Importing Data from the Web I

Cosma Shalizi and Vincent Vu
November 16, 2011
Agenda

• Multiple capture groups
• Example: Scraping web links
• Using readLines() dynamically
• Example: random web surfing
• Digression on exception handling
• Do all roads lead to Facebook?
• * Code from examples on course webpage
Tagged Expressions

• Enclose **capture groups** with parentheses

• Can be extracted separately using `regexec()`

• Returns multiple positions

  • Match, group 1, group 2, ...
Example: Web links
Brittanie Boone (Econ-Stat, 2012) has been chosen as a 2011-2012 Andrew Carnegie Society Scholar. This award is for seniors who “embody high standards of academic excellence combined with multi-dimensional characteristics such as volunteerism, involvement in student organizations, participation in sports or the arts and leadership.” Only nine H&SS students were chosen as ACS scholars this year. Besides receiving monetary awards, ACS Scholars work together during the school year to contribute to philanthropic activities on campus.

Congratulations to Brittanie!

Applications are invited for possible tenure-track, lecturer, and visiting positions. Carnegie Mellon offers a collegial faculty environment, emphasizing a combination of disciplinary and cross-disciplinary research and teaching. All areas of statistics are welcome, and joint appointments with other units in the Pittsburgh area are possible. We especially encourage women and minorities to apply. For further information, please contact

Carnegie Mellon University
Department of Statistics
RESEARCH SHOWCASE
Thursday, November 10th
4:30pm - 6:30pm
Click for more details.

Thursday November 17th
5:00 - 7:00 PM
ASA Pittsburgh Chapter Fall Mixer
U of Pitt University Club

Founded in 1966, Carnegie Mellon’s Department of Statistics evolved separately from the traditional umbrella of Mathematical Sciences in a University environment that emphasized computation, the understanding of human behavior and decision-making, and cross-disciplinary research. This led the Department to define a path for itself by

PhD Application Deadline is December 15th
Apply Online Now!

Master’s in Statistical Practice (MSP) application for Fall 2012 will be available November 15, 2011. The application deadline is February 15, 2011. For more information about the MSP program please see the MSP Information Page

Our Research Environment
Schedule of Classes
Our Computing Environment

TA and grading employment opportunities available. Apply now!

Expense Reimbursement Information:
- Business & Travel Policy - READ ME FIRST
- Reimbursement Procedures
- Reimbursement FAQs
Goal

- Extract all links on the CMU Stat Dept Homepage
- Create a data frame with columns:
  - url, link.text
Example matches

<a href="/programs/graduate/research-environment">Our Research Environment</a>

<a href="http://hss.cmu.edu">Humanities & Social Sciences</a>

Regular expression

\[<a[^\s\]*\s*href[^\s]*=([^\s]*[^\s]*[^>])[^\s]*>([^\s]*[^\s]*[^>])[^\s]*</a>\]

Two capture groups
linkpat <- paste(
  '<a[^>]*[^>]*href[^>]*>',
  '([^>]*)',
  '[^>]*>',
  '(.*)',
  '</a[^>]*>', sep = '')
Live Demo
readLines()

```r
y <- readLines(con, warn = TRUE)
```

- `con` url string specifying location of text file
- `warn` warn if last line does not end with a return? (ok to set to `FALSE` in many cases)
Dynamic readLines()

• The URL does not have to be static / hardcoded.

• Dynamically generated URLs are very useful for web scraping
Example: World’s Most Powerful People
The World’s Most Powerful People

The 70 Who Matter
Heads of state, business and religious leaders, opinion makers and criminals. Continue
Edited by Nicole Perlroth and Michael Noer | 11.2.11

20 Power Lists Of Their Own

With Vaccines, Bill Gates Changes The World Again

Behind Our List Of The World’s Most Powerful
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Profession</th>
<th>Country</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Pope Benedict XVI</td>
<td>Pope</td>
<td>Roman Catholic Church</td>
<td>84</td>
</tr>
<tr>
<td>8</td>
<td>Ben Bernanke</td>
<td>Chairman of the Federal Reserve</td>
<td>United States of America</td>
<td>57</td>
</tr>
<tr>
<td>9</td>
<td>Mark Zuckerberg</td>
<td>Founder</td>
<td>Facebook</td>
<td>27</td>
</tr>
<tr>
<td>10</td>
<td>David Cameron</td>
<td>Prime Minister</td>
<td>United Kingdom</td>
<td>45</td>
</tr>
</tbody>
</table>

Each name is also a link to person’s Forbes.com profile
Example

• Scrape data from the list
  • name, title, organization, age, link

• Use readLines() to download profile page for each link
Example: Do all roads lead to Rome Facebook?
Random Surfer

• Probabilistic model of what web surfing
• Given current web page
  • Choose next web page at random from links on current page
• Will we eventually reach Facebook?
• How many moves will it take?
Random Surfer in R

- Build on web links example
- Use `readLines()` to read web page
- `extractLinks()` from web page
- Choose next link at random
- *First order Markov Chain*
extractLinks <- function(html) {
  require(plyr)

  linkpat <- '<a\s+\s*href\s*=\s*"([^"]\)*\"[^>]*>(.*)</a\s*>'
  m <- gregexpr(linkpat, html, ignore.case = TRUE)
  links <- regmatches(html, m)
  links <- do.call(c, links)

  m <- regexec(linkpat, links, ignore.case = TRUE)
  links <- regmatches(links, m)

  df <- ldply(links, function(x) data.frame(url = x[2], link.text = x[3], stringsAsFactors = FALSE))

  return(df)
}
randomSurf <- function(url) {
  cat('Visiting', url, '\n')
  html <- readLines(url, warn = FALSE)
  links <- extractLinks(html)

  # Only look at fully-qualified, non-encrypted URLs
  # (because relative URLs are too much to deal with in this
  # example)
  j <- grepl('^http:', links$url, ignore.case = TRUE)
  links <- links[j, ]

  # Dead end?
  if(nrow(links) == 0) {
    stop('D\'oh! I\'m at a deadend')
  }

  # Draw a uniform random integer from 1 to nrow(links)
  i <- sample(nrow(links), size = 1)

  # Pick the next url at random
  nexturl <- links$url[i]

  return(nexturl)
}
Live Demo
Bouncy Random Surfer

- Some links lead to unreadable pages that forces readLines() to throw an error
- Bouncy surfer catches those errors and goes back to the previous link
- Second order Markov Chain
Bouncy Random Surfer

1. Choose next link at random
2. Try going to next link
3. If error
   – Go back to previous link
4. Else
   – Proceed to next link
Exception Handling

• Need automated way to handle error in a function call

• Can’t rely on return value if an error occurs inside the function before it completes

• Solution: Exception Handling
Simple exception handling with \texttt{try()}

- \texttt{result <- try( expression )}
- If expression fails (in error)
  - \texttt{class(result) == "try-error"}
- Otherwise,
  - \texttt{result is result of expression}
result <- try(readLines(someurl))
if(class(result) == "try-error") {
  # Bad news. Caught an error.
  # Do something about it.
} else {
  # Great! No error.
  # Proceed as usual
}
bouncySurf <- function(url, previousurl) {
  result <- try(randomSurf(url))
  if(class(result) == 'try-error') {
    nexturl <- previousurl
  } else {
    nexturl <- result
  }
  return(nexturl)
}
surfUntilFacebook <- function(start) {
    nsteps <- 0
    previous <- NULL
    current <- start

    while(grepl('facebook.com', current, ignore.case = TRUE) == FALSE) {
        nexturl <- bouncySurf(current, previous)
        previous <- current
        current <- nexturl

        nsteps <- nsteps + 1
        cat('Next:', nexturl)
    }

    cat('Hit Facebook after', nsteps, 'steps')
    return(nsteps)
}
Live Demo
Summary

• Multiple capture groups for extracting multiple pieces of data
• Programmatically create strings for readLines()
• Exception handling to programmatically handle errors

Next: Web page dissection lab (art of scraping)