Getting Started in R

Oct. 2019 STAT 206

What is R?

For now, we will use R as a fancy calculator. There are a lot of formulas in this class that would be tedious to do by hand or with a simple calculator. For instance, calculating the sample mean of a list of 100 numbers would take a while to do by hand, but seconds in R. Another convenient feature of R is its plotting capabilities. Using R one can produce clean looking plots very efficiently. Later this quarter, we will learn how to use various built in functions (such as *lm*) in R to perform data analysis.

Installation

- Installing R:
  1. Go to r-project.org.
  2. On the left side of the page, click the "CRAN" link.
  3. Scroll down to the USA section, and click a link (say, Berkeley).
  4. Click the download link for your operating system (Windows, Mac, or Linux).
  5. Follow the instructions to download the latest version of R.
- Installing RStudio (An integrated development environment for R): Go to www.rstudio.com and follow the instructions.

Using R

0.0.1 Getting Help

help function, R blog, etc.
Question mark + name of function

```
# find the detailed document of lm function
> ?lm
```

Assignment and Arithmetic

Assignment of variables means that in R you can create your own variables, vectors, matrices, etc. R can perform basic arithmetic operations with numbers or declared variables. Here are some examples:

```
#define variables x and y
> x = 4
> y = 3
> z <- 2

#perform arithmetic on variables as well as numbers
> x * y - 4

#output
[1] 8
```

In most cases, '=' and '<-' can be used interchangeably.
R can perform arithmetic on numbers, but it is really designed to work with ***vectors***. Vectors are just lists of numbers, and in R we can define and perform operations on them. To create a vector, we use the concatenate function c():

```
#use concatenate function to make a 3-dimension vector called 'myvec'
> myvec = c(4,5,6)

#display myvec
> myvec
[1] 4 5 6
```

Once we have a vector, we can perform arithmetic on the entire vector at once. We can also add two vectors together.

```
#perform arithmetic on entire vectors at once
> myvec + 3
[1] 7 8 9
> myvec * 2
[1]  8 10 12
> myvec^2
[1] 16 25 36

#create another vector called 'myvec2'
```

```
> myvec2 = c(1,2,1)

#add my vectors together
> myvec + myvec2
[1] 5 7 7
Caution: if two vectors have different length,
R will calculate the result by recircling the shorter one.
> myvec3 = c(1,2,3,4,5,6)
> myvec3+myvec
[1]  5  7  9  8 10 12
> myvec3 = c(1,2,3,4)
> myvec3+myvec
[1] 5 7 9 8
here you will get a warning message,
because the length of longer one is not multiple of the shorter one.
```

There are built in functions that we can perform on a vector as well.

```
#sum the elements of a vector
> sum(myvec)
[1] 15

#take the product of all elements of a vector
> prod(myvec)
[1] 120

#calculate the sample mean of a vector
> mean(myvec)
[1] 5

#calculate the median of a vector
> median(myvec)
[1] 5

#calculate the sample standard deviation of a vector
> sd(myvec)
[1] 1

#calculate the sample variance of a vector
```

```
> var(myvec)
[1] 1

#get the length of a vector
> length(myvec)
[1] 3
```

As an example, consider the toy example in Lecture 2.

```
#define vectors X and Y
> X = c(1.86,.22,3.55,3.29,1.25)
> Y = c(3.34,1.79,5.66,5.83,4.74)

#Xbar is the sample mean of X
> Xbar = mean(X)
> Xbar
[1] 2.034

#similarly for Y
> Ybar = mean(Y)
> Ybar
[1] 4.272

#more advanced calculations (^ means exponent)
> sum((X - Xbar)^2)
[1] 7.81132
> sum((X - Xbar)*(Y - Ybar))
[1] 8.35866
```

Here we have everything calculated, so we can easily obtain

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} = \frac{8.35866}{7.81132} = 1.07,$$
$$\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1\overline{X} = 4.272 - 1.07(2.034) = 2.09.$$

```
> beta1 = 8.35866 / 7.81132
> beta0 = Ybar - beta1*Xbar
```

We can now use this to find the fitted values $\hat{Y}_i$ and residuals e_i:

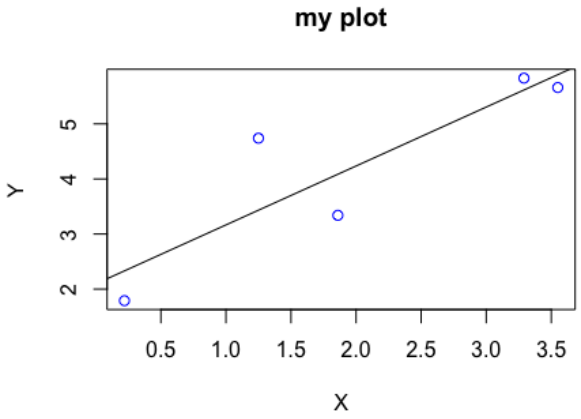
```
### fitted values
> Yhat = beta0 + beta1*X
> Yhat
[1] 4.085808 2.330893 5.894226 5.616008 3.433065

## residuals
> e=Y-Yhat
> e
[1] -0.7458078 -0.5408928 -0.2342263  0.2139920  1.3069350


### check properties of residuals:
#the summation of residuals is equal to 0 when intercept is included in regression
> sum(e)
[1] 3.108624e-15
# residuals and covariates are linearly uncorrelated.
> sum(X*e)
[1] 5.925815e-15
# residuals  and fitted values are linearly uncorrelated.
> sum(Yhat*e)
[1] 1.24345e-14
```

Plotting

Now that we have our data, and the fitted regression line, we would like to plot them both to see what we have done. The plot function in R is fairly simple. There are several options one can use when plotting, but we start with the basic ones:

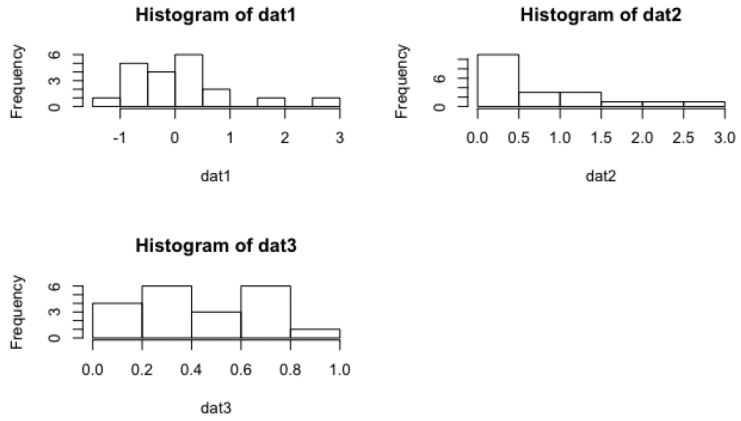
```
#main is the title of the plot, col is the color of the points
> plot(X,Y,main='my Plot',col='blue')
# for other arguments of plot, check by ?plot

#use abline to meake a line
> abline(beta0,beta1)
```

Here, the function abline() creates a line with a given slope and intercept. Here is the output for this simple example:



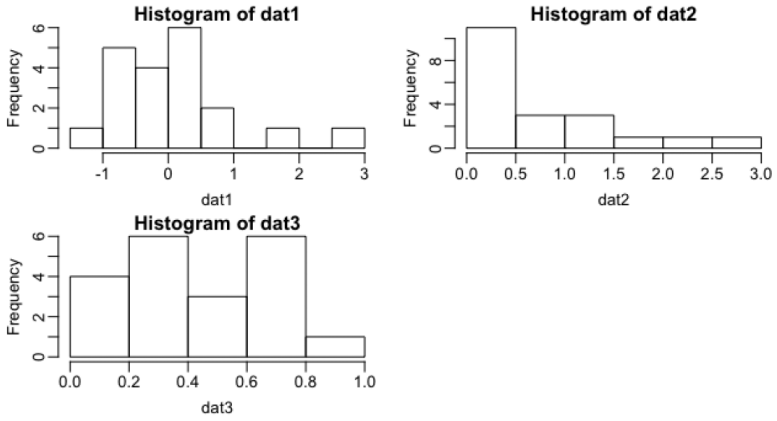
We can also show separate plots in one window by use of the *par* command:

```
par(mfrow=c(2,2))  # arrange plots in 2x2 matrix
hist(dat1)
hist(dat2)
hist(dat3)
```



we can further improve the plot by adjusting the default margin settings:

```
par(mfrow=c(2,2),mgp=c(1.8,.5,0), mar=c(3,3,1,1), oma=c(0,0,3,0), pty='m')
# for the meaning of these arguments, check in ?par
hist(dat1)
hist(dat2)
hist(dat3)
```



0.0.2 Save Workspace

save, save.image, etc.

```
# creating ".RData" in current working directory
> save.image()
```

We can also save a workspace in Rstudio by clicking "save" button in top right panel. We can load an R object by using "load" function.

```
> load(".RData")
```

Example: Inference

A bottle filling machine is set to fill bottles with soft drink to a volume of 500 ml. The actual volume is assumed to follow a normal distribution. The manufacturer believes the machine is **under-filling** bottles. A sample of 20 bottles is taken and the volume of liquid inside is measured.

Let's begin by loading the dataset (bottles.csv) into R. We can do this by the *read.table* command

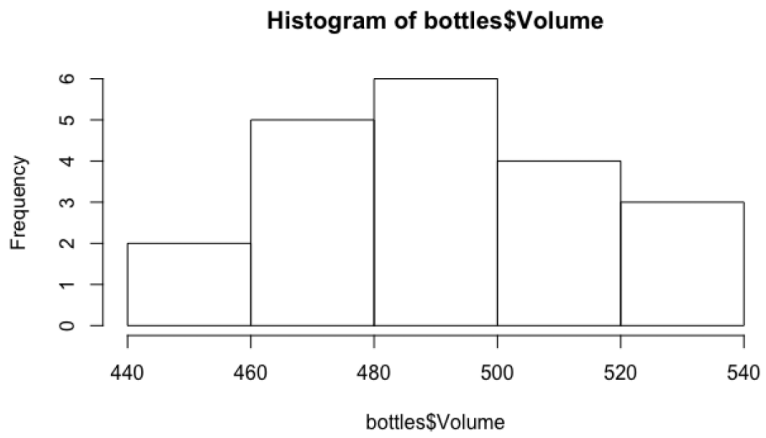
```
# set working directory
> setwd("C://Users//lf//Dropbox//TA//206FA2018//lab1") # for windows system
> setwd("/Users/lingfeicui/Dropbox/TA/206FA2018/lab1") # for mac system
# make sure the file is in working directory
> bottles = read.table(file="bottles.csv",header=TRUE)
```

We can also do this in Rstudio by the "import dataset" GUI. Either way, we should inspect the data after read-in to verify its contents.

```
> head(bottles)
# A tibble: 6 x 1
Volume
<dbl>
1 484.11
2 459.49
3 471.38
4 512.01
5 494.48
6 528.63
```

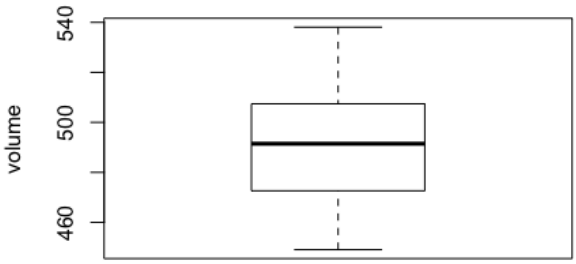
Now we can proceed to performing inference. First, we should check the assumption of normality by an exploratory plot. We get the following histogram:

```
> hist(bottles$Volume)
```



box plot, summary statistics
We also can get the following box plot by

```
> boxplot(bottles$Volume,ylab="volume")
```



Besides these exploratory plots, we can summary data by the following summary statistics.

```
> summary(bottles$Volume)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  449.1   473.3   491.5   491.6   505.1   538.1
```

We can clearly see that our sample may provide evidence of under-filling and seems to follow a normal distribution. To test the manufacturer's suspicion, we consider the following hypothesis test:

$$H_0 : \mu = 500 \; vs \; H_1 : \mu < 500$$

From previous statistics courses, we use the following test statistic

$$\frac{\overline{X} - 500}{S/\sqrt{20}} \overset{H_0}{\sim} t_{19}$$

with decision rule: reject H_0, at level α of significance, if

$$\overline{X} \leq 500 + t_{19}(\alpha) S/\sqrt{20}$$

or equivalently by use of p-values, reject if

$$P\left(t_{19} \leq \frac{\overline{x} - 500}{s/\sqrt{20}}\right) \leq \alpha.$$

(Note that $\overline{x}$ is the observed value of $\overline{X}$, similarly for s and S.).

With R, we can quickly carry out this hypothesis test. Let's test at $\alpha = 0.1$ level of signifiance. Calculating the necessary quantities:

```
> n = length(bottles$Volume)
> xbar = mean(bottles$Volume)
> shat = sd(bottles$Volume)
# Test Statistic
> sqrt(n)*(xbar - 500)/shat
[1] -1.520463

# Critical value
> qt(p=.1, df=n-1)
-1.327728

# P-value
> pt( sqrt(n)*(xbar - 500)/shat, df=n-1 )
[1] 0.07243113
```

From both calculations we can see that we can reject the null hypothesis at level 0.1 of signifiance. We can see that the lowest level of significance that the null hypothesis can be rejected at is around 0.07. It looks like the manufacturer's suspicion was correct.

Note that the function qnorm() gives the percentiles (a.k.a. quantiles) of the standard normal distribution if you provide the percentile you want. Sometimes we would like to find the percentiles for a t distribution or a Chisquare distribution or an F distribution, these are by the functions qt(), qchisq(), qf(), respectively.

```
##95% percentile of Chisquare distribution with 3 degrees of freedom
> qchisq(0.95, 3)
[1] 7.814728

### 95% percentile of t-distribution with 3 degrees of freedom
> qt(0.95, 3)
[1] 2.353363

## 95% percentile of F-distribution with (1,3) degrees of freedom
```

```
> qf(0.95, 1,3)
[1] 10.12796
```

Exercises: Calculate the 0.975 percentile for $N(0,1)$ – standard normal, $t_{(10)}$ – t-distribution with 10 degrees of freedom, $\chi^2_{(10)}$ – Chisquare distribution with 10 degrees of freedom, $F_{1,10}$ – F-distribution with $(1,10)$ degrees of freedom.