Agenda

- Pattern matching in strings – why?
- Basics of regular expressions
  - literals, classes, modifiers
- Search/Replace with grep() and gsub()
- Live examples
- Recommended reading: “?regexp” in R
Many text processing problems involving automatically matching different, but regularly structured parts of the text

Want an expressive method for describing these patterns of characters
Examples
Extract the MPG related columns

```r
> cars <- read.csv('http://www.stat.cmu.edu/~cshalizi/statcomp/labs/07/epafuel2010.csv')
> names(cars)
[1] "year"                  "manufacturer"
[3] "division"              "country"
[5] "carline"               "manufacturer.code"
[7] "model.type"            "displ"
[9] "cylinders"             "mpg.city"
[11] "mpg.hwy"               "mpg.combined"
[13] "mpg.city.unadjusted"   "mpg.hwy.unadjusted"
[15] "mpg.combined.unadjusted" "aspiration"
[17] "transmission"          "gears"
[21] "X2Dr.Pass.Vol"         "X2Dr.Lugg.Vol"
[23] "X4Dr.Pass.Vol"         "X4Dr.Lugg.Vol"
[27] "epa.annual.fuel.cost"  "vehicle.class"
[29] "vehicle.type"          "batteries"
```
Amateur approach: *scroll and count*

Pro approach: *grep* and *regular expressions*

```r
> names(cars)
[1] "year"                                "manufacturer"
[3] "division"                             "country"
[5] "carline"                             "manufacturer.code"
[7] "model.type"                           "displ"
[9] "cylinders"                           mpg.city
[15] mpg.combined.unadjusted                "aspiration"
[17] "transmission"                        "gears"
[21] "X2Dr.Pass.Vol"                       "X2Dr.Lugg.Vol"
[23] "X4Dr.Pass.Vol"                       "X4Dr.Lugg.Vol"
[27] "epaannualfuelcost"                   "vehicle.class"
[29] "vehicle.type"                        "batteries"
```

```r
> grep('^mpg', names(cars))
[1] 10 11 12 13 14 15
```
Find the e-mail addresses

B
Anthony Brockwell
Adjunct Associate Professor

C
Jodi Casabianca
Postdoctoral Fellow
Office: Wean Hall 8119
Phone: (412) 268-6328
Email: jodicasa @ andrew.cmu.edu

D
Bernie Devlin
Adjunct Associate Professor
Office: University of Pittsburgh
Phone: (412) 624-1432

George T. Duncan
Emeritus Professor
Office: Hamburg Hall 2102D
Phone: (412) 268-2172
Email: gd17 @ andrew.cmu.edu
Home Page
Where are the e-mail addresses?
<ul class="peoplelisting">
  <li>
    <a name="B"><br /></a>
    <h3>B</h3>
    <p></p>
    <ul>
      <li>Anthony Brockwell<br />
          Adjunct Associate Professor</li>
    </ul>
  </li>
  <li>
    <a name="C"><br /></a>
    <h3>C</h3>
    <p></p>
    <ul>
      <li>Jodi Casabianca<br />
          Postdoctoral Fellow<br />
          Office: Wean Hall 8119<br />
          Phone: (412) 268-6328<br />
          Email: jodicasa @ andrew.cmu.edu</li>
    </ul>
  </li>
  <li>
    <a name="D"><br /></a>
    <h3>D</h3>
    <p></p>
    <ul>
      <li>Bernie Devlin<br />
          Adjunct Associate Professor<br />
          Office: University of Pittsburgh<br />
          Phone: (412) 624-1432</li>
      <li>George T. Duncan<br />
          Emeritus Professor<br />
          Office: Hamburg Hall 2102D<br />
          Phone: (412) 268-2172<br />
          Email: gd17 @ andrew.cmu.edu<br />
          <a href="http://duncan.heinz.cmu.edu/GeorgeWeb/">Home Page</a></li>
    </ul>
  </li>
</ul>

Where are the e-mail addresses?
What’s an e-mail address?

vqv@cmu.edu
cshalizi@cmu.edu

[alphanumeric]@[alphanumeric].[alpha]
What’s an e-mail address?

vqv@stat.cmu.edu
cshalizi@stat.cmu.edu

[alphanumeric]@[alphanumeric or .].[alpha]
Cain Speaks Out to Deny Charges; 2nd Voice Heard
By MICHAEL D. SHEAR and JIM RUTENBERG 18 minutes ago
Moments after a woman spoke publicly for the first time about being harassed by Herman Cain, the candidate held a news conference to again deny the charges.

Ohio Repeals Law Limiting Union Rights
By SABRINA TAVERNISE 17 minutes ago
Voters in Ohio delivered their verdict on a centerpiece of the conservative legislative agenda.

Hard-Fought Contests Watched for 2012 Clues
By KATHARINE Q. SEELYE 24 minutes ago
In Mississippi, a ballot question that could all but ban abortion boosted turnout. Gov. Steve Beshear was re-elected in Kentucky.

OPINION
On War and Redemption
A veteran of the Iraq and Afghanistan wars writes that it is not the sights, sounds and carnage of war that linger. It's the morality.
Find the links

Joe Paterno’s time as football coach will soon be over in the wake of a sex-abuse scandal, people briefed on the matter said.

Study Debunks Operation to Prevent Strokes

Doctors had hoped the operation would prevent strokes in people with poor circulation to the brain.

Ohio Repeals Law Limiting Union Rights

Voters in Ohio delivered their verdict on a centerpiece of the conservative legislative agenda.

Hard-Fought Contests Watched for 2012 Clues
What’s a link?

The part in orange is anchor text – it surrounds what we want.
Search/Replace

- Search strings for a pattern
- Replace substrings that match a pattern
- Very flexible if when regular expressions are used for patterns
y <- grep(pattern, x, ignore.case = FALSE, value = FALSE, fixed = FALSE)
Other search functions

- `grepl()` – returns logical vector of matching elements (does x contain pattern?)
- `regexpr()` – returns positions of first match
- `gregexpr()` – returns positions of all matches
`grepl()`

\[ y \leftarrow \text{grepl}(\text{pattern}, \ x, \ \text{ignore.case} = \text{FALSE}) \]

- **pattern** pattern to match
- **x** string to search within
- **ignore.case** ignore case when matching?
- Returns a logical vector indicating elements of x that matched
grep()
> fruits <- c(
  "apples and oranges and pears and bananas",
  "pineapples and mangos and guavas"
)

> gregexpr("mango", fruits)
[[1]]
[1] -1
attr("match.length")
[1] -1

[[2]]
[1] 16
attr("match.length")
[1] 5
\texttt{gsub()} \\

\texttt{y <- gsub(pattern, replacement, x, ignore.case = FALSE)} \\

- \texttt{pattern} pattern to match \\
- \texttt{replacement} replacement string \\
- \texttt{x} string to search within \\
- \texttt{ignore.case} ignore case when matching? \\
- Returns a string with occurrences of pattern replaced by \texttt{replacement}
> fruits <- c(
  "apples and oranges and pears and bananas",
  "pineapples and mangos and guavas"
)

> gsub("and", ",", fruits)
[1] "apples , oranges , pears , bananas"
[2] "pineapples , mangos , guavas"

> gsub("apples", "nuts", fruits)
[1] "nuts and oranges and pears and bananas"
[2] "pinenuts and mangos and guavas"
gsub()

> gsub(pattern = 'aa', replacement = 'b',
       x = 'aaacc')
[1] "bacc"
Regular Expressions

- Method of expressing patterns in strings
- Formally
  - Describe a set of strings
  - Matching is checking whether a given (sub)string belongs to the set
- Are strings themselves!
Syntax

• Regular expression syntax varies somewhat between languages

• R (>2.10.0) tries to be flexible
  • supports Perl syntax
  • supports POSIX 1003.2 standard (we will use this one)
Components

- Literal characters – match a single character
- Character classes – match any character in class
- Modifiers – operate on the above
## Literals

<table>
<thead>
<tr>
<th>regular expression</th>
<th>example matches*</th>
</tr>
</thead>
<tbody>
<tr>
<td>“a”</td>
<td>“apple”, “car”, “orchestra”</td>
</tr>
<tr>
<td>“ee”</td>
<td>“eel”, “between”, “feed”</td>
</tr>
<tr>
<td>“cat”</td>
<td>“cat”, “catch”, “scatter”</td>
</tr>
<tr>
<td>“19”</td>
<td>“19”, “1984”, “4519”</td>
</tr>
</tbody>
</table>

*more precisely: strings that contain matching substrings*
Special characters

These characters have special meaning:

.  ^  $  +  ?  *  (  )  [  ]  {  }  \  \\

Use the backslash \ to signal whether you want them to be taken literally:

\.  \^  \$  \+  \?  \*  \(  \)  \[  \]  \{  \}  \\  \\

Actually, 2 backslashes \\ in R, since \ is special for R too.
# Literals

<table>
<thead>
<tr>
<th>regular expression</th>
<th>example matches*</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;.com&quot;</td>
<td>&quot;google.com&quot;,&quot;apple.com&quot;</td>
</tr>
<tr>
<td>&quot;$100&quot;</td>
<td>&quot;$100&quot;,&quot;$100,999&quot;</td>
</tr>
<tr>
<td>&quot;!&quot;</td>
<td>&quot;!!!&quot;,&quot;bang!&quot;,&quot;omg!&quot;</td>
</tr>
</tbody>
</table>
Character classes

- Specified with square brackets [ ]
- Match any characters within square brackets
- Use dash “-” to specify a range of characters
- Use [^ ] to invert the class – all characters not inside[^ ]
# Character classes

<table>
<thead>
<tr>
<th>regular expression</th>
<th>example matches*</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ab]</td>
<td>“apple”, “bump”, “abacus”</td>
</tr>
<tr>
<td>[a-zA-Z]</td>
<td>“car”, “Vince Q.Vu”</td>
</tr>
<tr>
<td>[^0-9]</td>
<td>“apple”, “car”, “pic.jpg”</td>
</tr>
</tbody>
</table>
## Named classes

For convenience many commonly used classes can be referred to by name

<table>
<thead>
<tr>
<th>named class</th>
<th>members</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:alnum:]</td>
<td>alphanumeric characters</td>
</tr>
<tr>
<td>[:alpha:]</td>
<td>alphabetic characters</td>
</tr>
<tr>
<td>[:digit:]</td>
<td>digits 0 1 2 ... 9</td>
</tr>
<tr>
<td>[:space:]</td>
<td>whitespace characters</td>
</tr>
<tr>
<td>[:punct:]</td>
<td>punctuation characters</td>
</tr>
<tr>
<td>[:graph:]</td>
<td>[:alnum:] and [:punct:]</td>
</tr>
</tbody>
</table>

* see help for ‘regex’ in R for more
Dot

“.” matches any single character except newline

<table>
<thead>
<tr>
<th>regular expression</th>
<th>example matches*</th>
</tr>
</thead>
<tbody>
<tr>
<td>“c.t”</td>
<td>“cat”, “mascot”, “c0t”</td>
</tr>
<tr>
<td>“.ing”</td>
<td>“king”, “ding”, “9ing”</td>
</tr>
<tr>
<td>“[:alnum:]”</td>
<td>“hw5”, “vqv1978”</td>
</tr>
</tbody>
</table>
Modifiers

Allow us to construct more complex patterns

<table>
<thead>
<tr>
<th>modifier</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>anchor expression to beginning</td>
</tr>
<tr>
<td>$</td>
<td>anchor expression to end</td>
</tr>
<tr>
<td>*</td>
<td>match 0 or more occurrences of preceding</td>
</tr>
<tr>
<td>?</td>
<td>match 0 or 1 occurrences of preceding</td>
</tr>
<tr>
<td>+</td>
<td>match 1 or more occurrences of preceding</td>
</tr>
<tr>
<td>{n}</td>
<td>match exactly $n$ occurrences of preceding</td>
</tr>
<tr>
<td>{n,}</td>
<td>match at least $n$ occurrences of preceding</td>
</tr>
<tr>
<td>{n,m}</td>
<td>match between $n$ and $m$ occurrences</td>
</tr>
<tr>
<td>( )</td>
<td>group patterns together</td>
</tr>
</tbody>
</table>
# Modifiers

<table>
<thead>
<tr>
<th>regular expression</th>
<th>example matches*</th>
</tr>
</thead>
<tbody>
<tr>
<td>“aa+”</td>
<td>“aardvark”, “aaa”</td>
</tr>
<tr>
<td>“[alnum:]+.jpg”</td>
<td>“picture01.jpg”, “filename.jpg”</td>
</tr>
</tbody>
</table>
Warning!

The quantifiers

+ * {n,}

are greedy!
**Modifiers**

<table>
<thead>
<tr>
<th>regular expression</th>
<th>example matches*</th>
</tr>
</thead>
<tbody>
<tr>
<td>“&lt;.*&gt;”</td>
<td>“&lt;July 11&gt; &lt;June 10&gt;”</td>
</tr>
</tbody>
</table>
Alternatives (ORing)

- Use the vertical bar ‘|’ to separate alternative patterns
- This is like the logical OR operation, or taking the union of sets
## Alternatives

<table>
<thead>
<tr>
<th>regular expression</th>
<th>example matches*</th>
</tr>
</thead>
<tbody>
<tr>
<td>“apple</td>
<td>orange”</td>
</tr>
<tr>
<td>“cat</td>
<td>dog”</td>
</tr>
<tr>
<td>“.edu</td>
<td>.org</td>
</tr>
</tbody>
</table>
Regular Expressions in R

• Regular expressions are simply character strings in R

• Need to double backslash \ to signal specials to be taken literally – because \ is a special character in R too

• Can use R to programmatically construct regular expressions
Summary

- Use regular expressions to describe patterns
- `grep()` to search strings for matches to a pattern
- `gsub()` to replace matching patterns in a string
- Next: Regular Expressions II