Welcome!

Intro to R for Text Mining

Folder: https://tinyurl.com/TinkerR2019

Adrienne Canino
Science & Data Outreach Librarian

The Goals

Our basic introductory tour of R and RStudio is meant to be from the very beginning.

We'll rely mostly on live demo.

- ➔ Load up R
- ➔ Load up RStudio
- ➔ Learn some basics
- ➔ Start data manipulation

The Tools

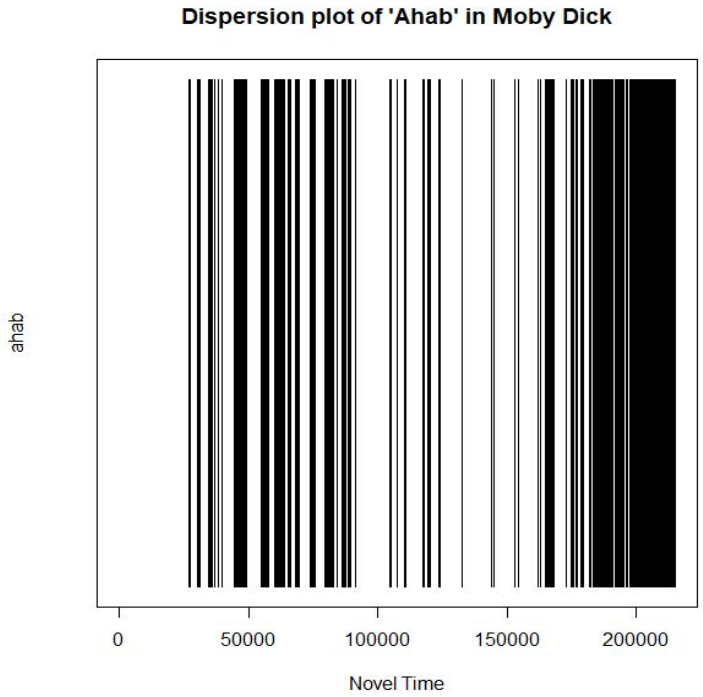
This is a very high-level overview of R.

How am I going to do this with you all today?

- ➔ Worksheets & Glossary
- ➔ Google Folder (materials)
- ➔ Live-Demo
- ➔ Refer out to more materials throughout

Folder: https://tinyurl.com/TinkerR2019

Getting Started - Where is Ahab?



Why use R for text analysis?

Ever use control+f in a web browser to get right to the letters/words that you need?

How great would it be to do that to a text with over 200,000 words?

And then plot it!

Folder: https://tinyurl.com/TinkerR2019

Getting Started - Download

[R]

- Download from cran.r-project.org
  - Select a ‘mirror’ link in your country
- You can run through R app or through Terminal
  - We’re going to use an integrated design environment (IDE)

RStudio

- Download from rstudio.org
- Upload and run
- If you’d like to set yours up so it looks like mine with the fancy colors, it’s a fun choice to make!

(tour is next)

Folder: https://tinyurl.com/TinkerR2019

Getting Started: What can R do?

[R]

- Named for its creators, Robert Gentleman and Ross Ihaka
  - University of Auckland, New Zealand in 1995
- First released as an open source package in 2000
  - GNU GPL Version 2 License
- Took language, ‘S’, up a level

RStudio

- Founded in 2008
  - First public release in 2011
- Built to provide an integrated R tool, “enterprise” ready - and to be Open Source
- **Take a tour of the screens!**

Folder: https://tinyurl.com/TinkerR2019

Data Management Starts Now

"In this world you've just got to hope for the best and prepare for the worse..."
~ Anne Of Avonlea, by Lucy Maud Montgomery

- Save early, often, and in versions
- File names that make sense (See Smithsonian's helpful doc)
- Master version, working versions, and archives out of your directory
- Track and document... everything
  - Data dictionary
  - Central and consistent metadata (Intro to Metadata)
- Plan for storage and backup
  - When your computer crashes, it's too late to have a backup

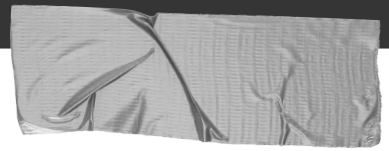
Folder: https://tinyurl.com/TinkerR2019

Easy maths and logicals

A ~fancy~ calculator

Recap:

- R is a math language, so it gobbles up math exercises (much easier than ‘print’ or language exercises)
- R functions typically work off objects, and the values stored in them.
- run == ctrl + enter
- R is Case Sensitive!



Code:

```
#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
```

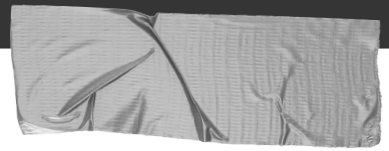
Folder: https://tinyurl.com/TinkerR2019

Hello World!

Embrace the tradition.

Recap:

- run == ctrl + enter
- print command - you must feed it a string
- color changes to indicate your code's readiness
- assignment operator arrow builds objects
- In this case, we made a variable to hold a phrase, which is a string data type
- So, variables are containers in R that hold things for us to do stuff to them later.



Code:

```
#the print function --
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!
#why print() ?
```

Folder: https://tinyurl.com/TinkerR2019

Data types - and why get to know ‘em

- **Numeric**
  - integer
  - float
  - continuous
- **Categorical**
- Spatial
- Boolean (logical)
- **String**
- Timeseries

- Not all data-types fit a visualization type:
  Data-To-Viz

  Periodic Table of Visualizations

- A very good blogpost on types of data:

  Data Types

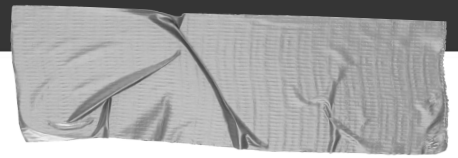
Folder: https://tinyurl.com/TinkerR2019

Easy Plot and Getting Help

Get graphical.

Recap:

- plot command - you must feed it numbers
- You can tell R to use a range, with the colon notation (i.e., 1:20)
- The plot shows in RStudio's viewer.
- Notice anything about that plot?
- R is Case Sensitive!
- you can copy the line of code down onto a new line with Alt + Shift + ↓ (down arrow)



Code:

```
#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
??plots
```

Folder: https://tinyurl.com/TinkerR2019

Objects in R

Vectors, Variables and Matrices, oh my! - Myself, a paraphrasing.

The flashcards I'm handing out all have a different object type in R. Make friends with a neighbor for this activity.

Using the 'help' command and documentation in RStudio (and, if you need to, Google Search), define this object, type it as a #comment in your script, and report it back to the class.

Don't cheat with the glossary!

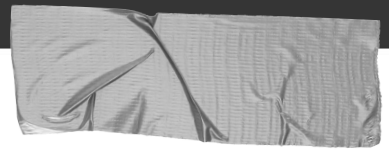
(2 minutes)

Subsetting and Indices

or, getting to a value

Recap:

- The assignment operator builds these objects in R
- c function is ‘Combine’ and allows you to assign multiple values (like a list) into an object
- In this case, we built a vector holding 3 values, they have been indexed in the order fed to Combine, starting at 1
- So ‘blue’ is 3
- My code is 2/3 comments, 1/3 code!



Code:

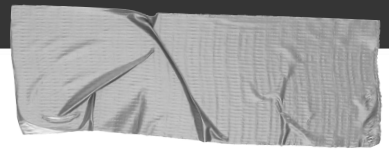
```
#R indexing starts at 1 (not 0)
#remember, building an object in R
requires the <- ‘assignment
operator’

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’
```

Folder: https://tinyurl.com/TinkerR2019

Bonus! Practice Subsetting

Don't peek ahead!

1. Build 2 vectors:
   a. f (list 5 favorite foods)
   b. n (list 5 favorite numbers)
2. Display both to check their validity.
3. Now we can select, through the index, which starts with 1.
4. Using square brackets, [], subset something from each vector to test.
5. Try to get this prompt to work!



Code:

```
# build the 2 vectors, your
favorite food and favorite numbers.
# Check them
#practice subsetting. R indices
start with 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]))
```

Folder: https://tinyurl.com/TinkerR2019

Regex in R

Remember that wordsearch?

"Regular expressions" or *regex* is a set of tools for sifting through text data.

They are fairly consistent across most programming languages

Symbols represent patterns or characters.

So we can put together a series of symbols and characters to match patterns (and words are patterns).

More: https://www.regular-expressions.info/rlanguage.html

Regex Common Symbols

- . match any character except new line
    - Is it a whale? or a while? or whole?
    - wh.le
- + match preceding one or more times
    - Is it a mob? or is it moby? or a mobile?
    - mob+
- ^ match only if at the start of the line/value
    - Does the value start with
- [] match one of the listed characters
    - Is it a gray whale? or a grey whale?

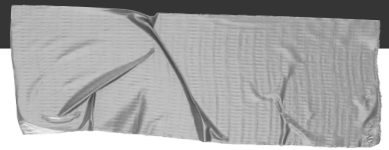
      gr[ae]y

Folder: https://tinyurl.com/TinkerR2019

REGEX practice

Recap :

- We made a character vector
- Assignment operator
- Clean code - copy and pasting!
- A string variable
- What's the index number?
- why translate it to lower case?



Code:

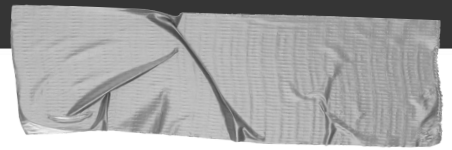
```
#Let's start with a short vector we
made ourselves

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

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

#level up
#we don't want cases!

fox <- tolower(fox)
```

Folder: https://tinyurl.com/TinkerR2019

REGEX practice

Recap:

- Remember - I want a character vector with string values
- REGEX is case sensitive
- What happened to the punctuation?
- What if I had just unlisted it?
  - the function returns the value, but you haven't assigned that value to anything - so it doesn't save it anywhere really, it just has this thing it can do



Code:

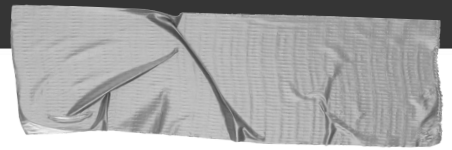
```
#In reality, we want each word to be a value
in this object
?strsplit

fox1 <- strsplit(fox,"\\W")

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

class(fox1)

#nope! strsplit makes lists. Unlisting it
and assigning (that unlisted object) will do
the trick

fox1 <- unlist(fox1)
fox1
```

Folder: https://tinyurl.com/TinkerR2019

REGEX practice - 10 minutes

Get familiar with GREP function. It is not the only REGEX function for R, but it's the one we'll use today.

- Can you call out the words that start with d?
  - can you assign that to the 'animals' object?
- Can you call out the words that end with d?
- What about any string that matches having an 'o' in the middle?



Code:

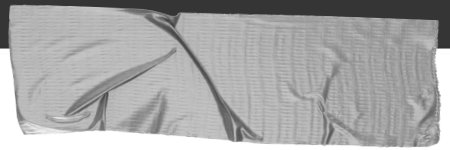
```
#Let's practice with REGEX
?grep
#let's start with words that start with f

grep('^f', fox1)

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

animals <- fox1[4]
animals
```

Folder: https://tinyurl.com/TinkerR2019

REGEX practice - Answers

Get familiar with GREP function. It is not the only REGEX function for R, but it's the one we'll use today.

- Can you call out the words that start with d?
  - can you assign that to the 'animals' object?
- Can you call out the words that end with d?
- What about any string that matches having an 'o' in the middle?



Code:

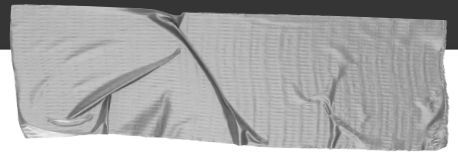
```
#All highlighted text at the REGEX operators
that allow for string matching

#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)
```

Folder: https://tinyurl.com/TinkerR2019

Let's look at Moby Dick!

Recap:

- what kind of work did I do to make this data file?
- build the novel_timeline
- *tokens* is a common reference in text analysis
  - from tidy text analysis:
  - 'tokens are meaningful units of text' (words) that we can search out and compare
- we made a list of a tokens starting with a - but, they are the index vales then we SUBSET those index values out of the character dataset
- Then we could store the actual words in an object with an assignment operation



Code:

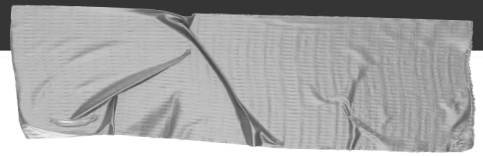
```
#load in the data from the data file

load("~/moby_data.RData")
moby_words

#take a look - how long is this book?
length(moby_words)

#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)
starts_a <- grep('^a', moby_words)
moby_words[starts_a]
a_words <- moby_words[starts_a]
```

Folder: https://tinyurl.com/TinkerR2019

Let's look at Moby Dick!

Recap:

- we compared how often 'ahab' token appears
  - We have to use the 'which' call instead of subsetting because character vectors vs. index number
- We made an object that holds the index values of where Ahab appears. Compare that to the index of every single word, and we have dispersion.
- You can feed arguments for labels and such into the plot function
  - ?plot for more!
- we technically did not have to call out x in this plot, but it's good to see where's what
- *Could you do the same for 'whale' ?*



Code:

```
#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')
```

Folder: https://tinyurl.com/TinkerR2019

Wrap Up

Save protocols:

- Save script and raw data files in the same place!
- Save documentation there too
- Writing out data objects with 'save()' function

Documenting protocols:

- **2/3 comments, 1/3 code**
- **readme files, data dictionaries**

Keep learning:

- https://www.rstudio.com/online-learning/
- https://flowingdata.com/category/tutorials/
- https://www.r-bloggers.com/how-to-learn-r-2/
- "Text Analysis with R for Students of Literature" by Matthew Jockers (eBook)
- Text Mining with R by Julia Silge and David Robinson (free eBook, linked earlier)

Folder: https://tinyurl.com/TinkerR2019

“There are 3 kinds of lies: lies, damned lies, and statistics.”
-Mark Twain

Thank you!

acanino@library.rochester.edu