

## 4-more dplyr

### Rearranging Data

Iowa State University

# Outline

- ▶ working with ddp1y: an example
- ▶ plyr variations

# French Fries

- ▶ 10 week experiment
- ▶ 12 participants
- ▶ 3 types of oil
- ▶ 2 replicates for each individual, each week
- ▶ Participants asked to evaluate how potato-y, buttery, grassy, rancid, and paint-y the french fries taste

What would be interesting to investigate?

# What would be interesting to investigate?

- ▶ How have average ratings changed over time?
- ▶ Do certain participants give higher ratings than others?
- ▶ How similar are different replicates for each subject?
- ▶ Do different oils have different rating trajectories over time?

All of these questions can be answered using ddply.

# Your Turn

Answer one or more of the questions you came up with using `ddply`

Sample Questions:

- ▶ How do average ratings change over time for each variable?
- ▶ How similar are different replicates for each subject?
- ▶ Do different oils have different average rating trajectories over time?

# stringr package

## The str\_command philosophy for more consistency with string operators

- Basics
  - `str_c`  
Join multiple strings into a single string.
  - `str_join`  
Join multiple strings into a single string.
  - `str_length`  
The length of a string (in characters).
  - `str_dup`  
Duplicate and concatenate strings within a character vector.
  - `str_trim`  
Trim whitespace from start and end of string.
  - `word`  
Extract words from a sentence.
  - `str_split`  
Split up a string into a variable number of pieces.
- Pattern Detection
  - `str_detect`  
Detect the presence or absence of a pattern in a string.
- `str_locate`, `str_locate_all`  
Locate the position of the first, all occurrence of a pattern in a string.
- `str_count`  
Count the number of matches in a string.
- `str_match`, `str_match_all`  
Extract first, all matched group from a string.
- Pattern Extraction & Replacement
  - `str_extract`, `str_extract_all`  
Extract first, all piece of a string that matches a pattern.
  - `str_sub`  
Extract substrings from a character vector.
  - `str_sub_replace`  
Replace substrings in a character vector.  
'str\_sub<.' will recycle all arguments to be the same length as the longest argument.
  - `str_replace`, `str_replace_all`  
Replace first occurrence of a matched pattern in a string.

# transform

Transform allows multiple statements with the same dataset (shortcut)

```
data$newvar1 = ...  
data$newvar2 = ...  
data$newvar3 = ...
```

With transform, this becomes...

```
data <- transform(data,  
  newvar1 = ...,  
  newvar2 = ...,  
  newvar3 = ...  
)
```

ddply and transform together allow us to perform group-wise operations



# OkCupid Data

OkCupid is an online dating site. This dataset contains demographic information (age, gender, location, relationship status, religious affiliation) from a subset of OkCupid users, along with their response to one essay prompt.

[http:](http://heike.github.io/R-workshops/03-r-format/data/OkCupid.csv)

[//heike.github.io/R-workshops/03-r-format/data/OkCupid.csv](http://heike.github.io/R-workshops/03-r-format/data/OkCupid.csv)

Read the data into R:

```
profile <- read.csv("http://heike.github.io/R-workshops/03-r-format/dat
```

## Your Turn

Use `Stringr` and `transform` to create separate variables for the state, city, and height in inches of each user.

Hints:

- ▶ `word()` will give you single items in a string, so if you define `sep=" ", "`, you can extract the city and state separately by changing start and end
- ▶ Use `word()` with a different `sep=""`, twice. The second time, specify start.  
Alternately, use `str_sub()` to get feet and/or inches, with `str_locate()` to get the relevant start/end locations.

## ddply + separate function

Instead of using an existing function, we can have more flexibility and write our own function:

```
data <- ddply(data, .(id),  
              function(x){  
                ...  
              })
```

This function should return a data frame (for now).

Notice that you no longer have to use one of “summarize”, “transform”, etc. after your `.(id)` statement.

## Your Turn

Use `ddply` to create a dataset that examines the proportion of users of each gender and one other variable in each state that are on OkCupid.

**Hint:** Variables with a small number of choices, such as orientation, status, Education, Drinks, Smokes are good options.

**Hint 2:** Use `table()` and `as.data.frame` to convert the table into something we can deal with.

## Variations on dply

- ▶ plyr commands have the format `xply`
- ▶ x and y are letters representing different object types
- ▶ x is “going in” and y is “coming out”

letter	object
l	list
d	data frame
m	data frame or matrix
a	array (vector or matrix)
—	no output
r	input only

**Example:** `ldply` takes a list and returns a data frame

# Your Turn

Use one of the `plyr` commands to do each of the following:

1. For each state, return the longest essay (`essay_0`) for each gender
2. From the previous output, extract the most common word (use `table()` and `order()`) and count the number of sentences (or at least the number of `"."`'s. ).