Math 243 **Lab #1**

Bivariate Data in **R**: Scatterplots, Correlation and Regression

---

Overview
Thus far in the course, we have focused upon displays of univariate data: stem-and-leaf plots, histograms, density curves, and boxplots. In this lab we consider displays of bivariate data, which are instrumental in revealing relationships between variables.

Variables and data entry
Though there are other ways of assigning variables, the easiest way is probably with the assignment operator '<–'. (In general, '=' is *not* used.) In **R** variables must exclusively consist of *numeric* (quantitative) values, *string* values (categorical data), or boolean (TRUE or FALSE) values. (Missing entries are considered non-existent with value "NA".) Some simple examples:

```
> x <– c(1.72, 3.68, 5.91, -.27, 3.11)
> y <– x > 2
```

creates a numeric vector x and a boolean vector y. Try typing these in, and then type

```
> y
```

to see the contents of the vector y. One may similarly create a categorical variable:

```
> x <– c("Dallas", "Detroit", "Cedar Rapids", "Detroit")
```

A command like

```
> c("Dallas", "Detroit", "Cedar Rapids", "Detroit")
```

produces a string vector, but does not assign it to a variable, and so the input is echoed to the screen rather than stored anywhere.

If one wishes to create a numeric vector x whose values are evenly-spaced, an easier method is through a call to the **seq()** function. For example, the following two commands have the same effect:

```
> x <– c(1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2)
> x <– seq(1, 2, .1)
```

In the above example, the spacing between successive values was 0.1. When the spacing is to be 1, as in the list of values 0, 1, 2, . . . , 10, the command "0:10" will do. So, another way to get the above is

```
> x <– 1 + 0:10/10
```

Variable names must begin with a letter. Names may be quite lengthy if so desired, but they may not contain spaces, hyphens, underscores, or any other type of punctuation other than a period. So, the variable name "store.name" is permissible, whereas "store_name" is not. Also, keep in mind that variable names, just like all command/function calls in **R**, are case-sensitive. **R** does not protect its functions from being redefined. Since 'c' is the name of a function, 'T' and 'F' are tags for the boolean values "TRUE" and "FALSE", it is quite important not to use 'c', 'T' or 'F' as variable names. Another name one might be tempted to use but should not is 'data' (a command in **R** that loads pre-entered data sets).

If you wish to know how many entries are in a vector x, the command length(x) will reveal this. If the vector x has 15 entries, then x[1] returns the first entry in the list. The commands x[seq(3, 8)] and x[3:8] both output a subvector of x consisting of the 3rd, 4th, ..., 8th entries in x. If x is a numeric vector and you want the subvector consisting only of those entries in x whose values are greater than 5, type $x[x >$ 5]. To add a 16th entry from the command line, one may simply type

```
> x[16] = 31.7
```

(Notice that, in this case, "=" is the appropriate assignment operator.) You may add a 19th entry in the same way, even if nothing has been done about x[17] and x[18], in which case both of those entries are viewed as missing ("NA"). You might also use the "x[k] = ..." command to make changes to a data point, or use the **edit()** command described below.

For bivariate data, it is preferable to place the data in a *data frame*, essentially a table of data. If one needs to enter data from the keyboard into a data frame, perhaps the easiest way is to create a blank data frame and then use the **edit** command. Enter the data of Example 3.2, p. 100, in the following way:

```
> sugarbush <- data.frame()
> sugarbush <- edit(sugarbush)
```

The first of these commands makes *sugarbush* the name of a blank data frame. The **edit** command brings up a spreadsheet-type window in which the data may be entered. It is a good idea to click on the the top cell of each column and rename the variable. For this data, "soil.pH" and "crown.dieback" seem like good names. Go ahead and enter the data, clicking on "Quit" when you are done. (Incidentally, to edit the data in the frame "sugarbush" you might simply repeat the edit command as it appears above. The command "edit(sugarbush)" is not enough, in that whatever changes you make will be echoed to the screen, not stored.) The commands

```
> sugarbush
> sugarbush$soil.pH
> sugarbush$crown.dieback
```

display the entire data frame, the first column vector and the second column vector respectively. If one were going to do much with this data, it would quickly become tedious to type such long names. In fact, it is only necessary to type enough to distinguish the column from other columns (so sugarbush$cr would be enough to get the 2nd column), but one may give special status to a data frame by "bfattaching" it. See how **R** responds to these commands:

```
> attach(sugarbush)
> soil.pH
> detach(sugarbush)
> soil.pH
```

Scatterplots
In statistics, a plot of one quantitative variable against another is called a *scatterplot*. One simply chooses to consider one of the variables as *explanatory* and the other as *response*. Having made these assignments, one plots on the plane each coordinate pair: (*expl. value*, *resp. value*). Figure 3.3 on p. 101 is a scatterplot of the data in Example 3.2. To plot it yourself, type

> attach(sugarbush)
> plot(soil.pH, crown.dieback)

In my display, the dots are plotted as large circles with clear centers. Try the command

> plot(soil.pH, crown.dieback, cex=.5, pch=19)

which employs the **cex** option to change the size of the points and the **pch** option to change the type of point displayed. For more on these and other options, see "help(points)".

By default, **plot()** displays points, as above. One may change the type of plot with the **type** option. Compare the results from the commands

> plot(soil.pH, crown.dieback, type="p")
> plot(soil.pH, crown.dieback, type="l")
> plot(soil.pH, crown.dieback, type="h")

To see other possible plot types, along with methods for labeling x and y axes, providing title and subtitle to the plot, etc., look at "help(plot)".

Incidentally, the method for plots of standard functions in **R** is the same as that for making scatterplots. Should we wish, for example, to see the graph of $\sin x, 0 \leq x \leq 2\pi$, we might first create a list, probably evenly-spaced, of x-values where points will be plotted. We may then obtain our plot as before, using a call to the **sin()** function to create the corresponding y-list. (Note that, if x is a vector (list of values), **R** responds to $\sin(x)$ with another vector of values. This is true for other **R** functions as well, such as $\text{sqrt}(x)$, x^3, $3 * x - 1/x$, etc.)

> x <– seq(0, 2*pi, 0.1)
> plot(x, sin(x), type="l")

Should you wish to plot any of the density functions we have discussed, all you need to know is how to refer to the desired pdf in place of **sin()**. In **R**, the convention is to put the letter '**d**' in front of the distribution type: dnorm(x, μ, σ), for example, in the case of a normal distribution.

As with any display of data, the work begins after the scatterplot has been produced. The display is just an aid to understand what the data tells us. Key features we look for/at:

1. **Association**: Do large explanatory values tend to go with large response ones and small with small (positive association)? Is the situation reversed—i.e., small explanatory values go with large response values and large with small (negative association)?

2. **Form**: Is there a clear shape to the data? Is it linear? Can it be made linear via some sort of transformation on the variables (i.e., $\text{sqrt}(y)$ vs. x)?

3. **Strength**: Are deviations from the pattern quite small (strong)? Is there a good deal of variation away from the pattern (weak)?

Quantile plots

A *quantile plot* (also called a *probability plot*) can be thought of as a particular type of scatterplot. It serves a particular purpose, that of determining whether the sampled values of a (single) quantitative variable fit a particular type of distribution.

To explore this idea, let us consider whether the univariate data of Example 2.3 (p. 62) might be normally distributed. This data set has 20 observations, and so each one represents 5% of the overall sample. The sample has mean $\bar{x} = 965.0$ and standard deviation $s = 178.30$. If we wished to check whether the data behaves the way we would expect a sample from a normal distribution with $\mu = 965.0$ and $\sigma = 178.30$, we might convert each of these data points to a z-score in the usual way $z = (x - \mu)/\sigma$:

Value	612	623	666	744	883	898	964	970	983	1003
z-score in N(965.0, 178.30)	-1.98	-1.92	-1.68	-1.24	-0.46	-0.38	-0.01	0.03	0.10	0.21
Value	1016	1022	1029	1058	1085	1088	1122	1135	1197	1201
Quantile (z-score)	0.29	0.32	0.36	0.52	0.67	0.69	0.88	0.95	1.30	1.32

We might then look at the table of standard normal values to get the quantiles of N(0, 1) in increments of 5%, perhaps starting with the 0.025th quantile (2.5th percentile):

position	0.025	0.075	0.125	0.175	0.225	0.275	0.325	0.375	0.425	0.475
Quantile in N(0, 1)	-1.96	-1.44	-1.15	-0.93	-0.76	-0.60	-0.45	-0.32	-0.19	-0.06
position	0.525	0.575	0.625	0.675	0.725	0.775	0.825	0.875	0.925	0.975
Quantile in N(0, 1)	0.06	0.19	0.32	0.45	0.60	0.76	0.93	1.15	1.44	1.96

If the z-scores from the two tables matched in all 20 cases, then a scatterplot of our 2 sets of z-scores would lie precisely along the line $y = x$, and we could feel pretty good that this data was a sample from N(965.0, 178.30). It is rare, of course, that they match exactly and, even if the fit is poor, there might be other normal distributions (centered elsewhere, with wider or thinner spreads) for which the match with our data would be better.

To embark on seeing whether the above process, applied with many other choices of μ and σ, yields a scatterplot close to the line $y = x$ would be a hopeless undertaking. Fortunately, in one fell swoop it is possible to determine whether the data seems to suggest itself as a sample from *any* normal distribution. All that is really necessary is to make a scatterplot of the data itself against the appropriate quantiles (corresponding to the sample size n) from the standard normal distribution. If the points have a strong linear relationship, they are approximately normal (some μ and σ). Even more fortunately, software packages such as **R** have the entire process programmed into them, requiring only that we supply the data to be tested. To get a quantile plot in **R** testing the plausability that the data in soil.pH is distributed normally, we type the first of the commands below. The second command would not produce any plot itself, but it adds the best fit line to pass through the points drawn by **qqnorm()**.

```
> qqnorm(soil.pH)
> qqline(soil.pH)
```

One may sometimes wish to determine if sample data fits some other type of distribution. The process would be similar to that described above. **R** has the **qqplot()** command for these more general situations.

Correlation and regression

When a relationship between variables appears to be linear, the *correlation* provides a numeric measure of the *strength* of that relationship. While we will learn the details of the correlation in Section 3.2, for now it suffices to say that the correlation is always a number $-1 \leq r \leq 1$, r-values close to (-1) or $(+1)$ indicate a strong relationship, the closer r is to zero the less strength in the linear relationship, and negative/positive values go with negative/positive associations (see above). In **R**, one may obtain the correlation between two quantitative variables in a data frame with the **cor()** command. Type

```
> cor(soil.pH, crown.dieback)
```

to see the correlation between the two variables of Example 3.2. The formula for r, given on p. 106, allows us to calculate r for any pair of quantitative variables regardless of whether the relationship between them is linear. Nevertheless, as we shall see, there are instances of very strong relationships (albeit nonlinear ones) in which r may be near or equal to zero.

In Section 3.3 we will learn the theoretical aspects behind *simple linear regression*, the name given to determining the line that "best fits" a seemingly linear relationship between two quantitative variables. In **R**, the **lm()** command will give us the slope and intercept of this line. In the case of data for Example 3.2, type

```
> lm(crown.dieback ~ soil.pH)
```

In this case the slope is (-5.792) and intercept is (31.040). It would be nice to view the scatterplot of the data and this line together on the same graph. One might use the commands

```
> plot(soil.pH, crown.dieback, pch=19, ecx=0.5)
> plot(seq(3, 5.5, 0.1), 31.04 - 5.792*seq(3, 5.5, 0.1), type="l")
```

but, unfortunately, **R** opens only one plot window at a time, and the second command will produce a plot that replaces, instead of adds to, the first. We should go ahead with the first of these commands, and then run a low-level plotting command that adds elements to the original. The various low-level commands include **points()** (to add points to an existing graph), **lines()** (to add a curve), **text()** (to add text at a specific location), **abline()** (to add a straight line), **polygon()** (to add a polygon), **legend()** (to add a box indicating the meaning of symbols/curve types), **title()** (to add a title, subtitle, and/or labels along x and y-axes) and **axis()** (to add an axis). Several of these may be used to add our regression line, but perhaps the easiest is

```
> abline(lm(crown.dieback ~ soil.pH))
```

In fact, if you immediately precede the **abline()** call with the **lm** command, you can use the **.Last.value** variable.

```
> plot(soil.pH, crown.dieback, pch=19, ecx=0.5)
> lm(crown.dieback ~ soil.pH)
> abline(.Last.value)
```

Other bivariate displays

So long as you have an internet connection, you may download the data frame that contains the information you and members of statistics classes collected with your survey. Type

```
> surveyInfo <- read.table(
    "http://www.calvin.edu/~scofield/courses/stats/data/surveys/surveyDataForLab.data",
    sep=',', header=T)
> attach(surveyInfo)
```

With this data set, let's look at several other types of displays. For instance, we may wish to look at the number of cds owned (a quantitative variable) as it relates to gender. *Side-by-side boxplots* can be helpful for comparisons of quantitative variables displayed separately according to the values of a categorical variable. (Since "gender" has only two values, the screen will not be filled up with boxplots.) Type

> boxplot(cds ~ gender)

Write a command that allows you to see if there is much difference in the number of cds owned based upon the type of region in which one lives.

One often explores relationships between two categorical variables using a *two-way table*. Look at the output of

> table(region, smoker)

There are three columns of numbers. I believe the column of zeros is there because some people did not respond to one or both of the questions "Do you smoke?" and "What kind of environment do you come from?" The other columns give counts of all combinations of answers to these two questions. For instance, there are 49 respondents who are nonsmokers and who come from a rural area.

Exercises

For these exercises, you will be working with the data from your survey. (See above for how to load that data.) It is understandable when some respondents are reluctant to fill in answers to all of the questions, as has happened here. This means that some variables have missing entries, which can be a bit of a pain when using some of **R**'s commands. For instance, a number of commands produce succinct output for quantitative variables—**min()**, **max()**, **summary()**, **fivenum()**, **mean()**, **median()**, and **sd()**, to mention a few. You should look at the help on these commands (if you have not discovered them already) to see what they do, and try them out on the variable "cds", which has 5 missing entries. Certain of them should work fine, while others will complain about "missing observations" and/or give no meaningful output. For these latter types, try adding the option "na.rm = T", as in

> max(cds, na.rm=T)

Then, using **R** on this survey dataset, answer the following questions. You should type up your answers, including graphs and/or output from **R** to support your claims.

1. Look at a histogram of the random numbers supplied by respondents. Recall that these random numbers were supposed to be between 0 and 20, so the ones higher than 20 should be filtered out. If the number selection were truly random, what kind of distribution would you expect these to tend to in the long run? Does this histogram support the ideas that such choices are truly random?

2. Is there a difference in the distributions of male and female respondents in the amount paid for their last haircuts?

3. Generate two two-way tables, one showing how "selfhandedness" relates to "momhandedness", the other showing how "selfhandedness" relates to "dadhandedness". Does it appear that a respondent is left-handed more often with a left-handed mother or with a left-handed father? Support your answer.

4. Generate scatterplots of every combination of the following quantitative variables:

   height, pulse, hourssleep, gpa.

   Do any of them appear to be strongly linear? Back up your answer with correlation values.