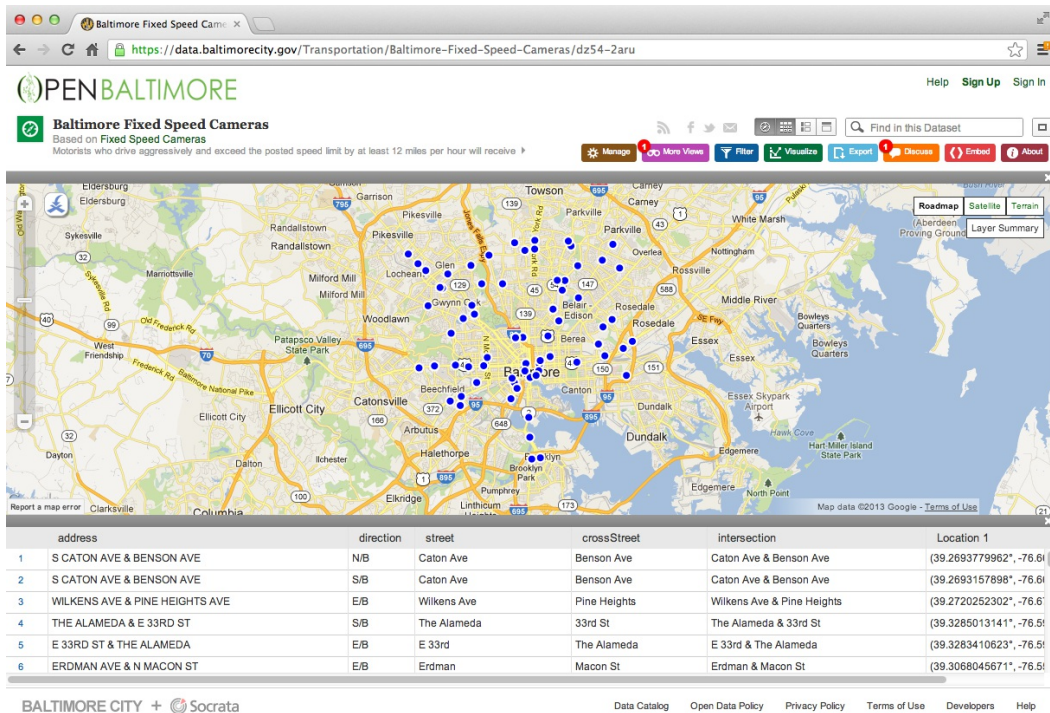




# Editing text variables

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# 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"       "crossStreet"  "intersection" "Location.1"
```

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

```
[1] "address"      "direction"    "street"       "crossstreet"  "intersection" "location.1"
```

# 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

# Quick aside - lists

```
mylist[1]
```

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

```
mylist$letters
```

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

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

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

# 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"
```

# Peer review experiment data

The screenshot shows a web browser displaying a PLOS ONE article. The browser's address bar shows the URL: [www.plosone.org/article/info:doi/10.1371/journal.pone.0026895](http://www.plosone.org/article/info:doi/10.1371/journal.pone.0026895). The page features a header with the PLOS ONE logo and navigation links: Articles, For Authors, About Us, and a Search bar. A banner at the top reads "Simplify your research with automatic and continuous dosing" with an image of medical syringes and pills. Below the header, a statistics bar shows: 6,497 VIEWS, 2 CITATIONS, 61 ACADEMIC BOOKMARKS, and 108 SOCIAL SHARES. The article title is "Cooperation between Referees and Authors Increases Peer Review Accuracy" by Jeffrey T. Leek, Margaret A. Taub, and Fernando J. Pineda. The article is categorized as "RESEARCH ARTICLE" and is "OPEN ACCESS" and "PEER-REVIEWED". Below the title, there are tabs for Article, About the Authors, Metrics, Comments, and Related Content. The Article tab is active, showing a diagram of the peer review process and a series of line graphs. To the right of the article content, there are buttons for Download, Print, and Share, and a section for Comments.

PLOS ONE

Articles For Authors About Us Search

OPEN ACCESS PEER-REVIEWED

6,497 VIEWS 2 CITATIONS 61 ACADEMIC BOOKMARKS 108 SOCIAL SHARES

RESEARCH ARTICLE

**Cooperation between Referees and Authors Increases Peer Review Accuracy**

Jeffrey T. Leek, Margaret A. Taub, Fernando J. Pineda

Article About the Authors Metrics Comments Related Content

Download Print Share

Comments

Media Coverage of This Article  
Posted by PLoS\_ONE\_Group

<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)
```

	id	solution_id	reviewer_id	start	stop	time_left	accept
1	1	3	27	1304095698	1304095758	1754	1
2	2	4	22	1304095188	1304095206	2306	1

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

	id	problem_id	subject_id	start	stop	time_left	answer
1	1	156	29	1304095119	1304095169	2343	B
2	2	269	25	1304095119	1304095183	2329	C

# Fixing character vectors - sub()

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

```
names(reviews)
```

```
[1] "id"          "solution_id" "reviewer_id" "start"      "stop"      "time_left"  
[7] "accept"
```

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

```
[1] "id"          "solutionid" "reviewerid" "start"      "stop"      "timeleft"   "accept"
```

# Fixing character vectors - gsub()

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

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

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

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

# Finding values - grep(),grepl()

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

```
[1]  4  5 36
```

```
table(grepl("Alameda",cameraData$intersection))
```

```
FALSE  TRUE  
   77    3
```

```
cameraData2 <- cameraData[!grepl("Alameda",cameraData$intersection),]
```

# More on grep()

```
grep("Alameda", cameraData$intersection, value=TRUE)
```

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

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

```
integer(0)
```

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

```
[1] 0
```

# More useful string functions

```
library(stringr)  
nchar("Jeffrey Leek")
```

```
[1] 12
```

```
substr("Jeffrey Leek", 1, 7)
```

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

```
paste("Jeffrey", "Leek")
```

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

# More useful string functions

```
paste0("Jeffrey", "Leek")
```

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

```
str_trim("Jeff      ")
```

```
[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)