**Coursera Data Science Course 3 Week 4**

**1. Editing text variables**

**Example:** Baltimore Traffic Data (Again!)  
//download data from the internet  
**tolower(names(cameraData))** // makes every element lower case

**Strsplit()** – good for splitting variable names  
**splitNames = strsplit(names(cameraData),”**[**\\.**](file:///\\.)**”)  
splitNames[[5]]** //gives u the name of the 5th element  
**splitNames[[6]]** // splits Location.1 into “Location” “1”

**Lists are also very easy….  
myList <- list(letters = c(“A”, “b”, “c”), numbers = 1:3, matrix(1:25, ncol = 5))** // passes letters, numbers, and a matrix  
**myList[1]** OR **myList$letters** OR **myList[[1]]** // return the same thing ( “A” “b” “c”)

**Sapply()** – apply some function to each element in a vector / list  
Parameters: X, FUN  
**splitNames[[6]][1]** // takes the first element of the vector splitNames[[6]] out **firstElement <- function(x){x[1]}** // this function takes the first element of any vector out **sapply(splitNames,firstElement)** // applies the firstElement function to the entire list of vectors

**Example:** Peer Review Data Frame (again)

**sub()** – substitute out some characters. Lets say if we want to remove underscore (\_)  
**sub(“\_”,””,names(reviews),)**

**gsub()** – takes out every character in the same variable  
**testName <- “this\_is\_a\_test”  
sub(“\_”,””,testName)** // returns thisis\_a\_test  
**gsub(“\_”,””,testName)** // returns thisisatest

**Finding Values!  
grep(“Alameda”,cameraData$intersection)** // find all the intersection that contains Alameda. It’s names all the # of elements that return true **table(grepl(“Alameda”,cameraData$intersection))** // will return the amount of values that are true and false for the rest  
**cameraData2 <- cameraData[!grepl(“Alameda”,cameraData$intersection),]** // saves all the data that DON’T contain Alameda in their intersection  
**grep(“Alameda”,cameraData$intersection,value=TRUE)** // will return the actual value (aka the name of the intersections themselves)  
**grep(“JeffStreet”,cameraData$intersection)** // return integer(0) because it doesn’t exist **length(grep(“JeffStreet”,cameraData$intersection))** // returns 0

**Other useful string functions  
library(stringr)  
nchar(“Jeffrey Leek”)** //tells you the number of characters in a string  
**substr(“Jeffrey Leek”,1,7)** //gives you the first 7 characters of the string Jeffrey Leek  
**paste(“Jeffrey”,”Leek”)** // prints the string “Jeffrey Leek”  
**paste0(“Jeffrey”,”Leek”)** // prints the string “JeffreyLeek”  
**str\_trim(“Jeff “)** // trims all the extra spaces at the end

**Things to Remember**For names of variables:  
-lower case whenever possible  
-descriptive  
-not duplicated  
-not underscored or dots or white spaces

For variables with character values  
-made into factor variables  
-descriptive

**2. Regular Expressions**-combination of literals and metacharacters  
-literal text forming the words of this language, and the metacharacters defining its grammar  
-regular expressions have rich sets of metacharacters

-simplest pattern consists of only literals; a match occurs if the sequence of literals occurs anywhere in the text being tested

-i.e. search for all lines that start with “**I think** “  
-$ represents the end of a line. i.e. “**morning$**” search for all lines that end with morning  
-you can search for the combination of either case **([Bb] [Uu] [Ss] [Hh])**  
-**^[Ii] am** – the I will be either case and then the am is specifically just “am”  
-**^[0-9][a-zA-Z]** – a line that starts with some number and some letter immediately after  
-**[^?.]$** - looking for lines that end in anything other than ? or .

-**“.”** refers to any character.   
**9.11** searches anything with 9(some character)11  
**flood|fire** searches anything that has flood or fire in the line // can include any amount of alternatives  
**^[Gg]ood|[Bb]ad**  searches for anything that begins with Good/good OR anything with the word “bad” or “Bad” in the line  
**[Gg]eorge( [Ww]\.)? [Bb]ush** the question mark indicates that the expression is optional. So the Ww is optional. The backlash allows us to search for an actual ., not the “any character” .

**(.\*)** - \* means any number, including none, of the item. Searching for the combination of ( and ) in the line with any amount of characters in betwee  
+ means at least one of the item

**[0-9]+ (.\*)[0-9]+** - a number followed by some number of characters then another number

**[Bb]ush( +[^ ]+ +){1,5} debate** // bracket gives the minimum and maximum number of matches of an expression. Bush (some characters) and then at the end is debate. We wanna see something like space word space up between one and five times.

M,n – at least m but not more than n matches  
m – exactly m matches  
m; - at least m matches

\1 is like a space followed by some, at least one, characters followed by at least one space

**+([a-zA-Z]+) +\1 +** // a repetition of word space word

**^s(.\*)s** - \* matches the longest possible string that satisfies the regular expression.  
**^s(.\*?)s$** - doesn’t only search for the longest possible string that satisfies the regular expression anymore

**3. Working with Dates**

**d1 = date()** // gives the date **class(d1)** // “character”

**d2 = Sys.Date()** // Sys.Date()  
**class(d2)** // “Date”

**format(d2, %a %b %d”)** // %d = day as number, %a = abbreviated weekday, %A = unabbreviated weekday, %m = month, %b = abbreviated month, %B = unabbreviated month, %y = 2 digit year, %Y = four digit year

**x = c(“1jan1960”, “2jan1960”, “31nar1960”, “30jul1960”); z = as.Date(x, “%d%b%Y”)  
z** // will print out 1960-01-01 etc  
**z[1] – z[2]** can calculate the time difference of the two dates  
**as.numeric(z[1]-z[2])** // gives -1

**weekdays(d2)** // gives the week day  
**months(d2)** // “January”  
**Julian(d2)** // how many days have occurred since the origin date which was 1/1/1970

**library(ludribate); ymd(“20140108”)**// allows ymd, mdy, dmy (writing dates in different order) and just give some number

**ymd\_hms(“2011-08-03 10:15:03”)** //can also change time zone  
check **?Sys.timezone**

**Some functions have slightly different syntax  
x = dmy(c(“1jan2013”, “2jan2013”, “31mar2013”, “30jul2013”))  
wday(x[1])** // 3  
**wday(x[1], label=TRUE)** // Sun < Mon < Tues < Wed < Thur < Fri < Sat

**LAST, CHECK WEEK 4 FOR A LIST OF DATA RESOURCES, THEY’RE SO USEFUL**