Text Manipulation

These slides are a combination of Deb's, Duncan's and Mark's materials

Why teach it?

- Expand horizon about what are data
- Text data are plentiful
- These sources can be more compelling for students; in some sense they are closer to home, are more recognizable than certain scientific data sources.
  - Google search results
  - State of Union speeches
  - Spam and ham email
  - Abstracts database

Why we teach it

- From early problems in authorship attribution to more recent work in large-scale text mining, there's plenty of interesting problems and data sources to analyze
- Artifacts from computer-mediated communication (web logs, bulletin boards, chat transcripts, email) all provide complex and socially interesting data for students to work with
- Data often don't come in the format that we would like them and text manipulation is required to get them in shape
- Language for searching files with patterns

Spam filtering

Header from three email messages.

```
Date: Tue, 02 Jan 2007 12:17:45 -0800
From: Duncan Temple Lang <duncan@wald.ucdavis.edu>
To: Deborah Nolan <nolan@stat.Berkeley.EDU>
Subject: Re: 90 days
```

HAM:
Subject line has "Re:"

```
Date: Sat, 27 Jan 2007 16:28:48 +0800
From: remade SSE <glzmeqrxr99@embarqhsd.net>
To: depchairs03-04@uclink.berkeley.edu
Subject: [SPAM:XXXXXXXXX]
```

SPAM: Yelling in the subject line

```
Date: Thu, 03 Apr 2008 09:24:53 +0700
From: Faustino Britt <Faustino@sfera.umk.pl>
To: Brice Frederick <nolan@stat.Berkeley.EDU>
Subject: Fancy rep1!c@ted watches
```

SPAM: There is an ! instead of i and @ not a

Web logs

Date always in []

```
169.237.46.168 - - [26/Jan/2004:10:47:58 -0800]
    "GET /stat141/Winter04 HTTP/1.1" 301 328
    "http://anson.ucdavis.edu/courses/"
    "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)"

169.237.46.168 - - [26/Jan/2004:10:47:58 -0800]
      "GET /stat141/Winter04/ HTTP/1.1" 200 2585
     "http://anson.ucdavis.edu/courses/"
     "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)"
```

Information is not consistently separated by, e.g. comma

Text Mining
State of the Union Addresses

```
***
```

Speeches all in one file separated by ***

State of the Union Address
George Washington
December 8, 1790

Three lines:
State of the union address
President
Date

Fellow-Citizens of the Senate and House of Representatives:

In meeting you again I feel much satisfaction in being able to repeat my congratulations on the favorable prospects which continue to distinguish our public affairs. The abundant fruits of another year have blessed our country with plenty and with the means of a flourishing commerce. ...

Merging data to place on a map

```
"De Witt County",IL,40169623,-88904690
"Lac qui Parle County",MN,45000955,-96175301
"Lewis and Clark County",MT,47113693,-112377040
"St John the Baptist Parish",LA,30118238,-90501892
```

County Centers

```
"St. John the Baptist Parish","43,044","52.6","44.8",...
"De Witt County","16,798","97.8","0.5", ...
 "Lac qui Parle County","8,067","98.8","0.2", ...
 "Lewis and Clark County","55,716","95.2","0.2", ...
```

Census
Noted the "St."

```
 DeWitt 23 23 4,920 2,836 0
Lac Qui Parle 31 31 2,093 2,390 36
Lewis & Clark 54 54 16,432 12,655 386
St. John the Baptist 35 35 9,039 10,305 74
```

Election results
Note "County" and "Parish" missing in county name

String Manipulation

- substr(x, start, stop) - Extract or replace substrings in a character vector
- nchar(x) – x is a character vector; returns a vector of the number of characters in each element of x
- strsplit(x, split) - Split the elements of x into substrings at the split values
- paste(..., sep = " ", collapse = NULL) - Concatenate character vectors
- tolower(x) and toupper(x) – translate character to lower case / to upper case

Web Caching

1.000000 http://a.hatena.ne.jp/yamagen2001/ 72 1,42,55,57,68,69

Viewed 72 times
Changed at visits 1, 42, 55

1.000000 http://aces.boom.ru/all4/grebenyv.htm 59

Never Changed over 59 visits

1.000000 http://africa.oneworld.net/article/rssheadlines/512 72
1,4,6,8,15,20,23,25,39,40,53,66,71,72

Changed 14 times over the course of the 72 visits

Structure

- Tab-delimited
- First piece can be thrown away
- Keep:
  - URL
  - Number of visits
  - Times of change
- How to handle the variable number of times of change?

Ideas

- Split the string on tab character
  - Will have 3 or 4 pieces
  - Pick up the second as URL
  - Pick up the third as number of times visited
- Split the 4th piece on commas
  - Will have a variable number of times of change
- Pull it all into a data frame

```
txt =  readLines(filename)
els = strsplit(txt,\\t)

urls = sapply(els, `[`, 2)
numIntervals = as.integer(sapply(els, `[`, 3))

intervals = lapply(els, function(x)
                    if(length(x) == 3)
                      integer(0)
                    else
                      as.integer(strsplit(x[4], ",")[[1]]))

n = sapply(intervals, length)
refreshEvents = data.frame(url = rep(urls, n),
                    numIntervals = rep(numIntervals, n),
                    intervals = unlist(intervals))
```

Merging data to place on a map

```
"De Witt County",IL,40169623,-88904690
"Lac qui Parle County",MN,45000955,-96175301
"Lewis and Clark County",MT,47113693,-112377040
"St John the Baptist Parish",LA,30118238,-90501892
```

County Centers

```
"St. John the Baptist Parish","43,044","52.6","44.8",...
"De Witt County","16,798","97.8","0.5", ...
 "Lac qui Parle County","8,067","98.8","0.2", ...
 "Lewis and Clark County","55,716","95.2","0.2", ...
```

Census
Noted the "St."

```
 DeWitt 23 23 4,920 2,836 0
Lac Qui Parle 31 31 2,093 2,390 36
Lewis & Clark 54 54 16,432 12,655 386
St. John the Baptist 35 35 9,039 10,305 74
```

Election results
Note "County" and "Parish" missing in county name

Find/eliminate the word "County"

Try to do this by using only these string manipulation functions:

strsplit(), nchar(), substr(), paste()

… groups come up with pseudo-code/code

```
> ctyNames
[1] "De Witt County"  "Lac qui Parle County"
[3] "Lewis and Clark County"   "St John the Baptist Parish"
```

Example – Find the word "County"

Use nchar() to find the length of the string and then substr() to drop the last 6 characters (or 7?)

```
substr(ctyNames, 1, nchar(ctyNames)-7)
[1] "De Witt"   "Lac qui Parle"  "Lewis and Clark"
[4] "St John the Baptist"
```

That was lucky – both Parish and County have 6 letters…

Example – Find the word "County"

strsplit() on blank character, examine the last element, drop it, and paste() back together

```
> words = strsplit(countynames, split=" ")
> sapply(words, function(x) {paste(x[-length(x)],
   collapse=" ")})
[1] "De Witt" "Lac qui Parle"
[3] "Lewis and Clark" "St John the Baptist"
```

Example – Find the word "County"

strsplit() on "" and search through for "C" followed by "o" followed by "u" …

```
> letters = unlist(strsplit(string, ""))
> el = substring(pattern, 1, 1)
> possibles = which(letters == el)
> possibles
[1] 9
> pattern == substring(string, possibles, possibles +
    nchar(pattern) - 1)
[1] TRUE
```

Regular Expressions

- Domain specific language - What do you need?
- Atoms or units (e.g. a character)
- Compose into patterns – a sequence of characters – match literals
- Express other types of structure – non-literal characters, meta-characters
  - e.g. beginning of a line or word,
  - Short-hand for a concept – equivalence class of characters
- Other shorthand – 0-9 is easier to read and less likely to make a mistake than 0123456789

Regular Expressions

```
> regexpr("County", ctyNames)
[1]  9 15 17 -1
attr(,"match.length")
[1]  6  6  6 -1
> grep("County", ctyNames)
[1] 1 2 3
> gsub("County", "", ctyNames)
[1] "De Witt "              "Lac qui Parle "
[3] "Lewis and Clark "         "St John the Baptist
    Parish"
```

Match the literal string "County" character by character

The functions

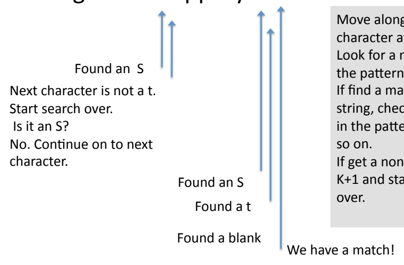
grep(), regexpr(), gsub()

can be used with fixed = TRUE to apply to literal pattern matching rather than treating the contents of the pattern as from the regular expression language.

Regular Expression matching

Pattern : "St " - note that there are 3 characters

String: "The Slippery St Francis"



Move along string left to right, one character at a time.
Look for a match to the first literal in the pattern.
If find a match at position K in the string, check that the second literal in the pattern is in position K+1, and so on.
If get a non match, return to position K+1 and start the matching process over.

Meta Characters

We need a way to express

- white space
- word boundaries
- sets or classes of literals
- the beginning and end of a line
- alternatives (”war” or “peace”)

Types of Meta Characters

- Start or end of line – e.g. find , "***" at the beginning of the string
- Start or end of word – e.g. match , "St" in , "St. John" and not , "St" in , "Street"
- Equivalent characters – e.g. match either c or C in
- Multiplicities, e.g. any number of punctuation marks in rep1!c@ted
- Sub-expressions and alternatives – e.g. match , "GET" or , "POST"

Anchors

^ anchor the search to the beginning of the string

$ anchor the search to the end of the string

\b start end of word

"County$" – pattern that matches a string with County at the end of the string.

"\bCounty" – pattern that matches a word that starts with County

Equivalence Classes

- [and] to express character classes (or equivalence classes) , e.g. Mc[cC]ain
- [^xyz] any characters but x, y, and z
- Named equivalence classes, e.g.

[:alpha:] – any alphabetic character
[:digit:] – any single digit
[:space:] – white space (e.g. blank, tab, return)
[:punct:] – any punctuation symbol

Equivalence Classes

```
subjects
[1] "Subject: Re: 90 days"
[2] "Subject: [SPAM:XXXXXXX]"
[3] "Subject: Fancy rep1!c@ted watches"

grep("[[:alpha:]][!,;:.?][[:alpha:]]", subjects)
[1] 2
 grep("[[:alpha:]][[:punct:]][[:alpha:]]", subjects)
[1] 2 3
grep("[[:alpha:]][!,;:.?[:digit:]][[:alpha:]]", subjects)
[1] 2
grep("[[:alpha:]][!,;:.?[:digit:]]+[[:alpha:]]", subjects)
[1] 2 3
```

Sub-patterns and alternatives

- (and) delimits a sub-pattern
- | defines alternatives

gsub(" County| Parish", "", countynames)
Why the blank before County and Parish?

gsub(" (County)|(Parish)", "", countynames)
Is this call to gsub the same as the previous?

Multiplicities

Preceding character or sub-pattern appears:
* 0 or more times
+ 1 or more times
? 0 or 1 time
{n} exactly n times
{m,n} between m and n times, inclusive

Escaping Meta Characters

"." The literal may be anything

If you want to search for a literal that is a meta character, then precede it with a \

In R, \ is a control character so any \ must be preceded by a \ in order to denote a backslash.

How would you search for a square bracket in a string? What about a backslash?

Greedy matches

Be careful with "." and "*"

For example, .* would match any literal any number of times, and so the entire string would be a match

Question: what does "<.*>" match in the following string:

"<p> This is a short paragraph.</p>"

Variables

It is possible to reference a matched pattern.

```
> web
[1] "169.237.46.168 - - [26/Jan/2004:10:47:58 -0800]
   \"GET /stat141/Winter04 HTTP/1.1\" 301 308"
```

```
gsub('(.*) - - \\[(.*) [-+][0-9]{4}\\] "(GET|POST).*',
       '\\1, \\2, \\3', web)
[1] "169.237.46.168, 26/Jan/2004:10:47:58, GET"
```

Each parenthetical pattern is treated as a variable.
The first is \\1, the second is \\2. What is gsub() doing?

Text Mining
Sample - State of the Union Addresses

***

State of the Union Address
George Washington
December 8, 1790

Fellow-Citizens of the Senate and House of Representatives:

In meeting you again I feel much satisfaction in being able to repeat my congratulations on the favorable prospects which continue to distinguish our public affairs. The abundant fruits of another year have blessed our country with plenty and with the means of a flourishing commerce.

State of the Union Address

The goal of this homework is for you to analyze the "State of the Union" speeches from 1776 to 2008. To do this you will need to prepare the data in a form that is suitable for statistical analysis. In particular, you will be examining the words that each president used in his address and their frequency of use. With this information, you can compare the presidents and see how they differ across time and party.

- **Preparation**: Use R to chop the document up into a list called speeches, where each speech is a character vector, one element for each sentence
- **Explore**: Use summary statistics and plots to compare the speeches. Does any president stand out? Do presidents from the same era give similar speeches?
- **Word Vectors**: There are over 12,000 unique (stemmed) words in all speeches. Create a word vector for each speech. Also compute the document frequency for each word.

- **Distances between speeches**: Use the Shannon-Jensen metric to compute the distance between the word vectors
- **Analysis:** Analyze the speeches according to their word frequencies. Try using multi-dimensional scaling and clustering. Produce a visualization of the results.
- **Turn In:** Write up one page on your findings in the exploratory and final analysis, include 2 exploratory plots and 2 analysis plots. Include all code in a plain text file. Your write up should contain six interesting observations about the presidents' speeches. Use historical background to help corroborate your findings.