

Lab 1

Psychology 310

Instructions. Work through the lab, saving the output as you go. If you work in Microsoft Word, you can easily copy any graph to Word via the clipboard. Numerical output may also be copied easily by highlighting, moving it to the clipboard, then copying into Word. However, you should format R output in TrueType Courier New font so that it is *monospaced*. In general, lab assignments are due within ten days after being assigned.

Preamble. Today's assignment involves course grades – getting them into R, analyzing them, rescaling them, and checking them for errors.

Assume that the professor you work with hands you the following grade sheet that he has been maintaining on a piece of paper in his office:

Last.Name	First.Name	ID	MidTerm	Final
Allison	Bartley	57743227	87	79
Zukhov	Marsha	76767802	82	88
Chou	Patricia	13248884	76	69
Fillion	Herve	19092881	100	88
Braich	Kenny	76589944	90	85
Farce	Stanley	89000347	97	68
Farce	Max	88292233	65	96
Yang	Ming-Mei	56092221	80	80
Edwards	Victoria	08911090	93	97
Campbell	Kimberly	10001010	62	58
Trudeau	Pierre	19972323	55	59

1. The professor wants these grades analyzed. Your first step is to get them into R. One of the things you discover quickly in graduate school is that getting the data into a statistical program can sometimes be more trouble than it should be. R is extraordinarily powerful, but it can sometimes be difficult to figure out, because it is not menu-driven.

There are several ways of organizing data in R, but perhaps the most effective and the most common is the “data frame,” which is essentially a matrix with columns that can be different variable types.

R distinguishes between `character` and `numeric` variables, and you will quickly become familiar with them.

Here is a hint. Some universities have ID numbers with leading zeros. If you are at such a university, and you enter the ID as a numeric variable, your statistical program will remove the leading zero. For example, if the student's ID number is 00932883 in a system where all ID numbers have 8

digits, you might see 932883 in your data file. In this case, you could make your ID number a **character** variable and solve the problem, because now the software would treat the ID number as a string of characters, and maintain all the characters you enter.

There are two ways you could get the information into R relatively quickly. One way is to create a data frame and enter the numbers using the R edit facility. The other way is to copy the numbers to the clipboard (if they are in a PDF file, at least), paste them to the windows notepad, and save the resulting text file, then import it back into R.

You can try doing that in this case. If you have a problem, you can download the file *Grades.txt* from the website and use that file.

Create a working directory called *Lab01* and switch R to that working directory, using the *File->ChangeDir* command from the menu. Make sure the *Grades.txt* file is in that directory.

Import the file into R using the following syntax.

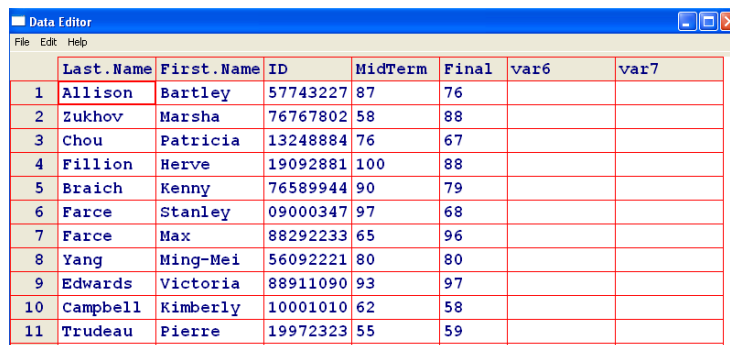
```
> grades <- read.table("Grades.txt",header=TRUE,  
+ colClasses=c("character","character","character",  
+ "numeric","numeric"))
```

The filename comes first *in quotes*, followed by the statement **header=TRUE**, which tells R that the first row of data is actually a list of variable names. Then we provided a vector of column classes, defining which kind of variable is in each column. Notice that we forced the variable ID to be a character variable.

To verify that your data have successfully been entered, type the following command:

```
> fix(grades)
```

This will open up an editor window.



	Last.Name	First.Name	ID	MidTerm	Final	var6	var7
1	Allison	Bartley	57743227	87	76		
2	Zukhov	Marsha	76767802	58	88		
3	Chou	Patricia	13248884	76	67		
4	Fillion	Herve	19092881	100	88		
5	Braich	Kenny	76589944	90	79		
6	Farce	Stanley	09000347	97	68		
7	Farce	Max	88292233	65	96		
8	Yang	Ming-Mei	56092221	80	80		
9	Edwards	Victoria	88911090	93	97		
10	Campbell	Kimberly	10001010	62	58		
11	Trudeau	Pierre	19972323	55	59		

If you just want to view data without risking any change, define the following function

```
> view <- function(x) {  
+ on.exit(sink())  
+ invisible(suppressWarnings(edit(x))) }  
}
```

Then type

```
> view(grades)
```

to view the file without changing it.

Now, add one line of data to the data frame

```
Wasserman Iggy 23148878 80 82
```

2. *Finding an error.* Congratulations! Your data frame is now ready to analyze. Check your data frame using the `view` function, and see if in fact you can see the additional line.

Next, let's call for a quick numerical summary of the grades with the following command:

```
> summary(grades)
```

```
      Last.Name      First.Name      ID      MidTerm
Length:12      Length:12      Length:12      Min.   : 55.00
Class :character Class :character Class :character 1st Qu.: 73.25
Mode  :character Mode  :character Mode  :character Median : 81.00
                                           Mean  : 80.58
                                           3rd Qu.: 90.75
                                           Max.   :100.00

      Final
Min.   :58.00
1st Qu.:68.75
Median :81.00
Mean   :79.08
3rd Qu.:88.00
Max.   :97.00
```

You can see that the performance on the exams was rather similar. You can get graphical representations of that in several ways. You can also, of course, call for summary statistics of the variables. For example, suppose you wanted to know the mean of the `MidTerm` exam grades. Note: Unless you have attached the data frame, you will need to tell the `mean` command the name of the data frame where the variable `MidTerm` is located, using the following syntax.

```
> mean(grades$MidTerm)
```

```
[1] 80.58333
```

Generally, it is better to `attach` the file. This makes all variable names in this file immediately available for analysis.

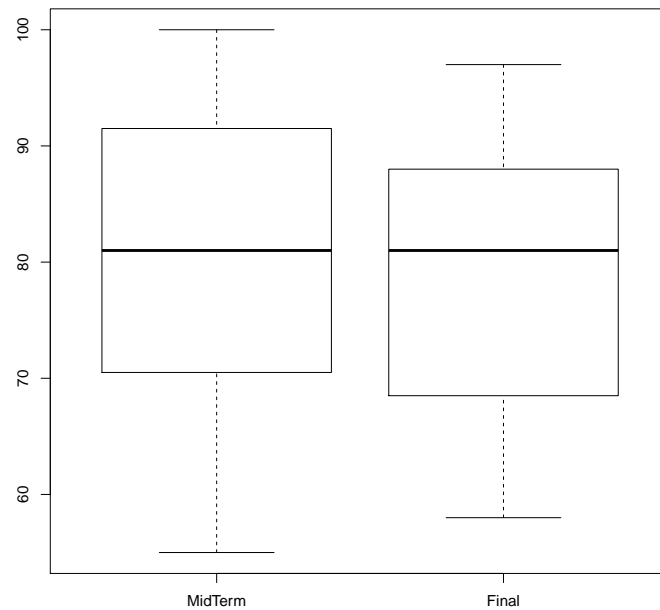
```
> attach(grades)
```

```
> mean(MidTerm)
```

```
[1] 80.58333
```

Using the help file facility in R, find out how to duplicate the following side-by-side boxplot of the `MidTerm` and `Final` grades, with the correct

axis labels as shown. Read about boxplots in Glass and Hopkins or on the internet. Which exam showed more variability?



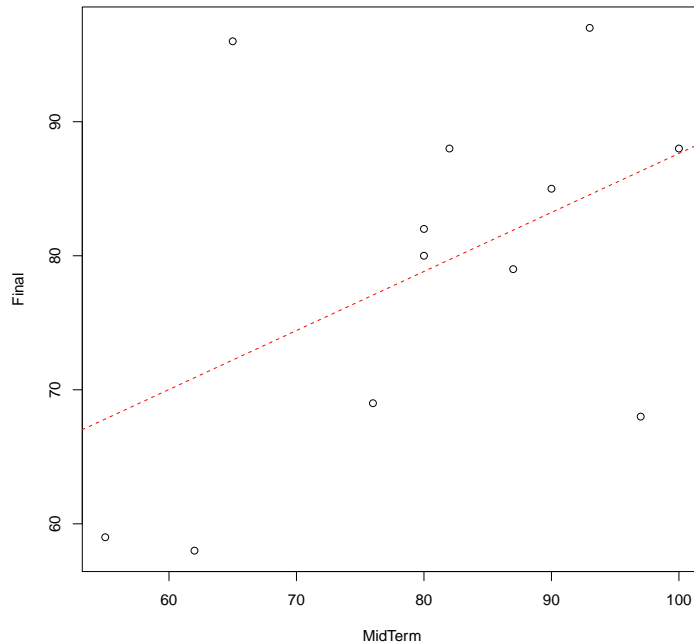
We can also examine the distributions of values in a variety of ways. One way is to look at the joint distribution of MidTerm and Final Grades with a scatterplot. You can plot the regression line, and get its slope and intercept, from the following commands.

```
> fit <- lm (Final ~ MidTerm)
> fit
```

```
Call:
lm(formula = Final ~ MidTerm)
```

```
Coefficients:
(Intercept)      MidTerm
   43.5983      0.4404
```

```
> plot(MidTerm,Final)
> abline(fit,lty=2, col='red')
```



If you look carefully at the scatterplot, you can see two points that are fairly far removed from the others. They are at the upper left and lower right of the plot. Who are they? In this case, it is pretty easy to tell because there are very few data points in the file. However, R provides a special function to help you identify points on a scatterplot.

Enter the following command:

```
> identify(MidTerm,Final,plot=TRUE,labels=Last.Name)
```

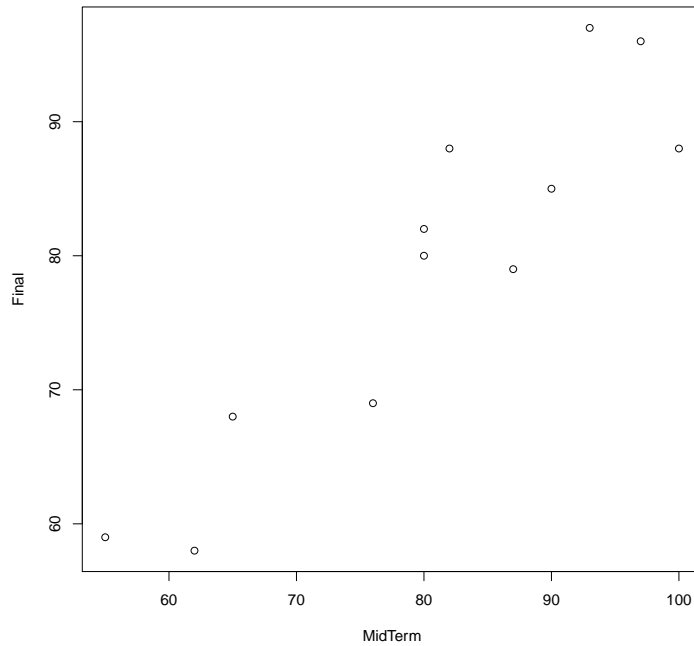
Move the cross-hairs over each of the two outlier points and click on them. You will see the last names of the two individuals represented in the points appear.

See something interesting? Looks like there was almost certainly a recording error. What do you think happened?

The problem is, who made the error, and who has the wrong data? Let's assume that the error was made by the professor when recording the data on the second midterm for these two individuals. Fix the error using the `fix` function discussed earlier.

Then redo the scatterplot. It should look like the one below.

```
> plot(MidTerm,Final)
```



2. *Sorting*. Now let's sort the grades. We want to sort by ascending order. In large classes, I have on one or two occasions had individuals with identical first *and* last names. So sorting should be done on `Last.Name,First.Name`, and ID in that order.

```
> sorted.grades <- grades[order>Last.Name,First.Name,ID),]
> sorted.grades
```

	Last.Name	First.Name	ID	MidTerm	Final
1	Allison	Bartley	57743227	87	79
5	Braich	Kenny	76589944	90	85
10	Campbell	Kimberly	10001010	62	58
3	Chou	Patricia	13248884	76	69
9	Edwards	Victoria	88911090	93	97
7	Farce	Max	88292233	65	68
6	Farce	Stanley	09000347	97	96
4	Fillion	Herve	19092881	100	88
11	Trudeau	Pierre	19972323	55	59
12	Wasserman	Iggy	23148878	80	82
8	Yang	Ming-Mei	56092221	80	80
2	Zukhov	Marsha	76767802	82	88

It worked!

3. *Z-Scoring*. You can add new variables to the data frame in numerous ways. If you are transforming data already in the file, you can create the new variable by referencing it in the transformation formula. However, you must reference it with the data frame name included. For example, to create a new variable that is the sum of the midterm and the final grades, you would issue the command `sorted.grades$sum = MidTerm + Final`

Here's what I want you to do next:

- (a) Create two new variables called `Z.Midterm` and `Z.Final` that contain *z*-scores for the two variables.
- (b) Then, compute the mean and variance of these two new variables to make sure they are actually in *z*-score form.
- (c) Next, compute the correlation between these two variables.
- (d) Then, compute their covariance.
- (e) Next, create a variable called `z.grade` that is compute with the formula
$$\text{z.grade} = (1/3)*\text{z.MidTerm} + (2/3)*\text{z.Final}$$
- (f) Create a single formula in R that will transform these `z.grade` scores to have a mean of 78 and a standard deviation of 12, and store these final grades in a variable called `Course.Grade`.
- (g) Verify that these grades have a mean of 78 and a standard deviation of 12.
- (h) Finally, go to www.statmethods.net, and figure out how to save your data frame to a file, and save it.

Remember, save all your input, output, and graphics files, and your final data file.