http://stat405.had.co.nz/utah

Data manipulation with plyr

Hadley Wickham

Rice University / RStudio

November 2012

Visualise

Model

Transform

1. Baby names data
2. Four tools for data manipulation
3. Groupwise operations
4. `ddply`
5. Challenges

Baby names

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

258,000 records (1000 * 2 * 129)

But only five variables: `year`, `name`, `soundex`, `sex` and `prop`.

```
install.packages(c("plyr", "ggplot2"))

library(plyr)
library(ggplot2)
```

```
options(stringsAsFactors = FALSE)

bnames <- read.csv("bnames2.csv.bz2")
bnames <- read.csv(choose.file())
```

```
head(bnames)
  year    name      prop sex soundex
1 1880    John 0.081541 boy    J500
2 1880 William 0.080511 boy    W450
3 1880   James 0.050057 boy    J520
4 1880 Charles 0.045167 boy    C642
5 1880  George 0.043292 boy    G620
6 1880   Frank 0.027380 boy    F652
```

```
tail(bnames)
       year     name      prop  sex soundex
257995 2008     Diya 0.000128 girl    D000
257996 2008 Carleigh 0.000128 girl    C642
257997 2008    Iyana 0.000128 girl    I500
257998 2008   Kenley 0.000127 girl    K540
257999 2008   Sloane 0.000127 girl    S450
258000 2008  Elianna 0.000127 girl    E450
```

Your turn

Extract your name from the dataset. Plot the trend over time.

```
garrett <- bnames[bnames$name == "Garrett", ]

qplot(year, prop, data = garrett, geom = "line")
```

subset
summarise
mutate
arrange

Revision

Recall the four functions that filter rows, create summaries, add new variables and rearrange the rows.

You have 30 seconds!

Function	Package
`subset`	`base`
`summarise`	`plyr`
`mutate`	`plyr`
`arrange`	`plyr`

They all have similar syntax. The first argument is a data frame, and all other arguments are interpreted in the context of that data frame. Each returns a data frame.

color	value
blue	1
black	2
blue	3
blue	4
black	5

color	value
blue	1
blue	3
blue	4

```
subset(df, color == "blue")
```

color	value
blue	1
black	2
blue	3
blue	4
black	5

double
2
4
6
8
10

```
summarise(df, double = 2 * value)
```

color	value
blue	1
black	2
blue	3
blue	4
black	5

total
15

```
summarise(df, total = sum(value))
```

color	value
blue	1
black	2
blue	3
blue	4
black	5

color	value	double
blue	1	2
black	2	4
blue	3	6
blue	4	8
black	5	10

```
mutate(df, double = 2 * value)
```

color	value
4	1
1	2
5	3
3	4
2	5

color	value
1	2
2	5
3	4
4	1
5	3

```
arrange(df, color)
```

color	value
4	1
1	2
5	3
3	4
2	5

color	value
5	3
4	1
3	4
2	5
1	2

```
arrange(df, desc(color))
```

Your turn

Using the data frame containing your name:

Reorder from highest to lowest popularity.

Calculate the min, mean, and max proportions for your name over the years. Return these in a data frame.

Add a new column that changes the proportion to a percentage

```
arrange(garrett, desc(prop))

summarise(garrett,
  min = min(prop),
  mean = mean(prop),
  max = max(prop))

mutate(garrett, perc = prop * 100)
```

Group wise operations

Total number of people per name

	name	total
1	Aaden	959
2	Aaliyah	39665
3	Aarav	219
4	Aaron	509464
5	Ab	25
6	Abagail	2682
7	Abb	16
8	Abbey	14348
9	Abbie	16622
10	Abbigail	6800

Total number of people per name

	name	total
1	Aaden	959
2	Aaliyah	39665
3	Aarav	219
4	Aaron	509464
5	Ab	25
6	Abagail	2682
7	Abb	16
8	Abbey	14348
9	Abbie	16622
10	Abbigail	6800

Do we have enough information to calculate this?

Your turn

In groups, calculate the totals for each name. If it is difficult, just devise a strategy for doing it.

Hint: Start by calculating the total for a single name (e.g, your name).

	name	total
1	Aaden	959
2	Aaliyah	39665
3	Aarav	219
4	Aaron	509464
5	Ab	25
6	Abagail	2682
7	Abb	16
8	Abbey	14348
9	Abbie	16622
10	Abbigail	6800

```r
garrett <- subset(bnames, name == "garrett")
sum(garrett$n)

# Or
summarise(garrett, total = sum(n))

# But how could we do this for every name?
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_total <- summarise(john, total = sum(n))
william_total <- summarise(william, total = sum(n))
james_total <- summarise(james, total = sum(n))
...

# Combine
totals <- rbind(john_total, william_total, james_total...)
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_total <- summarise(john, total = sum(n))
william_total <- summarise(william, total = su
james_total <- summarise(james, total = sum(
...

# Combine
totals <- rbind(john_total, willi                mes_total...)
```

The approach is sound, but there are 6782 names :(

Your turn

Look at the previous code and answer these three questions:

What data set did we split into pieces?

What variable did we use to split up the data set?

What function did we apply to each piece?

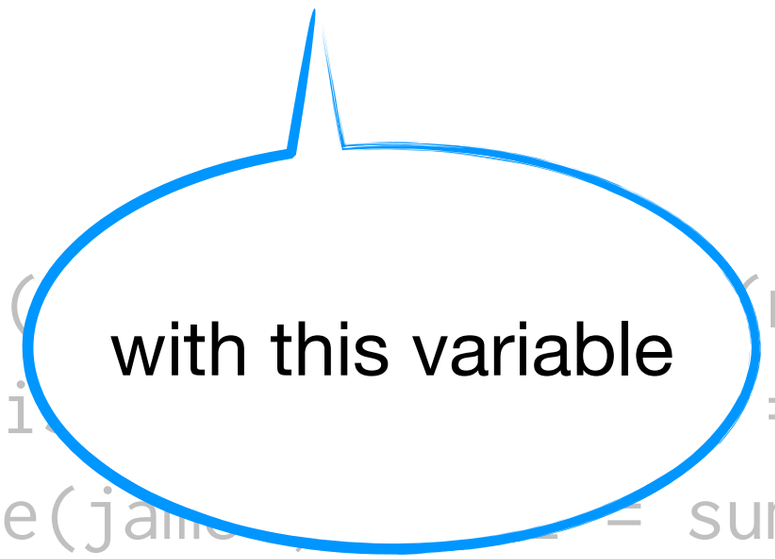
```r
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_total <- summarise(john, total = sum(n))
william_total <- summarise(william, total = sum(n))
james_total <- summarise(james, total = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_total <- summarise(john, total = sum(n))
william_total <- summarise(william, total = sum(n))
james_total <- summarise(james, total = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

Split this data set

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_total <- summarise(john, total = sum(n))
william_total <- summarise(william, total = sum(n))
james_total <- summarise(james, total = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_total <- summarise(                    n))
william_total <- summari                    = sum(n))
james_total <- summarise(jam                = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```



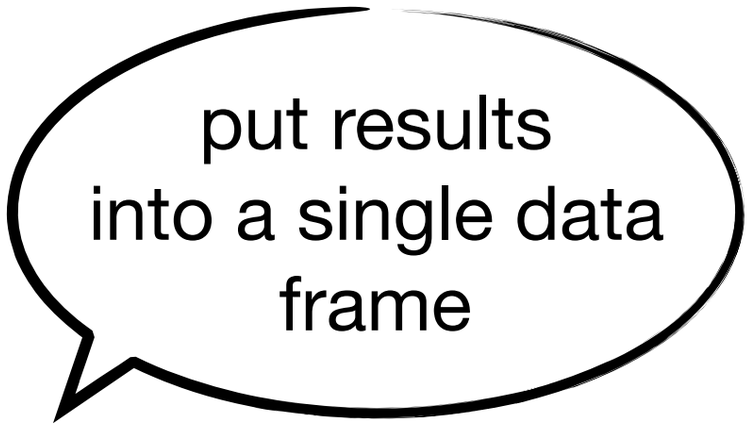
```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- summarise(john, total = sum(n))
william_years <- summarise(william, total = sum(n))
james_years <- summarise(james, total = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- summarise(john, total = sum(n))
william_years <- summarise(william, total = sum(n))
james_years <- summarise(james, total = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total,
```

then apply this function

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- nrow(john)
william_years <- nrow(william)
james_years <- nrow(james)
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- nrow(john)
william_years <- nrow(william)
james_years <- nrow(james)
...
```



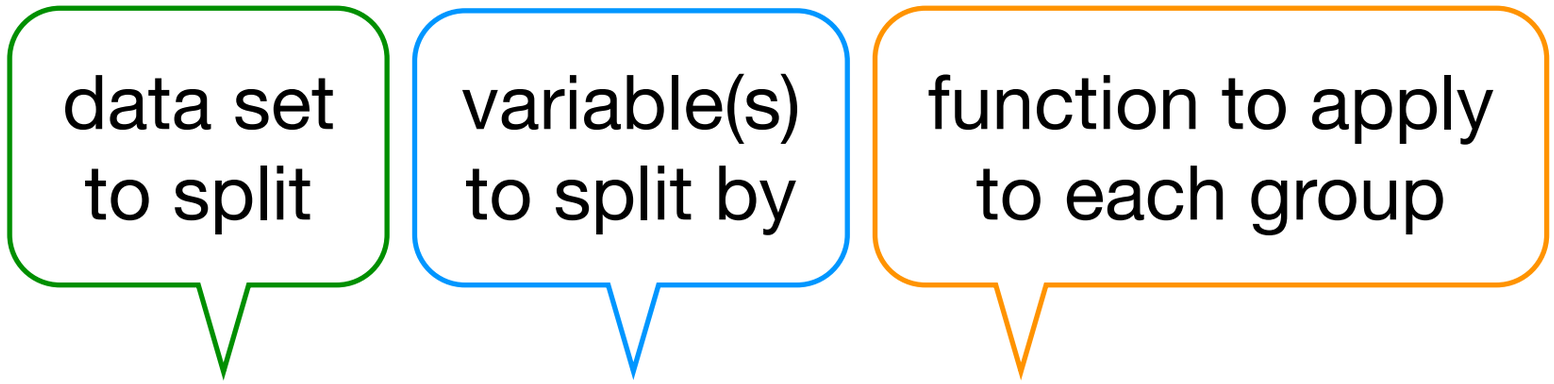

```
# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

ddply

split-combine-apply with **ddply**

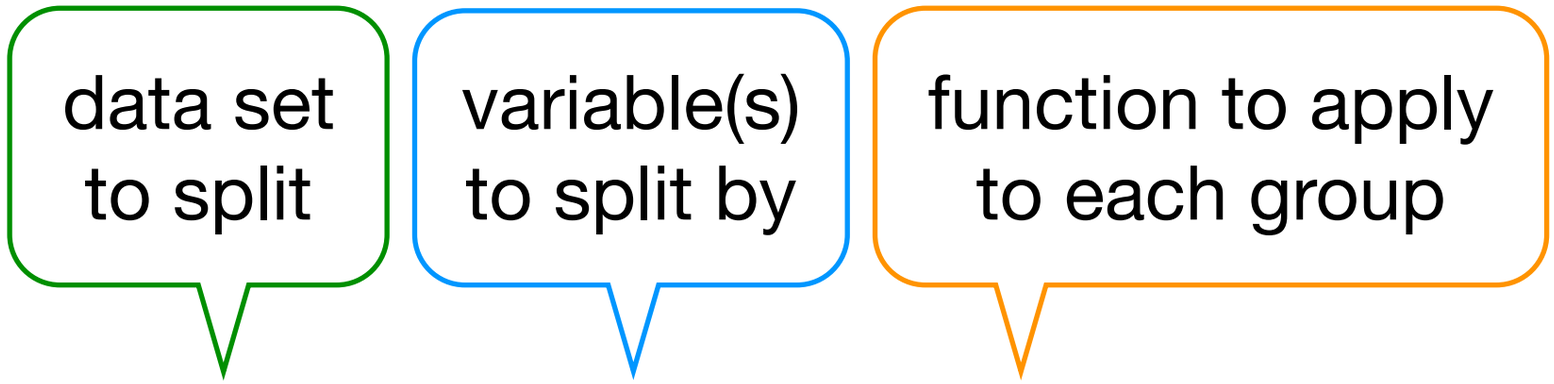
```
ddply( )
```

split-combine-apply with **ddply**



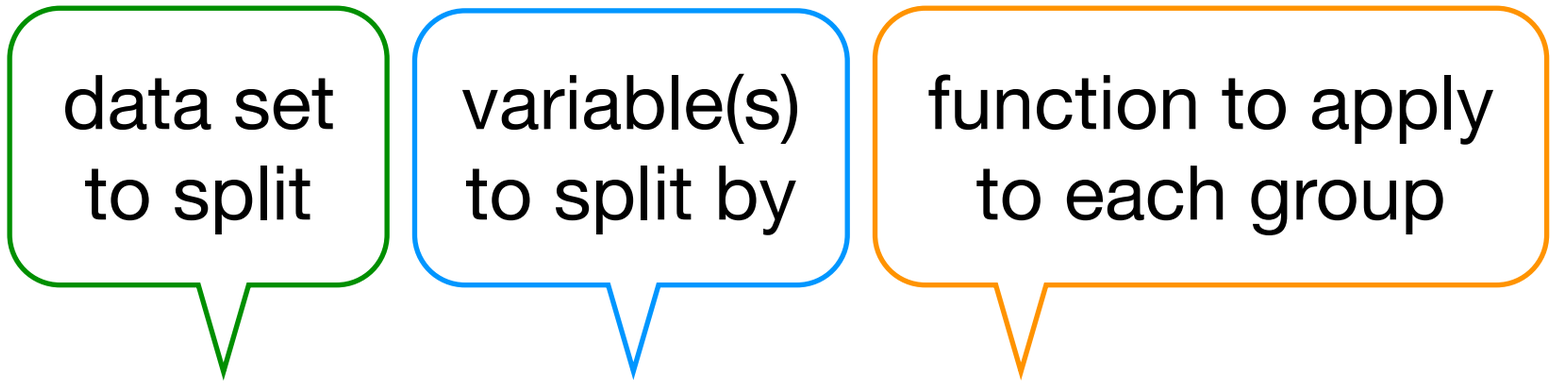

```
ddply(                                    )
```

split-combine-apply with **ddply**



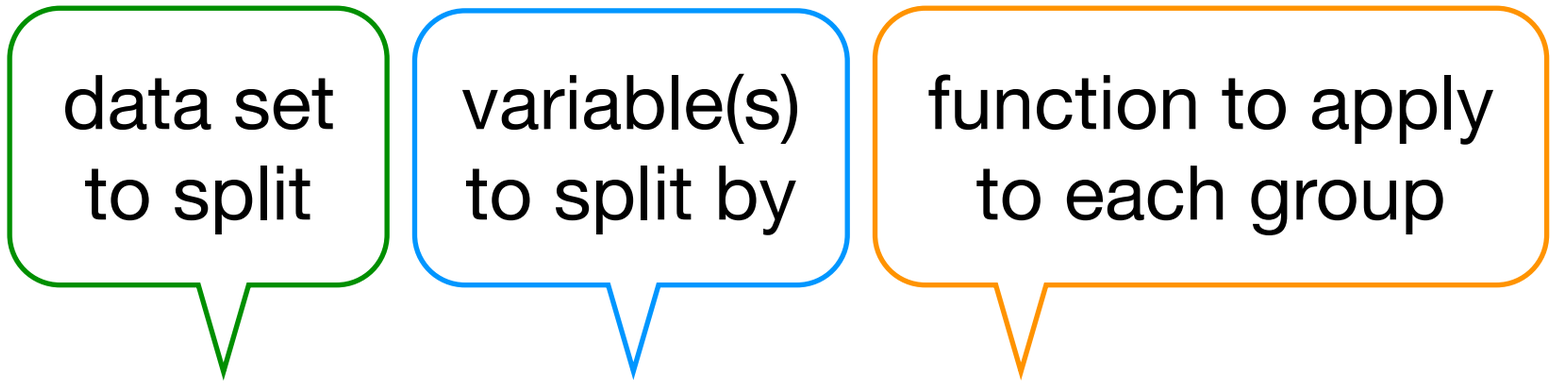

```
ddply(bnames2                                    )
```

split-combine-apply with **ddply**



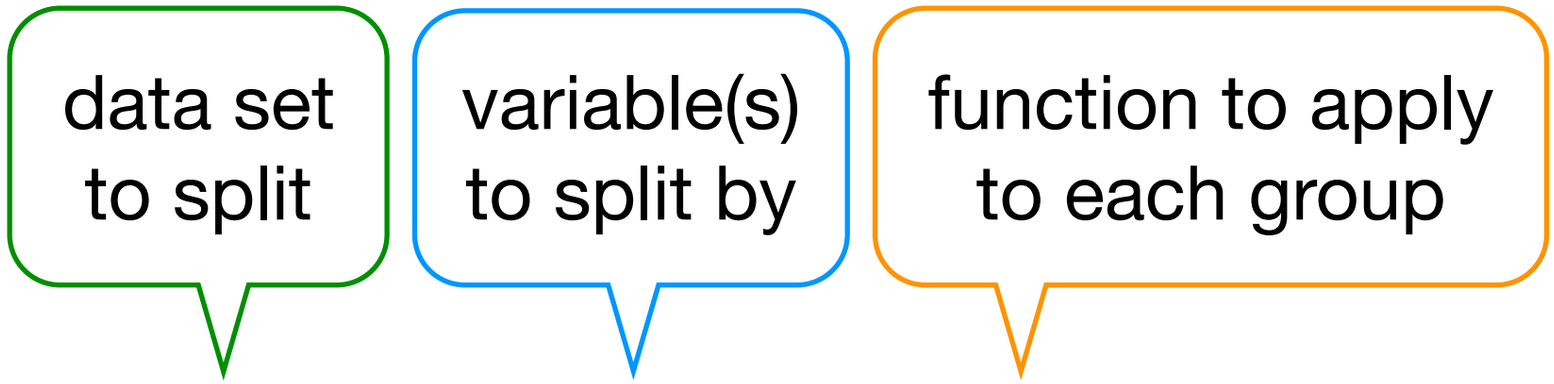

```
ddply(bnames2, "name"                    )
```

split-combine-apply with **ddply**




```
ddply(bnames2, "name", summarise)
```

split-combine-apply with **ddply**

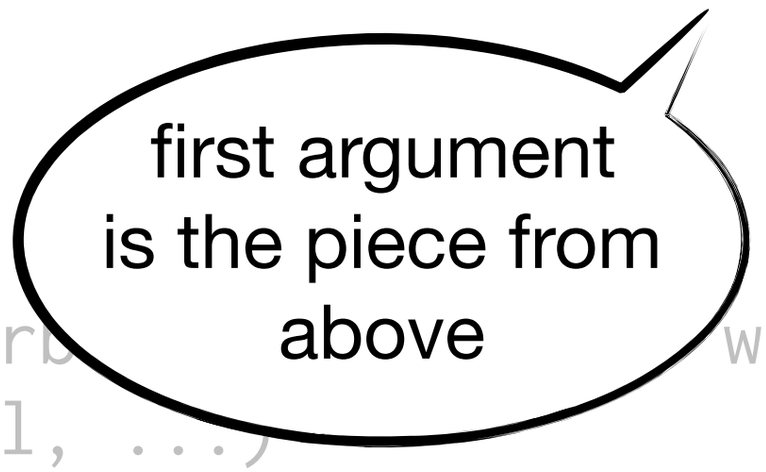



```
ddply(bnames2, "name", summarise)
```

Q: How many arguments did summarise get in our previous code? Will it know what they are above?

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- summarise(john, total = sum(n))
william_years <- summarise(william, total = sum(n))
james_years <- summarise(james, total = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- summarise(john, total = sum(n))
william_years <- summarise(william, total = sum(n))
james_years <- summarise(james, total = sum(n))
...

# combine
totals <- rb                    william_total,
james_total, ...)
```



```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- summarise(john, total = sum(n))
william_years <- summarise(william, total = sum(n))
james_years <- summarise(james, total = sum(n))
...

# combine
totals <- rbind(john_total, william_total,
james_total, ...)
```

```
# Split
john <- subset(bnames2, name == "John")
william <- subset(bnames2, name == "William")
james <- subset(bnames2, name == "James")
...

# Apply
john_years <- summarise(john, total = sum(n))
william_years <- summarise(william, total = sum(n))
james_years <- summarise(james, total = sum(n))
...

# combine
totals <- rb                    , william_to
james_total, ...)
```

first argument is the piece from above

the second argument is what we want summarise to do

split-combine-apply with **ddply**

```
ddply(bnames2, "name", summarise, ... )
```

split-combine-apply with **ddply**

data set to split

variable(s) to split by

function to use on each group

```
ddply(bnames2, "name", summarise, ... )
```

split-combine-apply with **ddply**

data set to split

variable(s) to split by

function to use on each group

extra arguments for function

```
ddply(bnames2, "name", summarise, ... )
```

Your turn

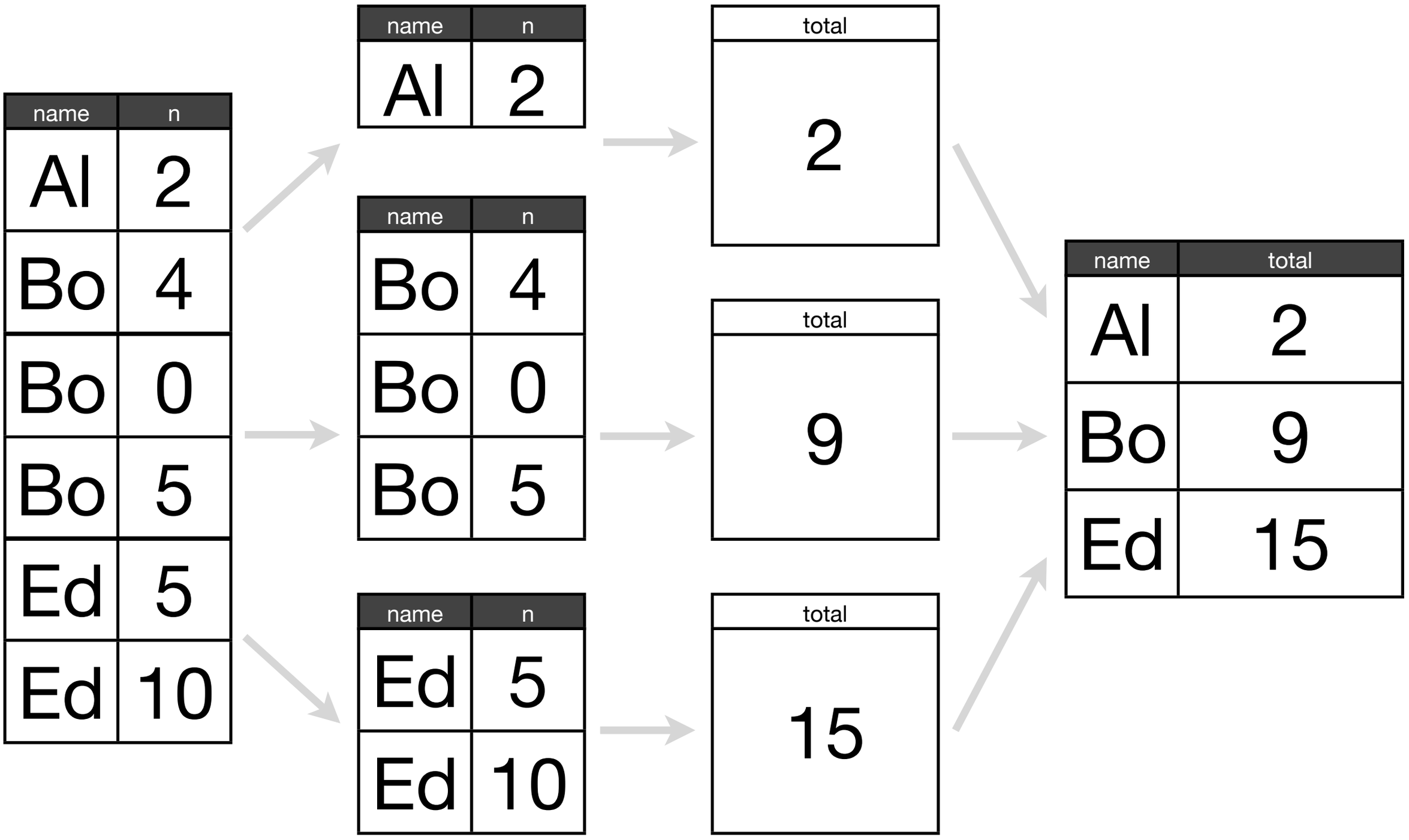
In your group, use `ddply` to finish calculating the totals for each name.

Hint: be sure to save your output to something.

	name	total
1	Aaden	959
2	Aaliyah	39665
3	Aarav	219
4	Aaron	509464
5	Ab	25
6	Abagail	2682
7	Abb	16
8	Abbey	14348
9	Abbie	16622
10	Abbigail	6800

```
totals <- ddply(bnames2, "name", summarise,
  total = sum(n))
```

Split Apply Combine




```
ddply(bnames2, "name", summarise, total = sum(n))
```

Number per soundex

- How can we compute the total number of people who ever had each *soundex*?

Number per soundex

- How can we compute the total number of people who ever had each *soundex*?

Predict (in your head) the code that will do this.

Your turn

Repeat the same operation, but use soundex instead of name. This tells us the total number of people whose names ever sounded like each soundex.

What is the most common sound? What name does it correspond to?

```
stotals <- ddply(bnames2, "soundex", summarise,
  total = sum(n))
stotals <- arrange(stotals, desc(n))
```

```
subset(bnames, soundex == "W450")
```

Workflow

1. Extract a single group
2. Find a function that solves it for just that group
3. Combine the function with ddply to solve it for all groups

Transformations

- How can we make a new variable that shows the rank of each name for each year? We should treat boys names as distinct from girls names (even if they're spelled the same).

Transformations

- How can we make a new variable that shows the rank of each name for each year? We should treat boys names as distinct from girls names (even if they're spelled the same).

Hint: We'll use rank()

Your turn: Step 1

Extract a single year's worth of names. Make sure they are all the same sex. Try year = 2008 and sex = boy.

Your turn: Step 2

Add a rank variable to our subset. Use rank().

Your turn: Step 3

Combine

```
mutate(boys2008, rank = rank(desc(prop)))
```

with ddply and apply to the entire data set.

```
boys2008 <- subset(bnames2,
  year == 2008 & sex == "boy")

mutate(boys2008, rank == rank(desc(prop)))

ranks <- ddply(bnames2, c("year", "sex"),
   mutate, rank = rank(desc(prop)))
```

Challenges

Solving new problems

- Simple problems: identify which one of `subset`, `summarise`, `mutate` or `arrange` that you need. Combine with `ddply`.
- Complex problems: figure out a sequence of simple problems that leads you from the raw data to the final problem.

Warmups

Which names were most popular in 1999?

Work out the average yearly usage of each name.

List the 10 names with the highest average proportions.

```r
# Which names were most popular in 1999?
subset(bnames2, year == 1999 & rank < 10)
n1999 <- subset(bnames2, year == 1999)
head(arrange(n1999, desc(prop)), 10)

# Average usage
overall <- ddply(bnames2, "name", summarise,
   prop1 = mean(prop),
   prop2 = sum(prop) / 129)

# Top 10 names
head(arrange(overall, desc(prop)), 10)
```

Set up for challenges

```
library(stringr)

# extracts first and last letter of each name
bnames2 <- mutate(bnames2,
   first = str_sub(name, 1, 1),
   last = str_sub(name, -1, -1))
```

Challenge 1

How has the total proportion of babies with names in the top 1000 changed over time?

How has the popularity of different initials changed over time?

```
top1000 <- ddply(bnames2, c("year","sex"), summarise,
  prop = sum(prop),
  npop = sum(prop > 1/1000))

qplot(year, prop, data = top1000, colour = sex,
  geom = "line")
qplot(year, npop, data = top1000, colour = sex,
  geom = "line")
```

```
init <- ddply(bnames2, c("year","first"), summarise,
  prop = sum(prop)/2)
```

```
qplot(year, prop, data = init, colour = first,
  geom = "line")
```

Challenge 2

For each name, find the year in which it was most popular, and the rank in that year. (Hint: you might find which.max useful).

Print all names that have been the most popular name at least once.

```
most_pop <- ddply(bnames2, "name", summarise,
  year = year[which.max(prop)],
  rank = min(rank))
most_pop <- ddply(bnames2, "name", subset,
  prop == max(prop))

subset(bnames2, rank == 1)
```

Challenge 3

What two names (boy and girl) have been in the top 10 most often?

(Hint: you'll have to do this in three steps. Think about what they are before starting)

```r
top10 <- subset(bnames2, rank <= 10)
counts <- ddply(top10, c("sex", "name"), nrow) # or
counts <- count(top10, c("sex", "name"))
```

```r
ddply(counts, "sex", subset, freq == max(freq)) # or
head(arrange(counts, desc(freq)), 10
```

Challenge 4

For each soundex, find the most common name in that group.

Hint: use count (see ?count)

```
names <- count(bnames2, c("soundex", "name"), "n")
```

```
ddply(names, "soundex", subset, freq == max(freq))
```

More about plyr

	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

http://plyr.had.co.nz