Strings: A Brief Review

Intuitively, a string is a series of letters, numbers, and other symbols. R stores these using the R character type.

```r
my.strings = character(3)
my.strings[1] = "Strings"
my.strings[2] = "Are"
my.strings[3] = "Fantastic!"
my.strings
```

```r
## [1] "Strings" "Are" "Fantastic!"
```

```r
class(my.strings)
```

```r
## [1] "character"
```

```r
typeof(my.strings)
```

```r
## [1] "character"
```

More About Strings

R treats strings and vectors differently, which means we have new functions for manipulating strings. `nchar` instead of `nrow` or `length`

```r
nchar(my.strings)
```

```r
## [1] 7 3 10
```

```r
nrow(my.strings)
```

```r
## NULL
```

```r
length(my.strings)
```

```r
## [1] 3
```
### Substring

We used slices of vectors, matrices, etc. For strings, we use substrings.

```r
code
substr(my.strings, start=1, stop=4)
```

## 

```
# [1] "Stri" "Are" "Fant"
```

```r
code
substring(my.strings, first=5)
```

## 

```
# [1] "ngs" "" "astic!"
```

`substr` and `substring` have some subtle differences.

---

### Regular Expressions: A Brief Review

We’ll often want to extract interesting parts of a string. Regular expressions provide a way to describe those interesting parts. You’ve had a comprehensive introduction to these in lecture, I’ll just highlight a few important points.

We can use plain strings:

```r
code
look.for = "ing"
grepl(look.for, my.strings)
```

## 

```
# [1] TRUE FALSE FALSE
```

We can specify ranges of characters, e.g. `[A-Z]1, [1-9], [A-z]`. Can also use POSIX expressions like `[:space:]` and `[:punct:]`. Can also look for anything besides a newline, using `'`

We can specify how many times we want to see something, e.g. `*` `[A-Z]` (*Any capital letters zero or more times*) `'[1-9]+` (Any numbers one or more times)

---

### Excluding Things

We can exclude things. This is important because regex matching is greedy.
my.string = "<HTML>words<TAG>"
strsplit(my.string,"<.*>"

## [[1]
## [1] ""
strsplit(my.string,"<[^>]*>"

## [[1]
## [1] ""  "words"

Our text for the day: The second Republican primary debate (courtesy CNN):

linc = readLines("~/Documents/CompStats/RepublicanDebate.clean.txt")
length(linc)

## [1] 894

hist(nchar(linc))

Histogram of nchar(linc)

Each line is a paragraph. We can have some fun parsing the text!
The transcript has a separate line for APPLAUSE. How many times did the audience applaud?

```
grep('APPLAUSE',linc)
```

```r
## [1] 4 8 10 17 19 21 23 25 27 29 36 38 42 55 77 128 138
## [18] 142 154 165 173 184 188 220 242 251 266 269 292 295 337 341 343 347
## [35] 364 370 393 418 441 447 471 476 495 504 548 550 557 581 589
## [52] 596 605 614 618 634 636 637 660 667 671 686 704 709 717 724 753 756
## [69] 789 802 819 822 825 829 835 840 849 853 862 872 874 876 878 880 882
## [86] 884 886 888 890 892 894

length(grep('APPLAUSE',linc))
```

```r
## [1] 91
```

Here's another way to get a number:

```
head(grepl('APPLAUSE',linc),10)
```

```r
## [1] FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE TRUE

sum(grepl('APPLAUSE',linc))
```

```r
## [1] 91
```

Note that `grep` and `grepl` return the same information, but in different forms.

How many times did each candidate speak?

```
candidates = c('BUSH', 'TRUMP', 'FIORINA', 'RUBIO', 'CARSON', 'CRUZ', 'WALKER', 'KASICH', 'CHRISTIE', 'PAUL', 'HUCKABEE')
spoke = rep(0,length(candidates))
names(spoke)=candidates
for (name in candidates){
  spoke[name] = length(grep(name,linc))
}
spoke
```

```r
## BUSH TRUMP FIORINA RUBIO CARSON CRUZ WALKER KASICH
## 99 108 51 30 30 26 30 25
## CHRISTIE PAUL HUCKABEE
## 31 35 20
```

This code searches the full text for lines that contain the candidate names (in ALL CAPS).

How many times did each candidate mention jobs?
jobscount = rep(0,length(candidates))
names(jobscount) = candidates
for (name in candidates){
    name.index = grepl(name,linc)
    jobscount[name] = sum(grepl('[Jj]obs',linc[name.index]))
}
jobscount

## BUSH TRUMP FIORINA RUBIO CARSON CRUZ WALKER KASICH
## 0 1 1 2 0 1 2 2
## CHRISTIE PAUL HUCKABEE
## 0 1 1

Does this actually count the number of times each candidate used the word?

Another pass at counting jobs references:

realjobscount = rep(0,length(candidates))
names(realjobscount) = candidates
for (name in candidates){
    name.index = grepl(name,linc)
    allwords = strsplit(linc[name.index],"(\[[[:space:]]|\[[[:punct:]]]\])+")
    allwords = unlist(allwords)
    word.table = table(allwords)
    jobsct = 0
    if (!is.na(word.table['jobs'])){
        jobsct = jobsct + word.table['jobs']
    }
    if (!is.na(word.table['Jobs'])){
        jobsct = jobsct + word.table['Jobs']
    }
    realjobscount[name] = jobsct
}
realjobscount

## BUSH TRUMP FIORINA RUBIO CARSON CRUZ WALKER KASICH
## 0 1 5 3 0 1 10 2
## CHRISTIE PAUL HUCKABEE
## 0 2 2

Note that here, `strsplit` returns a list of lists. `unlist` converts that into one flat list, which is more convenient for `table`. What would happen if we ran this code without the `if` statements?
Other things we could do:

- Frequently used words for each candidate
- Unusual words for each candidate
- Time series for words of interest
- ???

Another source: MLB Standings (via ESPN)

```r
linc = readLines("http://espn.go.com/mlb/standings")

## Warning in readLines("http://espn.go.com/mlb/standings"): incomplete final
## line found on 'http://espn.go.com/mlb/standings'

length(linc)

## [1] 407

line = linc[249]

nchar(line)

## [1] 49141

In the source for this page all of the data are in one line.

Let’s split up the line with all the data:

Here’s an excerpt from that line:

```html
<span class="teams sprite-mlb-teams-25 sprite-25-team-14"></span><a name="&lpos=mlb:standings:team" href="/mlb/team/_/name/nyy"><span>
```

What can we use to split?

Splitting the string

```r
teams = strsplit(line,"class=""")[[1]]
teams[[2]]

## [1] "Toronto Blue Jays"<abbr title="Toronto Blue Jays">TOR</abbr>
```
teams = teams[2:length(teams)]
num.teams = length(teams)
num.teams

## [1] 30

Note that there are 30 MLB teams. Phew!

The data are surrounded by HTML tags – let’s split on the tags.

Tags start with “<” and end with “>”.

teaminfo = strsplit(teams,"<[^>]*>")
teaminfo[[1]]

## [1] "Toronto Blue Jays" "" "TOR"
## [4] "" "" ""
## [7] "" "90" ""
## [10] "65" "" ".581"
## [13] "" "" ""
## [16] "53-28" "" "37-37"
## [19] "" "853" ""
## [22] "631" "" "+222"
## [25] "" ".4" ""
## [28] "7-3" "" "100.0%"
## [31] "" "" ""
## [34] "" "" ""
## [37] "" "" ""
## [40] "<span "

teaminfo[[2]]

## [1] "New York Yankees" "" "NYY"
## [4] "" "" ""
## [7] "" "86" ""
## [10] "69" "" ".555"
## [13] "" "4" ""
## [16] "44-33" "" "42-36"
## [19] "" "741" ""
## [22] "651" "" "+90"
## [25] "" ".2" ""
## [28] "6-4" "" "99.9%"
## [31] "" "" ""
## [34] "" "" ""
## [37] "" "" ""
## [40] "<span "
Let’s put some of the data into a data frame:

```r
teamnames = character(num.teams)
wins = rep(0, num.teams)
losses = rep(0, num.teams)
for (i in 1:30){
    teamnames[i] = teaminfo[[i]][[1]]
    wins[i] = as.numeric( teaminfo[[i]][[8]] )
    losses[i] = as.numeric( teaminfo[[i]][[10]] )
}
w1 = data.frame(wins=wins, losses=losses)
rownames(w1) = teamnames
w1
```

```
## wins losses
## Toronto Blue Jays 90 65
## New York Yankees 86 69
## Baltimore Orioles 76 79
## Boston Red Sox 75 80
## Tampa Bay Rays 75 81
## Kansas City Royals 90 65
## Minnesota Twins 80 75
## Cleveland Indians 77 77
## Chicago White Sox 73 83
## Detroit Tigers 72 83
## Texas Rangers 84 71
## Houston Astros 82 74
## Los Angeles Angels 81 74
## Seattle Mariners 74 82
## Oakland Athletics 65 91
## New York Mets 89 67
## Washington Nationals 79 76
## Miami Marlins 69 87
## Atlanta Braves 62 94
## Philadelphia Phillies 59 97
## St. Louis Cardinals 98 58
## Pittsburgh Pirates 95 61
## Chicago Cubs 90 65
## Milwaukee Brewers 66 90
## Cincinnati Reds 63 92
## Los Angeles Dodgers 87 68
## San Francisco Giants 81 74
## Arizona Diamondbacks 75 81
## San Diego Padres 73 83
## Colorado Rockies 66 90
```

Which teams have 90 or more wins?
<table>
<thead>
<tr>
<th></th>
<th>wins</th>
<th>losses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toronto Blue Jays</td>
<td>90</td>
<td>65</td>
</tr>
<tr>
<td>Kansas City Royals</td>
<td>90</td>
<td>65</td>
</tr>
<tr>
<td>St. Louis Cardinals</td>
<td>98</td>
<td>58</td>
</tr>
<tr>
<td>Pittsburgh Pirates</td>
<td>95</td>
<td>61</td>
</tr>
<tr>
<td>Chicago Cubs</td>
<td>90</td>
<td>65</td>
</tr>
</tbody>
</table>

Let’s go Bucs!