R for Text Mining Coding Examples

\*These are only for reference, as copying and pasting can cause some complications for RStudio\*

[Getting Started - Word Search](#_dkby3gd2a5u6)

[Data Management Starts Now](#_g6d5ppc85uvq)

[Easy maths](#_1fr9zhzhzhsa)

[Hello World](#_ndkk2t2lkfd2)

[Data types](#_6w1uosa1bqdf)

[Easy Plot and Getting Help](#_vfz51grbc2lf)

[Subsetting and Indices: or, getting to a value](#_9k1ik5dnercb)

[Bonus: Practice and Subsetting](#_2jlftf4san4n)

[Regular Expressions in R](#_x39nprm0az2l)

[More REGEX](#_qcr8672hjrqn)

[REGEX Practice](#_6rxwsvbmh7sz)

[Let’s look at Moby Dick - importing your own data](#_fqyj28q8yik8)

[Let’s look at a plot of Moby Dick](#_k97k4b1ssv0l)

[Index/Glossary of Functions and Operators](#_wco62my5txf9)

[A Quick Glossary of R Objects](#_8gqj5xlwiw6c)

[R For Text Mining - Coding Examples (mostly in order of their use)](#_f8jsswgnes4r)

# 

# 

# Getting Started - Word Search

Find the following:

ahab, corpus ,dog, fox, lazy ,mining, quick, rstudio, text, whale

r k s y k o n w u a t y v c e i o y z y

q s v x m g s q n w z y o u q l c x g u

f v t u g z n e f a x r m k i n a d f x

d q v u s q j m l c p a t i o q f h g c

l r j r d e i i y u i f i u t m x z w b

l z j l c i f n s k g z n v x j q x s r

z f z l d d o i q k m l s q e v f s m s

k t d q o i e n a h a b l g t c i o x k

e p w k g v y g k c i u q a n g s a x z

v m n b s n m n u d h f o q v z o c e t

# Data Management Starts Now

Use this table to keep track of different objects you make, what kind it is, and description/comments. It is the draft of your data dictionary.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Object | Type | Comments |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Last backup:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Easy maths

#this is a comment

#Let’s do some math. Run each of these lines of code in your script:

2+2

100/5

33\*3

1 > 0

#level up! Use the assignment operator and range operator to build a bigger object

x <- 1:4

# Hello World

print(“Hello World”)

#level up! We can build an object here too, with a different kind of data inside

greeting<-“Hello World”

print(greeting)

#well done on a classic exercise!

# Data types

**Numeric** - integers, float (decimals), continuous

**Categorical** - Economy plane ticket, Business plane ticket, First Class Plane Ticket

**Spatial** - coordinates, 101 Trustee lane

**Boolean** - TRUE, FALSE

**String** - “Call Me Ishmael.”

**Timeseries** - data with built in understanding over time (rise of minimum wage, etc)

# Easy Plot and Getting Help

#I’d like a scatter plot - comments do not calculate

plot(1:20)

#level up, feed additional arguments to the function

plot(1:20, type="s")

#but Adrienne, how did you know that?

#let’s see the help documentation

?help

?plot

#double the ? to force a search of the documentation

??plots

# Objects in R

See Glossary-- after the exercise!

# Subsetting and Indices: or, getting to a value

#R indexing starts at 1 (not 0)

#remember, building an object in R requires the <- ‘assignment operator’

colors <- c(‘red’, ‘green’, ‘blue’)

#that object has an index, so we can access the values based on a number

#the square brackets, directly next to your object, indicate subsetting

colors[1]

#should return ‘red’

# Bonus: Practice and Subsetting

Prompt:

# build the 2 vectors, your favorite food and favorite numbers.

# Check them

#practice subsetting. R indices start with 1.

f[2]

n[1]

#level up. Try this prompt with your own vector subsets to print a funny sentence

print(c("I’m going to the picnic and I’m bringing",n[1],f[1]))

Sample Answer:

# build the 2 vectors, your favorite food and favorite numbers.

f <- c(“cookies”, “apples”, “chocolate”, “salmon”, “tea”)

n <- c(1, 8, 111, a kajillion, 42)

# Check them by running them on their lines

#practice subsetting. R indices start with 1.

f[2]

n[1]

#pull a sample into the prompt.

print(c("‘I’m going to the picnic and I’m bringing",n[1],f[2]))

# Regular Expressions in R

#“Regular expressions” or regex is a set of tools for sifting through text data.

#Symbols represent patterns or characters.

#So we can put together a series of symbols and characters to match patterns.

#Let’s start with a short vector we made ourselves

fox <- “The quick brown fox jumped over the lazy hound dog.”

#how long is this (how many values)? What type of data is it?

class(fox)

#level up

#we don’t want cases!

fox <- tolower(fox)

# More REGEX

#In reality, we want each word to be a value in this object

?strsplit

fox1 <- strsplit(fox,”\\W”)

#how long is this now? But wait, is it still a character vector?

class(fox1)

#nope! strsplit makes lists. We want character type data

#Unlisting it and assigning (that unlisted object) will do the trick

fox1 <- unlist(fox1)

fox1

# REGEX Practice

#Let’s practice with REGEX

?grep

#let’s start with words that start with f

grep(‘^f’, fox1)

#Value return is indexed. I can assign it to something

animals <- fox1[4]

animals

#Can you call out the words that start with d?

#Can you call out the words that end with d?

#What about any string that matches ‘o’ in the middle?

Answers:

#All highlighted text at the REGEX operators that allow for string matching

#starts with f or d

grep(‘^f’, fox1)

grep('^d', fox1)

animals <- fox1[10]

animals

#ends with d

grep('d$', fox1)

#strings that match ‘o’ in the middle

grep('.o.', fox1)

# Let’s look at Moby Dick - importing your own data

#visit <https://tinyurl.com/TinkerR2019> and visit the data folder

# download the **moby\_data.RData** file to your desktop. Don’t open!

#get **the file directory** for your file names to use in the load() command

load(“~/moby\_data.RData”)

moby\_words

#take a look - how long is this book?

length(moby\_words))

#now to crunch it, how do we want to do that?

#create a vector holding the index of words 1 to 214944

novel\_timeline <- seq(1:length(moby\_words))

#How many words start with a?

grep('^a', moby\_words)

#that’s a lot,all index values, let’s store it somewhere managable!

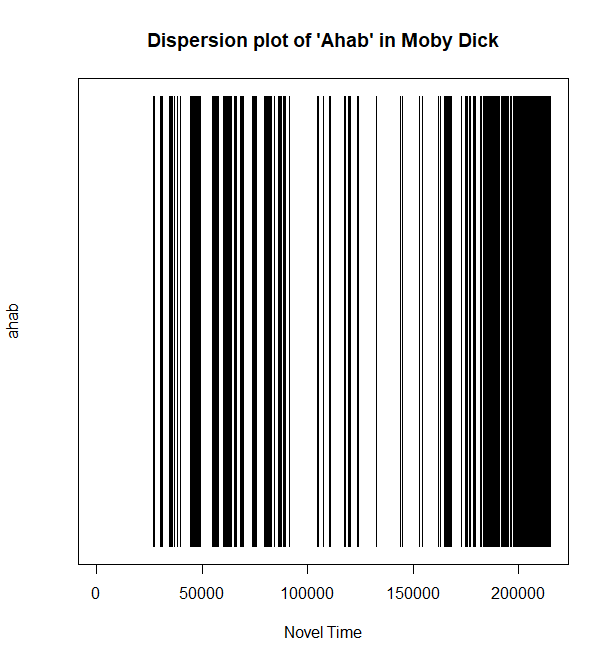
starts\_a <- grep('^a', moby\_words)

#now I can subset with those index values and get the actual words (the values at those index points) into an object

moby\_words[starts\_a]

a\_words <- moby\_words[starts\_a]

# Let’s look at a plot of Moby Dick

#how often does ahab appear?

which(moby\_words == "ahab")

ahab\_index <- which(moby\_words == "ahab")

#with a novel plotline and an ‘ahab index’ we can do a dispersion plot of ‘ahab’ across the story

#in our plot, x axes will be novel timeline, y will be Ahab occurrences, but it must be the same length as the x

y <- rep(NA, 214944)

y[ahab\_index] <- 1

plot(x=novel\_timeline, y=y, main="Title", xlab="Novel Time", ylab="ahab", type='h', ylim=c(0,1), yaxt='n')

# BONUS - Frequently used words?

#what are the most frequently used words?

#make a table object with table function

moby\_freqs <- table(moby\_words)

#sort the table by highest to lowest (decreasing), and assign that to the object

moby\_freqs <-sort(moby\_freqs, decreasing=T)

#subset the top 20 values in that table

moby\_freqs[1:20]

#interesting, let’s plot the more exciting words

plot(moby\_freqs[c(2,5:12, 15:25)])

# Index/Glossary of Functions and Operators

|  |  |  |
| --- | --- | --- |
| A Quick Glossary of R Objects | | |
| Object | Description | Example |
| Vector | a group of variables of the same value type  can hold primitive values (numbers, T/F, text,) | greeting - 3 values:  *“call”,”me”,”ishmael”* |
| matrix | a vector represented and accessible in two dimensions, must be the same data types within | a matrix of page numbers  *1 2 3*  *16 20 17* |
| dataframe | a set of data with a number of rows and columns, not necessarily the same type. A spreadsheet or table type thing. | Moby\_df - 2 observations of 2 variables  *age reading level*  *2 NA*  10 5th grade |
| list | a generic vector that is allowed to include different types of objects, including other lists | a list of chapters I read:  *Loomings, 2, 3, 4:6* |
| R For Text Mining - Coding Examples (mostly in order of their use) | | |
| Function | Description | Example |
| print() | Displays the string or full variable named in the console | print(“Hello World”) |
| <- | Assignment Operator (name of object on left, values within on right) | greeting <- “Hello World”  x <- 1:4 |
| plot() | Creates a visual graphic of the data or function named | plot(1:20) |
| c() #Combine | Combines the listed items into a vector | c(1, 2, 3, 4) |
| ? | Help documentation for value following to see R documentation on a function, object, dataset, etc. | ?plot |
| [ ] | Square brackets appended to an object will subset the row, column, indexed items in that obect | iris[42, 5] |
| install.packages() | Installs a new package to your machine, typically fetching packages automatically.  You must then call them into the working environment with ‘library()’ | install.packages(“fun”) |
| class() | Returns the class or datatype of the object | class(fox) |
| tolower() | Takes a character vector and returns the same with all lower case | fox <- tolower(fox) |
| strsplit() | This function takes strings and splits them per the REGEX pattern you specify | strsplit(fox, “\\W”) |
| unlist() | This function will return the unlisted values of the list fed into it . you have to assign a new object to get it to stick around | fox1 <- unlist(fox1) |
| grep() | This function searches the given vector for matches in the pattern. It takes many arguments | grep(‘^f’, fox1) |
| load() | This function loads up a data object from a directory you specify | load(“~/moby\_data.RData”) |
| seq() | This is sequence generation, and it will build a sequence between the values you specify, in the pattern dictated. Typically builds an object | ten\_count <- seq(1:10) |
| which() | Returns a boolean T or F off the logical index argument fed | which(moby\_words == "ahab") |
| rep() | Replicate. It builds an object off the given value, for the pattern and length specified | y <- rep(NA, 214944) |
| plot() | Base plotting function in R, takes many arguments. Data fed must be of the same length/dimensions, or you will get an error. Defaults to x=y for x values | plot(1:20) |