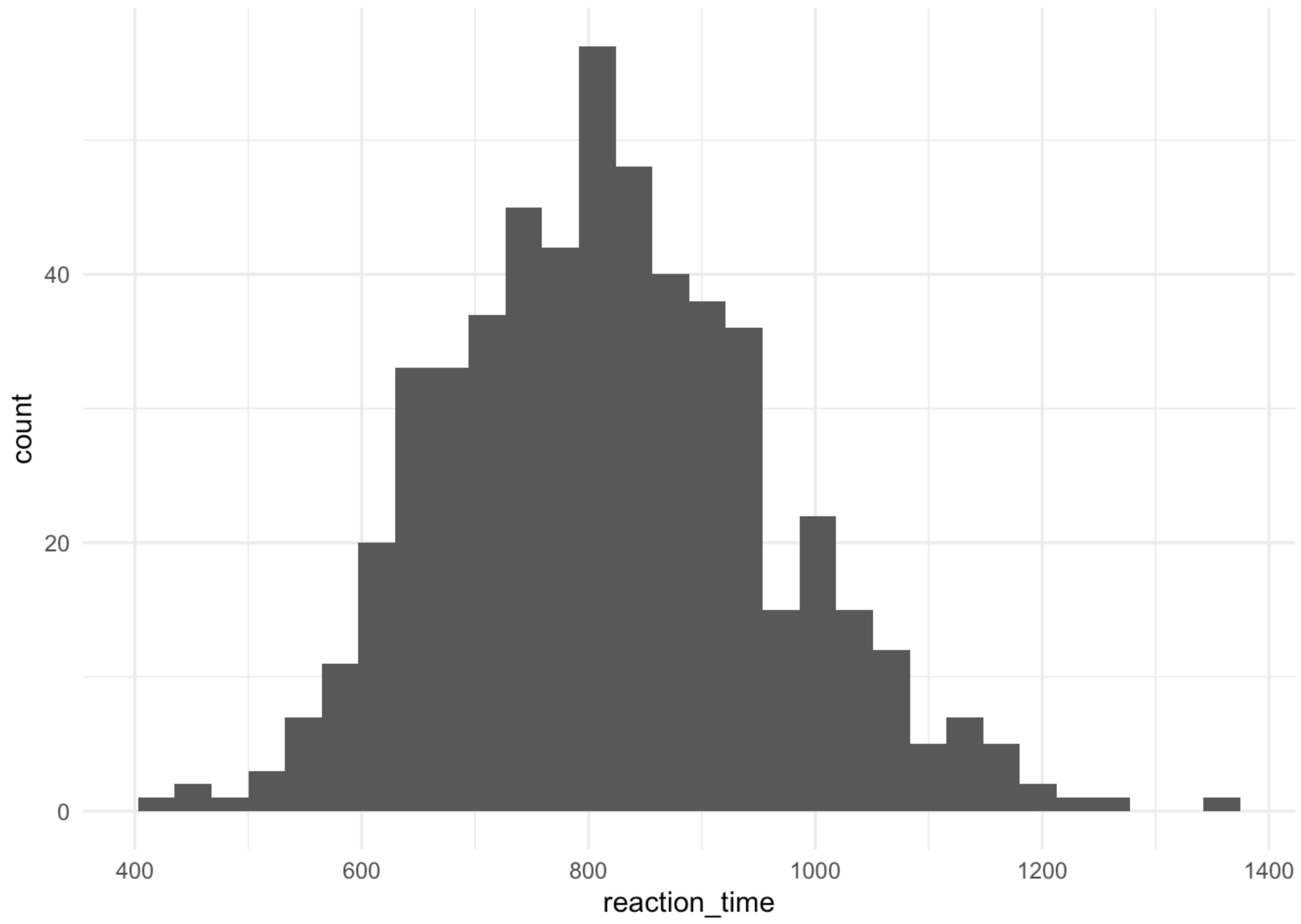

Figure 6.1: Histogram of `reaction_time` in the Stroop `experiment-data.csv` dataset. Reaction times are in milliseconds.

6.1 Counting observations

Drawing pictures of the data, as I did in Figure 6.1 is an excellent way to convey the “gist” of what the data is trying to tell you, it’s often extremely useful to try to condense the data into a few simple “summary” statistics. In most situations, the first thing that you’ll want to know is how many datapoints are included in your dataset.

```
glimpse(stroop_data)
```

```
## Rows: 540
## Columns: 3
## $ participant_id <int> 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 1…
## $ condition      <chr> "congruent", "incongruent", "congruent", "incongruent",…
## $ reaction_time  <dbl> 847.0311, 910.3084, 748.1366, 967.4626, 786.2370, 975.7…
```

The output from `glimpse()` tells us that there are 540 rows… but that may not mean there are 540 participants. In fact, each participant is listed twice, as they have a value for each of two conditions. In other datasets, you may even have a row for each participant for each trial!

One way to count participants would be to get the number of unique participant ID values using `distinct()` and then the count using `n()` .

```
stroop_data %>%
  distinct(participant_id) %>%
  summarize(n_participants = n())
```

```
##   n_participants
## 1            270
```

6.2 Measures of central tendency

After counting, you'll most likely want to calculate a measure of ***central tendency***. That is, you'd like to know something about the "average" or "middle" of your data lies. The most commonly used measures are the mean, median and mode.

6.2.1 Mean

The ***mean*** of a set of observations is just a normal, old-fashioned average: add all of the values up, and then divide by the total number of values. The first five reaction times are 847.0, 910.3, 748.1, 967.5, and 786.2 milliseconds, so the mean of these observations is:

$$\frac{847.0 + 910.3 + 748.1 + 967.5 + 786.2}{5} = 851.82$$

This is, of course, exactly the kind of average we deal with in everyday life. To extend this to create a general formula, we need some mathematical notation. We'll use N to denote the number of observations that we're averaging, and it's traditional to use X to describe the individual observations themselves, with subscripts identifying individual data points. That is, X_1 is the first observation, X_2 is the second, and X_N is the Nth, or last. The mean itself is usually denoted $\bar{X}$, so we could express the general formula for the mean as

$$\bar{X} = \frac{X_1 + X_2 + \ldots + X_{N-1} + X_N}{N}$$

There are several ways to compute a mean in R. First, you could just treat R as a calculator:

```
(847.0 + 910.3 + 748.1 + 967.5+ 786.2) / 5
```

```
## [1] 851.82
```

If your observations are stored in a vector, you can use the `mean()` function:

```
data <- c(847.0, 910.3, 748.1, 967.5, 786.2)
mean(data)
```

```
## [1] 851.82
```

However, you will usually want to wrap `mean()` inside the `tidyverse` function `summarize()` [8], which computes summary statistics on a data frame or tibble.

```
data <- data.frame(reaction_time = c(847.0, 910.3, 748.1, 967.5, 786.2))
data %>% summarize(mean_rt = mean(reaction_time))
```

```
##   mean_rt
## 1  851.82
```

6.2.2 Median

The second measure of central tendency that people use a lot is the ***median***, and it's even easier to describe than the mean. The median of a set of observations is just the middle value. As before let's imagine we were interested only in the first 5 reaction times: 847.0, 910.3, 748.1, 967.5, and 786.2 milliseconds. To determine the median, we sort these numbers into ascending order:

$$748.1, 786.2, \mathbf{847.0}, 910.3, 967.5$$

It's obvious that the median reaction time is 847.0, since that's the middle of the sorted list (and I've put it in bold to make it even more obvious). But what should we do if we were interested in an even number of observations? The first six reaction times, sorted into ascending order, are

$$748.1, 786.2, \mathbf{847.0}, \mathbf{910.3}, 967.5, 975.7$$

and since six is an even number, there are two middle values. The median is defined as the average of those two numbers, 878.65. It's very tedious to do this by hand when you have lots of numbers. but in real life nobody does it that way, we just ask our handy statistical software to compute a median.

```
data <- data.frame(reaction_time = c(847.0, 910.3, 748.1, 967.5, 786.2, 975.7))
data %>% summarize(median_rt = median(reaction_time))
```

```
##   median_rt
## 1    878.65
```

Let’s apply this to the full dataset rather than just the first few observations.

```
stroop_data %>% summarize(
  mean_rt = mean(reaction_time),
  median_rt = median(reaction_time),
  N = n()
)
```

```
##    mean_rt median_rt   N
## 1 817.7276  811.0353 540
```

6.2.3 Mean or median? What’s the difference, and which should I use?

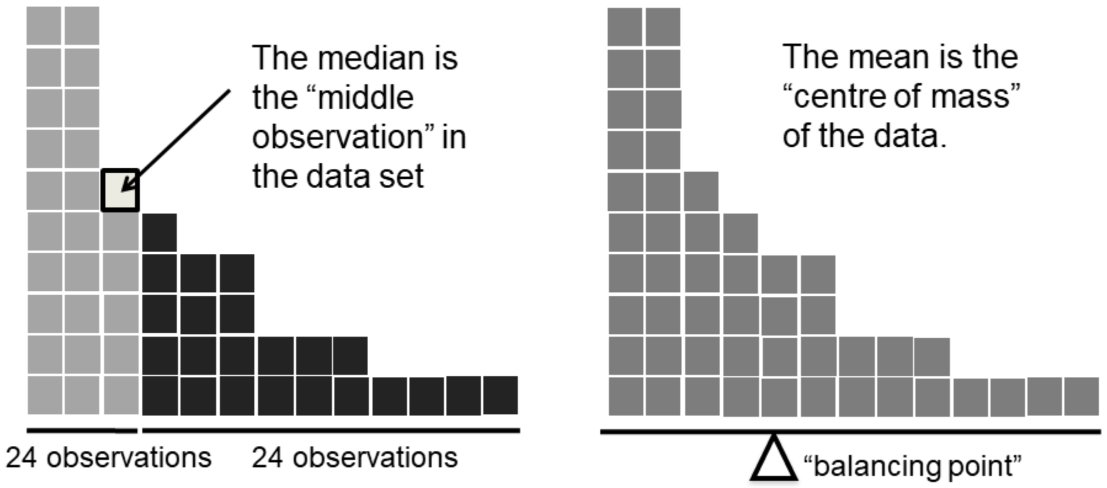
The median is the “middle observation” in the data set
24 observations
24 observations
The mean is the “centre of mass” of the data.
“balancing point”

Figure 6.2: An illustration of the difference between how the mean and the median should be interpreted. The mean is basically the “center of mass” of the data set: if you imagine that the histogram of the data is a solid object, then the point on which you could balance it (as if on a see-saw) is the mean. In contrast, the median is the middle observation. Half of the observations are smaller, and half of the observations are larger.

Knowing how to calculate means and medians is the easy part. You also need to be able to turn that skill into a conceptual understanding of when each should be used. As illustrated in Figure 6.2, the mean identifies the data's center of mass, while the median identifies the exact middle.

- If your data are nominal/categorical, you probably shouldn't be using either. These are for meaningful numbers. If your numbering scheme is arbitrary, use the mode instead.[9]
- If your data are ordinal, you most likely want the median. Because it just takes the middle value, it doesn't depend on the precise numbers involved, and isn't affected by unequal differences between the values.
- If your data are interval or ratio, it is most common to want the mean. The mean takes into account every data point, and is mathematically important in a number of statistical calculations. You may want the median instead if your dataset is skewed, with most outliers on one tail of the distribution. For example, variables like "household income" and "reaction time" tend to have high outliers.

Because of this last point, if you notice that the mean and median of a dataset are quite different from one another, that can imply that the data distribution is asymmetric, skewed in the direction of the mean.

6.2.4 Mode

The mode of a sample is very simple: it is the value that occurs most frequently. To illustrate this, we will turn away from the Stroop experiment data, and look at a second datafile that includes demographic information about the participants in this study.

```r
datafile <- here("datasets/stroop/participant-data.csv")
demographics <- read.csv(datafile)
glimpse(demographics)
```

```
## Rows: 270
## Columns: 3
## $ participant_id <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, …
## $ gender         <chr> "Man", "Man", "Man", "Man", "Man", "Man", "Man", "Man",…
## $ age            <int> 20, 20, 27, 19, 23, 28, 22, 21, 23, 26, 27, 28, 22, 20,…
```

We can see that there is a row for each participant, with their unique ID, gender, and age.[10] To create a frequency table we can use the tidyverse `count()` function.

```
demographics %>%
  count(gender)
```

```
##       gender   n
## 1        Man 100
## 2 Non-Binary  50
## 3      Woman 120
```

Now we can look it over and see that, for the 270 participants in this dataset, the most commonly reported gender is `Woman` .

Unfortunately, unlike the mean and the median, there is not a basic R function to compute the mode. We'll need to work a little harder. The code below first sorts the frequency values in descending order using `arrange()` ; then extracts any row(s) whose frequency matches the first value in the sorted dataset using `filter()` .

```
demographics %>%
  count(gender) %>%
  arrange(desc(n)) %>% # sort by n in descending order
  filter(n == n[1]) # filter for row/s
```

```
##   gender   n
## 1  Woman 120
```

Why did I use `filter(n==n[1])` rather than just `filter(row==1)` ? Well, it's possible for there to be a tie for highest frequency, and it's not clearly specified how the mode should be defined in this case. In psychological research, we are most often computing the mode to report some characteristic of our dataset, and want to know if there are multiple modal values. If you were writing a computational script that takes the mode and then uses it elsewhere, you would want to think about how to handle cases when your data have more than one mode.

One last point. While it generally true that the mode is most often calculated when you have nominal scale data (because means and medians are useless for those sorts of variables), there are some situations in which you really do want to know the mode of an ordinal, interval or ratio scale variable. For instance, supposed you are challenged to a bet to guess the exact magnitude of a friend's Stroop effect (to the nearest 10 ms). If you get it wrong, you lose $1; if you get it right, you win $100. There are no consolation prizes for getting "almost" the right answer. For this bet, the mean and median are useless; it is the mode that you should bet on.

```
stroop_effect_mode <- stroop_data %>%
  mutate(reaction_time = round(x = reaction_time, digits = -1)) %>%
  count(reaction_time) %>%
  arrange(desc(n)) %>%
  filter(n==n[1])

stroop_effect_mode
```

```
##   reaction_time  n
## 1           810 19
```

Because these data are approximately normally distributed, the mode is close to the mean and median. There is, however, no guarantee that this will always be true!

6.3 Measures of variability

The statistics that we've discussed so far all relate to central tendency. That is, they all talk about which values are "in the middle" or "popular" in the data. However, central tendency is not the only type of summary statistic that we want to calculate. The second thing that we really want is a measure of the variability of the data. That is, how “spread out” are the data? How “far” away from the mean or median do the observed values tend to be? For now, let’s assume that the data are interval or ratio scale, so we’ll continue to use the Stroop data. We’ll use this data to discuss several different measures of spread, each with different strengths and weaknesses.

6.3.1 Range

The ***range*** of a variable is very simple: it’s the biggest value minus the smallest value. The R function `range()` gives the minimum and maximum value in the dataset, but not the difference between them, so we need one more computation. (The code below uses `reframe()` rather than `summarize()` because the output has more than 1 row.

```
stroop_data %>%
  reframe(min_max = range(reaction_time)) %>%
  # take the difference between each row and the row immediately preceding
  reframe(range_rt = min_max-lag(min_max,1))
```

```
##   range_rt
## 1       NA
## 2 939.6367
```

(If we wanted to remove the row whose value is `NA`, we could add one more pipe, sending the output to `filter(is.na(range_rt) == FALSE)`.)

Although the range is the simplest way to quantify the notion of “variability”, it’s one of the worst. Recall from our discussion of the mean that we want our summary measure to be robust. If the data set has one or two extremely bad values in it, the range can change by huge amounts.

6.3.2 Interquartile range

The ***interquartile range*** (IQR) is like the range, but instead of calculating the difference between the biggest and smallest value, it calculates the difference between the 25th quantile and the 75th quantile. Probably you already know what a ***quantile*** is (they’re more commonly called percentiles), but if not: the 10th percentile of a data set is the smallest number x such that 10%

of the data is less than x. In fact, we've already come across the idea: the median of a data set is its 50th percentile! R actually provides you with a way of calculating quantiles, using the `quantile()` function.

```
stroop_data %>%
  summarize(median_rt = quantile(reaction_time,probs=.5))
```

```
##   median_rt
## 1  811.0353
```

Reassuringly, this agrees with the answer we got above using `median()`. We can use the same approach to find multiple percentile values in the data. In particular, we can select the 25th, 50th, and 75th percentiles, called ***quartiles*** because they are the values that split the data into quarters.

```
stroop_data %>%
  reframe(quartiles_rt = quantile(reaction_time,probs=c(.25, .50, .75)))
```

```
##   quartiles_rt
## 1     716.7179
## 2     811.0353
## 3     913.5901
```

We could use the same approach as above to compute the interquartile range, but R actually has a built-in function `IQR()` to take that difference. Check for yourself that the IQR output is the same as the difference between the 75th and 25th percentiles above.

```
stroop_data %>%
  summarize(iqr_rt = IQR(reaction_time))
```

```
##     iqr_rt
## 1 196.8722
```

While it’s obvious how to interpret the range, it’s a little less obvious how to interpret the IQR. The simplest way to think about it is like this: the interquartile range is the range spanned by the “middle half” of the data. That is, one quarter of the data falls below the 25th percentile, one quarter of the data is above the 75th percentile, leaving the “middle half” of the data lying in between the two. And the IQR is the range covered by that middle half.

6.3.3 Variance

The two measures we’ve looked at so far, the range and the interquartile range, both rely on the idea that we can measure the spread of the data by looking at the quantiles of the data. However, this isn’t the only way to think about the problem. A different approach is to select a meaningful reference point (usually the mean or the median) and then report the “typical” deviations from that reference point.

While you can compute a mean or median absolute deviation, there are some solid mathematical reasons to prefer the mean squared deviation instead, giving us a measure called the **variance**. This has several nice mathematical properties, as well as one massive psychological flaw. The nice properties are first, that by squaring the deviations, we are giving “extra weight” to the values that are furthest from the mean, which turns out to be important later in things like model fitting. Second, variances are additive. If we have two variables X and Y with variances $\mathrm{Var}(X)$ and $\mathrm{Var}(Y)$, and we want to compute a new variable $Z = X + Y$, then the variance $\mathrm{Var}(Z) = \mathrm{Var}(X) + \mathrm{Var}(Y)$. This is a very useful property in statistics!

You can denote the variance as $\mathrm{Var}(X)$ as I’ve done here, but it’s more often denoted as s^2, for reasons that will hopefully become clear in the next section.

The definition of variance as *mean squared deviation* helpfully tells us exactly how to compute it. We first take each observation X_i’s deviation from the mean $\bar{X}$. Those deviations are squared, summed, and divided by the number of observations N, just like computing any other mean.

$$s^2 = \frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + \cdots + (X_N - \bar{X})^2}{N}$$

To do it in R, we use the `var()` function.

```
stroop_data %>%
  summarize(variance_rt = var(reaction_time))
```

```
##   variance_rt
## 1     20956.6
```

The variance of reaction time in our dataset is 20,956.6 … milliseconds squared? For the moment, let's ignore the burning question that you're all probably thinking (i.e., what the heck does this actually mean?) and instead talk a bit more about how to R is doing these calculations, because something weird and important is happening behind the scenes. If I compute the variance "by hand", it might look something like this:

```
stroop_data %>%
  # compute each observation's squared deviation from the mean
  mutate(squared_deviation_rt = (reaction_time - mean(reaction_time))^2) %>%
  summarize(variance_rt = mean(squared_deviation_rt))
```

```
##   variance_rt
## 1    20917.79
```

And the output is a different number! Is R broken? Is there a typo?

It's not a typo, and R is not making a mistake. What R is doing is evaluating a slightly different formula to the one I showed you above. Instead of averaging the squared deviations, which requires you to divide by the number of data points N, R has divided by $N - 1$. We can confirm that if we switch to dividing by N-1, we get the same output as we did by calling `var()`.

```
stroop_data %>%
  # compute each observation's squared deviation from the mean
  mutate(squared_deviation_rt = (reaction_time - mean(reaction_time))^2) %>%
  summarize(variance_rt = sum(squared_deviation_rt) / (n()-1) )
```

```
##   variance_rt
## 1     20956.6
```

The real question is *why* R is dividing by $N-1$ and not by $N$. After all, the variance is supposed to be the mean squared deviation, right? So shouldn't we be dividing by $N$, the actual number of observations in the sample? Well, yes, we should. However, as we'll discuss later, there's a key distinction between "describing a sample" and "making guesses about the population from which the sample came". Up to this point, it's been a distinction without a difference. Regardless of whether you're describing a sample or drawing inferences about the population, the mean is calculated exactly the same way. Not so for the variance, or the standard deviation, or for many other measures besides. What I outlined to you initially (i.e., take the actual average, and thus divide by
$N$) assumes that you only intend to calculate the variance of the sample. Most of the time, however, you're not terribly interested in the sample in and of itself. Rather, the sample exists to tell you something about the world. If so, you're actually starting to move away from calculating a *sample statistic*, and towards the idea of estimating a *population parameter*. However, I'm getting ahead of myself. For now, let's just take it on faith that R knows what it's doing, and we'll revisit the question later on.

The final question you may have is, how do you *interpret* the variance? Descriptive statistics are supposed to describe things, after all, and right now the variance is really just a gibberish number. Unfortunately, the reason why I haven't given you the human-friendly interpretation of the variance is that there really isn't one. This is a serious problem with the variance. Although it has some elegant mathematical properties that suggest that it really is a fundamental quantity for expressing variation, it's completely useless if you want to communicate with an actual human… variances are completely uninterpretable in terms of the original variable! All the numbers have been squared, and they don't mean anything anymore. This is a huge issue. What does it mean to know that the typical participant's reaction time is 20,000 milliseconds-squared away from the mean? It's *not* a real unit of measurement, and since the variance is expressed in terms of this gibberish unit, it is totally meaningless to a human.

6.3.4 Standard deviation

Suppose that you like the idea of using the variance because of those nice mathematical properties (that I haven't talked about), but – since you're a human and not a robot – you'd like to have a measure that is expressed in the same units as the data itself (i.e., milliseconds, not milliseconds-squared). What should you do? The solution to the problem is obvious: take the square root of the variance, known as the ***standard deviation***, also sometimes called the "root mean squared deviation", or RMSD. This solves our problem fairly neatly: while nobody has a clue what "a variance of 20,000 milliseconds-squared" really means, it's much easier to understand "a standard deviation of 145 milliseconds", since it's expressed in the original units.

It is traditional to refer to the standard deviation of a sample of data as s, though "sd" and "std dev." are also used at times. Because the standard deviation is equal to the square root of the variance, you probably won't be surprised to see that the formula is:

$$s = \sqrt{s^2} = \sqrt{\frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + \cdots + (X_N - \bar{X})^2}{N}}$$

Computing standard deviations in R is relatively simple, using the `sd()` function.

```
stroop_data %>%
  summarize(sd_rt = sd(reaction_time))
```

```
##      sd_rt
## 1 144.7639
```

As you might have guessed from our discussion of the variance, R has assumed you want to estimate the population standard deviation, and divided the sum of squared deviations by $N-1$ before taking the square root. Because we're estimating a population parameter, it is more correct to denote it as $\hat{\sigma}$ rather than s.[11]

$$\hat{\sigma} = \sqrt{\frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + \cdots + (X_N - \bar{X})^2}{N-1}}$$

Interpreting standard deviations is slightly more complex. Because the standard deviation is derived from the variance, and the variance is a quantity that has little to no meaning that makes sense to us humans, the standard deviation doesn't have a simple interpretation. As a consequence, most of us just rely on a simple rule of thumb: in general, you should expect 68% of the data to fall within 1 standard deviation of the mean, 95% of the data to fall within 2 standard deviation of the mean, and 99.7% of the data to fall within 3 standard deviations of the mean. This rule tends to work pretty well most of the time, but it's not exact: it's actually calculated based on an *assumption* that the histogram is symmetric and "bell shaped."[12] If you look back at the histogram in Figure6.1, you should be able to convince yourself that about 2/3 of the data fall between 673 and 962 milliseconds, or 1 standard deviation below and above the mean.

6.3.5 Which measure of variability should I use?

We've discussed quite a few measures of spread, and hinted at their strengths and weaknesses. Here's a quick summary:

- Range. Gives you the full spread of the data. It's very vulnerable to outliers, and as a consequence it isn't often used unless you have good reasons to care about the extremes in the data.
- Interquartile range. Tells you where the "middle half" of the data sits. It's pretty robust, and complements the median nicely. This is used a lot.
- Variance. Tells you the average squared deviation from the mean. It's mathematically elegant, and is probably the "right" way to describe variation around the mean, but it's completely uninterpretable because it doesn't use the same units as the data. Almost never used except as a mathematical tool; but it's buried "under the hood" of a very large number of statistical tools.
- Standard deviation. This is the square root of the variance. It's fairly elegant mathematically, and it's expressed in the same units as the data so it can be interpreted pretty well. In situations where the mean is the measure of central tendency, this is the default. This is by far the most popular measure of variation.

In short, the IQR and the standard deviation are easily the two most common measures used to report the variability of the data; but there are situations in which the others are used.

6.4 Using `summarize()` to compute descriptive statistics

As you may have noticed above, we can use the R function `summarize()` (part of the `tidyverse` packages) to compute most of these values. There are a two useful things I want to show you how to do with this function.

6.4.1 Computing multiple descriptives at once

Above, we generally used `summarize` to compute only one summary statistic at a time, but R can also do a bunch at once. If I want to know the mean, median, standard deviation, IQR, and count of observations, I can run

```
stroop_data %>%
  summarize(
    mean_rt = mean(reaction_time),
    median_rt = median(reaction_time),
    sd_rt = sd(reaction_time),
    iqr_rt = IQR(reaction_time),
    n_observations = n()
  )
```

```
##    mean_rt median_rt    sd_rt   iqr_rt n_observations
## 1 817.7276  811.0353 144.7639 196.8722            540
```

6.4.2 Computing descriptive statistics for groups

In the Stroop effect data we've been looking at, there's one observation for each participant on *congruent* trials, and one for each participant on *incongruent* trials. So far, we've been computing summary statistics as if each observation is independent, and collapsing across all conditions. However, we might want to compute these summaries for each condition independently. Using `group_by()` tells R how to break our dataset into groups before applying the computations from `summarize()` . This lets us see the mean, median, standard deviation, etc for congruent trials and incongruent trials separately.

```
stroop_data %>%
  group_by(condition) %>%
  summarize(
    mean_rt = mean(reaction_time),
    median_rt = median(reaction_time),
    sd_rt = sd(reaction_time),
    iqr_rt = IQR(reaction_time),
    n_observations = n()
  )
```

```
## # A tibble: 2 × 6
##   condition   mean_rt median_rt sd_rt iqr_rt n_observations
##   <chr>         <dbl>     <dbl> <dbl>  <dbl>          <int>
## 1 congruent      744.      746.  105.   151.            270
## 2 incongruent    891.      901.  142.   187.            270
```

6.4.2.1 Collapsing across participants

You may have noticed above that our original estimates of standard deviation will be using the 540 observations in the whole dataset. However, those datapoints aren't independent! Some of them are drawn from the same participant. The *better* way to compute descriptive statistics would be to first average the two values for each participant, and then compute our summary.

```
stroop_data %>%
  group_by(participant_id) %>%
  summarize(participant_rt = mean(reaction_time)) %>%
  summarize(
    mean_rt = mean(participant_rt),
    median_rt = median(participant_rt),
    sd_rt = sd(participant_rt),
    iqr_rt = IQR(participant_rt),
    n_observations = n()
  )
```

```
## # A tibble: 1 × 5
##   mean_rt median_rt sd_rt iqr_rt n_observations
##     <dbl>     <dbl> <dbl>  <dbl>          <int>
## 1    818.      816.  86.7   116.            270
```

If you compare this output to our descriptive stats above, you'll notice that the mean doesn't change at all, the median changes slightly, and the measurements of variability shrink dramatically. This makes sense, because by averaging fast congruent and slow incongruent trials for each participant, we've turned two more-extreme values into one more-moderate one. And the number of observations is now 270, the number of participants in the experiment.

6.5 Good descriptive statistics are descriptive!

6.6 Epilogue: Good descriptive statistics are descriptive!

> *The death of one man is a tragedy. The death of millions is a statistic.*
>
> – Josef Stalin, Potsdam 1945

> *950,000 – 1,200,000*
>
> – Estimate of Soviet repression deaths, 1937-1938 (Ellman 2002)

Stalin's infamous quote about the statistical character death of millions is worth giving some thought. The clear intent of his statement is that the death of an individual touches us personally and its force cannot be denied, but that the deaths of a multitude are incomprehensible, and as a consequence mere statistics, more easily ignored. I'd argue that Stalin was half right. A statistic is an abstraction, a description of events beyond our personal experience, and so hard to visualise. Few if any of us can imagine what the deaths of millions is "really" like, but we can imagine one death, and this gives the lone death its feeling of immediate tragedy, a feeling that is missing from Ellman's cold statistical description.

Yet it is not so simple: without numbers, without counts, without a description of what happened, we have *no chance* of understanding what really happened, no opportunity event to try to summon the missing feeling. And in truth, as I write this, sitting in comfort on a Saturday morning, half a world and a whole lifetime away from the Gulags, when I put the Ellman estimate next to the Stalin quote a dull dread settles in my stomach and a chill settles over me. The Stalinist repression is something truly beyond my experience, but with a combination of statistical data and those recorded personal histories that have come down to us, it is not entirely beyond my comprehension. Because what Ellman's numbers tell us is this: over a two year period, Stalinist repression wiped out the equivalent of every man, woman and child currently alive in the Pittsburgh metropolitan area. Each one of those deaths had its own story, was its own tragedy, and only some of those are known to us now. Even so, with a few carefully chosen statistics, the scale of the atrocity starts to come into focus.

Thus it is no small thing to say that the first task of the statistician and the scientist is to summarize the data, to find some collection of numbers that can convey to an audience a sense of what has happened. This is the job of descriptive statistics, but it's not a job that can be told solely using the numbers. You are a data analyst, not a statistical software package. Part of your job is to take these *statistics* and turn them into a *description*. When you analyse data, it is not sufficient to list off a collection of numbers. Always remember that what you're really trying to do is communicate with a human audience. The numbers are important, but they need to be put together into a meaningful story that your audience can interpret. That means you need to think about framing. You need to think about context. And you need to think about the individual events that your statistics are summarizing.

7. Navarro is Australian, and her original text uses British/Commonwealth spellings and idioms. I (Noyce) am American, and use US ones. Apologies for the occasionally-jarring combination; editing for a consistent set of usages is on the list for a future edition of these notes.↩
8. The American spelling is `summarize()` ; the British spelling is `summarise()` . Both work.↩
9. It has gotten much, much easier to make your statistical software show category names rather than arbitrary numbers, and this reminder almost feels superfluous. I can't bring myself to remove it, though.↩
10. I'm guessing this is in years, but do not actually have a unit!↩

11. Population parameters are generally denoted with Greek letters: σ (sigma) is the Greek s, and the little hat ^ denotes that it is an estimated value. More about this in the next chapter.↩

12. Strictly, the assumption is that the data are *normally* distributed.↩