Professor Bear :: Regular Expressions in R

Nik Bear Brown

September 1, 2018

In this lesson we'll learn regular expressions.

Additional packages needed

To run the code in this lesson you may need additional packages.

- If necessary install the followings packages.

```
None
```

Data

We will be using a list of bears, some poems, some cartoons, as well as Dr Suess quotes.

```
bears <- c("bear", "Winnie-the-Pooh", "Yogi", "Smokey the Bear", "teddy
bear","Baloo","Iorek Byrnison","Rupert the dapper bear","John Lewis
bear","Fozzie","Teddy Ruxpin","Aloysius teddy bear","Lots-O'-Huggin' Bear")
seuss <- c("You have brains in your head.",
           "You have feet in your shoes.",
           "You can steer yourself any direction you choose.",
           "You're on your own.",
           "And you know what you know.",
           "And YOU are the one who'll decide where to go...",
           "- Dr. Seuss")
cummings<-c("(I do not know what it is about you that closes",
"and opens;only something in me understands",
"the voice of your eyes is deeper than all roses)",
"nobody,not even the rain,has such small hands",
"-- e. e. cummings,",
"*somewhere i have never travelled,gladly beyond*")
title <- c("Steven Universe","Regular Show","The Amazing World of
Gumball","Adventure Time","Adventure Time","Clarence","Rick and Morty","Rick
and Morty","Simpsons","Simpsons","Bob's Burgers","Bob's Burgers","Steven
Universe","Regular Show","The Amazing World of Gumball","Adventure
Time","Adventure Time","Clarence","Rick and Morty","Rick and
Morty","Simpsons","Bob's Burgers")
date <-c("Jan 5, 2016","Aug 6, 2015","Aug 19, 2016","Apr 23, 2016","Jan 13,
2016","Mar 24, 2016","Apr 7, 2014","Apr 14, 2014","Sep 25, 2016","May 22,
2016","Apr 3, 2016","Apr 17, 2016","Jan 7, 2016","Jun 23, 2015","Sep 5,
2016","May 22, 2013","May 27, 2013","Mar 25, 2016","Sep 13, 2015","Oct 4,
2015"," May 15, 2016","May 22, 2016")
episode <- c("Steven's Birthday","The Parkie Awards","The Detective; The Fury
Part 1","The Great Birdman; Simon & Marcy","Princess Potluck; James Baxter
the Horse","Sneaky Peeky","Close Rick-Counters of the Rick Kind","Ricksy
Business","Monty Burns' Fleeing Circus","Orange Is the New Yellow","Stand by
Gene","The Hormone-iums","Message Received","Death Kwon Do-Livery","The Re-
Run","I Am a Sword; Ghost Fly","Blank Eyed Girl; President Porpoise Is
Missing","Clarence Wendle and the Eye of Coogan","Big Trouble in Little
Sanchez","The Wedding Squanchers","Simprovised","Glued, Where's My Bob?")
season <- c(2,7,4,5,5,2,1,1,28,28,6,6,2,6,5,7,7,2,2,2,27,7)
cartoons <- data.frame(title,episode,date,season)
bears
```

```
##  [1] "bear"                    "Winnie-the-Pooh"
##  [3] "Yogi"                    "Smokey the Bear"
##  [5] "teddy bear"              "Baloo"
##  [7] "Iorek Byrnison"          "Rupert the dapper bear"
##  [9] "John Lewis bear"         "Fozzie"
## [11] "Teddy Ruxpin"            "Aloysius teddy bear"
## [13] "Lots-O'-Huggin' Bear"
```

```
seuss
```

```
## [1] "You have brains in your head."
## [2] "You have feet in your shoes."
## [3] "You can steer yourself any direction you choose."
## [4] "You're on your own."
## [5] "And you know what you know."
## [6] "And YOU are the one who'll decide where to go..."
## [7] "- Dr. Seuss"
```

```
cummings
```

```
## [1] "(I do not know what it is about you that closes"
## [2] "and opens;only something in me understands"
## [3] "the voice of your eyes is deeper than all roses)"
## [4] "nobody,not even the rain,has such small hands"
## [5] "-- e. e. cummings,"
## [6] "*somewhere i have never travelled,gladly beyond*"
```

```
cartoons
```

```
##                            title
## 1                Steven Universe
## 2                   Regular Show
## 3   The Amazing World of Gumball
## 4                 Adventure Time
## 5                 Adventure Time
## 6                       Clarence
## 7                  Rick and Morty
## 8                  Rick and Morty
## 9                    Simpsons
## 10                   Simpsons
## 11              Bob's Burgers
## 12              Bob's Burgers
## 13            Steven Universe
## 14               Regular Show
## 15 The Amazing World of Gumball
## 16             Adventure Time
## 17             Adventure Time
## 18                   Clarence
## 19             Rick and Morty
## 20             Rick and Morty
## 21                   Simpsons
## 22              Bob's Burgers
##                                        episode         date season
## 1                            Steven's Birthday  Jan 5, 2016      2
## 2                            The Parkie Awards  Aug 6, 2015      7
## 3                The Detective; The Fury Part 1 Aug 19, 2016      4
## 4              The Great Birdman; Simon & Marcy Apr 23, 2016      5
## 5        Princess Potluck; James Baxter the Horse Jan 13, 2016      5
## 6                                 Sneaky Peeky Mar 24, 2016      2
## 7            Close Rick-Counters of the Rick Kind  Apr 7, 2014      1
## 8                              Ricksy Business Apr 14, 2014      1
## 9                     Monty Burns' Fleeing Circus Sep 25, 2016     28
## 10                   Orange Is the New Yellow May 22, 2016     28
## 11                               Stand by Gene  Apr 3, 2016      6
## 12                           The Hormone-iums Apr 17, 2016      6
## 13                           Message Received  Jan 7, 2016      2
## 14                      Death Kwon Do-Livery Jun 23, 2015      6
## 15                                 The Re-Run  Sep 5, 2016      5
## 16                      I Am a Sword; Ghost Fly May 22, 2013      7
## 17 Blank Eyed Girl; President Porpoise Is Missing May 27, 2013      7
## 18          Clarence Wendle and the Eye of Coogan Mar 25, 2016      2
## 19                 Big Trouble in Little Sanchez Sep 13, 2015      2
## 20                      The Wedding Squanchers  Oct 4, 2015      2
## 21                                 Simprovised May 15, 2016     27
## 22                        Glued, Where's My Bob? May 22, 2016      7
```

Regular Expressions and Grep

In theoretical computer science and formal language theory, a regular expression (abbreviated RegEx or RegExp and sometimes called a rational expression) is a sequence of characters that define a search pattern, mainly for use in pattern matching with strings, or string matching, i.e. “find and replace”-like operations. The concept arose in the 1950s, when the American mathematician Stephen Kleene formalized the description of a regular language, and came into common use with the Unix text processing utilities ed, an editor, and grep (global regular expression print), a filter.

grep is a command-line utility for searching plain-text data sets for lines matching a regular expression. Grep was originally developed for the Unix operating system, but is available today for all Unix-like systems and is built in to languages like python and Perl.

Basic Regex Syntax

Basic regular expression syntax

```
.   Normally matches any character except a newline.

When you match a pattern within parentheses, you can use any of $1, $2, ...
later to refer to the previously matched pattern.

+   Matches the preceding pattern element one or more times.
?   Matches the preceding pattern element zero or one times.
*   Matches the preceding pattern element zero or more times.
|   Separates alternate possibilities.

\w  Matches an alphanumeric character, including "_";  same as [A-Za-z0-9_]
in ASCII, and
[\p{Alphabetic}\p{GC=Mark}\p{GC=Decimal_Number}\p{GC=Connector_Punctuation}]

\W  Matches a non-alphanumeric character, excluding "_";
same as [^A-Za-z0-9_] in ASCII, and
[^\p{Alphabetic}\p{GC=Mark}\p{GC=Decimal_Number}\p{GC=Connector_Punctuation}]

\s  Matches a whitespace character,
which in ASCII are tab, line feed, form feed, carriage return, and space;

\S  Matches anything BUT a whitespace.

\d  Matches a digit;
same as [0-9] in ASCII;

\D  Matches a non-digit;

^   Matches the beginning of a line or string.

$   Matches the end of a line or string.
```

R’s basic RegEx commands

Let us start with a list of *bears*. Let’s use a RegEx to find all items in this list containing ‘bear’. The basic RegEx command in R is grep, which simply returns the index of the matching elements:

```
bears
```

```
##  [1] "bear"                     "Winnie-the-Pooh"
##  [3] "Yogi"                     "Smokey the Bear"
##  [5] "teddy bear"               "Baloo"
##  [7] "Iorek Byrnison"           "Rupert the dapper bear"
##  [9] "John Lewis bear"          "Fozzie"
## [11] "Teddy Ruxpin"             "Aloysius teddy bear"
## [13] "Lots-O'-Huggin' Bear"
```

```
m=grep(pattern = "bear", bears)
m
```

```
## [1]  1  5  8  9 12
```

```
bears[m]
```

```
## [1] "bear"                   "teddy bear"
## [3] "Rupert the dapper bear" "John Lewis bear"
## [5] "Aloysius teddy bear"
```

Note that *teddy bear* was included whereas *Smokey the Bear* was not: the matching pattern can appear anywhere in the text string but is case sensitive.

To make the search ignore cases, simply add the `ignore.case = T` argument.

```
m=grep("bear", bears, ignore.case = T)
m
```

```
## [1]  1  4  5  8  9 12 13
```

```
bears[m]
```

```
## [1] "bear"                   "Smokey the Bear"
## [3] "teddy bear"             "Rupert the dapper bear"
## [5] "John Lewis bear"        "Aloysius teddy bear"
## [7] "Lots-O'-Huggin' Bear"
```

`grepl` is the same as `grep`, only it outputs a yes/now output for each element:

```
grepl("bear", bears, ignore.case = T)
```

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

Besides *grep* and *grepl* there are a number of functions that take RegEx patterns as arguments.

```
grep(pattern, x, ignore.case = FALSE, perl = FALSE, value = FALSE,
     fixed = FALSE, useBytes = FALSE, invert = FALSE)

grepl(pattern, x, ignore.case = FALSE, perl = FALSE,
      fixed = FALSE, useBytes = FALSE)

sub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

gsub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE,
     fixed = FALSE, useBytes = FALSE)

regexpr(pattern, text, ignore.case = FALSE, perl = FALSE,
        fixed = FALSE, useBytes = FALSE)

gregexpr(pattern, text, ignore.case = FALSE, perl = FALSE,
         fixed = FALSE, useBytes = FALSE)

RegExec(pattern, text, ignore.case = FALSE, perl = FALSE,
        fixed = FALSE, useBytes = FALSE)
```

We often want to "tokenize" strings with regular expressions. For example. we may want to split on white space.

```
strsplit(bears, split = "[ ]+") # splits on one or more spaces

## [[1]]
## [1] "bear"
##
## [[2]]
## [1] "Winnie-the-Pooh"
##
## [[3]]
## [1] "Yogi"
##
## [[4]]
## [1] "Smokey" "the"    "Bear"
##
## [[5]]
## [1] "teddy" "bear"
##
## [[6]]
## [1] "Baloo"
##
## [[7]]
## [1] "Iorek"    "Byrnison"
##
## [[8]]
## [1] "Rupert" "the"    "dapper" "bear"
##
## [[9]]
## [1] "John"  "Lewis" "bear"
##
## [[10]]
## [1] "Fozzie"
##
## [[11]]
## [1] "Teddy"  "Ruxpin"
##
## [[12]]
## [1] "Aloysius" "teddy"    "bear"
##
## [[13]]
## [1] "Lots-O'-Huggin'" "Bear"
```

Finding and replacing in R

To search and replace the first instance of a pattern, use sub. Much more useful is gsub, which replaces all instances. To replace all instances of *bear* with *Teddy*, use the following:

```
gsub(pattern = "bear", replacement = "Teddy", bears, ignore.case = T)

##  [1] "Teddy"                        "Winnie-the-Pooh"
##  [3] "Yogi"                         "Smokey the Teddy"
##  [5] "teddy Teddy"                  "Baloo"
##  [7] "Iorek Byrnison"               "Rupert the dapper Teddy"
##  [9] "John Lewis Teddy"             "Fozzie"
## [11] "Teddy Ruxpin"                 "Aloysius teddy Teddy"
## [13] "Lots-O'-Huggin' Teddy"
```

What if we want to replace something by keeping what's there and adding something to it. For example, to add a space after a comma, colon or semi-colon.

```
gsub("([,:;])", "\\1 ", cummings, perl=T)

## [1] "(I do not know what it is about you that closes"
## [2] "and opens; only something in me understands"
## [3] "the voice of your eyes is deeper than all roses)"
## [4] "nobody, not even the rain, has such small hands"
## [5] "-- e. e. cummings, "
## [6] "*somewhere i have never travelled, gladly beyond*"

cummings

## [1] "(I do not know what it is about you that closes"
## [2] "and opens;only something in me understands"
## [3] "the voice of your eyes is deeper than all roses)"
## [4] "nobody,not even the rain,has such small hands"
## [5] "-- e. e. cummings,"
## [6] "*somewhere i have never travelled,gladly beyond*"
```

The above syntax is complicated so let's explain it.

- We have specified that we want Perl-esque RegEx with *perl=T*, allowing us to use groups.

- The paranthesis indicate the group to match. *([,:;])* reads as a group of comma, colon or semi-colon.

- The brackets refer to a list of characters we want to match *[,:;]* reads as comma or colon or semi-colon.

- the \\1 symbol means "replace this with the value captured in group 1", group 1 is the first pattern surrounded by paranthesis *()*

Regular Expression Examples

```
letters

##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"

grep("[a-z]", letters)

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26

grep("[A-Z]", letters)

## integer(0)

grep("[AB]", letters)

## integer(0)

grep("[AB]", letters,ignore.case = TRUE)

## [1] 1 2

grep("[a-zA-Z]", letters)

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26

grep("[azAZ]", letters)

## [1]  1 26

seuss

## [1] "You have brains in your head."
## [2] "You have feet in your shoes."
## [3] "You can steer yourself any direction you choose."
## [4] "You're on your own."
## [5] "And you know what you know."
## [6] "And YOU are the one who'll decide where to go..."
## [7] "- Dr. Seuss"

grep("you", seuss)
```

```
## [1] 1 2 3 4 5

if(length(i <- grep("in", seuss)))
  cat("'in' appears at least once in\n\t", seuss[i], "\n")

## 'in' appears at least once in
##   You have brains in your head. You have feet in your shoes.

i # 1 and 2

## [1] 1 2

seuss[i]

## [1] "You have brains in your head." "You have feet in your shoes."

 ## Modify all 'a' or 'b's;  "\" must be escaped
s<-gsub("([ab])", "\\1-z\\1_s", "abc vs, ABC  - Oh, A B C. It's easy as, 1 2
3. As simple as, do re mi. a b c, 1 2 3")
s

## [1] "a-za_sb-zb_sc vs, ABC  - Oh, A B C. It's ea-za_ssy a-za_ss, 1 2 3. As
simple a-za_ss, do re mi. a-za_s b-zb_s c, 1 2 3"

s<-gsub("([ab])", "-z_s", "abc vs, ABC  - Oh, A B C. It's easy as, 1 2 3. As
simple as, do re mi. a b c, 1 2 3")
s

## [1] "-z_s-z_sc vs, ABC  - Oh, A B C. It's e-z_ssy -z_ss, 1 2 3. As simple
-z_ss, do re mi. -z_s -z_s c, 1 2 3"

s<-gsub("([ab])", "\\a\\b", "abc vs, ABC  - Oh, A B C. It's easy as, 1 2 3.
As simple as, do re mi. a b c, 1 2 3")
s

## [1] "ababc vs, ABC  - Oh, A B C. It's eabsy abs, 1 2 3. As simple abs, do
re mi. ab ab c, 1 2 3"

stop.words <- c("The", "for", "are",
          "to",  "your",
         "to", "and", "the",
         "is", "your",  "and", "is",
         "it", "for", "all", "its", "it's")



seuss.re<-regexpr("[ ]+",seuss)
seuss.re

## [1] 4 4 4 7 4 4 2
## attr(,"match.length")
## [1] 1 1 1 1 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

```
seuss.re<-gregexpr("[ ]+",seuss)
seuss.re
```

```
## [[1]]
## [1]  4  9 16 19 24
## attr(,"match.length")
## [1] 1 1 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
## 
## [[2]]
## [1]  4  9 14 17 22
## attr(,"match.length")
## [1] 1 1 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
## 
## [[3]]
## [1]  4  8 14 23 27 37 41
## attr(,"match.length")
## [1] 1 1 1 1 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
## 
## [[4]]
## [1]  7 10 15
## attr(,"match.length")
## [1] 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
## 
## [[5]]
## [1]  4  8 13 18 22
## attr(,"match.length")
## [1] 1 1 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

```
##
## [[6]]
## [1]  4  8 12 16 20 27 34 40 43
## attr(,"match.length")
## [1] 1 1 1 1 1 1 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
##
## [[7]]
## [1] 2 6
## attr(,"match.length")
## [1] 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

```
## trim trailing white space
str <- " You have brains in your head      "
str
```

```
## [1] " You have brains in your head      "
```

```
str<-sub("[ ]+$", "", str)  ## trailing spaces only
str
```

```
## [1] " You have brains in your head"
```

```
str<-sub("^[ ]+", "", str)  ## leading spaces only
str
```

```
## [1] "You have brains in your head"
```

```
sub("[[:space:]]+$", "", str) ## white space, POSIX-style
```

```
## [1] "You have brains in your head"
```

```
sub("\\s+$", "", str, perl = TRUE) ## Perl-style white space
```

```
## [1] "You have brains in your head"
```

```
## capitalizing
txt <- "You have brains in your head"
gsub("(\\w)(\\w*)", "\\U\\1\\L\\2", txt, perl=TRUE)
```

```
## [1] "You Have Brains In Your Head"
```

```
gsub("\\b(\\w)",    "\\U\\1",       txt, perl=TRUE)
```

```
## [1] "You Have Brains In Your Head"
```

```
txt2 <- "You have feet in your shoes."
gsub("(\\w)(\\w*)(\\w)", "\\U\\1\\E\\2\\U\\3", txt2, perl=TRUE)

## [1] "YoU HavE FeeT IN YouR ShoeS."

sub("(\\w)(\\w*)(\\w)", "\\U\\1\\E\\2\\U\\3", txt2, perl=TRUE)

## [1] "YoU have feet in your shoes."

## Decompose a URL into its components.
url <- "http://nikbearbrown/machine/learning/"
m <- regexec("^(([^:]+)://)?([^:/]+)(:([0-9]+))?(/.*)", url)
m

## [[1]]
## [1]  1  1  1  8  0  0 20
## attr(,"match.length")
## [1] 37  7  4 12  0  0 18
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE

regmatches(url, m)

## [[1]]
## [1] "http://nikbearbrown/machine/learning/"
## [2] "http://"
## [3] "http"
## [4] "nikbearbrown"
## [5] ""
## [6] ""
## [7] "/machine/learning/"

## parts of a URL:
URL_parts <- function(u) {
  m <- regexec("^(([^:]+)://)?([^:/]+)(:([0-9]+))?(/.*)", u)
  parts <- do.call(rbind,
                   lapply(regmatches(u, m), `[`, c(3L, 4L, 6L, 7L)))
  colnames(parts) <- c("protocol","host","port","path")
  parts
}
URL_parts(url)

##      protocol host           port path
## [1,] "http"   "nikbearbrown" ""   "/machine/learning/"

# Spilt text
seuss

## [1] "You have brains in your head."
## [2] "You have feet in your shoes."
## [3] "You can steer yourself any direction you choose."
## [4] "You're on your own."
## [5] "And you know what you know."
## [6] "And YOU are the one who'll decide where to go..."
## [7] "- Dr. Seuss"
```

```
seuss[1]
```

```
## [1] "You have brains in your head."
```

```
sp<-strsplit(seuss[1], "a") # "You have brains in your head."
s<-"You    have   brains  in  your head. "
sp<-strsplit(seuss[1], " ")
sp
```

```
## [[1]]
## [1] "You"    "have"   "brains" "in"     "your"   "head."
```

```
sp<-strsplit(s," ")
sp
```

```
## [[1]]
##  [1] "You"    ""       ""       ""       "have"   ""       ""
##  [8] "brains" ""       "in"     ""       "your"   "head."
```

```
sp<-strsplit(s,"[ ]+")
sp
```

```
## [[1]]
## [1] "You"    "have"   "brains" "in"     "your"   "head."
```

```
sp<-strsplit(s,"[ ]+",perl = TRUE)
sp
```

```
## [[1]]
## [1] "You"    "have"   "brains" "in"     "your"   "head."
```

Cartoon data set

Episodes from 2015

```
cartoons
```

```
##                            title
## 1                Steven Universe
## 2                   Regular Show
## 3  The Amazing World of Gumball
## 4                 Adventure Time
## 5                 Adventure Time
## 6                       Clarence
## 7                 Rick and Morty
## 8                 Rick and Morty
## 9                     Simpsons
## 10                    Simpsons
## 11               Bob's Burgers
## 12               Bob's Burgers
## 13             Steven Universe
## 14                Regular Show
## 15 The Amazing World of Gumball
## 16              Adventure Time
## 17              Adventure Time
## 18                    Clarence
## 19              Rick and Morty
## 20              Rick and Morty
## 21                    Simpsons
## 22               Bob's Burgers
##                                       episode         date season
## 1                           Steven's Birthday  Jan 5, 2016      2
## 2                           The Parkie Awards  Aug 6, 2015      7
## 3                The Detective; The Fury Part 1 Aug 19, 2016      4
## 4              The Great Birdman; Simon & Marcy Apr 23, 2016      5
## 5     Princess Potluck; James Baxter the Horse Jan 13, 2016      5
## 6                                Sneaky Peeky Mar 24, 2016      2
## 7         Close Rick-Counters of the Rick Kind  Apr 7, 2014      1
## 8                             Ricksy Business Apr 14, 2014      1
## 9                 Monty Burns' Fleeing Circus Sep 25, 2016     28
## 10                  Orange Is the New Yellow May 22, 2016     28
## 11                              Stand by Gene  Apr 3, 2016      6
## 12                           The Hormone-iums Apr 17, 2016      6
## 13                           Message Received  Jan 7, 2016      2
## 14                       Death Kwon Do-Livery Jun 23, 2015      6
## 15                                 The Re-Run  Sep 5, 2016      5
## 16                     I Am a Sword; Ghost Fly May 22, 2013      7
## 17 Blank Eyed Girl; President Porpoise Is Missing May 27, 2013      7
## 18          Clarence Wendle and the Eye of Coogan Mar 25, 2016      2
## 19                 Big Trouble in Little Sanchez Sep 13, 2015      2
## 20                      The Wedding Squanchers  Oct 4, 2015      2
## 21                                 Simprovised May 15, 2016     27
## 22                       Glued, Where's My Bob? May 22, 2016      7
```

```
m<-grep("2015",cartoons$date) # Episodes from 2015
m
```

```
## [1]  2 14 19 20
```

```
cartoons[m,]
```

```
##             title                       episode         date season
## 2    Regular Show             The Parkie Awards  Aug 6, 2015      7
## 14   Regular Show          Death Kwon Do-Livery Jun 23, 2015      6
## 19 Rick and Morty Big Trouble in Little Sanchez Sep 13, 2015      2
## 20 Rick and Morty        The Wedding Squanchers  Oct 4, 2015      2
```

Episodes after 2010

We could try “201” as pattern but might get some false postives. What we want is“201” followed by one digit:

```
m<-grep("201.",cartoons$date) # "201" followed by one digit
m

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22

cartoons[m,]

##                           title
## 1               Steven Universe
## 2                  Regular Show
## 3  The Amazing World of Gumball
## 4                Adventure Time
## 5                Adventure Time
## 6                      Clarence
## 7                Rick and Morty
## 8                Rick and Morty
## 9                      Simpsons
## 10                     Simpsons
## 11               Bob's Burgers
## 12               Bob's Burgers
## 13              Steven Universe
## 14                 Regular Show
## 15 The Amazing World of Gumball
## 16               Adventure Time
## 17               Adventure Time
## 18                     Clarence
## 19               Rick and Morty
## 20               Rick and Morty
## 21                     Simpsons
## 22               Bob's Burgers
##                                      episode         date season
## 1                          Steven's Birthday   Jan 5, 2016      2
## 2                          The Parkie Awards   Aug 6, 2015      7
## 3                The Detective; The Fury Part 1  Aug 19, 2016      4
## 4              The Great Birdman; Simon & Marcy  Apr 23, 2016      5
## 5        Princess Potluck; James Baxter the Horse  Jan 13, 2016      5
## 6                                Sneaky Peeky  Mar 24, 2016      2
## 7              Close Rick-Counters of the Rick Kind   Apr 7, 2014      1
## 8                            Ricksy Business  Apr 14, 2014      1
## 9                   Monty Burns' Fleeing Circus  Sep 25, 2016     28
## 10                   Orange Is the New Yellow  May 22, 2016     28
## 11                               Stand by Gene   Apr 3, 2016      6
## 12                           The Hormone-iums  Apr 17, 2016      6
## 13                           Message Received   Jan 7, 2016      2
## 14                       Death Kwon Do-Livery  Jun 23, 2015      6
## 15                                 The Re-Run   Sep 5, 2016      5
## 16                        I Am a Sword; Ghost Fly  May 22, 2013      7
## 17 Blank Eyed Girl; President Porpoise Is Missing  May 27, 2013      7
## 18          Clarence Wendle and the Eye of Coogan  Mar 25, 2016      2
## 19                  Big Trouble in Little Sanchez  Sep 13, 2015      2
## 20                         The Wedding Squanchers   Oct 4, 2015      2
## 21                                    Simprovised  May 15, 2016     27
## 22                         Glued, Where's My Bob?  May 22, 2016      7
```

The "." acts as a wild card that matches any single character. Note that [0-9] stands for any digit (so does). Hence "20[0-9][0-9]" and "20" would work better (more restrictive) for all tests later than 2000.

Episodes from 2013 or 2014

We could try "201" as pattern but might get some false postives. What we want is "201" followed by one digit of 3 or 4.

```
m<-grep("201[34]",cartoons$date) # "201" followed by one digit of 3 or 4
m

## [1]  7  8 16 17

cartoons[m,]

##              title                                        episode
## 7  Rick and Morty           Close Rick-Counters of the Rick Kind
## 8  Rick and Morty                                Ricksy Business
## 16 Adventure Time                        I Am a Sword; Ghost Fly
## 17 Adventure Time Blank Eyed Girl; President Porpoise Is Missing
##            date season
## 7   Apr 7, 2014      1
## 8  Apr 14, 2014      1
## 16 May 22, 2013      7
## 17 May 27, 2013      7
```

Find all episodes in May 2013

The dates are in this form "May 22, 2013". The RegEx _"May[ ]+[0-9]+[,]*[ ]+2013"_ reads as May then one or more spaces *[ ]+* then one or more digits *[0-9]+* the zero or more commas *[,]** then one or more spaces *[ ]+* then 2013.

```
m<-grep("May[ ]+[0-9]+[,]*[ ]+2013",cartoons$date) # Find all episodes in May
2013
m

## [1] 16 17

cartoons[m,]

##              title                                        episode
## 16 Adventure Time                        I Am a Sword; Ghost Fly
## 17 Adventure Time Blank Eyed Girl; President Porpoise Is Missing
##             date season
## 16 May 22, 2013      7
## 17 May 27, 2013      7
```

Some simple RegEx examples

```
 {^[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?$}  # Floating Point Number

{^[A-Za-z]+$}   # Only letters.

 {^[[:alpha?:]]+$} # Only letters, the Unicode way.

 {(.)\1{3}} $string {\1} result # Back References

(\[0-9]{1,3})\.(\[0-9]{1,3})\.(\[0-9]{1,3})\.(\[0-9]{1,3}) # IP Numbers
```

Regular expression testing websites

There are a number of regular expression testing websites. It a good way to check if your *pattern* does what you think it does. Many also generate an english summary of what a regular expression does so they can be used to understand regular expressions that one might find on the web.

Regular expression testing websites
* regex101.com
* RegExpal.com
* regextester.com
* regexr.com

Resources

- regular-expressions.info
- Tech Stuff - Regular Expressions - A Gentle User Guide and Tutorial
- Learning to Use Regular Expressions - Gnosis.cx