Stat 407 Lab 1 Introduction to R
Due: Wednesday, August 22, 2012, in class. Hand in one solution per group.

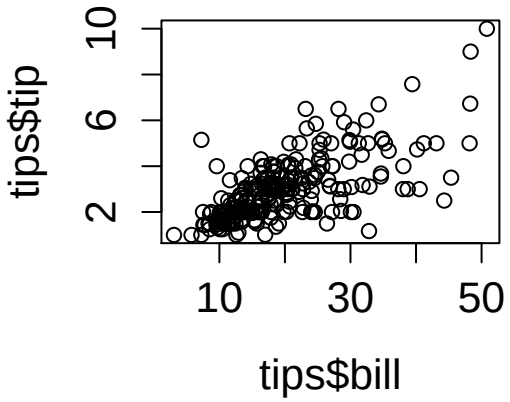
Purpose: To learn about the R language for data analysis. At the end of this lab, you will be able to start R, read in data, conduct simple calculations, install a package, and write a function. You will also learn to edit code in the editor window, run it, and save a copy of your code.

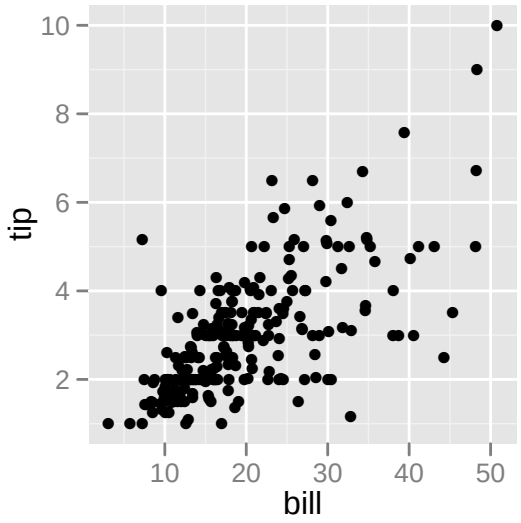
Let's get started! (These symbols at the start of a line have meaing for understanding the example code: ">" means line of R code, "+" line extended from previous line, "#" is a comment in the code, the rest is the output from R.)

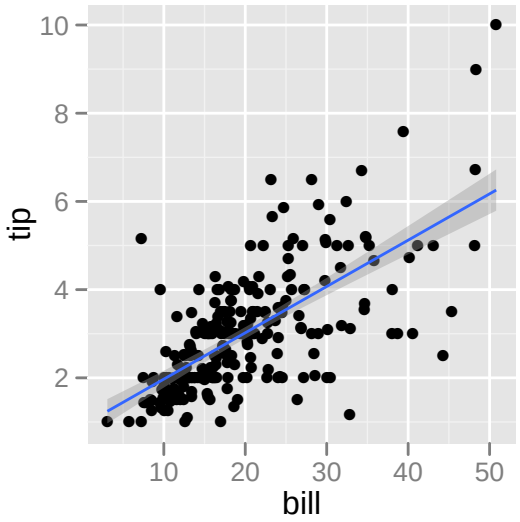
Demo: Let's use R to look at some data The tips data contains records on tips in a restaurant, collected by a waiter over a couple of months.

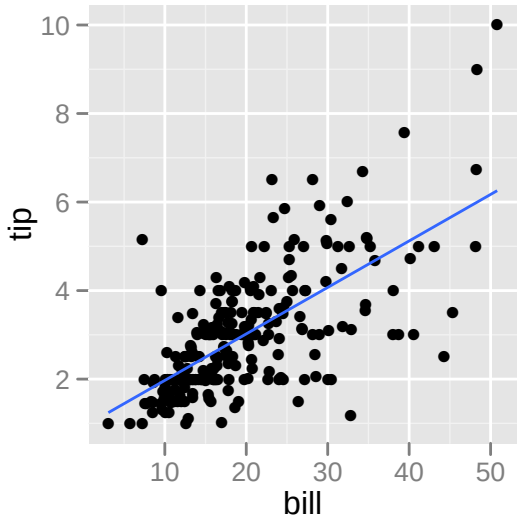
```
# Read in the data, and take a quick peek at it
> tips <- read.csv(file.choose()) # <- is the assignment operater, think of it as =, and you ca
> tips
# Whoa! That's a whole lotta lines in the console.
# Let's just look at the first few rows
> head(tips)
# Hmm... the first column looks awkward, why is it named X,
# and the same as the row number?
# Let's read it in by setting that column to be row labels
> tips <- read.csv(file.choose(), row.names=1)
> head(tips)
        bill  tip    sex smoker day   time size
1      16.99 1.01 Female     No Sun Dinner    2
2      10.34 1.66   Male     No Sun Dinner    3
3      21.01 3.50   Male     No Sun Dinner    3
4      23.68 3.31   Male     No Sun Dinner    2
5      24.59 3.61 Female     No Sun Dinner    4
6      25.29 4.71   Male     No Sun Dinner    4
# Much better!
# How big is tips? What types of variables are in the columns?
> str(tips)
'data.frame': 244 obs. of  7 variables:
 $ bill  : num  17 10.3 21 23.7 24.6 ...
 $ tip   : num  1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...
 $ sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...
 $ smoker: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ day   : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ time  : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...
 $ size  : int  2 3 3 2 4 4 2 4 2 2 ...
# It looks like I have 3 numerical variables and 4 character variable
# Lets get a summary of these columns of my data set
```

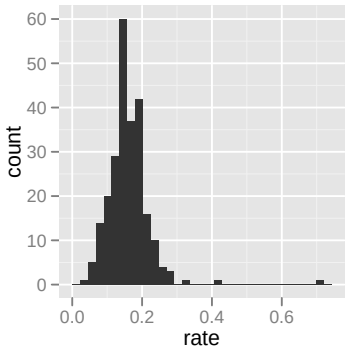
```
> summary(tips)
      bill             tip             sex      smoker      day         time          size
 Min.   : 3.07   Min.   : 1.000   Female: 87   No :151   Fri :19   Dinner:176   Min.   :1.00
 1st Qu.:13.35   1st Qu.: 2.000   Male  :157   Yes: 93   Sat :87   Lunch : 68   1st Qu.:2.00
 Median :17.80   Median : 2.900                          Sun :76                Median :2.00
 Mean   :19.79   Mean   : 2.998                          Thur:62                Mean   :2.57
 3rd Qu.:24.13   3rd Qu.: 3.562                                                 3rd Qu.:3.00
 Max.   :50.81   Max.   :10.000                                                 Max.   :6.00
# Look at the way that cost of the total bill might effect the tip value
#   using base R function plot()
> plot(tips$bill, tips$tip)
# Uggh, awful!
# Let's make the plot in style. Load the ggplot2 library.
> library(ggplot2)
> qplot(bill, tip, data=tips)
# What's the relationship? Fit a linear model.
> lm(tips$tip ~ tips$total_bill)

Call:
lm(formula = tips$tip ~ tips$bill)

Coefficients:
(Intercept)    tips$bill
     0.9203       0.1050
# Let's plot the model on the data
> qplot(bill, tip, data=tips, geom=c("point", "smooth"), method="lm")
# I don't like the grey error bar.
> qplot(bill, tip, data=tips, geom=c("point", "smooth"), method="lm", se=F)
# I am really more interested in tip as a % though
# Create a new variable using two of the existing variable
> tips$rate <- tips$tip / tips$bill
> summary(tips$rate)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.03564 0.12910 0.15480 0.16080 0.19150 0.71030
> qplot(rate, data=tips, geom="histogram")
# Let's see if there is a difference by gender
> qplot(rate, data=tips, geom="histogram", facets=sex~., fill=sex)
# Not much!
# Oh, by the way, we could have read the data directly from the internet
> tips <- read.csv("http://www.ggobi.org/book/data/tips.csv")
> head(tips)
```

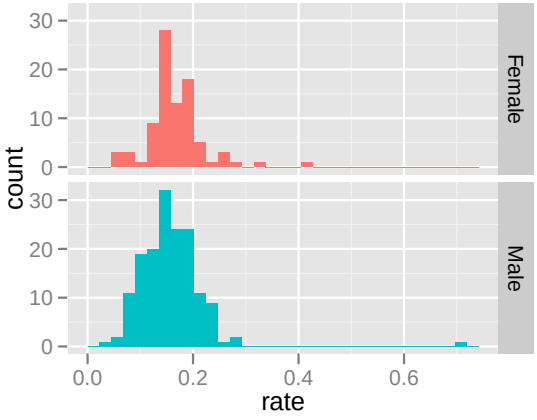

tips$tip
2
6
10
10
30
50
tips$bill

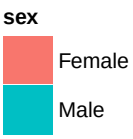

tip
2
4
6
8
10
10
20
30
40
50
bill


tip
2
4
6
8
10
10
20
30
40
50
bill


tip
2
4
6
8
10
10
20
30
40
50
bill


count
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
rate


count
0
10
20
30
0.0
0.2
0.4
0.6
rate
Female
Male


sex
Female
Male

Your turn! Open a Word document, and add your answers to these questions, including your code, results and plots. You'll need to print this off and turn one copy in from your working group by Wednesday's class.

Open RStudio. And also type your code into a new source Window file. You can use the code in the 1-Tips.R file as an example. Highlighting a line of code, and going to the menu "Code", and selecting "Run" will execute the code. You will also see a shortcut for doing this there, which will differ by operating system that you are running.

1. Read in the music data from the URL "http://www.ggobi.org/book/data/music-sub.csv".
2. How big is the data?
3. What type of variables?
4. What is the range of the variable called lvar?
5. How many different artists are in the data set? Name them, if you can.
6. How many different types of music are there?
7. Make a plot of lvar and lave coloured by the music type.

R is an overgrown calculator:

```
# Addition
> 2 + 2
[1] 4
# Subtraction
> 15 - 7
[1] 8
# Multiplication
> 109*23452
[1] 2556268
# Division
> 3/7
[1] 0.4285714
# Integer division
> 7 %/% 2
[1] 3
# Modulo operator (Remainder)
> 7 %% 2
[1] 1
# Powers
> 1.5^3
[1] 3.375
# Roots
```

```
> sqrt(25)
[1] 5
> 25^.5
[1] 5
> 25^(1/2)
[1] 5
# constant e, and powers of e
> exp(1)
[1] 2.718282
> exp(2)
[1] 7.389056
# same as
> exp(1)^2
[1] 7.389056
# logarithms
# log gives natural log
> log(exp(2))
[1] 2
# log10 gives log base 10
> log10(100)
[1] 2
# log base 2
> log2(10)
[1] 3.321928
# Can't do it for every base
> log3(10)
Error: could not find function "log3"
# gives error, use this
> log(10,base = 3)
[1] 2.095903
# Trigonometric functions
> pi
[1] 3.141593
> sin(pi)
[1] 1.224647e-16
> cos(pi)
[1] -1
> tan(pi)
[1] -1.224647e-16
# Cleaning up
> round(2.579)
[1] 3
> round(-2.579)
[1] -3
> round(-2.579, digits = 2)
```

```
[1] -2.58
> signif(2.579, 2)
[1] 2.6
> signif(23511,2)
[1] 24000
```

Your turn!

8. What is the sqrt of 10089?
9. What is the remainder when 1000001 is divided by 333?

****** When in doubt about syntax of how to use a command, look up the help information, for example *******

```
?str
?signif
```

R is a scientific calculator, and vector and matrix calculator:

```
# Basic stats
> sum(c(1, 5, 7)) # c means collect these elements together
[1] 13
> mean(c(1, 5, 7))
[1] 4.333333
> min(c(1, 5, 7))
[1] 1
# Indexing elements of a vector or matrix
# Generate a sample of size 50 from a normal distribution
> x <- rnorm(50)
> x
 [1] -0.226436014 -1.498103790  2.944261809  1.323891388 -0.872518718  0.308926695 -1.4182
 [9] -1.235539939  0.148650344  0.565105157 -0.460853689 -0.788546041 -2.337625021 -1.0426
[17] -0.442493006 -1.318793031 -0.191522992  0.575256334 -0.067072088 -0.300533095  0.9421
[25]  1.040933305 -2.700650261  0.216129746 -0.651531240  0.418958460  0.546047907  0.4465
[33]  0.607964620  0.007097171 -0.408626145 -0.372814345  2.429979631  1.181046768 -0.6036
[41]  0.683728733 -0.087632857 -1.776944087 -0.195147207 -0.179291936  0.269600814  1.2888
[49]  0.985053799 -2.457906688
# 10th element
> x[10]
[1] 0.1486503
# 10-15th elements
> x[10:15]
[1]  0.1486503  0.5651052 -0.4608537 -0.7885460 -2.3376250 -1.0426392
# every second element
```

```
> x[seq(1, 100, 2)]
 [1] -0.22643601  2.94426181 -0.87251872 -1.41820877 -1.23553994  0.56510516 -0.78854604 -
 [9] -0.44249301 -0.19152299 -0.06707209  0.94215559  1.04093331  0.21612975  0.41895846
[17]  0.60796462 -0.40862615  2.42997963 -0.60365294  0.68372873 -1.77694409 -0.17929194
[25]  0.98505380          NA          NA          NA          NA          NA          NA
[33]          NA          NA          NA          NA          NA          NA          NA
[41]          NA          NA          NA          NA          NA          NA          NA
[49]          NA          NA
# Generate another factor variable
> y <- c(rep("male", 35), rep("female", 45))
# Subset x by y
> x[y=="male"]
 [1] -0.226436014 -1.498103790  2.944261809  1.323891388 -0.872518718  0.308926695 -1.4182
 [9] -1.235539939  0.148650344  0.565105157 -0.460853689 -0.788546041 -2.337625021 -1.0426
[17] -0.442493006 -1.318793031 -0.191522992  0.575256334 -0.067072088 -0.300533095  0.9421
[25]  1.040933305 -2.700650261  0.216129746 -0.651531240  0.418958460  0.546047907  0.4465
[33]  0.607964620  0.007097171 -0.408626145
# Find the sample quantiles/percentiles of x
> quantile(x, seq(0, 1, 0.1))
         0%         10%         20%         30%         40%         50%         60%
-2.70065026 -1.42619827 -0.80534058 -0.44800121 -0.25607485 -0.07735247  0.23751817  0.551
        90%        100%
 1.19182390  2.94426181
```

Your turn!

10. For the tips data, find the mean of tiprate for a party of size 2, and then 3.

11. What is the 40th and 70th percentiles of bill?

Using the package plyr to rearrange data, and make calculations on subsets

```
> library(plyr)
# Calculate means by gender
> tips.new <- ddply(tips[c("bill","tip","sex","rate")], "sex", mean)
> tips.new
# In a way that it doesn't complain
> tips.new <- ddply(tips, "sex", summarize, mean.bill=mean(bill),
     mean.tip=mean(tip),
     mean.rate=mean(rate))
```

Your turn!

12. Find the mean of the variables in the music data by type of music.

13. Create a new variable in the music data which is log, base 10, of the lvar variable, and summarize these values.

Writing your own functions. The R language allows the user to create objects of mode function. These are true R functions that are stored in a special internal form and may be used in further expressions and so on. In the process, the language gains enormously in power, convenience and elegance, and learning to write useful functions is one of the main ways to make your use of R comfortable and productive. It should be emphasized that most of the functions supplied as part of the R system, such as mean(), sd(), and so on, are themselves written in R and thus do not differ materially from user written functions.

```
# This is a function that calculates a 95\% confidence interval
# for a vector of data values.
> ci95 <- function(x) {
+   xmn <- mean(x)
+   xsd <- sd(x)
+   qt <- qt(0.975, length(x))
+   return(data.frame(lower=xmn-qt*xsd/sqrt(length(x)),
+     upper=xmn+qt*xsd/sqrt(length(x))))
+ }
> ci95(tips$rate)
      lower     upper
1 0.1531014 0.1685037
# But this throws an error if we try to run it on anything but a numeric vector
> ci95(tips$day)
  lower upper
1    NA    NA
Warning messages:
1: In mean.default(x) : argument is not numeric or logical: returning NA
2: In var(as.vector(x), na.rm = na.rm) : NAs introduced by coercion
# An improved function that has safety checks
> ci95 <- function(x) {
+   if (is.numeric(x)) {
+     xmn <- mean(x)
+     xsd <- sd(x)
+     qt <- qt(0.975, length(x))
+     return(data.frame(lower=xmn-qt*xsd/sqrt(length(x)),
+       upper=xmn+qt*xsd/sqrt(length(x))))
+   }
+   else
+     cat("x needs to be a numeric vector\n")
+ }
> ci95(tips$rate)
      lower     upper
1 0.1531014 0.1685037
> ci95(tips$day)
x needs to be a numeric vector
# Allowing users to change the % value
> ci95 <- function(x, pct=0.95) {
```

```
+   if (is.numeric(x)) {
+     xmn <- mean(x)
+     xsd <- sd(x)
+     qt <- qt((1-pct)/2, length(x))
+     return(data.frame(lower=xmn-qt*xsd/sqrt(length(x)),
+       upper=xmn+qt*xsd/sqrt(length(x))))
+   }
+   else
+     cat("x needs to be a numeric vector\n")
+ }
> ci95(tips$rate)
      lower     upper
1 0.1685037 0.1531014
> ci95(tips$rate, pct=0.8)
      lower     upper
1 0.1658267 0.1557784
```

Your turn!

14. Write a function that returns values in a vector that are larger than the 80th percentile, sorted from lowest to highest. Use it to compute the top 20% of values for tiprate.

15. Modify your function so that it takes the the percentile value as an additional argument, and checks that the input is numeric.

Installing packages. There are hundreds of contributed packages for R, written by many different authors. Some of these packages implement specialized statistical methods, others give access to data or hardware, and others are designed to complement textbooks.

Using the "Tools" menu find and install the packages. Choose an R mirror close to you, such as the one at Iowa State University. OR you can also use the function `install.packages()` on the R command line to install a package.

Finally TURN OFF THE MONITOR REFLECTING THE INSTRUCTOR'S COMPUTER. You will need to do this every week.