Introduction to Data

Some define Statistics as the field that focuses on turning information into knowledge. The first step in that process is to summarize and describe the raw information - the data. In this lab, we will utilize a dataset about diabetic hospital encounters provided on behalf of the Center for Clinical and Translational Research at the Virginia Commonwealth University (*http://www.cctr.vcu.edu/*) to the UCI Machine Learning Repository (*http://archive.ics.uci.edu/ml/*). As this is a large data set, along the way you'll also learn the indispensable skills of data processing and subsetting.

Getting Started

This dataset represents diabetic hospital encounters from a 10 year span (1999-2008) from 130 US hosptials and integrated delivery netowrks. The original data contains over 50 variables that describe both patient and hospital outcomes. A full description of the data is provided on the UCI website (*http://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008*).

For the purposes of this lab, we will only focus on a small subset of the variables provided.

We begin by loading the full dataset into the R workspace. After launching RStudio, enter the following command. Be sure to change the pathway in the quotation marks to where the data is saved on your device. **Note:** If you downloaded the module from statsTeachR and saved it on your desktop, the path will be the same except for your username after */Users/*.

```
diabetic <- read.csv("/Users/nunezsa/Desktop/OpenIntro Lab1/source/labs/diabetic_data.csv",
    header = TRUE)
```

The data set diabetic that shows up in your workspace is a *data matrix*, with each row representing a *case* and each column representing a *variable*. R calls this data format a *data frame*, which is a term that will be used throughout the labs.

To view the names of the variables, type the command

```
names(diabetic)
```

This returns the 50 variables contained in this dataset. The first couple names returned are encounter_id, patient_nbr, race, gender, age, and weight.

Notice that each patient has two identifying numbers assigned to them. The first, encounter_id, is assigned each visit. The second, patient_nbr, is only assigned to the patient once. Therefore, if one patient has multiple diabetic encounters, they will appear in this data multiple times, each time with a different encounter_id but with the same patient_nbr.

In this lab, we will only consider unique cases and 7 variables. Let's upload this modified version of this data and call it *diab*. Again, be sure to change the path for which the data is located.

```
diab <- read.csv("/Users/nunezsa/Desktop/OpenIntro Lab1/source/labs/lab1_diabetes.csv",
    header = TRUE)
```

To view the names of the variables in this dataset, type the command

```
names(diab)
```

Because this data is a subset of the larger data frame, x here refers to the row number from the original data.

To make these variables easier to work with, let's rename them:

```
colnames(diab) <- c("x", "gender", "race", "labs", "meds", "diags", "insulin")
```

> **Exercise 1** How many cases are there in this data set? How many variables? For each variable, identify its data type (e.g. categorical, discrete). What does each variable describe?

We can have a look at the first few entries (rows) of our data with the command

```
head(diab)
```

and similarly we can look at the last few by typing

```
tail(diab)
```

You could also look at *all* of the data frame at once by typing its name into the console, but that might be unwise here. We know diab has 54,745 rows, so viewing the entire data set would mean flooding your screen. It's better to take small peeks at the data with **head**, **tail** or the subsetting techniques that you'll learn in a moment.

Summaries and tables

A good first step in any analysis is to distill all of the provided information into a few summary statistics and graphics. As a simple example, the function **summary** returns a numerical summary: minimum, first quartile, median, mean, third quartile, and maximum. For labs this is

```
summary(diab$labs)
```

R also functions like a very fancy calculator. If you wanted to compute the interquartile range for the patients number of labs, you would look at the output from the summary command above and then enter

```
57 - 31
```

R also has built-in functions to compute summary statistics one by one. For instance, to calculate the mean, median, and variance of labs, type

```
mean(diab$labs)

var(diab$labs)

median(diab$labs)
```

While it makes sense to describe a quantitative variable like labs in terms of these statistics, what about categorical data? We would instead consider the sample frequency or relative frequency distribution. The function **table** does this for you. For example, to see the distribution of gender in this population (the number of males and females), type the command

```
table(diab$gender)
```

or instead look at the relative frequency distribution by typing

```
(table(diab$gender)/54745) * 100
```

Notice how R automatically divides all entries in the table by 54,745 and multiplies the frequencies by 100 in the command above. This gives the percentages of the population that are male and female.

Next, we make a bar plot of the entries in the table by putting the table inside the barplot command.

```
barplot(table(diab$gender))
```

Notice what we've done here! We've computed the table of diab$gender and then immediately applied the graphical function, **barplot**. This is an important idea: R commands can be nested. You could also break this into two steps by typing the following:

```
gender <- table(diab$gender)

barplot(gender)
```

Here, we've made a new object, a table, called gender (the contents of which we can see by typing gender into the console) and then used it in as the input for **barplot**. The special symbol <- performs an *assignment*, taking the output of one line of code and saving it into an object in your workspace. This is another important idea that we'll return to later.

> **Exercise 2** Create a numerical summary for meds and diags, and compute the interquartile range for each. Compute the relative frequency distribution for race and insulin. How many known Caucasians are in the sample? What percentage of the sample has steady insulin levels?

The **table** command can be used to tabulate any number of variables that you provide. For example, to see the insulin levels across each gender, we could use the following.

```
table(diab$gender, diab$insulin)
```

Here, the rows refer to gender and the columns refer to insulin levels. To create a mosaic plot of this table, we would enter the following command.

```
mosaicplot(table(diab$gender, diab$insulin))
```

We could have accomplished this in two steps by saving the table in one line and applying **mosaicplot** in the next (see the table/barplot example above).

Exercise 3 What does the mosaic plot reveal about insulin levels and gender?

Interlude: How R thinks about data

We mentioned that R stores data in data frames, which you might think of as a type of spreadsheet. Each row is a different observation (a different respondent) and each column is a different variable (the first is `x`, the second `gender` and so on). We can see the size of the data frame next to the object name in the workspace or we can type

```
dim(diab)
```

which will return the number of rows and columns, respectively. Now, if we want to access a subset of the full data frame, we can use row-and-column notation. For example, to see the sixth variable of the 567th patient, use the format

```
diab[567, 6]
```

which means we want the element of our data set that is in the 567th row (meaning the 567th person or observation) and the 6th column (in this case, number of diagnoses). We know that `diags` is the 6th variable because it is the 6th entry in the list of variable names

```
names(diab)
```

To see the number of diagnoses for the first 10 patients we can type

```
diab[1:10, 6]
```

In this expression, we have asked just for rows in the range 1 through 10. R uses the ":" to create a range of values, so 1:10 expands to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. You can see this by entering

```
1:10
```

Finally, if we want all of the data for the first 10 patients, type

```
diab[1:10, ]
```

By leaving out an index or a range (we didn't type anything between the comma and the square bracket), we get all the columns. When starting out in R, this is a bit counterintuitive. As a rule, we omit the column number to see all columns in a data frame. Similarly, if we leave out an index or range for the rows, we would access all the observations, not just the 567th, or rows 1 through 10. Try the following to see the number of diagnoses for all 54,745 patients fly by on your screen

```
diab[, 6]
```

Recall that column 6 represents a patients' number of diagnoses, so the command above reported all of them in the data set. An alternative method to access the `diags` data is by referring to the name. Previously, we typed

names(diab) to see all the variables contained in the diabetes data set. We can use any of the variable names to select items in our data set.

```
diab$diags
```

The dollar-sign tells R to look in data frame diab for the column called diags. Since that's a single vector, we can subset it with just a single index inside square brackets. We see the number of diagnoses for the 567th patient by typing

```
diab$diags[567]
```

Similarly, for just the first 10 patients

```
diab$diags[1:10]
```

The command above returns the same result as the diab[1:10,6] command. Both row-and-column notation and dollar-sign notation are widely used, which one you choose to use depends on your personal preference.

A little more on subsetting

It's often useful to extract all individuals (cases) in a data set that have specific characteristics. We accomplish this through *conditioning* commands. First, consider expressions like

```
diab$gender == "Male"
```

or

```
diab$labs > 30
```

These commands produce a series of TRUE and FALSE values. There is one value for each patient, where TRUE indicates that the person was male (via the first command) or had more than 30 labs done (second command).

Suppose we want to extract just the data for the men in the sample, or just for those with more than 30 labs. We can use the R function **subset** to do that for us. For example, the command

```
mdata <- subset(diab, diab$gender == "Male")
```

will create a new data set called mdata that contains only the men from the diab data set. In addition to finding it in your workspace alongside its dimensions, you can take a peek at the first several rows as usual

```
head(mdata)
```

This new data set contains all the same variables but just under half the rows. It is also possible to tell R to keep only specific variables, which is a topic we'll discuss in a future lab (it has been done for you in this lab to get the

smaller dataset). For now, the important thing is that we can carve up the data based on values of one or more variables.

As an aside, you can use several of these conditions together with & and |. The & is read "and" so that

```
m_and_over30 <- subset(diab, diab$gender == "Male" & diab$labs > 30)
```

will give you the data for men with more than 30 lab procedures. The | character is read "or" so that

```
m_or_over30 <- subset(diab, diab$gender == "Male" | diab$labs > 30)
```

will take people who are men *or* have more than 30 labs (why that's an interesting group is hard to say, but right now the mechanics of this are the important thing). In principle, you may use as many "and" and "or" clauses as you like when forming a subset.

> **Exercise 4** Create a new object called over50_and_female that contains all observations of female patients that are on more than 50 medications. Write the command you used to create the new object as the answer to this exercise.

Quantitative data

With our subsetting tools in hand, we'll now return to the task of the day: making basic summaries of the diabetes dataset. We've already looked at categorical data such as insulin and race so now let's turn our attention to quantitative data. Two common ways to visualize quantitative data are with box plots and histograms. We can construct a box plot for a single variable with the following command.

```
boxplot(diab$diags)
```

You can compare the locations of the components of the box by examining the summary statistics.

```
summary(diab$diags)
```

Confirm that the median and upper and lower quartiles reported in the numerical summary match those in the graph. The purpose of a boxplot is to provide a thumbnail sketch of a variable for the purpose of comparing across several categories. So we can, for example, compare the heights of men and women with

```
boxplot(diab$diags ~ diab$gender)
```

The notation here is new. The ~ character can be read "versus" or "as a function of". So we're asking R to give us a box plots of the number of diagnoses where the groups are defined by gender.

> **Exercise 5** Pick another categorical variable from the data set and see how it relates to number of medications. List the variable you chose, why you might think it would have a relationship to number of medications, and indicate what the figure seems to suggest.

Finally, let's make some histograms. We can look at the histogram for the number of lab procedures done on the patients with the command

```
hist(diab$labs)
```

Histograms are generally a very good way to see the shape of a single distribution, but that shape can change depending on how the data is split between the different bins. You can control the number of bins by adding an argument to the command. In the next two lines, we first make a default histogram of meds and then one with 50 breaks.

```
hist(diab$meds)

hist(diab$meds, breaks = 50)
```

Note that you can flip between plots that you've created by clicking the forward and backward arrows in the lower right region of RStudio, just above the plots. How do these two histograms compare?

At this point, we've done a good first pass at analyzing the information in the diabetes dataset. We've also picked up essential computing tools – summary statistics, subsetting, and plots – that will serve us well in future labs.

On Your Own

1. Make a scatterplot of the number of labs and the number of medications someone is on for the first 100 patients. Describe the relationship between these two variables.
2. Create a new categorical variable that splits the patient population into two groups, patients who are on 16 or less medications, and those who are on more than 16 medications. Create a mosaic plot of these two groups as a function of gender. Describe the plot. Are you suprised by the results? Why or why not. (Hint: you can create a new variable called, say meds16, by typing diab$med16. Just as we call on variables in the data using a dollar sign, we can similarly create new ones.)
3. Are African Americans more represented in the 16 or less medications population or in the more than 16 medications population? Create a visual or figure that best supports your answer.
4. Create a subset of the data for African American females. Describe the shape and center of the distribution of the number of labs in this population. Compare this distribution to that of African American males. Is there any difference? Can you compare these two distributions easily with side-by-side boxplots? Why or why not.