C3-W4-Editing text variables

Jeffrey Leek (Hugh Nguyen edited)

08/18/2020

Example - Baltimore camera data

https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru

---

Fixing character vectors - tolower(), toupper()

```
# if(!file.exists("./data")){dir.create("./data")}
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl,destfile="./data/cameras.csv",method="curl")
cameraData <- read.csv("./data/cameras.csv")
names(cameraData)
```

```
[1] "address"                       "direction"                     "street"
[4] "crossStreet"                   "intersection"                  "Location.1"
[7] "X2010.Census.Neighborhoods"    "X2010.Census.Wards.Precincts" "Zip.Codes"
```

```
tolower(names(cameraData))
```

```
[1] "address"                       "direction"                     "street"
[4] "crossstreet"                   "intersection"                  "location.1"
[7] "x2010.census.neighborhoods"    "x2010.census.wards.precincts" "zip.codes"
```

Fixing character vectors - strsplit()

- Good for automatically splitting variable names
- Important parameters: *x*, *split*

```
splitNames = strsplit(names(cameraData),"\\.")
splitNames[[5]]
```

```
[1] "intersection"
```

```
splitNames[[6]]
```

```
[1] "Location" "1"
```

Quick aside - lists

```
mylist <- list(letters = c("A", "b", "c"), numbers = 1:3, matrix(1:25, ncol = 5))
head(mylist)
```

```
$letters
[1] "A" "b" "c"

$numbers
[1] 1 2 3

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

http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%203.pdf

---

Quick aside - lists

```
mylist[1]
```

```
$letters
[1] "A" "b" "c"
```

```
mylist$letters
```

```
[1] "A" "b" "c"
```

```
mylist[[1]]
```

```
[1] "A" "b" "c"
```

http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%203.pdf

---

Fixing character vectors - sapply()

- Applies a function to each element in a vector or list
- Important parameters: *X*,*FUN*

```
splitNames[[6]][1]
```

```
[1] "Location"
```

```
firstElement <- function(x){x[1]}
sapply(splitNames,firstElement)
```

```
[1] "address"      "direction"    "street"       "crossStreet"  "intersection" "Location"
[7] "X2010"        "X2010"        "Zip"
```

Peer review experiment data

http://www.plosone.org/article/info:doi/10.1371/journal.pone.0026895

Peer review data

```
fileUrl1 <- "https://dl.dropboxusercontent.com/u/7710864/data/reviews-apr29.csv"
fileUrl2 <- "https://dl.dropboxusercontent.com/u/7710864/data/solutions-apr29.csv"
download.file(fileUrl1,destfile="./data/reviews.csv",method="curl")
download.file(fileUrl2,destfile="./data/solutions.csv",method="curl")
reviews <- read.csv("./data/reviews.csv"); solutions <- read.csv("./data/solutions.csv")
head(reviews,2)
```

```
                                                        X..DOCTYPE.html.
1                                                                  <html>
2 <head><meta http-equiv=Content-Type content=text/html; charset=utf-8>
```

```
head(solutions,2)
```

```
                                                        X..DOCTYPE.html.
1                                                                  <html>
2 <head><meta http-equiv=Content-Type content=text/html; charset=utf-8>
```

Fixing character vectors - sub()

- Important parameters: *pattern*, *replacement*, *x*

```
names(reviews)
```

```
[1] "X..DOCTYPE.html."
```

```
sub("_","",names(reviews),)
```

```
[1] "X..DOCTYPE.html."
```

Fixing character vectors - gsub()

```
testName <- "this_is_a_test"
sub("_","",testName)
```

```
[1] "thisis_a_test"
```

```
gsub("_","",testName)
```

```
[1] "thisisatest"
```

Finding values - grep(),grepl()

```
# this will return all the indices in this col that contians 'Alameda'
grep("Alameda",cameraData$intersection)
```

```
[1] 65 69 79
```

```
# GREPL: wtf bro... l stands for loop through the col and record contains and not
# contains Alameda
table(grepl("Alameda",cameraData$intersection))
```

```
FALSE  TRUE
   77     3
```

```
#m subset where Alameda does not appear
cameraData2 <- cameraData[!grepl("Alameda",cameraData$intersection),]
```

More on grep()

```
# instead of return the indices, returns the values themselves
grep("Alameda",cameraData$intersection,value=TRUE)
```

```
[1] "E 33rd  & The Alameda"    "The Alameda  & 33rd St"   "Harford \n & The Alameda"
```

```
grep("JeffStreet",cameraData$intersection)
```

```
integer(0)
```

```
length(grep("JeffStreet",cameraData$intersection))
```

```
[1] 0
```

http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%203.pdf

---

More useful string functions

```
library(stringr)
nchar("Jeffrey Leek") # number of characters
```

```
[1] 12
```

```
substr("Jeffrey Leek",1,7) # slice the string: 1 to 7 (INCLUSIVE)
```

```
[1] "Jeffrey"
```

```
paste("Jeffrey","Leek") # " ".join(["Jeffrey","Leek"])
```

```
[1] "Jeffrey Leek"
```

More useful string functions

```
paste0("Jeffrey","Leek") # "".join(["Jeffrey","Leek"])
```

```
[1] "JeffreyLeek"
```

```
str_trim("Jeff      ") #.strip()
```

```
[1] "Jeff"
```

Important points about text in data sets

- Names of variables should be
  - All lower case when possible
  - Descriptive (Diagnosis versus Dx)
  - Not duplicated
  - Not have underscores or dots or white spaces
- Variables with character values
  - Should usually be made into factor variables (depends on application)
  - Should be descriptive (use TRUE/FALSE instead of 0/1 and Male/Female versus 0/1 or M/F)