# Transformations

## Hadley Wickham

**October 2009**

1. US baby names data

2. Transformations

3. Summaries

4. Doing it by group

# Baby names

Top 1000 male and female baby names in the US, from 1880 to 2008.

258,000 records (1000 * 2 * 129)

But only four variables: year, name, sex and prop.

```
> head(bnames, 15)                      > tail(bnames, 15)
     year     name  percent sex               year     name  percent  sex
1    1880     John 0.081541 boy        257986 2008   Neveah 0.000130 girl
2    1880  William 0.080511 boy        257987 2008   Amaris 0.000129 girl
3    1880    James 0.050057 boy        257988 2008 Hadassah 0.000129 girl
4    1880  Charles 0.045167 boy        257989 2008    Dania 0.000129 girl
5    1880   George 0.043292 boy        257990 2008   Hailie 0.000129 girl
6    1880    Frank 0.027380 boy        257991 2008   Jamiya 0.000129 girl
7    1880   Joseph 0.022229 boy        257992 2008    Kathy 0.000129 girl
8    1880   Thomas 0.021401 boy        257993 2008   Laylah 0.000129 girl
9    1880    Henry 0.020641 boy        257994 2008     Riya 0.000129 girl
10   1880   Robert 0.020404 boy        257995 2008     Diya 0.000128 girl
11   1880   Edward 0.019965 boy        257996 2008 Carleigh 0.000128 girl
12   1880    Harry 0.018175 boy        257997 2008    Iyana 0.000128 girl
13   1880   Walter 0.014822 boy        257998 2008   Kenley 0.000127 girl
14   1880   Arthur 0.013504 boy        257999 2008   Sloane 0.000127 girl
15   1880     Fred 0.013251 boy        258000 2008  Elianna 0.000127 girl
```

# Getting started

```
library(plyr)
bnames <- read.csv("baby-names.csv",
  stringsAsFactors = FALSE)


library(ggplot2)
qplot(year, prop, colour = sex,
  data = subset(bnames, name == "Hadley"),
  geom = "line")
```

# Your turn

Extract your name from the dataset. Plot the trend over time (hint: use `geom="line"`).

Create a new variable that contains the first three (or four, or five) letters of each name. How many names start the same as yours? Plot the trend over time.

```r
dian <- subset(bnames, substr(name, 1, 4) == "Dian")
qplot(year, prop, data = dian, geom = "line")
qplot(year, prop, data = dian, geom = "line",
  group = name)
qplot(year, prop, data = dian, geom = "line",
  group = interaction(name, sex))

qplot(year, prop, data = dian, geom = "line",
  colour = sex) + facet_wrap(~ name)
last_plot() + geom_point()
```

# Brainstorm

Thinking about the data, what are some of the trends that you might want to explore?  What additional variables would you need to create?  What other data sources might you want to use?

Pair up and brainstorm for 2 minutes.

# Some of my ideas

- First/last letter

- Length

- Number/percent of vowels

- Biblical names?

- Hurricanes?

- Rank

- Ecdf (how many babies have a name in the top 2, 3, 5, 100 etc)

```r
letter <- function(x, n = 1) {
  if (n < 0) {
    nc <- nchar(x)
    n <- nc + n + 1
  }
  tolower(substr(x, n, n))
}
vowels <- function(x) {
  nchar(gsub("[^aeiou]", "", x))
}

bnames$length <- nchar(bnames$name)
table(bnames$length)
bnames[bnames$length == 2, ]
bnames[bnames$length == 10, ]

bnames$first <- letter(bnames$name, 1)
bnames$last <- letter(bnames$name, -1)
bnames$vowels <- vowels(bnames$name)
```

Very verbose!

# Transform, summarise & subset

```
subset(df, subset)

transform(df, var1 = expr1, ...)

summarize(df, var1 = expr1, ...)
```

**Subset** selects rows from a data frame. **Transform** modifies an existing data frame. **Summarise** creates a new data frame. All evaluate arguments in scope of df.

```
bnames <- transform(bnames,
  first = letter(name, 1),
  last = letter(name, -1),
  vowels = vowels(name),
  length = nchar(name)
)

summarise(bnames,
  min_length = min(length),
  max_length = max(length)
)

subset(bnames, length == 2)
subset(bnames, length == 10)
```

# Aside:
# never use attach!

Non-local effects; not symmetric; implicit, not explicit.

Makes it very easy to make mistakes. Use `with()` instead.

```
with(bnames, table(year, length))
```

# Group-wise

What about group-wise **transformations** or **summaries**?  e.g. what if we want to compute the rank of a name within a sex and year?

This task is easy if we have a single year & sex, but hard otherwise.

# Group-wise

What about group-wise **transformations** or **summaries**? e.g. what if we want to compute the rank of a name within a sex and year?

This task is easy if we have a single year & sex, but hard otherwise.

Take two minutes to sketch out an approach

```
one <- subset(bnames, sex == "boy" & year == 2008)
one$rank <- rank(-one$prop,
    ties.method = "first")


# or
one <- transform(one,
    rank = rank(-prop, ties.method = "first"))
head(one)
```

What if we want to transform every sex and year?

```r
# Split
pieces <- split(bnames,
  list(bnames$sex, bnames$year))

# Apply
results <- vector("list", length(pieces))
for(i in seq_along(pieces)) {
  piece <- pieces[[i]]
  piece <- transform(piece,
    rank = rank(-prop, ties.method = "first"))
  results[[i]] <- piece
}

# Combine
result <- do.call("rbind", results)
```

```
# Or equivalently

bnames <- ddply(bnames, c("sex", "year"), transform,
   rank = rank(-prop, ties.method = "first"))
```

Input data

Way to split up input

Function to apply to each piece

```
# Or equivalently

bnames <- ddply(bnames, c("sex", "year"), transform,
  rank = rank(-prop, ties.method = "first"))
```

2nd argument to transform()

# Summaries

In a similar way, we can use `ddply()` for group-wise summaries.

There are many base R functions for special cases. Where available, these are often much faster; but you have to know they exist, and have to remember how to use them.

```r
# Explore average length

wtd.mean <- function(x, weights)
  sum(weights * x) / sum(weights)

sy <- ddply(bnames, c("sex", "year"), summarise,
  avg_length = wtd.mean(length, prop))

qplot(year, avg_length, data = sy, colour = sex,
  geom = "line")
```

```
# Explore number of names of each length

syl <- ddply(bnames, c("sex", "length", "year"),
  summarise, prop = sum(prop))
qplot(year, prop, data = syl, colour = sex,
  geom = "line") + facet_wrap(~ length)


twoletters <- subset(bnames, length == 2)
unique(twoletters$name)
qplot(year, prop, data = twoletters, colour = sex,
  geom = "line") + facet_wrap(~ name)
```

# Your turn

Use these tools to explore how the following have changed over time:

The total proportion of babies with names in the top 1000.

The number of vowels in a name.

The distribution of first (or last) letters.

```
sy <- ddply(bnames, c("year","sex"), summarise,
   prop = sum(prop),
   npop = sum(prop > 1/1000))


qplot(year, prop, data = sy, colour = sex,
   geom = "line")
qplot(year, npop, data = sy, colour = sex,
   geom = "line")



syl <- ddply(bnames, c("sex", "last", "year"),
   summarise, prop = sum(prop))
qplot(year, prop, data = syl, colour = sex,
   geom = "line") + facet_wrap(~ last)
```

# More about plyr

Many problems involve splitting up a large data structure, operating on each piece and joining the results back together:

# split-apply-combine

How you split up depends on the type of input: **arrays**, **data frames**, **lists**

How you combine depends on the type of output: **arrays**, **data frames**, **lists**, **nothing**

|  | array | data frame | list | nothing |
|---|---|---|---|---|
| array | aaply | adply | alply | a_ply |
| data frame | daply | ddply | dlply | d_ply |
| list | laply | ldply | llply | l_ply |
| n replicates | raply | rdply | rlply | r_ply |
| function arguments | maply | mdply | **mlply** | m_ply |

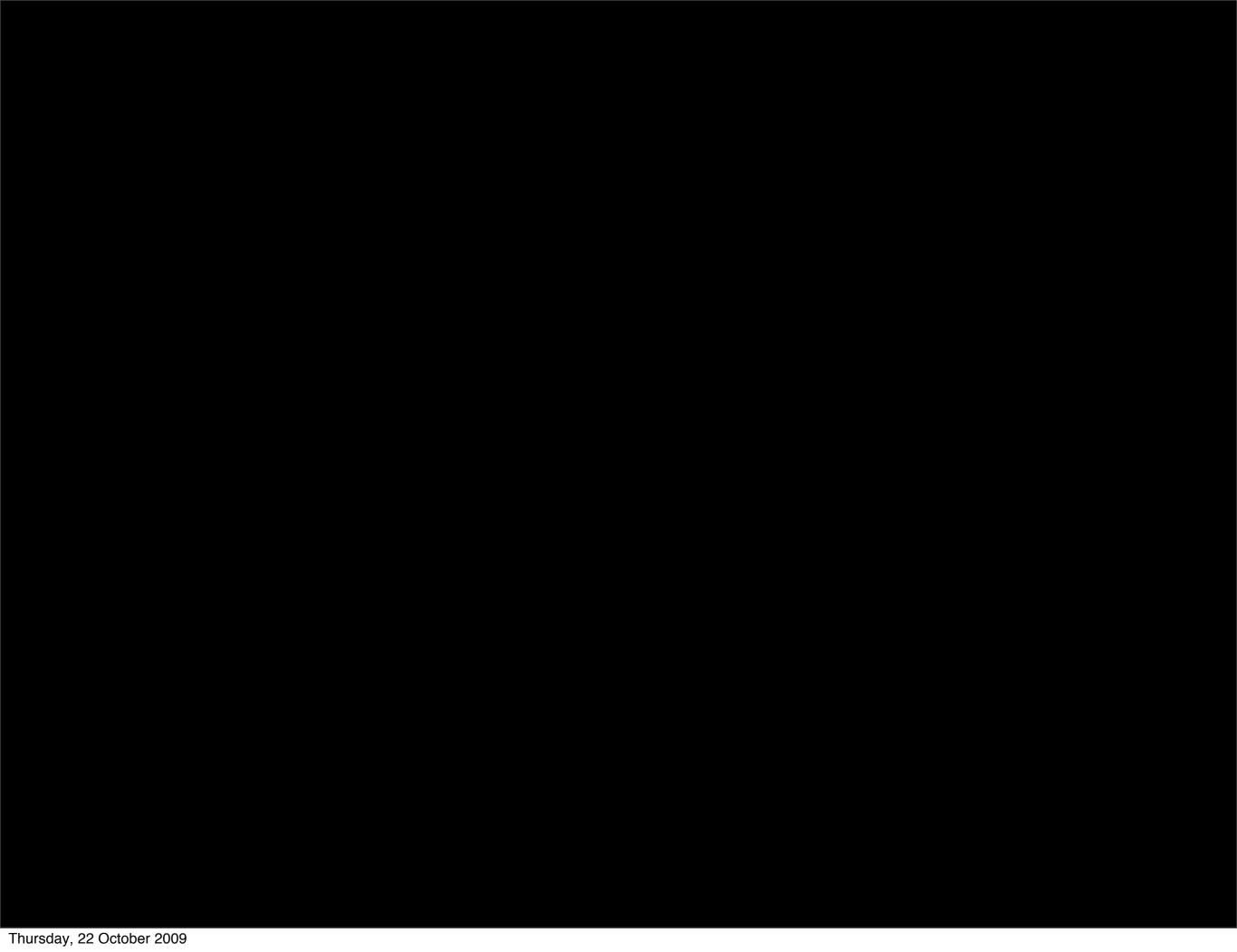|  | array | data frame | list | nothing |
|---|---|---|---|---|
| array | apply | adply | alply | a_ply |
| data frame | daply | *aggregate* | by | d_ply |
| list | sapply | ldply | lapply | l_ply |
| n replicates | replicate | rdply | replicate | r_ply |
| function arguments | mapply | mdply | mapply | m_ply |

# Fiddly details

Labelling

Progress bars

Consistent argument names

Missing values / Nulls

# http://had.co.nz/plyr

This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/us/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.