# Introduction to `dplyr` and other data manipulation techniques

Kiva Oken

April 4, 2017

# The tidyverse

## Group of 9 packages

dplyr, forcats, ggplot2, haven, purrr, readr, stringr, tibble, tidyr

# The tidyverse

## Group of 9 packages

dplyr, forcats, ggplot2, haven, purrr, readr, stringr, tibble, tidyr

# The tidyverse

## Group of 9 packages

dplyr, forcats, ggplot2, haven, purrr, readr, stringr, tibble, tidyr

```
install.packages('tidyverse')
```

```
require(tidyverse)
```

# Packages for another time

- `forcats`: Tools for dealing with factors
- `ggplot2`: Plotting
- `haven`: Read files from SAS, SPSS, Stata. NOT Matlab.
- `purrr`: Functional programming
- `readr`: Reading in rectangular data files
- `stringr`: Tools for dealing with strings.

# dplyr

- **Split** up a dataset
- **Apply** a function to each piece
- **Combine** pieces back together

Always returns a data frame!

# Basic `dplyr` verbs

- **Filter:** filter rows, like `subset()`
- **Arrange:** reorder rows according to an index, like `df[order(index),]`
- **Select:** select certain columns, like `select` argument to `subset()` (also `$`, `df[,index]`)
- **Mutate:** Add new columns that are a function of other columns, like `transform()`
- **Summarize:** Collapses a data frame (or subset of a data frame) into a single row

# filter() and slice()

```
olaf.assessments <- filter(assessment,
                           recorder == 'JENSEN')
```

# filter() and slice()

```
olaf.assessments <- filter(assessment,
                           recorder == 'JENSEN')
```

```
cod <- filter(stock, grepl('Gadus', scientificname))
```

# filter() and slice()

```
olaf.assessments <- filter(assessment,
                           recorder == 'JENSEN')
```

```
cod <- filter(stock, grepl('Gadus', scientificname))
```

```
slice(assessment, 5:8)
```

# Basic `dplyr` verbs

- **Filter:** filter rows, like `subset()`
- **Arrange:** reorder rows according to an index, like `df[order(index),]`
- **Select:** select certain columns, like `select` argument to `subset()` (also `$`, `df[,index]`)
- **Mutate:** Add new columns that are a function of other columns, like `transform()`
- **Summarize:** Collapses a data frame (or subset of a data frame) into a single row

# arrange()

```
arrange(olaf.assessments, daterecorded)
arrange(olaf.assessments, desc(daterecorded))
```

# Basic `dplyr` verbs

- **Filter:** filter rows, like `subset()`
- **Arrange:** reorder rows according to an index, like `df[order(index),]`
- **Select:** select certain columns, like `select` argument to `subset()` (also `$`, `df[,index]`)
- **Mutate:** Add new columns that are a function of other columns, like `transform()`
- **Summarize:** Collapses a data frame (or subset of a data frame) into a single row

# select() and rename()

```
select(olaf.assessments, stockid)
select(olaf.assessments, stock.id = stockid)
rename(olaf.assessments, stock.id = stockid)
select(olaf.assessments, assessid:stockid)
select(olaf.assessments, -recorder)
```

# Basic `dplyr` verbs

- **Filter:** filter rows, like `subset()`
- **Arrange:** reorder rows according to an index, like `df[order(index),]`
- **Select:** select certain columns, like `select` argument to `subset()` (also `$`, `df[,index]`)
- **Mutate:** Add new columns that are a function of other columns, like `transform()`
- **Summarize:** Collapses a data frame (or subset of a data frame) into a single row

# mutate() and transmute()

```
olaf.delay <- mutate(olaf.assessments,
                     delay = dateloaded -
                       daterecorded)
select(olaf.delay, delay)

transmute(olaf.assessments,
                     delay = dateloaded -
                       daterecorded)

toothfish.ssb <- filter(timeseries, assessid ==
                          'CCAMLR-ATOOTHFISHRS-1995-2007-JH
                        tsid == 'SSB-MT')
mutate(toothfish.ssb,
       zscore = (tsvalue - mean(tsvalue)) /
         sd(tsvalue))
```

# Basic `dplyr` verbs

- **Filter:** filter rows, like `subset()`
- **Arrange:** reorder rows according to an index, like `df[order(index),]`
- **Select:** select certain columns, like `select` argument to `subset()` (also `$`, `df[,index]`)
- **Mutate:** Add new columns that are a function of other columns, like `transform()`
- **Summarize:** Collapses a data frame (or subset of a data frame) into a single row

# summarize()

```
summarize(toothfish.ssb, mean(tsvalue, na.rm = TRUE))

##   mean(tsvalue, na.rm = TRUE)
## 1                    67070.98

summarize(toothfish.ssb, n_distinct(tsvalue), n())

##   n_distinct(tsvalue) n()
## 1                  11  13
```

# summarize()

```
do_something <- function(vec) {
  sum(vec, na.rm = TRUE)/5
}
summarize(toothfish.ssb, do_something(tsvalue))
summarise(toothfish.ssb, do_something(tsvalue))
```

# Aggregating functions

Accept vectors. Return scalars. (True?)

- Used for `summarize()`
- Base R: `min()`, `max()`, `mean()`, etc.
- From dplyr:
  `n()`, `n_distinct()`, `first()`, `last()`, `nth()`
- Write your own, can use C++ for speed

# And one more: `Do()`

- **Do:** general purpose verb to complement the specialized functions, like `dlply()`

```
assessors <- do(olaf.assessments,
                assessor = unique(.$assessorid))
assessors
assessors$assessor
class(assessors)

distinct(olaf.assessments, assessorid)
```

# Practice

1. Create a data frame in R that contains the time series data of Atlantic Amberjack using `filter()`. **Hint:** this stock is in `olaf.assessments`.
2. Using `filter()` and one other `dplyr` function, determine in which year Amberjack had the highest recruitment (R-E00).

# Pipes! (%>%)

`f(x) %>% g(y)` $\Leftrightarrow$ `g(f(x), y)`

# Pipes! (%>%)

$$\texttt{f(x) \%>\% g(y)} \Leftrightarrow \texttt{g(f(x), y)}$$

Three ways to get the same result:

# Pipes! (%>%)

$$\texttt{f(x) \%>\% g(y)} \Leftrightarrow \texttt{g(f(x), y)}$$

Three ways to get the same result:

```
1 x <- rnorm(100)
  x.mat <- matrix(x, nrow = 10)
  x.mns <- apply(x.mat, 1, mean)
```

# Pipes! (%>%)

$$\texttt{f(x) \%>\% g(y)} \Leftrightarrow \texttt{g(f(x), y)}$$

Three ways to get the same result:

1.
```
x <- rnorm(100)
x.mat <- matrix(x, nrow = 10)
x.mns <- apply(x.mat, 1, mean)
```

2.
```
x.mns <- apply(matrix(rnorm(100), nrow = 10),
               1, mean)
```

# Pipes! (%>%)

$$\texttt{f(x) \%>\% g(y)} \Leftrightarrow \texttt{g(f(x), y)}$$

Three ways to get the same result:

1.
```
x <- rnorm(100)
x.mat <- matrix(x, nrow = 10)
x.mns <- apply(x.mat, 1, mean)
```

2.
```
x.mns <- apply(matrix(rnorm(100), nrow = 10),
               1, mean)
```

3.
```
x.mns <- rnorm(100) %>%
  matrix(nrow = 10) %>%
  apply(1, mean)
```

# group_by()

The workhorse of dplyr, like tapply(), ddply()

```
toothfish <- filter(timeseries, assessid ==
    'CCAMLR-ATOOTHFISHRS-1995-2007-JENSEN') %>%
  select(tsid:tsvalue) %>%
  group_by(tsid)

summarize(toothfish, mn = mean(tsvalue, na.rm = TRUE),
          stdev = sd(tsvalue, na.rm = TRUE))
slice(toothfish, 1)
mutate(toothfish, z.score =
         (tsvalue - mean(tsvalue, na.rm=TRUE)) /
         sd(tsvalue, na.rm=TRUE)) %>%
  View()
```

# Join

Join multiple data frames together based on a common variable (e.g., species)

# Join

Join multiple data frames together based on a common variable (e.g., species)

- **Inner join:** rows with matching values in both data frames, columns from both data frames
- **Left join:** all rows from first (left) data frame, columns from both data frames
- **Semi join:** rows with matching values in both data frames, columns from first data frame
- **Anti join:** rows from first data frame with*out* matching values in second, columns from first data frame

# Join example

```
select(stock, stockid, scientificname, commonname) %>%
  inner_join(assessment) %>%
  View()

## Joining, by = "stockid"
```

# Tidy data

**Untidy data (wide)**

| Control | Treatment |
|---|---|
| $c_1$ | $t_1$ |
| $\vdots$ | $\vdots$ |
| $c_n$ | $t_n$ |

**Tidy data (long)**

| Condition | Value |
|---|---|
| Control | $c_1$ |
| $\vdots$ | $\vdots$ |
| Control | $c_n$ |
| Treatment | $t_1$ |
| $\vdots$ | $\vdots$ |
| Treatment | $t_n$ |

# The tidyr package

Update of reshape2

- gather(): Wide to long data frame
- spread(): Long to wide data frame

```
wide.toothfish <- ungroup(toothfish) %>%
  mutate(tsid = gsub('-', '_', tsid)) %>%
  spread(key = 'tsid', value = 'tsvalue')
long.toothfish <- gather(wide.toothfish,
                         key = tsid, value = tsvalue,
                         BdivBmgttouse_dimensionless:
                           Utouse_index)
```

# Practice II

1. Create a data frame in R of data for Pacific herring (*Clupea pallasii*) that is grouped by stock and population metric (SSB, recruitment, etc.). You will need to join information from all three of the data tables in the RAM database to do this.
2. Using your data frame, calculate the mean and standard deviation of each population metric (SSB, recruitment, etc.) for each area. Note that the database contains NAs.
3. Plot the time series of spawning stock biomass of Pacific herring to compare across regions using either `do()` with base graphics or `ggplot()`.
4. Bonus: Color the lines produces above by exploitation rate (ER-ratio). You will probably need to use `tidyr`.

# The end

Further resources:

- Package vignettes for `dplyr`, `tidyr`
- www.tidyverse.org
- Trevor Branch Super Advanced R course webpage
- Google, Stack Exchange, etc.