

Data munging basics

Jeffrey Leek, Assistant Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Recall Tidy Data

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
id	problem_id	subject_id	start	stop	time_left	answer									
1	498	17	1307119989	1307120016	2369	A									
2	150	15	1307119991	1307120009	2376	D									
3	313	16	1307119994	1307120009	2376	E									
4	12	13	1307119995	1307120019	2366	B									
5	273	14	1307119996	1307120028	2357	A									
6	101	19	1307119996	1307120021	2364	B									
7	105	18	1307119998	1307120048	2337	B									
8	162	12	1307120004	1307120042	2343	C									
9	70	15	1307120011	1307120038	2347	C									
10	300	16	1307120012	1307120092	2293	B									
11	494	17	1307120017	1307120075	2310	D									
12	357	13	1307120021	1307120118	2267	A									
13	522	19	1307120025	1307120152	2233	D									
14	232	14	1307120030	1307120158	2227	C									
15	344	15	1307120041	1307120117	2268	B									
16	160	17	1307120079	1307120249	2136	D									
17	516	16	1307120094	1307120159	2226	B									
18	472	12	1307120119	1307120170	2215	A									
19	43	15	1307120122	1307120140	2245	C									
20	353	13	1307120144	1307120199	2186	C									
21	218	15	1307120152	1307120272	2113	E									
22	69	16	1307120163	1307120188	2197	D									
23	562	16	1307120190	1307120301	2084	D									
24	121	19	1307120253	1307120294	2091	E									
25	297	15	1307120277	1307120342	2043	B									
26	495	13	1307120281	1307120353	2032	E									
27	84	14	1307120288	1307120343	2042	E									
28	22	18	1307120310	1307120365	2020	C									
29	64	19	1307120310	1307120385	2000	B									
30	502	16	1307120323	1307120336	2049	B									
31	44	16	1307120339	1307120352	2033	A									
32	315	14	1307120348	1307120362	2023	B									
33	385	15	1307120352	1307120553	1832	E									
34	550	13	1307120356	1307120444	1941	B									
35	92	14	1307120368	1307120397	1988	B									
36	395	16	1307120377	1307120426	1959	D									
37	267	17	1307120382	1307120515	1870	E									
38	257	14	1307120401	1307120427	1958	C									
39	312	19	1307120407	1307120548	1837	D									
40	321	18	1307120431	1307120449	1936	A									
41	220	16	1307120437	1307120510	1878	A									

1. Each variable forms a column
2. Each observation forms a row
3. Each table/file stores data about one kind of observation (e.g. people/hospitals).

<http://vita.had.co.nz/papers/tidy-data.pdf>

[Leek, Taub, and Pineda 2011 PLoS One](#)

Where we would like to be

- [Tidy data](#) refers to the shape of the data
 - Variables in columns
 - Observations in rows
 - Tables holding elements of only one kind
- Plus
 - Column names are easy to use and informative
 - Row names are easy to use and informative
 - Obvious mistakes in the data have been removed
 - Variable values are internally consistent
 - Appropriate transformed variables have been added

A partial list of munging operations

- Fix variable names
- Create new variables
- Merge data sets
- Reshape data sets
- Deal with missing data
- Take transforms of variables
- Check on and remove inconsistent values

These steps must be recorded

90% of your effort will often be spent here

A partial list of munging operations

- Fix variable names
- Create new variables
- Merge data sets
- Reshape data sets
- Deal with missing data
- Take transforms of variables
- Check on and remove inconsistent values

Fixing character vectors - tolower(), toupper()

```
cameraData <- read.csv("../data/cameras.csv")  
names(cameraData)
```

```
[1] "address"      "direction"    "street"       "crossStreet"  
[5] "intersection" "Location.1"
```

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

```
[1] "address"      "direction"    "street"       "crossstreet"  
[5] "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"
```

Fixing character vectors - sapply()

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

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

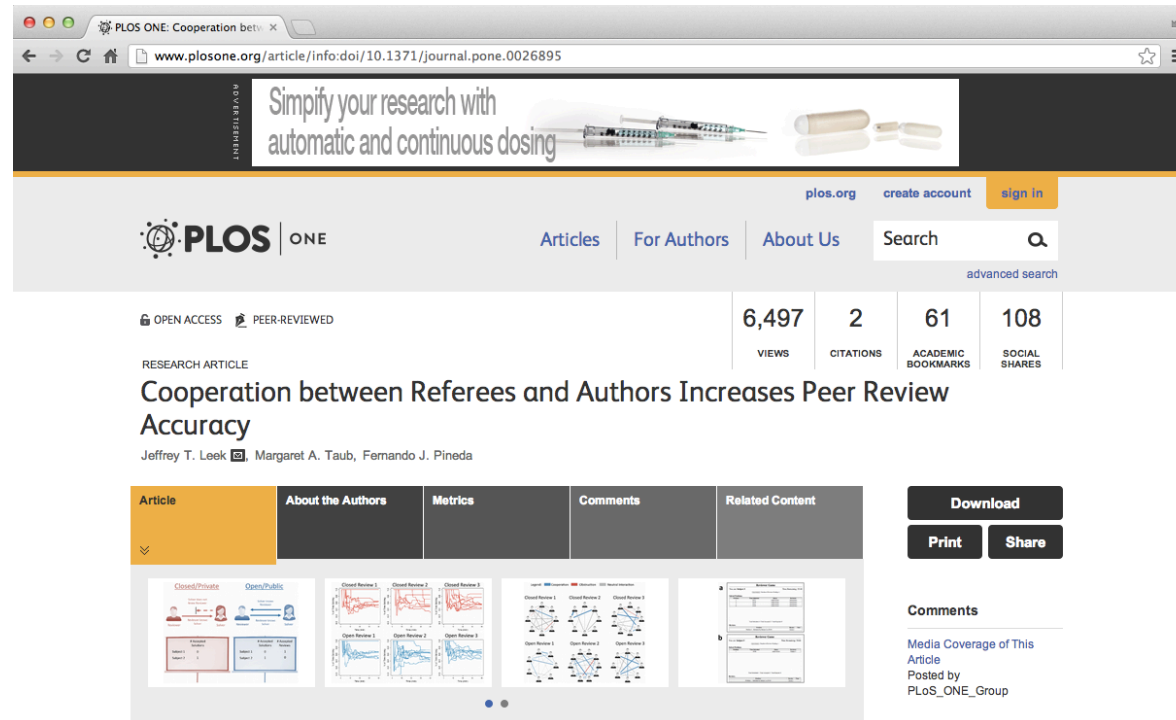
```
[1] "Location"
```

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

```
[1] "address"      "direction"    "street"       "crossStreet"  
[5] "intersection" "Location"
```


Peer review experiment data

- Data on submissions/reviews in an experiment



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

Peer review data

```

fileUrl1 <- "https://dl.dropbox.com/u/7710864/data/reviews-apr29.csv"
fileUrl2 <- "https://dl.dropbox.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(),gsub()

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

```
names(reviews)
```

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

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

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

Fixing character vectors - sub(),gsub()

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

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

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

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

Quantitative variables in ranges - - cut()

- Important parameters: *x*, *breaks*

```
reviews$time_left[1:10]
```

```
[1] 1754 2306 2192 2089 2043 1999 2130    NA 1899 2024
```

```
timeRanges <- cut(reviews$time_left,seq(0,3600,by=600))  
timeRanges[1:10]
```

```
[1] (1.2e+03,1.8e+03] (1.8e+03,2.4e+03] (1.8e+03,2.4e+03]  
[4] (1.8e+03,2.4e+03] (1.8e+03,2.4e+03] (1.8e+03,2.4e+03]  
[7] (1.8e+03,2.4e+03] <NA>                (1.8e+03,2.4e+03]  
[10] (1.8e+03,2.4e+03]  
6 Levels: (0,600] (600,1.2e+03] ... (3e+03,3.6e+03]
```

Quantitative variables in ranges - - cut()

```
class(timeRanges)
```

```
[1] "factor"
```

```
table(timeRanges, useNA="ifany")
```

```
timeRanges
```

(0,600]	(600,1.2e+03]	(1.2e+03,1.8e+03]
30	32	25
(1.8e+03,2.4e+03]	(2.4e+03,3e+03]	(3e+03,3.6e+03]
28	0	0
<NA>		
84		

Quantitative variables in ranges - cut2() {Hmisc}

```
library(Hmisc)
timeRanges<- cut2(reviews$time_left,g=6)
table(timeRanges,useNA="ifany")
```

```
timeRanges
[ 22, 384) [ 384, 759) [ 759,1150) [1150,1496) [1496,1909)
      20      19      19      19      19
[1909,2306] <NA>
      19      84
```

Adding an extra variable

```
timeRanges<- cut2(reviews$time_left,g=6)
reviews$timeRanges <- timeRanges
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


```
timeRanges
1 [1496,1909)
2 [1909,2306]
```


Merging data - merge()

- Merges data frames
- Important parameters: *x,y,by,by.x,by.y,all*

```
names(reviews)
```

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

```
names(solutions)
```

```
[1] "id"           "problem_id" "subject_id" "start"       "stop"  
[6] "time_left"   "answer"
```

Merging data - merge()

```
mergedData <- merge(reviews,solutions,all=TRUE)
head(mergedData)
```

	id	start	stop	time_left	solution_id	reviewer_id	accept
1	1	1304095119	1304095169	2343	NA	NA	NA
2	1	1304095698	1304095758	1754	3	27	1
3	2	1304095119	1304095183	2329	NA	NA	NA
4	2	1304095188	1304095206	2306	4	22	1
5	3	1304095127	1304095146	2366	NA	NA	NA
6	3	1304095276	1304095320	2192	5	28	1

	timeRanges	problem_id	subject_id	answer
1	<NA>	156	29	B
2	[1496,1909)	NA	NA	<NA>
3	<NA>	269	25	C
4	[1909,2306]	NA	NA	<NA>
5	<NA>	34	22	C
6	[1909,2306]	NA	NA	<NA>

Merging data - merge()

```
mergedData2 <- merge(reviews,solutions,by.x="solution_id",by.y="id",all=TRUE)
head(mergedData2[,1:6],3)
```

	solution_id	id	reviewer_id	start.x	stop.x	time_left.x
1	1	4	26	1304095267	1304095423	2089
2	2	6	29	1304095471	1304095513	1999
3	3	1	27	1304095698	1304095758	1754

```
reviews[1,1:6]
```

	id	solution_id	reviewer_id	start	stop	time_left
1	1	3	27	1304095698	1304095758	1754

Sorting values - sort()

- Important parameters: *x*, *decreasing*

```
mergedData2$reviewer_id[1:10]
```

```
[1] 26 29 27 22 28 22 29 23 25 29
```

```
sort(mergedData2$reviewer_id)[1:10]
```

```
[1] 22 22 22 22 22 22 22 22 22 22
```

Ordering values - order()

- Important parameters: *list of variables to order, na.last, decreasing*

```
mergedData2$reviewer_id[1:10]
```

```
[1] 26 29 27 22 28 22 29 23 25 29
```

```
order(mergedData2$reviewer_id)[1:10]
```

```
[1] 4 6 14 22 23 24 27 32 37 39
```

```
mergedData2$reviewer_id[order(mergedData2$reviewer_id)]
```

```
[1] 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22
[22] 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23 23 23 23 23 23 23
[43] 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 24 24 24 24 24
```

Reordering a data frame

```
head(mergedData2[,1:6],3)
```

	solution_id	id	reviewer_id	start.x	stop.x	time_left.x
1	1	4	26	1304095267	1304095423	2089
2	2	6	29	1304095471	1304095513	1999
3	3	1	27	1304095698	1304095758	1754

```
sortedData <- mergedData2[order(mergedData2$reviewer_id),]  
head(sortedData[,1:6],3)
```

	solution_id	id	reviewer_id	start.x	stop.x	time_left.x
4	4	2	22	1304095188	1304095206	2306
6	6	16	22	1304095303	1304095471	2041
14	14	12	22	1304095280	1304095301	2211

Reordering by multiple variables

```
head(mergedData2[,1:6],3)
```

	solution_id	id	reviewer_id	start.x	stop.x	time_left.x
1	1	4	26	1304095267	1304095423	2089
2	2	6	29	1304095471	1304095513	1999
3	3	1	27	1304095698	1304095758	1754

```
sortedData <- mergedData2[order(mergedData2$reviewer_id,mergedData2$id),]  
head(sortedData[,1:6],3)
```

	solution_id	id	reviewer_id	start.x	stop.x	time_left.x
4	4	2	22	1304095188	1304095206	2306
14	14	12	22	1304095280	1304095301	2211
6	6	16	22	1304095303	1304095471	2041

Reshaping data - example

```
misShaped <- as.data.frame(matrix(c(NA,5,1,4,2,3),byrow=TRUE,nrow=3))
names(misShaped) <- c("treatmentA","treatmentB")
misShaped$people <- c("John","Jane","Mary")
misShaped
```

	treatmentA	treatmentB	people
1	NA	5	John
2	1	4	Jane
3	2	3	Mary

<http://vita.had.co.nz/papers/tidy-data.pdf>

Reshaping data - melt()

- Important parameters: *id.vars*, *measure.vars*, *variable.name*

```
melt(misShaped, id.vars="people", variable.name="treatment", value.name="value")
```

	people	treatment	value
1	John	treatmentA	NA
2	Jane	treatmentA	1
3	Mary	treatmentA	2
4	John	treatmentB	5
5	Jane	treatmentB	4
6	Mary	treatmentB	3

More resources

- [Tidy data and tidy tools](#)
- Andrew Jaffe's [Data Cleaning Lecture](#)
- Hadley Wickham on [regular expressions](#)
- Long, painful experience :-)