Lab 2: Arbuthnot

Dr. Arbuthnot's Christening Records

To get you started, run the following command to load the data.

```
data(arbuthnot)
```

The Arbuthnot data set refers to Dr. John Arbuthnot, an 18th century physician, writer, and mathematician. He was interested in the ratio of newborn boys to newborn girls, so he gathered the christening records for children born in London for every year from 1629 to 1710. We can view the data by typing its name into the console.

```
arbuthnot
```

An alternate way to look at the data is to use the data viewer. Click on the name `arbuthnot` in the *Environment* pane (upper right window) that lists the objects in your workspace. This will bring up an alternative display of the data set in the *Data Viewer* (upper left window). You can close the data viewer by clicking on the `x` in the upper lefthand corner.

What you should see are four columns of numbers, each row representing a different year: the first entry in each row is simply the row number (an index we can use to access the data from individual years if we want), the second is the year, and the third and fourth are the numbers of boys and girls christened that year, respectively. Use the scrollbar on the right side of the console window to examine the complete data set.

Note that the row numbers in the first column are not part of Arbuthnot's data. R adds them as part of its printout to help you make visual comparisons. You can think of them as the index that you see on the left side of a spreadsheet. In fact, the comparison to a spreadsheet will generally be helpful. R has stored Arbuthnot's data in a kind of spreadsheet or table called a *data frame*.

You can see the dimensions of this data frame as well as the names of the variables and the first few observations by typing:

```
glimpse(arbuthnot)
```

This command should output the following

```
## Rows: 82
## Columns: 3
## $ year  <int> 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639~
## $ boys  <int> 5218, 4858, 4422, 4994, 5158, 5035, 5106, 4917, 4703, 5359, 5366~
## $ girls <int> 4683, 4457, 4102, 4590, 4839, 4820, 4928, 4605, 4457, 4952, 4784~
```

We can see that there are 82 observations and 3 variables in this data set. The variable names are `year`, `boys`, and `girls`. At this point, you might notice that many of the commands in R look a lot like functions from math class; that is, invoking R commands means supplying a function with some number of arguments. The `glimpse` command, for example, took a single argument, the name of a data frame.

Some Exploration

Let's start to examine the data a little more closely. We can access the data in a single column of a data frame separately using a command like

```
select(arbuthnot, boys)
```

This command will only show the number of boys christened each year. This can be read as "select the boys column out of the arbuthnot data frame."

1. What command would you use to extract just the counts of girls christened? Try it!

Data visualization

R has some powerful functions for making graphics. We can create a simple plot of the number of girls christening per year with the command

```
ggplot(arbuthnot, aes(x = year, y = girls)) +
  geom_point()
```

In the `ggplot()` function we specify which columns of the data frame we wish to visualize, as well as our desired geometry (or geom). If we wanted to connect the data points with lines, we could instead use the `geom_line()`.

```
ggplot(arbuthnot, aes(x = year, y = girls)) +
  geom_line()
```

2. Is there an apparent trend in the number of girls christened over the years? How would you describe it? (To ensure that your lab report is comprehensive, be sure to include the code needed to make the plot as well as your written interpretation in markdown.)

Adding a new variable to the data frame

It will be useful for future analysis if we had access to additional column of information: the total number of births in every year. We can add this with the `mutate()` function.

```
arbuthnot <- arbuthnot %>%
  mutate(total = boys + girls)
```

The `%>%` operator is called the **pipe** operator. It takes the output of the previous expression and pipes it into the first argument of the function in the following one. To continue our analogy with mathematical functions, `x %>% f(y)` is equivalent to `f(x, y)`.

A note on piping: Note that we can read these three lines of code as the following:

*"Take the `arbuthnot` data set and **`pipe`** it into the `mutate` function. Mutate the `arbuthnot` data set by creating a new variable called `total` that is the sum of the variables called `boys` and `girls`. Then assign the resulting data set to the object called `arbuthnot`, i.e. overwrite the old `arbuthnot` data set with the new one containing the new variable."*

This is equivalent to going through each row and adding up the `boys` and `girls` counts for that year and recording that value in a new column called `total`.

You'll see that there is now a new column called `total` that has been tacked on to the data frame. The special symbol `<-` performs an *assignment*, taking the output of one line of code and saving it into an object in your workspace. In this case, you already have an object called `arbuthnot`, so this command updates that data set with the new mutated column.

We can make a plot of the total number of christenings per year with the command

```
ggplot(arbuthnot, aes(x = year, y = total)) +
         geom_line()
```

Similarly to how we computed the total number of births, we can compute the ratio of the number of boys to the number of girls christened in 1629 with

```
5218 / 4683
```

or we can act on the complete columns with the expression

```
arbuthnot <- arbuthnot %>%
  mutate(boy_to_girl_ratio = boys / girls)
```

We can also compute the proportion of newborns that are boys in 1629

```
5218 / (5218 + 4683)
```

or this may also be computed for all years simultaneously and append it to the data set:

```
arbuthnot <- arbuthnot %>%
  mutate(p_boys = boys / total)
```

Note that we are using the new `total` variable we created earlier in our calculations.

3. Now, generate a plot of the proportion of boys born over time. What do you see?

Finally, in addition to simple mathematical operators like subtraction and division, you can ask R to make comparisons like greater than, `>`, less than, `<`, and equality, `==`. For example, we can ask if boys outnumber girls in each year with the expression

```
arbuthnot <- arbuthnot %>%
  mutate(more_boys = boys > girls)
```

This command add a new variable to the `arbuthnot` dataframe containing the values of either `TRUE` if that year had more boys than girls, or `FALSE` if that year did not (the answer may surprise you). This variable contains a different kind of data than we have encountered so far. All other columns in the `arbuthnot` data frame have values that are numerical (the year, the number of boys and girls). Here, we've asked R to create *logical* data, data where the values are either `TRUE` or `FALSE`. In general, data analysis will involve many different kinds of data types, and one reason for using R is that it is able to represent and compute with many of them.

Fast forward to the present

In the previous few pages, you recreated some of the displays and preliminary analysis of Arbuthnot's christening data. Your assignment involves repeating these steps, but for present day birth records in the United States. Load the present day data with the following command.

```
data(present)
```

The data are stored in a data frame called `present`.

4. *What years are included in this data set? What are the dimensions of the data frame? What are the variable (column) names?

5. *How do these counts compare to Arbuthnot's? Are they of a similar magnitude?

6. *Make a plot that displays the proportion of boys born over time. What do you see? Does Arbuthnot's observation about boys being born in greater proportion than girls hold up in the U.S.? Include the plot in your response. *Hint:* You should be able to reuse your code from question 3 above, just replace the data frame name.

7. *In what year did we see the most total number of births in the U.S.? *Hint*: First calculate the totals and save it as a new variable. Then, sort your data set in descending order based on the total column. You can do this interactively in the data viewer by clicking on the arrows next to the variable names. To include the sorted result in your report you will need to use two new functions: `arrange()` (for

sorting). We can arrange the data in a descending order with another function: `desc()` (for descending order). To learn the syntax for this, bring up the help file for `arrange()` by clicking on the Help tab in the bottom right near your Files tab, and entering `arrange` into the search box.

These data come from reports by the Centers for Disease Control. You can learn more about them by searching for `present` in the Help tab.