ISSR Short Course

Gregory J. Matthews [1]

[1]Department of Mathematics and Statistics
Loyola University Chicago

June 2015

Outline

Reading data from URL

R can read data directly from the internet if the URL provides links directly to a data source that can be read by R.

```
#Used to be able to do this:
#titanic<-read.csv("http://bit.ly/1KSjVqg")
#Now you have to do this:
library(repmis)

## Warning:  package 'repmis' was built under R
version 3.1.3

titanic<-source_data("http://bit.ly/1KSjVqg")

## Downloading data from:  http://bit.ly/1KSjVqg
##
## SHA-1 hash of the downloaded data file is:
## ea08b483790c2a7bc9b95b0f923526f8e60eae44

class(titanic)
```

R XML package

- Usually this isn't the case though.
- Let's say we want to read data from a table that is in a web page.
- We can use functions in the R package XML to help here.

R XML package

Simple Example:

```
library(XML)
#Let's go look at this page
url<-"http://www.baseball-reference.com/teams/BOS/2014.shtml"
#Convert the web page tables into R data.frames
redSoxTable<-readHTMLTable(url)
class(redSoxTable)


## [1] "list"


names(redSoxTable)


## [1] "team_batting"            "team_pitching"
## [3] "standard_fielding"       "players_value_batting"
## [5] "players_value_pitching"


class(redSoxTable$team_batting)


## [1] "data.frame"
```

- The object redSoxTable is of class list.
- Each element of the list contains a data.frame object that corresponds to one of the tables on the web page.

R XML package

```
#Let's look in
redSoxTable$team_batting[1:3,1:8]

##   Rk Pos               Name Age   G  PA  AB  R
## 1  1   C A.J. Pierzynski*  37  72 274 256 19
## 2  2  1B       Mike Napoli  32 119 500 415 49
## 3  3  2B    Dustin Pedroia  30 135 609 551 72
```

- Some of the player's names have characters in them that I don't want. How do I get rid of '*' and '#' in some player's names?
- What if I only want to look at infielders?

R XML package

```
#Use gsub to remove unwanted character
redSoxTable$team_batting[,3]<-
  gsub("[*#]","",redSoxTable$team_batting[,3])
#Unwanted characters are removed
redSoxTable$team_batting[1:3,1:8]

##   Rk Pos             Name Age   G  PA  AB  R
## 1  1   C A.J. Pierzynski  37  72 274 256 19
## 2  2  1B      Mike Napoli  32 119 500 415 49
## 3  3  2B  Dustin Pedroia  30 135 609 551 72
```

R XML package

```
#To get just infielders, we use the function %in%
#Only pull out infielders
infInd <- redSoxTable$team_batting$Pos%in%c("1B","2B","3B"
infielders <- redSoxTable$team_batting[infInd,]
infielders[,1:8]

##    Rk Pos                Name Age   G  PA  AB  R
## 2   2  1B         Mike Napoli  32 119 500 415 49
## 3   3  2B      Dustin Pedroia  30 135 609 551 72
## 4   4  SS     Xander Bogaerts  21 144 594 538 60
## 5   5  3B Will Middlebrooks  25  63 234 215 14
## 17 16  SS        Stephen Drew  31  39 145 131 11
## 27 26  3B        Ryan Roberts  33   8  22  19  1
## 29 28  1B     Ryan Lavarnway  26   9  10  10  0
```

R XML package

- Now let's say I want to do something across every team.
- How do I do that?
- Notice that in the URL the only piece that changes is the team abbreviation.
- http://www.baseball-reference.com/teams/**BOS**/2014.shtml
- So first, let's get a vector of these abbreviations.
- We can do this by going to this url: http://www.baseball-reference.com/leagues/MLB/2014.shtml

R XML package

```
teamsData<-readHTMLTable("http://www.baseball-reference.com/leagues/MLB/2014.shtml")
class(teamsData)

## [1] "list"

names(teamsData)

## [1] "NULL"                     "teams_standard_batting"
## [3] "teams_standard_pitching" "team_output"
## [5] "teams_standard_fielding"

teamsData[["teams_standard_batting"]][1:3,1:5]

##    Tm #Bat BatAge  R/G   G
## 1 ARI   52   27.6 3.80 162
## 2 ATL   39   26.8 3.54 162
## 3 BAL   44   28.3 4.35 162

#Construct a vector of all the team abbreviations
teams<-as.character(teamsData[["teams_standard_batting"]]$Tm[1:30])
teams

##  [1] "ARI" "ATL" "BAL" "BOS" "CHC" "CHW" "CIN" "CLE" "COL" "DET" "HOU"
## [12] "KCR" "LAA" "LAD" "MIA" "MIL" "MIN" "NYM" "NYY" "OAK" "PHI" "PIT"
## [23] "SDP" "SEA" "SFG" "STL" "TBR" "TEX" "TOR" "WSN"
```

R XML package

```r
#resList will store the results
resList<-list()
#We will loop over all teams
for (t in teams){print(t)
  url<-paste("http://www.baseball-reference.com/teams/",t,"/2014.shtml",sep="")
  teamData<-readHTMLTable(url)
  #Pull out the team batting component
  output<-teamData$team_batting
  #Convert player name to a character
  output[,3]<-as.character(output[,3])
  output[,3]<-gsub("[*#]","",output[,3])
  #Store the output
  resList[[t]]<-output[output[,3]!="",]
}
```

R XML package

```
class(resList)
names(resList)
resList$BOS[1:3,1:8]
resList$NYY[1:3,1:8]
```

R XML package

```r
#Now I want to stack all of these
#data frames on top of each other
allTeams<-do.call(rbind,resList)
dim(allTeams)
```

- So far we have scraped data from tables online.
- What if we want to scrap unstructured data from the web.

- This first example involves scraping the content of presidential inaugural speeches.
- We're going to use the functions getURL and parseHTML here.

```
library(RCurl)

## Loading required package:  bitops

library(XML)
presList<-list()
url<-"http://www.presidentialrhetoric.com/historicspeeches/bush/
```

- We can read the content of the webpage.
- But it is a mess right now.

```
a<-getURL(url)
str(a)
```

- If we want to parse the html we can use the following commands.
- Either way (htmlParse or getURL) we can extract the text we want.

```
b<-htmlParse(a)
b
```

Before we were reading HTML tables from the web. Now we are reading an HTML table that is an object in R.

```r
x<-readHTMLTable(a)
x
```

let's clean it up.

```r
#clean it up a bit
text<-levels(x[[6]]$V1)
text<-gsub("\n","",text)
text<-gsub("[,.]","",text)
#Make it a plain text document and add it to a list
library(tm)
presList[["HWBush"]]<-PlainTextDocument(text)
```

- Now we have one speech in a list.
- Let's go get some more speeches and make a corpus.
- We're going to repeat what we just did for Clinton and W. Bush.
- Then we will make these three speeches into a Corpus.

```r
url<-"http://www.presidentialrhetoric.com/historicspeeches/
a<-getURL(url)
b<-htmlParse(url)
x<-readHTMLTable(b)
text<-levels(x[[6]]$V1)
text<-gsub("\n","",text)
text<-gsub("[,.]","",text)
presList[["Clinton"]]<-PlainTextDocument(text)
```

```
url<-"http://www.presidentialrhetoric.com/historicspeeches/
a<-getURL(url)
b<-htmlParse(url)
x<-readHTMLTable(b)
text<-levels(x[[6]]$V1)
text<-gsub("\n","",text)
text<-gsub("[,.]","",text)
presList[["Bush"]]<-PlainTextDocument(text)
```

- We have a list object called presList.
- Each element of the list is a PlainTextDocument.
- We can now turn the list into a Corpus.

```
presList<-Corpus(VectorSource(presList))
class(presList)
```

Basic Filtering

- Remove extra white space
- Remove “stop words” (i.e. “a”, “an”, and “the”)
- Stemming: Convert words to their stems (i.e “Fishing” and “Fished” become “Fish”)
- Convert to lower case: This way “Dog” at the beginning of a sentence is treated the same was as “dog” in the middle of the sentence.
- Remove sparse terms: remove terms that are only used in a small number of documents.

Transformations on Corpora

```r
#Removes extra whitespace
presList<-tm_map(presList,stripWhitespace)
#Stop words: Words filtered out before processing.
stopwords("english")[1:10]
presList<-tm_map(presList, removeWords, stopwords("english"))
#presList <- tm_map(presList, tolower) #OLD
presList <- tm_map(presList, content_transformer(tolower)) #NEW
#Stemming: reducing words to their stems.
presList<-tm_map(presList, stemDocument) #requires SnowballC package
presList<-tm_map(presList, PlainTextDocument)
```

```r
presTDM<-TermDocumentMatrix(presList)
presTDM<-removeSparseTerms(presTDM,0.5)
#1 means it has to be in all documents
#0 means we keep all words
#0.5 means we keep words that appear in
#at least half of the documents.
```

Term Frequency

- We may be interested in seeing how often certain words appear.
- This is analagous to the table function in R, but we can filter out certain terms that we may not want based on some criteria.

```r
#Set control parameters
ctrl<-list(removePunctuation =
             list(preserve_intra_word_dashes = TRUE),
           stopwords=c("bush","Bush"),
           stemming=TRUE,
           wordLengths=c(4,Inf))
#Look at a frequency table of words
sort(termFreq(presList[["HWBush"]]),control=ctrl)
```

```
#We can also do this:
findFreqTerms(presList[["HWBush"]],lowfreq=3,highfreq=10)
```

- We can also measure the similarity or disimilarity of the texts in the corpus.
- We also may be interested in looking for associations of words within certian texts. Simply how often do these words appear in the same text.

```r
library(proxy)
#Slot 6 contains the text.
presTDM<-TermDocumentMatrix(presList)
#
dissimilarity(presTDM,method="eJaccard")
findAssocs(presTDM,c("responsibility","vision"),corlimit=c(
```

- Right now we have three presidential inaugural addresses.
- But what if we want to get ALL the presidential inaugural addresses?
- We need to provide URLs for each speech and this can be tedious.
- Luckily, we can use R to get the list of URLs for us.

```
#Now get them all
#pull in the content of the page
#the parse the HTML
doc<-htmlParse(getURL("http://www.presidentialrhetoric.com/
#This next line pulls out all of the links
vec<-xpathSApply(doc, "//a/@href")
#Now we can pull out only the speeched that were inaugural
vecList<-vec[grep("inaugural",vec)]
```

- What do we have now?
- We have a vector containing all of the last pieces of the URLs that we will need for each presidential inaugural address.
- We can combine these pieces with the root URL using paste and loop across all of the speeches to get all of them.

```
presList<-list()
for (vvv in vecList){print(vvv)
url<-paste("http://www.presidentialrhetoric.com/historicspe
a<-getURL(url)
b<-htmlParse(url)
x<-readHTMLTable(b)
text<-levels(x[[6]]$V1)
text<-gsub("\n","",text)
text<-gsub("[,.]","",text)
presList[[vvv]]<-PlainTextDocument(text)
}
```

```
presCorpus <- Corpus(VectorSource(presList))
presCorpus <- tm_map(presCorpus, content_transformer(tolower))
presCorpus<-tm_map(presCorpus,removeWords,stopwords("english"))
presCorpus<-tm_map(presCorpus,removePunctuation)
presCorpus<-tm_map(presCorpus,removeNumbers)
presCorpus<-tm_map(presCorpus,stripWhitespace)
presTDM<-TermDocumentMatrix(presCorpus)
presTDM$dimnames$Docs<-names(presList)
```

```
findFreqTerms(presTDM,500)
#[1] "government" "people"     "will"
findAssocs(presTDM,"government",0.7)
#           government
#system           0.78
#states           0.73
#interests        0.72
#protection       0.71
#adoption         0.70
```

Let's try it:

- Get inaugural speeches of the last three presidents.
- Turn these speeches into a corpus.
- Get all inaugural presidential speeches.
- Turn these speeches into a corpus.
- Find frequntly used terms.
- See BushCode.R