lecture four

Yi (Chris) Chen

September 29, 2017

lecture four

character strings and string operations

1. string is a symbol in a written language that anything you can enter on a keyboard
2. a string is a sequence of characters.

```
#character can be many types: scalars, vectors, matrices,list or dataframe.
mode('d')
```

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

```
mode('cat,squirrel')
```

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

```
class('d')
```

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

```
class('cat,squirrel')
```

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

```
# nchar() is the function that calculate the number of characters in a given
string.
nchar(' ');nchar('  ');nchar('')
```

```
## [1] 1
```

```
## [1] 2
```

```
## [1] 0
```

```
class('\n')
```

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

```
class("\"")
```

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

```
class("\t")
```

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

```
nchar("\n") # create new line
```

```
## [1] 1
```

```
nchar("\"") # quotes within a string
```

```
## [1] 1
```

```
nchar("\t") # tab
```

```
## [1] 1
```

string as elements of a vector

```
# length calculate the number of string rather than the number of
elements(characters) in this string
length("cat,squirrel,hedgehog")
```

```
## [1] 1
```

```
length(c("cat,squirrel,hedgehog"))
```

```
## [1] 1
```

```
#nchar calculate the number of character in the string.
nchar("cat,squirrel,hedgehog")
```

```
## [1] 21
```

```
nchar(c("cat,squirrel,hedgehog"))
```

```
## [1] 21
```

display the string

```
print("cat,squirrel")
```

```
## [1] "cat,squirrel"
```

```
# cat() coerces its argument to strings, so can be useful when printing
warnings
cat('cat,squirrel')
```

```
## cat,squirrel
```

```
x <- 6
y <- 7
cat('I have',x,'cats and',y,'hedgehogs as pets')
```

```
## I have 6 cats and 7 hedgehogs as pets
```

```
print('cat, \nsquirrel')  #\n in print() will not change the line
```

```
## [1] "cat, \nsquirrel"
```

```
cat('cat, \nsquirrel')  # \n in cat() will change the line
```

```
## cat,
## squirrel

print('In R, an \"array"\ is a muti-dimension matrix.')

## [1] "In R, an \"array\" is a muti-dimension matrix."

cat("A group of hedgehogs is called an \"array\".")

## A group of hedgehogs is called an "array".

# \t means tab
print('columbia\tuniversity')

## [1] "columbia\tuniversity"

cat('columbnia\tuniversity')

## columbnia    university
```

substrings

```
phrase <- 'Christmas Bonus'
substr(phrase,start = 8,stop = 12)

## [1] "as Bo"

# take the substring of the string phrase from the 8th character to the 12ed
character
substr(phrase,start = 13, stop = 13) <- 'g'
# revalue the subtring
phrase

## [1] "Christmas Bogus"

fav_animals <- c('cat','squirrl','hedgehog')
# when use substr() for a list, it substring every element in the list
substr(fav_animals,start = 1, stop = 2)

## [1] "ca" "sq" "he"

substr(fav_animals,start = nchar(fav_animals)-1,stop = nchar(fav_animals))

## [1] "at" "rl" "og"

substr(fav_animals,start = 4, stop = 4)

## [1] ""  "i" "g"
```

dividing strings into vectors

```
# dividing the strings into vectors for different symbol
todo <- 'lecture, lab, homework'
strsplit(todo,split = ',')
```

```
## [[1]]
## [1] "lecture"   " lab"      " homework"
```

```
strsplit(c(todo,'midterm, final'),split = ',')
```

```
## [[1]]
## [1] "lecture"   " lab"      " homework"
##
## [[2]]
## [1] "midterm" " final"
```

```
# paste the elements together
paste('cat','squirrel','hedge')
```

```
## [1] "cat squirrel hedge"
```

```
paste('cat','squirrel','hedge',sep = ',')
```

```
## [1] "cat,squirrel,hedge"
```

```
animals <- c('cat','suqirrel','hedgrhog')
# paste can match the elements one to one automatically
paste(animals,1:3)
```

```
## [1] "cat 1"      "suqirrel 2" "hedgrhog 3"
```

```
paste(animals,"(",1:3,")")
```

```
## [1] "cat ( 1 )"      "suqirrel ( 2 )" "hedgrhog ( 3 )"
```

```
paste(animals, "(" , 1:3 , ")" ,sep= "" , callaps = ";" )
```

```
## [1] "cat(1);"      "suqirrel(2);" "hedgrhog(3);"
```

```
# test
lum_vector <- c("columbia","slumber party","sugarplum")
substr(lum_vector,start= c(3,2,7),stop = c(5,4,9))
```

```
## [1] "lum" "lum" "lum"
```

```
# paste: turns vectors into a string
strsplit(lum_vector,split = 'lum')
```

```
## [[1]]
## [1] "co"  "bia"
##
## [[2]]
## [1] "s"         "ber party"
##
## [[3]]
## [1] "sugarp"
```

```
paste(lum_vector,"[",c(3,5,7),"-",c(5,49),"]",collapse = ";")
```

```
## [1] "columbia [ 3 - 5 ];slumber party [ 5 - 49 ];sugarplum [ 7 - 5 ]"
```

Honor Code Example

```
setwd("C:/Users/cheny/Desktop/study/statistical computing and intro to data science/lecture four")
# readlines gives us the line which is always a line of sentence
HC <- readLines("HonorCode.txt")
# scan gives us the words
HC2 <- scan("HonorCode.txt",what = "")
length(HC);length(HC2)
```

```
## [1] 41
```

```
## [1] 443
```

```
head(HC,5);head(HC2,5)
```

```
## [1] "Students should be aware that academic dishonesty (for example,
plagiarism, cheating on an examination, or dishonesty in dealing with a
faculty member or other University official) or the threat of violence or
harassment are particularly serious offenses and will be dealt "
## [2] "with severely under Dean's Discipline."
## [3] ""
## [4] "Graduate students are expected to exhibit the high level of personal
and academic integrity"
## [5] "and honesty required of all members of an academic community as they
engage in scholarly"
```

```
## [1] "Students" "should"   "be"       "aware"    "that"
```

```
## FIND THE INDEX OF CERTAIN WORD
grep("students",HC)
```

```
## [1]  4 13 21 28
```

```
grep("students",HC2)
```

```
## [1]  48 142 232 310
```

```
# this return the sentence(lines) in which there have 'students'
HC[grep("students",HC)]
```

```
## [1] "Graduate students are expected to exhibit the high level of personal
and academic integrity"
## [2] "In practical terms, students must not cheat on examinations, and
deliberate plagiarism is of"
## [3] "Graduate students are responsible for proper citation and
paraphrasing, and must also take"
## [4] "All incoming doctoral and master's students in the Arts and Sciences
at Columbia are"
```

```
# this return the word 'students'
HC2[grep("students",HC2)]
```

```
## [1] "students" "students" "students" "students"
```

```
# grepl is the function return whether or not the word can be matched in the
given string.
head(grepl('students',HC),15)
```

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

```
# there two have the same function
HC[grepl('students',HC)] == HC[grep("students",HC)]
```

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

Using functions we've learned today, let's make HC a vector with each element a word of the Honor Code (instead of a line of text). Of course, could do this with scan().

```
# make the HC a long string
HC <- paste(HC,collapse = ' ')
HC
```

```
## [1] "Students should be aware that academic dishonesty (for example,
plagiarism, cheating on an examination, or dishonesty in dealing with a
faculty member or other University official) or the threat of violence or
harassment are particularly serious offenses and will be dealt  with severely
under Dean's Discipline.  Graduate students are expected to exhibit the high
level of personal and academic integrity and honesty required of all members
of an academic community as they engage in scholarly discourse and research.
Scholars draw inspiration from the work done by other scholars; they argue
their claims with reference to othersâ<U+0080><U+0099> work; they extract
evidence from the world or from earlier scholarly works. When a student
engages in these activities, it is vital to credit properly the source of his
or her claims or evidence. Failing to do so violates oneâ<U+0080><U+0099>s
scholarly responsibility.  In practical terms, students must not cheat on
examinations, and deliberate plagiarism is of course prohibited. Plagiarism
includes buying, stealing, borrowing, or otherwise obtaining all or part of a
paper (including obtaining or posting a paper online); hiring someone to
write a paper; copying from or paraphrasing another source without proper
citation or falsification of citations; and building on the ideas of another
without citation. Students also should not submit the same paper to more than
one class. This information is adapted from the material published by Purdue
University's Online Writing Lab.  Graduate students are responsible for
proper citation and paraphrasing, and must also take special care to avoid
even accidental plagiarism. The best strategy is to use great caution in the
handling of ideas and prose passages: take notes carefully and clearly mark
words and ideas not oneâ<U+0080><U+0099>s own. When in doubt, consult your
professor. Failure to observe these rules of conduct will result in serious
academic consequences, which can include dismissal from the university.  All
incoming doctoral and master's students in the Arts and Sciences at Columbia
are required to complete an Academic Integrity Tutorial prior to arrival on
campus.   Students engaging in research must be aware of and follow
University policies regarding intellectual and financial conflicts of
```

```
interest, integrity and security in data collection and management,
intellectual property rights and data ownership, and necessary institutional
approval for research with human subjects and animals.  Academic integrity
concerns honest research practices as much as avoiding plagiarism. Research
misconduct falls into three categories: plagiarism, falsification, and
fabrication. Falsification includes purposeful manipulation, modification, or
omission of data or results. Fabrication is the making up of data or results
and the recording or reporting thereof. The university does not tolerate any
form of research misconduct and violation of this policy may result in
serious sanctions, including termination."
```

```
# split every word in this long string
HC.words <- strsplit(HC,split = ' ')[[1]]
head(HC.words,10)
```

```
##  [1] "Students"    "should"      "be"          "aware"       "that"
##  [6] "academic"    "dishonesty"  "(for"        "example,"    "plagiarism,"
```

```
# we can count words using table()
word_count <- table(HC.words)
word_count <- sort(word_count, decreasing = TRUE)
head(word_count, 10)
```

```
## HC.words
##  and   of   or  the   to   in     from    a   is
##   23   17   16   13   11   10    9    6    5    5
```

```
tail(word_count, 10)
```

```
## HC.words
##   vital   which   words    work   work;  works.   world   write Writing
##       1       1       1       1       1       1       1       1       1
##    your
##       1
```

test

```
# grep : search for patterns in a string
semicolons <- grep(";",names(word_count))
names(word_count)[semicolons]
```

```
## [1] "citations;" "online);"   "paper;"     "scholars;"  "work;"
```

```
# gsub(pattern, replacement, x) function searches for a specific substring
given by pattern in a vector x of strings and replaces it with the substring
specified by replacement.

names(word_count)<-gsub(";","",names(word_count))
names(word_count)[semicolons]
```

```
## [1] "citations" "online)"   "paper"     "scholars"  "work"
```

Summary:

1. nchar(): Finds the length of a string.
2. substring(): Extracts substrings and substitutes.
3. strsplit(): Turns strings into vectors.
4. paste(): Turns vectors into a string.
5. grep(): Search for patterns in a string.
6. gsub(): Replaces patterns in a string with another string.

Regular Expressions

```
fav_animals <- "cat, squirrel,  hedgehog, octopus"

strsplit(fav_animals,split = ",")

## [[1]]
## [1] "cat"        " squirrel"  "  hedgehog" " octopus"

strsplit(fav_animals,split = ",  ")

## [[1]]
## [1] "cat, squirrel"     "hedgehog, octopus"
```

regular expressions

```
# | means or
grep("cat|dog", c("categorize", "work doggedly"))

## [1] 1 2

grep("A|b",c("Alabama","buleberry","categorize"))

## [1] 1 2

grep("A|b",c("Alabama","p(A|b)"))

## [1] 1 2

# \\| means |
grep("A\\|b", c("Alabama", "blueberry", "work doggedly", "P(A|b)"))

## [1] 4
```

rules for regular expression:

1. Indicate sets of characters with brackets **[]**.

- "[a-z]" matches any lower case letters.
- "[:punct:]" matches all punctuation marks.

2. The caret **^** negates a character range when in the leading position.

- "[^aeiou]" matches any characters except lower-case vowels.

3. The period . stands for any character and doesn't need brackets.

- "c..s" matches "cats", "class", "c88s", "c s", etc.

4. some 'how often' rules: | Quantier | Description | |------------|---------------------------------| | **+** |Repeated one or more times. | | ** * ** |Repeated zero or more times. | | **?** |Repeated zero or one times. | | **{n}** |Repeated exactly n times. | | **{n, }** |Repeated n or more times. | | **{n, m}** |Repeated between n and m times. |

- "[0-9][0-9][a-zA-z]+" matches strings with two digits followed by one or more letters.

- "(abc){3}" matches three consecutive occurences of "abc".

- "abc{3}" matches "abccc".
- "M[rs][rs]?nn.?" matches "Mr", "Ms", "Mrs", "Mr.", "Ms.", "Mrs.".
- The above also matches "Mrr", "Msr", "Mss", "Mrr.", "Msr.", "Mss." (and nothing else).

5. The dollar sign **$** means that a pattern only matches at the end of a line.

- "[a-z,]$" matches strings ending in lower-case letters or a comma.

6. The caret **^** outside of brackets means that a pattern only matches at the beginning of a line.

- "[1]" matches strings not beginning with capital letters.

7. the function that we can use regular expression:

- strsplit()

- grep()

- gsub()

```
grep("^[^A-Z]",c("Hello","cat","1dsad93"))
```

```
## [1] 2 3
```

```
grep("^[A-Z]",c("Hello","cat","1dsad93"))
```

```
## [1] 1
```

```
grep("[a-z,]$",c("Hello","cat","1dsad93"))
```

```
## [1] 1 2
```

---

[1] ^A-Z

honorcode example

```
HC <- readLines("HonorCode.txt")
HC <- paste(HC, collapse = " ") # One long string
HC.words <- strsplit(HC, split=" ")[[1]] # Last Time
head(HC.words, 10)
```

```
##  [1] "Students"    "should"      "be"          "aware"       "that"
##  [6] "academic"    "dishonesty"  "(for"        "example,"    "plagiarism,"
```

```
tail(HC.words, 10)
```

```
##  [1] "of"           "this"         "policy"       "may"
##  [5] "result"       "in"           "serious"      "sanctions,"
##  [9] "including"    "termination."
```

```
HC.words <- strsplit(HC, split="(\\s|[[:punct:]])+")[[1]]
head(HC.words, 10)
```

```
##  [1] "Students"   "should"     "be"         "aware"      "that"
##  [6] "academic"   "dishonesty" "for"        "example"    "plagiarism"
```

```
tail(HC.words, 10)
```

```
##  [1] "of"          "this"        "policy"      "may"         "result"
##  [6] "in"          "serious"     "sanctions"   "including"   "termination"
```

earthquakes

```
quakes <- readLines("NCEDC_Search_Results.html")
```

```
## Warning in readLines("NCEDC_Search_Results.html"): incomplete final line
## found on 'NCEDC_Search_Results.html'
```

```
tail(quakes)
```

```
## [1] "2016/12/21 00:17:14.99  -7.5082  127.9206 152.00  6.70   Mw       17
3 1.20  us  201612212002"
## [2] "2016/12/24 01:32:16.04  -5.2453  153.5754  35.00  6.00   Mw       13
2 0.91  us  201612242007"
## [3] "2016/12/25 14:22:27.05 -43.4029  -73.9395  38.00  7.60   Mw       29
0 0.80  us  201612252035"
## [4] "2016/12/29 22:30:19.30  -9.0283  118.6639  79.00  6.30   Mw       26
4 1.43  us  201612292025"
## [5] "</PRE>"
## [6] "</BODY></HTML>"
```

```
length(quakes) ## this contain some lines are not about data
```

```
## [1] 2426
```

```
data_regex <- "^[0-9]{4}/[0-9]{2}/[0-9]{2}" ## format of data

grep(quakes,pattern = data_regex)
```

```
##    [1]   12   13   14   15   16   17   18   19   20   21   22   23   24
##   [14]   25   26   27   28   29   30   31   32   33   34   35   36   37
##   [27]   38   39   40   41   42   43   44   45   46   47   48   49   50
##   [40]   51   52   53   54   55   56   57   58   59   60   61   62   63
##   [53]   64   65   66   67   68   69   70   71   72   73   74   75   76
##   [66]   77   78   79   80   81   82   83   84   85   86   87   88   89
##   [79]   90   91   92   93   94   95   96   97   98   99  100  101  102
##   [92]  103  104  105  106  107  108  109  110  111  112  113  114  115
##  [105]  116  117  118  119  120  121  122  123  124  125  126  127  128
##  [118]  129  130  131  132  133  134  135  136  137  138  139  140  141
##  [131]  142  143  144  145  146  147  148  149  150  151  152  153  154
##  [144]  155  156  157  158  159  160  161  162  163  164  165  166  167
##  [157]  168  169  170  171  172  173  174  175  176  177  178  179  180
##  [170]  181  182  183  184  185  186  187  188  189  190  191  192  193
##  [183]  194  195  196  197  198  199  200  201  202  203  204  205  206
##  [196]  207  208  209  210  211  212  213  214  215  216  217  218  219
##  [209]  220  221  222  223  224  225  226  227  228  229  230  231  232
##  [222]  233  234  235  236  237  238  239  240  241  242  243  244  245
##  [235]  246  247  248  249  250  251  252  253  254  255  256  257  258
##  [248]  259  260  261  262  263  264  265  266  267  268  269  270  271
##  [261]  272  273  274  275  276  277  278  279  280  281  282  283  284
##  [274]  285  286  287  288  289  290  291  292  293  294  295  296  297
##  [287]  298  299  300  301  302  303  304  305  306  307  308  309  310
##  [300]  311  312  313  314  315  316  317  318  319  320  321  322  323
##  [313]  324  325  326  327  328  329  330  331  332  333  334  335  336
##  [326]  337  338  339  340  341  342  343  344  345  346  347  348  349
##  [339]  350  351  352  353  354  355  356  357  358  359  360  361  362
##  [352]  363  364  365  366  367  368  369  370  371  372  373  374  375
##  [365]  376  377  378  379  380  381  382  383  384  385  386  387  388
##  [378]  389  390  391  392  393  394  395  396  397  398  399  400  401
##  [391]  402  403  404  405  406  407  408  409  410  411  412  413  414
##  [404]  415  416  417  418  419  420  421  422  423  424  425  426  427
##  [417]  428  429  430  431  432  433  434  435  436  437  438  439  440
##  [430]  441  442  443  444  445  446  447  448  449  450  451  452  453
##  [443]  454  455  456  457  458  459  460  461  462  463  464  465  466
##  [456]  467  468  469  470  471  472  473  474  475  476  477  478  479
##  [469]  480  481  482  483  484  485  486  487  488  489  490  491  492
##  [482]  493  494  495  496  497  498  499  500  501  502  503  504  505
##  [495]  506  507  508  509  510  511  512  513  514  515  516  517  518
##  [508]  519  520  521  522  523  524  525  526  527  528  529  530  531
##  [521]  532  533  534  535  536  537  538  539  540  541  542  543  544
##  [534]  545  546  547  548  549  550  551  552  553  554  555  556  557
##  [547]  558  559  560  561  562  563  564  565  566  567  568  569  570
##  [560]  571  572  573  574  575  576  577  578  579  580  581  582  583
##  [573]  584  585  586  587  588  589  590  591  592  593  594  595  596
##  [586]  597  598  599  600  601  602  603  604  605  606  607  608  609
##  [599]  610  611  612  613  614  615  616  617  618  619  620  621  622
##  [612]  623  624  625  626  627  628  629  630  631  632  633  634  635
##  [625]  636  637  638  639  640  641  642  643  644  645  646  647  648
##  [638]  649  650  651  652  653  654  655  656  657  658  659  660  661
```

```
##  [651]  662  663  664  665  666  667  668  669  670  671  672  673  674
##  [664]  675  676  677  678  679  680  681  682  683  684  685  686  687
##  [677]  688  689  690  691  692  693  694  695  696  697  698  699  700
##  [690]  701  702  703  704  705  706  707  708  709  710  711  712  713
##  [703]  714  715  716  717  718  719  720  721  722  723  724  725  726
##  [716]  727  728  729  730  731  732  733  734  735  736  737  738  739
##  [729]  740  741  742  743  744  745  746  747  748  749  750  751  752
##  [742]  753  754  755  756  757  758  759  760  761  762  763  764  765
##  [755]  766  767  768  769  770  771  772  773  774  775  776  777  778
##  [768]  779  780  781  782  783  784  785  786  787  788  789  790  791
##  [781]  792  793  794  795  796  797  798  799  800  801  802  803  804
##  [794]  805  806  807  808  809  810  811  812  813  814  815  816  817
##  [807]  818  819  820  821  822  823  824  825  826  827  828  829  830
##  [820]  831  832  833  834  835  836  837  838  839  840  841  842  843
##  [833]  844  845  846  847  848  849  850  851  852  853  854  855  856
##  [846]  857  858  859  860  861  862  863  864  865  866  867  868  869
##  [859]  870  871  872  873  874  875  876  877  878  879  880  881  882
##  [872]  883  884  885  886  887  888  889  890  891  892  893  894  895
##  [885]  896  897  898  899  900  901  902  903  904  905  906  907  908
##  [898]  909  910  911  912  913  914  915  916  917  918  919  920  921
##  [911]  922  923  924  925  926  927  928  929  930  931  932  933  934
##  [924]  935  936  937  938  939  940  941  942  943  944  945  946  947
##  [937]  948  949  950  951  952  953  954  955  956  957  958  959  960
##  [950]  961  962  963  964  965  966  967  968  969  970  971  972  973
##  [963]  974  975  976  977  978  979  980  981  982  983  984  985  986
##  [976]  987  988  989  990  991  992  993  994  995  996  997  998  999
##  [989] 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012
## [1002] 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025
## [1015] 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038
## [1028] 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051
## [1041] 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064
## [1054] 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077
## [1067] 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090
## [1080] 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103
## [1093] 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116
## [1106] 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129
## [1119] 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142
## [1132] 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155
## [1145] 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168
## [1158] 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
## [1171] 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194
## [1184] 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207
## [1197] 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220
## [1210] 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233
## [1223] 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246
## [1236] 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259
## [1249] 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272
## [1262] 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285
## [1275] 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298
## [1288] 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311
```

```
## [1301] 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324
## [1314] 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337
## [1327] 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
## [1340] 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363
## [1353] 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376
## [1366] 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389
## [1379] 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402
## [1392] 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415
## [1405] 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428
## [1418] 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441
## [1431] 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454
## [1444] 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467
## [1457] 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480
## [1470] 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493
## [1483] 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506
## [1496] 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519
## [1509] 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532
## [1522] 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545
## [1535] 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558
## [1548] 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571
## [1561] 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584
## [1574] 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597
## [1587] 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610
## [1600] 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623
## [1613] 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636
## [1626] 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649
## [1639] 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662
## [1652] 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675
## [1665] 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688
## [1678] 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701
## [1691] 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714
## [1704] 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727
## [1717] 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740
## [1730] 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753
## [1743] 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766
## [1756] 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779
## [1769] 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792
## [1782] 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805
## [1795] 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818
## [1808] 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831
## [1821] 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844
## [1834] 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857
## [1847] 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870
## [1860] 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883
## [1873] 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896
## [1886] 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909
## [1899] 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922
## [1912] 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935
## [1925] 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948
## [1938] 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961
```

```
## [1951] 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974
## [1964] 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
## [1977] 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
## [1990] 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
## [2003] 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026
## [2016] 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039
## [2029] 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052
## [2042] 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065
## [2055] 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078
## [2068] 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091
## [2081] 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104
## [2094] 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117
## [2107] 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130
## [2120] 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143
## [2133] 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156
## [2146] 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169
## [2159] 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182
## [2172] 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195
## [2185] 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208
## [2198] 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221
## [2211] 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234
## [2224] 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247
## [2237] 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260
## [2250] 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273
## [2263] 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286
## [2276] 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299
## [2289] 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312
## [2302] 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325
## [2315] 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338
## [2328] 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351
## [2341] 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364
## [2354] 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377
## [2367] 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390
## [2380] 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403
## [2393] 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416
## [2406] 2417 2418 2419 2420 2421 2422 2423 2424
```

```
head(grep(quakes,pattern = data_regex))
```

```
## [1] 12 13 14 15 16 17
```

```
tail(grep(quakes,pattern = data_regex,value = TRUE))
```

```
## [1] "2016/12/20 04:21:29.15 -10.1750  161.2271  20.00  6.40   Mw       21
1 0.87  us  201612202010"
## [2] "2016/12/20 12:33:14.24 -10.1785  160.9149  10.00  6.00   Mw       14
1 0.88  us  201612202041"
## [3] "2016/12/21 00:17:14.99  -7.5082  127.9206 152.00  6.70   Mw       17
3 1.20  us  201612212002"
## [4] "2016/12/24 01:32:16.04  -5.2453  153.5754  35.00  6.00   Mw       13
2 0.91  us  201612242007"
```

```
## [5] "2016/12/25 14:22:27.05 -43.4029  -73.9395  38.00  7.60   Mw       29
0 0.80  us  201612252035"
## [6] "2016/12/29 22:30:19.30  -9.0283  118.6639  79.00  6.30   Mw       26
4 1.43  us  201612292025"
```

```
# if invert is true, it return the element that do not match
grep(quakes,pattern = data_regex, invert = TRUE, value = TRUE)
```

```
##  [1] "<HTML><HEAD><TITLE>NCEDC_Search_Results</TITLE></HEAD><BODY>Your
search parameters are:<ul>"
##  [2] "<li>catalog=ANSS"
##  [3] "<li>start_time=2002/01/01,00:00:00"
##  [4] "<li>end_time=2017/01/01,00:00:00"
##  [5] "<li>minimum_magnitude=6.0"
##  [6] "<li>maximum_magnitude=10"
##  [7] "<li>event_type=E"
##  [8] "</ul>"
##  [9] "<PRE>"
## [10] "Date       Time             Lat       Lon  Depth   Mag Magt  Nst Gap
Clo  RMS  SRC   Event ID"
## [11] "------------------------------------------------------------------
-------------------------"
## [12] "</PRE>"
## [13] "</BODY></HTML>"
```

commom commands in grep() family:

1. **grep()** returns the indices containing a match.
2. **grepl()** returns a logical indicating which elements contain a match.
3. **regexpr()** returns the location of the rst match with attributes like the length of the match.
4. **gregexpr()** works similarly to regexpr(), but returns all matching locations. `g' for global.
5. **regmatches()** takes strings and the output of regexpr() or gregexpr() and returns the actual matching strings.

```
grep('a[a-z]',"Alabama")
```

```
## [1] 1
```

```
grepl('a[a-z]',"Alabama")
```

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

```
regexpr('a[a-z]',"Alabama")
```

```
## [1] 3
## attr(,"match.length")
## [1] 2
## attr(,"useBytes")
## [1] TRUE
```

```r
gregexpr('a[a-z]',"Alabama")
```

```
## [[1]]
## [1] 3 5
## attr(,"match.length")
## [1] 2 2
## attr(,"useBytes")
## [1] TRUE
```

```r
regmatches("Alabama",gregexpr('a[a-z]',"Alabama"))
```

```
## [[1]]
## [1] "ab" "am"
```

earth quake example

```r
coord_exp <- "-?[0-9]+\\.[0-9]{4}"
full_exp <- paste(coord_exp,"\\s+",coord_exp,sep = "")
head(grep(quakes,pattern = full_exp,value = TRUE),15)
```

```
##  [1] "2002/01/01 10:39:06.82 -55.2140 -129.0000  10.00  6.00   Mw   78
1.07  NEI 200201014017"
##  [2] "2002/01/01 11:29:22.73   6.3030  125.6500 138.10  6.30   Mw  236
0.90  NEI 200201014018"
##  [3] "2002/01/02 14:50:33.49 -17.9830  178.7440 665.80  6.20   Mw  215
1.08  NEI 200201024034"
##  [4] "2002/01/02 17:22:48.76 -17.6000  167.8560  21.00  7.20   Mw  427
0.90  NEI 200201024041"
##  [5] "2002/01/03 07:05:27.67  36.0880   70.6870 129.30  6.20   Mw  431
0.87  NEI 200201034024"
##  [6] "2002/01/03 10:17:36.30 -17.6640  168.0040  10.00  6.60   Mw  386
1.14  NEI 200201034040"
##  [7] "2002/01/10 11:14:56.93  -3.2120  142.4270  11.00  6.70   Mw  333
1.18  NEI 200201104038"
##  [8] "2002/01/13 14:10:56.52  -5.6510  151.0740  43.60  6.40   Mw  441
1.06  NEI 200201134060"
##  [9] "2002/01/15 04:47:59.85 -17.3340  167.7220  10.00  6.00   Mw  173
1.09  NEI 200201154016"
## [10] "2002/01/15 07:12:58.03  -6.3140  105.2050  10.00  6.10   Mw  209
1.23  NEI 200201154033"
## [11] "2002/01/15 09:01:15.95  -5.5270  151.0970  41.10  6.20   Mw  191
0.96  NEI 200201154045"
## [12] "2002/01/16 23:09:52.08  15.5020  -93.1330  80.20  6.40   Mw  431
0.94  NEI 200201164053"
## [13] "2002/01/22 04:53:52.65  35.7900   26.6170  88.00  6.20   Mw  390
0.95  NEI 200201224014"
## [14] "2002/01/28 13:50:28.72  49.3810  155.5940  33.00  6.10   Mw  528
0.82  NEI 200201284038"
## [15] "2002/01/28 15:09:55.89 -15.3040 -173.2250  33.00  6.20   Mw  192
1.01  NEI 200201284040"
```

```r
coord_log <- grepl(quakes,pattern = full_exp)
matches <- gregexpr(pattern = full_exp,text = quakes[coord_log])
head(matches,1)
```

```
## [[1]]
## [1] 24
## attr(,"match.length")
## [1] 18
## attr(,"useBytes")
## [1] TRUE
```

```r
coords <- regmatches(quakes[coord_log],matches)
head(coords,4)
```

```
## [[1]]
## [1] "-55.2140 -129.0000"
##
## [[2]]
## [1] "6.3030  125.6500"
##
## [[3]]
## [1] "-17.9830  178.7440"
##
## [[4]]
## [1] "-17.6000  167.8560"
```

```r
tail(coords,4)
```

```
## [[1]]
## [1] "-7.5082  127.9206"
##
## [[2]]
## [1] "-5.2453  153.5754"
##
## [[3]]
## [1] "-43.4029  -73.9395"
##
## [[4]]
## [1] "-9.0283  118.6639"
```

```r
coors_split <- sapply(coords,strsplit,split="\\s+")
head(coors_split)
```

```
## [[1]]
## [1] "-55.2140"  "-129.0000"
##
## [[2]]
## [1] "6.3030"   "125.6500"
##
## [[3]]
## [1] "-17.9830" "178.7440"
```

```
##
## [[4]]
## [1] "-17.6000" "167.8560"
##
## [[5]]
## [1] "36.0880" "70.6870"
##
## [[6]]
## [1] "-17.6640" "168.0040"
```

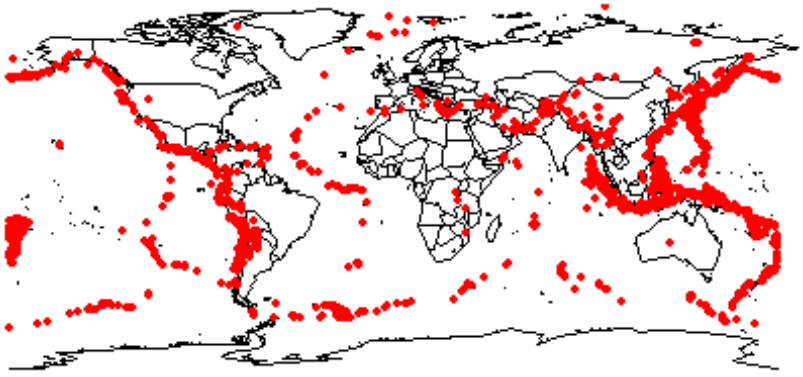
```
cs_unlisted <- unlist(coors_split)
coords_mat <- matrix(cs_unlisted,ncol=2,byrow = TRUE)
head(coords_mat)
```

```
##      [,1]       [,2]
## [1,] "-55.2140" "-129.0000"
## [2,] "6.3030"   "125.6500"
## [3,] "-17.9830" "178.7440"
## [4,] "-17.6000" "167.8560"
## [5,] "36.0880"  "70.6870"
## [6,] "-17.6640" "168.0040"
```

```
colnames(coords_mat) <- c("latitude","longitude")
head(coords_mat)
```

```
##      latitude   longitude
## [1,] "-55.2140" "-129.0000"
## [2,] "6.3030"   "125.6500"
## [3,] "-17.9830" "178.7440"
## [4,] "-17.6000" "167.8560"
## [5,] "36.0880"  "70.6870"
## [6,] "-17.6640" "168.0040"
```

```
library(maps)
map("world")
points(coords_mat[,"longitude"], coords_mat[,"latitude"],
pch = 19, col = "red", cex = .5)
```



read html

```
con <- url("http://www.columbia.edu", "r")
x <- readLines(con, warn = FALSE)
head(x,20)
```

```
##  [1] "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"
\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">"
##  [2] "<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\"
lang=\"en\" dir=\"ltr\">"
##  [3] ""
##  [4] "<!-- developed by CUIT -->"
##  [5] "<!-- 10/04/17, 12:36:41am --><head>"
##  [6] "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-
8\" />"
##  [7] "<meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\" >"
##  [8] "<meta name=\"msvalidate.01\"
content=\"DB472D6D4C7DB1E74C6D939F9C8AA8B4\" />"
##  [9] "<title>Columbia University in the City of New York</title>"
## [10] "<meta name=\"revisit-after\" content=\"1 day\" />"
## [11] "<link rel=\"shortcut icon\"
href=\"sites/all/themes/base/columbia2/images/favicon-crown.png\"
type=\"image/x-icon\" />"
## [12] "<script type=\"text/javascript\"
src=\"sites/all/modules/ias/mdetect/mdetect.js\"></script>"
## [13] "<meta name=\"viewport\" content=\"maximum-scale=1.0, user-
scalable=yes\" />"
```

```
## [14] "<link type=\"text/css\" rel=\"stylesheet\" media=\"all\"
href=\"modules/node/node.css\" />"
## [15] "<link type=\"text/css\" rel=\"stylesheet\" media=\"all\"
href=\"modules/system/defaults.css\" />"
## [16] "<link type=\"text/css\" rel=\"stylesheet\" media=\"all\"
href=\"modules/system/system.css\" />"
## [17] "<link type=\"text/css\" rel=\"stylesheet\" media=\"all\"
href=\"modules/system/system-menus.css\" />"
## [18] "<link type=\"text/css\" rel=\"stylesheet\" media=\"all\"
href=\"modules/user/user.css\" />"
## [19] "<link type=\"text/css\" rel=\"stylesheet\" media=\"all\"
href=\"sites/all/modules/contrib/cck/theme/content-module.css\" />"
## [20] "<link type=\"text/css\" rel=\"stylesheet\" media=\"all\"
href=\"sites/all/modules/contrib/ckeditor/ckeditor.css\" />"
```

```
length(x)
```

```
## [1] 420
```