Tidy data

Hadley Wickham

Assistant Professor / Dobelman Family Junior Chair

Department of Statistics / Rice University

August 2011

1. What is tidy data?
2. Five common causes of messiness
3. Tidying messy data (x5)
4. stringr review

What is tidy data?

- A step along the road to clean data
- Data that is easy to model, visualise and aggregate (i.e. works well with `lm`, `ggplot`, and `ddply`)
- Variables in columns, observations in rows, one type per dataset

	Pregnant	Not pregnant
Male	0	5
Female	1	4

There are three variables in this data set. What are they?

pregnant	sex	n
no	female	4
no	male	5
yes	female	1
yes	male	0

Storage	Meaning
Table / File	Data set
Rows	Observations
Columns	Variables

Common causes of messiness

- column headers are values, not variable names
- multiple variables are stored in one column
- variables are stored in both rows and columns
- multiple types of experimental unit stored in the same table
- one type of experimental unit stored in multiple tables

```
# Tools

library(reshape2)
?melt
?dcast
?col_split

library(stringr)
?str_replace
?str_sub
?str_split_fixed

library(plyr)
?arrange
```

Column headers values, not variable names

```
raw <- read.delim(”pew.txt”, check.names = F,
 stringsAsFactors = F)

head(raw)

# What are the variables in this dataset?
# Discuss with your neighbour for 1 minute
# Fixing this problem is easy.  We use melt, from
# reshape2, with two arguments, the input data, and
# the columns which are already variables:

library(reshape2)
tidy <- melt(raw, ”religion”)

head(tidy)

# We can now tweak the variable names
names(tidy) <- c(”religion”, ”income”, ”n”)
```

Multiple variables in one column

```
raw <- read.csv(”tb.csv”, stringsAsFactors = FALSE)
raw$new_sp <- NULL

names(raw) <- str_replace(names(raw), ”new_sp_”, ””)

# What are the variables in this dataset?
# Discuss with your neighbour for 1 minute
# Hint: f = female, u = unknown, 1524 = 15-25
```

Your turn

Use melt in the same way as for the religion-income data to get all variables in columns.

Think about how you might separate the "variable" variable into age and sex.

```
# na.rm = TRUE is useful if the missings don’t have
# any meaning
clean <- melt(raw, id = c(”iso2”, ”year”),
  na.rm = TRUE)
names(clean)[4] <- ”cases”

# Often a good idea to ensure the rows are ordered
# by the variables
clean <- arrange(clean, iso2, variable, year)
```

```
str_sub(clean$variable, 1, 1)
str_sub(clean$variable, 2)

ages <- c(”04” = ”0-4”, ”514” = ”5-14”, ”014” =
”0-14”, ”1524” = ”15-24”, ”2534” = ”25-34”, ”3544” =
”35-44”, ”4554” = ”45-54”, ”5564” = ”55-64”, ”65”=
”65+”, ”u” = NA)
ages[str_sub(clean$variable, 2)]

clean$sex <- str_sub(clean$variable, 1, 1)
clean$age <- factor(ages[str_sub(clean$variable, 2)],
   levels = ages)
clean$variable <- NULL

tidy <- tidy[c(”iso2”, ”year”, ”sex”, ”age”, ”cases”)]
```

Variables in rows and columns

```
raw <- read.delim(”weather.txt”,
  stringsAsFactors = FALSE)

# What are the variables in this dataset?
# Discuss with your neighbour for 1 minute
# Hint: TMIN = minimum temperature,
#       id = weather station identifier
```

Your turn

Melt the data, clean variables, and reorder rows and columns.

What do you need to do next?

```
raw1 <- melt(raw, id = 1:4, na.rm = T)
raw1$day <- as.integer(
  str_replace(raw1$variable, ”d”, ””))
raw1$variable <- NULL
raw1$element <- tolower(raw1$element)

raw1 <- raw1[c(”id”, ”year”, ”month”, ”day”,
  ”element”, ”value”)]
raw1 <- arrange(raw1, year, month, day, element)
```

```
# dcast shifts variables from rows to columns
tidy <- dcast(raw1, ... ~ element)

# casting syntax:
#   row_var1 + row_var2 ~ col_var1 + col_var2
#   ... = all variables not otherwise mentioned
```

Multiple types in the same table

Your turn

Practice everything you've learned so far to clean up billboard.csv.

(You might want to peek in `billboard-encoding.r`)

```
raw <- read.csv(”billboard.csv”,
  stringsAsFactors = F)
raw$date.peaked <- NULL
raw$artist.inverted <- iconv(raw$artist.inverted,
  ”MAC”, ”UTF-8”)
raw$track <- str_replace(raw$track,
  ” \\(.*?\\)”, ””)
names(raw)[-(1:6)] <- str_c(1:76)

tidy <- melt(raw, 1:6, na.rm = T)
tidy$week <- as.integer(tidy$variable)
tidy$variable <- NULL
```

```
# Fix dates (bonus)
library(lubridate)
tidy$date.entered <- ymd(tidy$date.entered)
tidy$date <- tidy$date.entered +
  weeks(tidy$week - 1)
tidy$date.entered <- NULL

# Tidy column names, order and row order
tidy <- rename(tidy, c(”value” = ”rank”,
  ”artist.inverted” = ”artist”))
tidy <- tidy[c(”year”, ”artist”, ”track”, ”time”,
   ”genre”, ”week”, ”date”, ”rank”)]
tidy <- arrange(tidy, year, artist, track, week)
```

Normalisation

Each fact about a song is repeated many many times. Sign that multiple types of experimental unit stored in the same table.

Need to separate out into song and rank tables.

```
song <- unrowname(unique(tidy[c(”artist”, ”track”,
”genre”, ”time”)]))
song$song_id <- 1:nrow(song)

rank <- join(tidy, song, match = ”first”)
rank <- rank[c(”song_id”, ”date”, ”rank”)]
```

One type in multiple tables

```
# Not shown, but easy with ldply
files <- dir(”path”, pattern = ”.csv”, full = T)
names(files) <- basename(files)

all <- ldply(files, read.csv)
```

stringr review

http://bit.ly/stringr

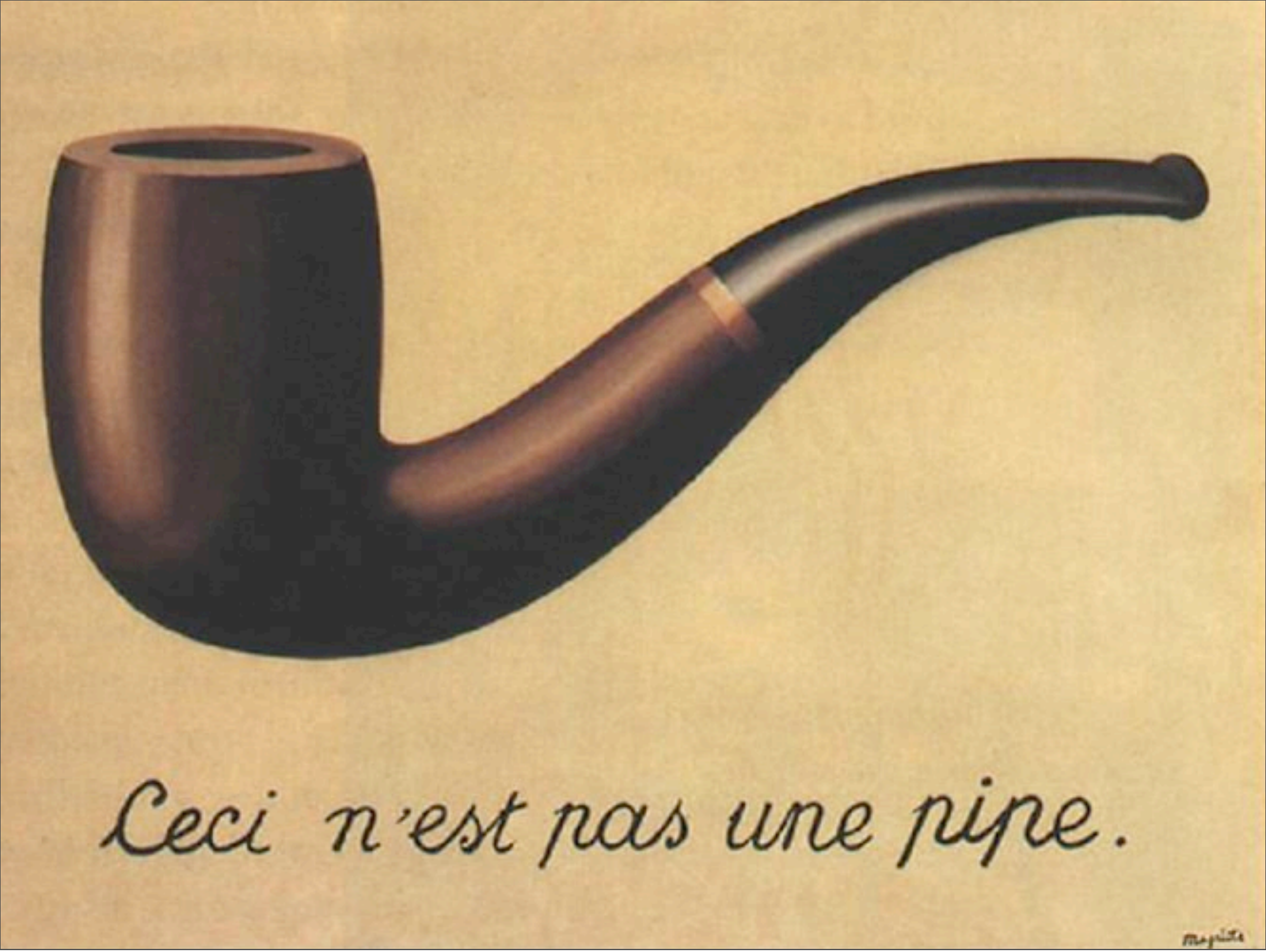
Function	Parameters	Result
`str_detect`	string, pattern	logical vector
`str_locate`	string, pattern	numeric matrix
`str_extract`	string, pattern	character vector
`str_replace`	string, pattern, **replacement**	character vector
`str_split_fixed`	string, pattern	character matrix

Single	Multiple (output usually a list)
`str_detect`	
`str_locate`	`str_locate_all`
`str_extract`	`str_extract_all`
`str_replace`	`str_replace_all`
`str_split_fixed`	`str_split`

Regular expressions

If you work with text data, I highly recommend learning at least a little about regular expressions.

They are complex, but very powerful.


Ceci n'est pas une pipe.
Magritte

```
# Special characters
a <- ”\\”
b <- ”\””
c <- ”a\nb\nc”

a
cat(a, ”\n”)
b
cat(b, ”\n”)
c
cat(c, ”\n”)
```

Special characters

- Use \ to “escape” special characters
  - \” = ”
  - \n = new line
  - \\ = \
  - \t = tab
- ?Quotes for more

Useful tools

- http://gskinner.com/RegExr/
- http://regexp.resource.googlepages.com/analyzer.html
- http://www.txt2re.com/
- http://www.regular-expressions.info/reference.html