BIFX 553 - Discussion 1

Randy Johnson

January 19, 2017

BIFX 552 Review and R Tips 'n Tricks

Pretest

1) What is a p-value?
2) Write a short R script to add all numbers from 1 to 100 using a for loop.

```
s <- 0
for(i in 1:100)
  s <- s + i
```

3) Repeat problem 2 using only one line of code (i.e. without using a for loop).

```
sum(1:100)
```

```
## [1] 5050
```

4) Given the data.frame, *dat*, with the variables {*x1*, *x2*, *x3*, *y*}, write the R command to create a linear model for *y* using *x1*, *x2* and *x3* as predictors.

```
lm(y ~ x1 + x2 + x3, data = dat)
```

5) Write an R command to plot the relationship between *x1* and *y*.

```
plot(x1, y, bty = 'l')
```

6) Given the relationship between *x1* and *y* in this figure, update the R command in question 4 with a more appropriate model.

```
dat$x1sqr <- dat$x1^2 lm(y ~ x1 + x1sqr + x2 + x3)
```

7) A collaborator presents a data analysis to you. The p-value they give you is p = 0.012. Is this statistically significant?

Yes

8) After discussing the results further, you discover that this p-value represents just one of 20 tests performed in the analysis. What is the Bonferroni threshold for significance if you want the family-wise error rate to be $\alpha = 0.05$? Is the result statistically significant?

```
(thresh <- 0.05 / 20)
```

```
## [1] 0.0025
```

No

9) What assumptions are made in a simple linear regression? What R command(s) can you use to get a quick look at whether those assumptions have been violated?

- Linear relationship
- Multivariate Normality
- No/little multicollinearity
- No autocorrelation
- Homoscedasticity

- General lm diagnostics
    - plot()
    - car package
    - gvlma package

10) Explain the relationship between confidence intervals and p-values. Why are confidence intervals more useful than p-values?

Git / GitHub

- Lecture notes are located in the course repository.
- For more information on using Git from the command line, you can access a tutorial/presentation I gave last year here along with a list of comands that accompanied the presenation. There are numerous other tutorials, best practices and answers to questions on the Atlassian tutorials page.
- We will use git to share course materials and submit some homework assignments.

SourceTree

I recommend using SourceTree to manage your git repositories. It can do nearly everything you might want to do on the command line, and is a more user-friendly way to enter the world of Git.

- Pulling: repository updates must be pulled down from the git server! Git will never presume that you want to apply someone else's updates to your local copy of the repository. This could end very badly if you have uncommitted changes you are working on.

- Committing: when you have changes you want to put up on the git server for the world (or perhaps just your development team) to see, you need to commit them.

    - Choose the changes you want to commit
    - Write a description of the changes you made (make the first line *short*, followed by additional text if you would like)
    - Commit away!

- Pushing: like pulling, your commits need to be pushed to the git server, or no one will ever know about them! Git will never presume that you are ready to share your commits with others until you tell it to do so.

- Forking: if you see a repository you would like to make your own, fork it! Not only will you be able to put your own stamp on the repository, but you can issue a pull request to the owner if you have something you think they would like to implement in their version. You will be notified of any commits they push as well, which you can apply to your version (or not) as you wish.

Assignment

Create a GitHub account (if you don't already have one) and fork the course repository before class next week.

Discussion of R programming basics

- What data types does R have?

- What are the standard data structures in R? How are they indexed?

- Examples of vector calculations

```
x <- 1:10
y <- rnorm(10)
x + y
```

```
##  [1] 1.714201 2.473387 3.287186 3.739490 4.599493 5.885009 7.878700
##  [8] 8.414016 9.447221 9.516530
```

```
mat <- cbind(x, y)
apply(mat, 1, sum)
```

```
##  [1] 1.714201 2.473387 3.287186 3.739490 4.599493 5.885009 7.878700
##  [8] 8.414016 9.447221 9.516530
```

```
lst <- list(rnorm(100),
            rnorm(100, 5),
            rnorm(100, 5, 10))
sapply(lst, mean)
```

```
## [1] 0.197392 4.993251 5.322037
```

```
lapply(lst, mean)
```

```
## [[1]]
## [1] 0.197392
##
## [[2]]
## [1] 4.993251
##
## [[3]]
## [1] 5.322037
```

```
ifelse(y > 0, 1, -1)
```

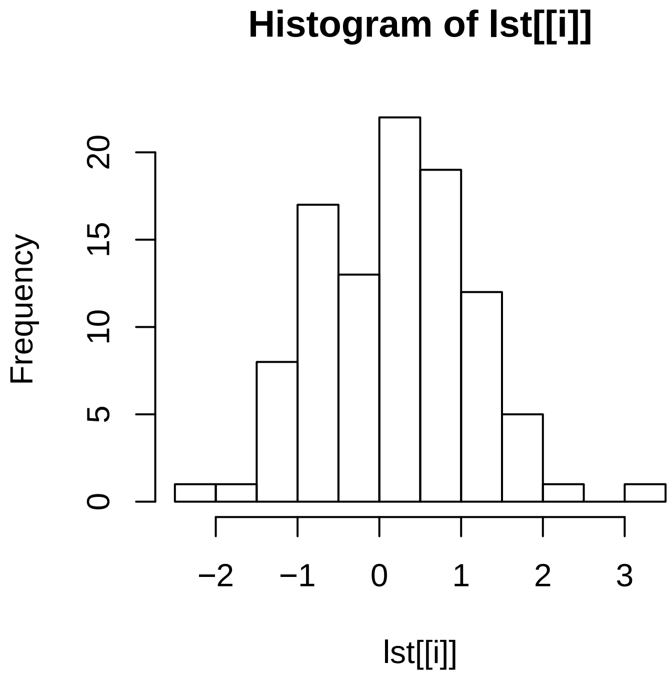
```
##  [1]  1  1  1 -1 -1 -1  1  1  1 -1
```

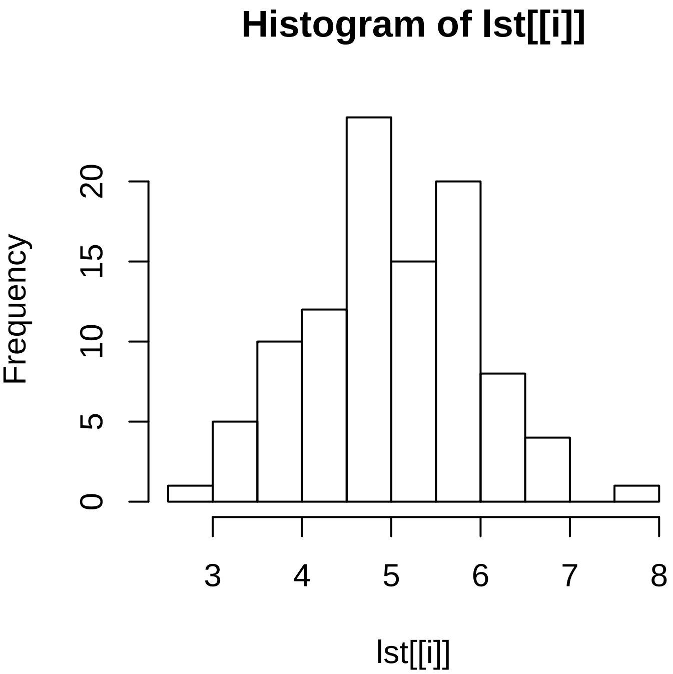
- Examples of control structures

```
test <- TRUE
if(test)
{
  print("This is executed when test is true.")
}else{
  print("This is executed when test is false.")
}
```

```
## [1] "This is executed when test is true."
```

```
for(i in 1:2)
{
  hist(lst[[i]])
}
```

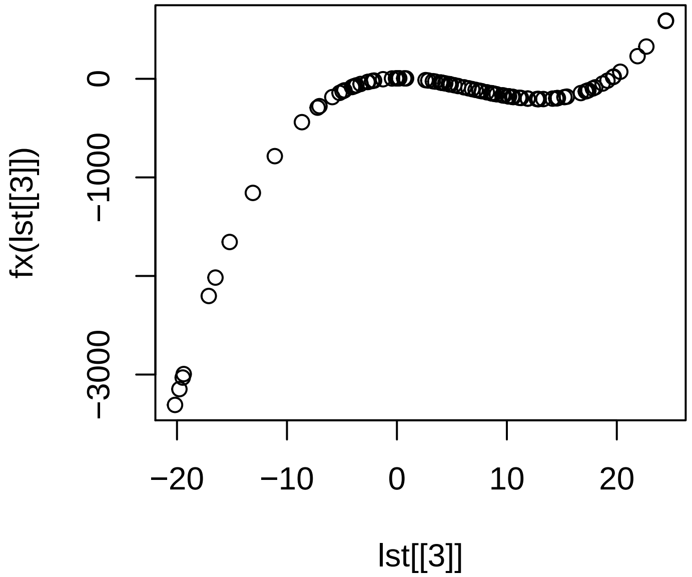




- There are other control structures, but you shouldn't need them very often (if ever).
- Functions

```
fx <- function(x)
{
  # do something here and return a value
  return(5 + 2*x - 4*x^2 + x^3/5)
}

plot(lst[[3]], fx(lst[[3]]))
```



Questions up to this point?

What would you like to learn to do in R?

Homework

- Create a GitHub account if you don't already have one and fork the class repository.
- Create a Gist with a randomly generated dataset containing 100 rows with the following variables:
  - x = rnorm(100)
  - y = x + rnorm(100)
- Create a graphic showing the regression line and confidence region using `ggplot`. Email me for hints if you get stuck.