

Lab 2: Exploratory Analysis (Part 1, 4 Questions)

Due 1/13

Throughout this first half of this lab, you will be generating simple graphical and numerical summaries of a dataset collected by the Centers for Disease Control and Prevention (CDC) and discussed several times in class. The Behavioral Risk Factor Surveillance System (BRFSS) is an annual telephone survey of 200,000 people in the United States. As its name implies, the BRFSS is designed to identify so-called *risk factors* in the adult population and report emerging trends. For example, respondents are asked about their diet and weekly physical activity, their HIV/AIDS status, possible tobacco use, and even their level of healthcare coverage. The BRFSS Web site (<http://www.cdc.gov/brfss>) contains a complete description of the survey, the questions that were asked and even research results that have been derived from the data.

We will focus on a sample of 20,000 people from the BRFSS survey conducted in 2000.¹ While there are over 200 questions or *variables* in this dataset, we are only going to work with a small subset. During this first part of the lab, we will walk you through the commands to recreate the graphical and numerical summaries in lecture and have you discuss what you see.

In the second half of the lab, you will work with the “student view” of the registrar’s data set. The questions will be somewhat open-ended, inviting you to examine the data on your own. In several places in the lab, you will be asked questions about the plots you create. If you choose to do all your work in our Computing Lab in Boelter Hall, the TA can show you how to prepare your report in Appleworks, a text editor similar to MS Word. He will also show you how to incorporate R graphics into your lab report. You are expected to answer all the questions in this document, providing whatever graphs or numerical summaries you think are necessary. You

¹Recall from lecture that we have added an extra column to the CDC survey, the degree to which a city or town is “sprawling.” This was added by a researcher at Johns Hopkins University and his sprawl values depended on data from the 2000 Census. Hence we are working with data from 2000. The BRFSS, however, is a yearly survey and you could analyze data from any (all?) years available on their web site.

should include an appendix with the commands you entered.

Note: I view computing as a collaborative, if not social exercise; You are strongly encouraged to work in groups, to consult the TAs, to talk to me, send me e-mail, IMs, you name it. I want computing to feel like a natural extension of the concepts you are learning. **You are, however, expected to write things up on your own, and to include with your assignment a list of the names of people you worked with.** While collaboration is encouraged, you are still individually responsible for the material; you will not be allowed to consult with your classmates during exams and the final.

Finally, this first half of the lab is a bit mechanical. You are both learning the statistical tools as well as the language for expressing them. It's hard to have both of these tasks take place at once, so in this first part you will be directed; in the second half, you will explore. Have fun!

1. Getting started

We begin by loading the dataset of 20,000 observations into your R workspace. After starting R (by clicking on the icon in your dock, or on your desktop if you have downloaded a copy of R onto a computer running some flavor of the Windows operating system), enter the following command.

```
source("http://www.stat.ucla.edu/~cocteau/stat13/data/cdc.R")  
  
ls()
```

The second command lists the *objects* that are now in your workspace. You should see the dataset² `cdc` and a vector called `index`. The BRFSS sample is stored in `cdc` and is the same data we analyzed in Lecture 2. Remember, you can have a dump of any object's contents printed to your screen by simply typing the name of the object and hitting return; for this object, lots and lots of data scroll by, and it's hard to actually learn something about the data this way. Instead, we will walk you

²In R lingo, `cdc` is called a *data frame*; it is basically just a matrix. If you ever want to know a bit more about the underlying structure of R, you can consult one of the online manuals available from the R Web site, cran.r-project.org, or talk to Fei, Zhangzhang or Mark during their office hours.

through some simple commands that will help you get the feeling for any dataset we will encounter this quarter.

Let's start with `cdc`. It contains responses from 20,000 people to a series of 12 questions; therefore, our *sample* consists of 20,000 people, each *observation* is person, and for each person we have 12 *variables*. You can see the names of the variables contained in a dataset by typing

```
dim(cdc)
names(cdc)
```

This should return the names `state`, `genhlth`, `physhlth`, `exerany`, `hlthplan`, `smoke100`, `height`, `weight`, `wtdesire`, `age`, `gender` and `sprawl`. In class, we described the questions leading to these variables: For example, `genhlth`, respondents were asked to evaluate their general health, giving a score from 1-5 (excellent = 1, very good = 2, good = 3, fair = 4 and poor = 5); `exerany`, this is 1 if the respondent exercised in the past month and 0 otherwise; `hlthplan`, this is a 1 if the respondent has some form of health coverage and 0 otherwise; `smoke100`, this is 1 if the respondent has smoked at least 100 cigarettes in their entire life and 0 otherwise; and finally, we have variables that record the respondent's `height` in inches, `weight` in pounds, `age` in years, and `gender`. Consult your lecture notes for a complete description of all the variables in our dataset. Also, the `state` variable is coded using the FIPS (Federal Information Processing Standards) code, described at <http://www.itl.nist.gov/fipspubs/fip5-2.htm> (scroll down the page a bit; you'll find a table of numeric codes and state names).

2. Summaries and tables

As we've noted several times in class, R has been designed so that you can easily "express" statistical calculations. As a simple example, the command `summary` returns the 5 number summary (minimum, first quartile, median, second quartile and maximum) as well as the arithmetic mean of a variable. For `weight` this is

```
summary(cdc$weight)
```

R also functions like a very fancy calculator; if you wanted to compute the inter-quartile range for the respondents' weight, you would look at the output from the

summary command above and then enter

```
190.0 - 140.0
```

Why? R also has built-in functions to compute the mean and variance (mentioned in Lecture 3 but we won't go into much depth until Lecture 5); for weight we can type

```
mean(cdc$weight)
```

```
var(cdc$weight)
```

and there is even a shortcut for the median

```
median(cdc$weight)
```

For categorical data, we would instead consider the sample frequency or relative frequency distribution; the command `table` does this for you by counting the number of times each kind of response was given. For example, to see the number of people who have smoked 100 cigarettes in their lifetime, you could type

```
table(cdc$smoke100)
```

or instead look at the *relative* frequency distribution by typing

```
table(cdc$smoke100)/20000
```

Why? (Since R is a big calculator; we can divide the totals from `table(cdc$smoke100)` by 20,000, the sample size.) Finally, a barplot is as simple as

```
barplot(table(cdc$smoke100))
```

Notice what we've done here! We've formed a table from the `smoke100` variable with `table(cdc$smoke100)` and we've provided that output to the command `barplot` that, in turn, makes a barplot of the counts in the table. This is an important idea. R's commands can be nested. You could also do the following:

```
smoke = table(cdc$smoke100)
```

```
barplot(smoke)
```

Here, we've made a new object, a table, called `smoke` (which we could see by typing `smoke` and hitting "enter") and then used it in a call to `barplot`. Either way will work, it's just good that you see how R's commands can be easily combined. We'll have more to say about creating new variables a little later.

Question 1. Create the five-number summary for height and age, and compute the inter-quartile range for each. Compute the relative frequency distribution for gender and exerany. How many males are in the sample? What proportion of the sample reports being in excellent health?

The `table` command can be used to tabulate any number of variables you provide it. That is, if we want to examine which participants have smoked broken down by gender, we could use the following.

```
table(cdc$gender,cdc$smoke100)
```

Here, we see column labels of 0 and 1; 0 meaning they have not smoked 100 cigarettes in their lifetime, and 1 meaning they have. The rows refer to gender. To create a mosaic plot of this table, we would enter the following command.

```
mosaicplot(table(cdc$gender,cdc$smoke100))
```

And we could have done this in two steps, as we did with the `barplot` example above.

Question 2. What does the mosaic plot reveal about smoking habits and gender?

3. Interlude: How R thinks about data

3a. Extracting subsets

R treats the dataset `cdc` as a kind of spreadsheet or a *matrix*. Each row is a different observation (a different respondent) and each column is a different variable (the first is `state`, the second `genhlth` and so on). To see the overall size of this table, we can type

```
dim(cdc)
```

which should return the number of rows and columns in the dataset. We can access the data using row-and-column notation. So we can see the weight of the 567th respondent with the command

```
cdc[567,8]
```

which means we want the element of our dataset that is in the 567th row (meaning the 567th person or observation) and the 8th column (meaning `weight`). To see the

weights for the first 10 respondents we could type

```
cdc[1:10,8]
```

In this expression, we have asked just for rows in the *range* 1 through 10 (R uses the “:” to create a range of values, 1:10 expanding 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 respondents, we could type

```
cdc[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 (compare what you see on your screen with the printout given in Lecture 2). 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 all of the weights for all 20,000 respondents fly by on your screen

```
cdc[,8]
```

Again, this is asking for all the data from variable or column 8, `weight`.

We can also access data for particular questions by referring to their names. On a previous page, we typed `names(cdc)` and saw all the variables contained in the dataset. We can use any of these to select items in our dataset. For example, we see the weight for the 567th respondent by typing

```
cdc$weight[567]
```

(compare this with what you got using row-and-column notation above) or for just the first 10 respondents

```
cdc$weight[1:10]
```

or all the weights

```
cdc$weight
```

And again, if you just type

```
cdc
```

all 8 variables for all 20,000 respondents will flash by. While we will try to teach you only as much about the R language as you absolutely need, we thought we should spend some time with these various ways to access information in a dataset; it's a pretty fundamental concept and hopefully the basic ideas make sense.

3b. A reminder about printing data

Recall from the last lab that the way R chooses to display data will differ depending on what you've asked it for. So, if we want to see the weights and the desired weights for the first 10 survey respondents we would type

```
cdc[1:10,8:9]
```

(because weight and desired weight are in columns 8 and 9, and we want just the first 10 rows or respondents) and we would see something like

```
      weight wtdesired
1       175        175
2       125        115
3       105        105
4       132        124
5       150        130
6       114        114
7       194        185
8       170        160
9       150        130
10      180        170
```

Recall that when printing data sets to the screen, R adds row labels (the first column above, the numbers 1 through 10). These are not part of the data, but are added to help you examine the printout.

On the other hand, when you print data for a single variable, you get a different kind of display. So, if we want the weights for the first 50 participants we would type the expression

```
cdc[1:50,8]
```

and we should see

```
[1] 175 125 105 132 150 114 194 170 150 180 186 168 185 170 170 185 156
[18] 185 200 125 200 160 160 165 105 190 190 160 115 185 166 180 182 185
[35] 220 117 160 190 125 160 124 143 118 210 200 145 175 130 112 141
```

The display is different now, with the numbers being stacked on top of each other in a compact way. Here R adds [1] because 175 is the weight of the first participant, [18] because 185 is the weight of the 18th, and [35] because 220 is the weight of the 35th. Again, we like this stacked print out because it lets us look at all the weight values more easily; if R printed them in one long column like it does for data sets, it would be much harder to compare different values or to get an overall sense of them visually (not that even this stacked business is very informative).

At a technical level, the real distinction here is that `cdc` is a data frame (a spreadsheet or a kind of matrix) whereas `cdc[1:50,8]` is a vector, a collection of one kind of data, in this case the weights of 50 people. R is making a choice. Depending on the kind of object you want printed, it is choosing a print format that the language designers thought was best. You are encouraged to question this choice!

As a last example, recall that we've also seen R exhibit or print simple tabulations of data, the output from the `table` command. The display for these objects is different than either of the two formats above, and is well-suited for 2x2 tables.

3c. A little more on subsetting

In class, we often performed our analyses on subsets of the data. While we have seen how to pull out the first 10 respondents from a data set or maybe just a single variable, we often want to extract portions of a data set based on the values of one or more variables. We'll do this in stages. First, consider expressions like

```
cdc$gender=="m"
```

or

```
cdc$age > 30
```

If you type these, you will see a series of `TRUE` and `FALSE` values. There should be

20,000 of them, and each is true or false depending on whether the corresponding respondent is over age 30 or is a male.

Now, suppose we want to extract just the data for the men in the sample, or just for those over 30. We can use the R command `subset` to do that for us. For example, the command

```
md = subset(cdc,cdc$gender=="m")
```

will give us the complete data for just the men, and create a new data set called `md`. You can see the new object if you type `ls()`. You can also see how big it is and the variables as usual

```
dim(md)
names(md)
```

So it contains all the same variables, but just under half the rows. It is also possible to tell R to not keep all the variables, but just a few. We'll get to that later. 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 string several of these conditions together with `&` and `|`; the `&` is read "and" so that

```
md = subset(cdc,cdc$gender=="m" & cdc$age > 30)
```

will give you the data for men over the age of 30; while `|` is read "or" so that

```
md = subset(cdc,cdc$gender=="m" | cdc$age > 30)
```

will take people who are either men or over the age of 30 (why that's an interesting group is hard to say, but right now the mechanics of this are the important thing).

4. Quantitative data

We will finish this part of the lab with a series of plots for quantitative data. Our main tools (for the moment) will be boxplots and histograms. Although we will describe how you can make the violin plots from lecture in a supplement.

We can construct a boxplot for a single variable with the following command.

```
boxplot(cdc$height)
```

This should give one box and whiskers construction; you can compare the locations of the components of the box by examining the five number summary

```
summary(cdc$height)
```

Do the median and upper and lower quartiles reported here match those in the graph? As we have seen in class, the purpose of a boxplot is to provide a thumbnail sketch of data set for the purpose of comparing across several categories. So, we can, for example, compare the heights of men and women with

```
boxplot(cdc$height~cdc$gender)
```

The notation here is new. The ~ can be read “related to” or “as a function of”. So we’re asking R to give us a boxplots of heights where the groups are defined by gender. Recall in class we compared BMI to general health. The following two lines first make a new object called `bmi` and then creates boxplots of these values, defining groups by the variable `cdc$genhlth`.

```
bmi = 703*(cdc$weight)/(cdc$height * cdc$height)
boxplot(bmi~cdc$genhlth)
```

Notice that the first line above is just some arithmetic, but done on all 20,000 numbers in the `cdc` data set. That is, for each of the 20,000 participants, we take their weight, divide by their height and then multiply by 703. The result is 20,000 BMI values, one for each respondent. This is one reason why we like R; it lets us perform computations like this using very “high level” expressions.

Question 3. What does this boxplot show? Pick another of the categorical variables in your data set and see how it relates to BMI. List the variable you chose, why you might think it would have a relationship to BMI and indicate what the figure seems to suggest.

Finally, we come to histograms. We can have a look at the histogram for the age of our respondents with the command

```
hist(cdc$age)
```

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 BMI and then one with

50 breaks.

```
hist(bmi)
hist(bmi,breaks=50)
```

How do these compare? Another interesting histogram came up in class. Let's consider the difference between desired weight and current weight. We can create a histogram with either

```
diffw = cdc$wtdesired-cdc$weight
hist(diffw)
```

where we created a new variable `diffw` to hold the 20,000 differences in weights and desired weights, or we can do it in one step

```
hist(cdc$wtdesired-cdc$weight)
```

Again, both work, it just depends on which is easier for you to read and interpret.

Question 4. Make a histogram of the heights of the women in our survey. To do this, first use the `subset` command in Section 3c to extract just the women in the survey. Then create a histogram of the `height` variable in the new data set. What do you see?

The second part of your lab will be handed out Wednesday in class. It will concern the registrar's data, and is largely open-ended, letting you use the tools you have learned here.