

Data Wrangling with Dplyr

**Symposium: Using RStudio for Visualization and Analysis
of Weed Science Experiments**

Maxwel Coura Oliveira, PhD

Department of Agronomy
University of Wisconsin-Madison

December 2019

Outline

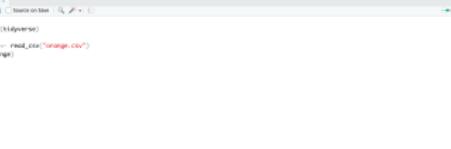
- 7 dplyr verbs for data manipulation
 - **select, filter, group_by, summarize, mutate, count, arrange**
- Combining verbs using pipes `%>%`
- 2 tidyr verbs to reshape your data (**spread, gather**)

Prerequisites

- Install R and R studio
- See the R basics lesson if you're unfamiliar with R or R studio

Create a new R script

- File > New File > R script
 - Save it in your project directory
 - Look on the top left of the R Studio window to find project directory



The screenshot shows the RStudio interface with the following details:

- Top Bar:** File, Edit, Code, View, Help, Profile, Session, Build, Debug, Profile, Tools, Window.
- Toolbar:** Run, Save, Stop, Reload, Hide, Help.
- Code Editor:** Shows R code for reading a CSV file named "orange.csv".
- Console:** Displays the output of the R code, showing the loading of the tidyverse package and its components: ggplot2, dplyr, purrr, tibble, stringr, forcats, and gridExtra.
- Status Bar:** tidyverse 1.2.1

What is the tidyverse?

- Packages for data manipulation
- Built for data tables
- Makes data manipulation easier than in base R
- Combine verbs using pipes



Installing packages

```
install.packages()
```

- Input: package name
- Downloads packages from CRAN (Comprehensive R Archive Network)
- Install once per machine

```
install.packages("tidyverse")
```

Loading packages

```
library(packagename)
```

```
library(tidyverse)
```

- **Input:** package name
- Gives R access to functions in the package
- Load packages every time you restart R

Data set: barley yields in Minnesota

- Stored in R as **barley** dataset at lattice package

```
library(lattice)
```

- Rows: observations of individual columns
- *Columns*: Variables that describe the experiment
 - yield, variety, year, site



barley data

barley

```
##      yield      variety year      site
## 1 27.00000 Manchuria 1931 University Farm
## 2 48.86667 Manchuria 1931          Waseca
## 3 27.43334 Manchuria 1931          Morris
## 4 39.93333 Manchuria 1931 Crookston
## 5 32.96667 Manchuria 1931 Grand Rapids
## 6 28.96667 Manchuria 1931 Duluth
## 7 43.06666 Glabron   1931 University Farm
## 8 55.20000 Glabron   1931          Waseca
## 9 28.76667 Glabron   1931          Morris
## 10 38.13333 Glabron   1931 Crookston
## 11 29.13333 Glabron   1931 Grand Rapids
## 12 29.66667 Glabron   1931 Duluth
## 13 35.13333 Svansota  1931 University Farm
## 14 47.33333 Svansota  1931          Waseca
## 15 25.76667 Svansota  1931          Morris
## 16 40.46667 Svansota  1931 Crookston
## 17 29.66667 Svansota  1931 Grand Rapids
## 18 25.70000 Svansota  1931 Duluth
```

Import data in tidyverse

- `read_csv()` – loads contents of a CSV file
- *Input:* a `file` path
- *Output* a “tibble”

Why not `read.csv()`?

- `read_csv()` is faster
- Create tibbles
- More reproducible

Data frame	Tibble
Strings to factors	Keeps character
Has row names	No row names
Changes column names	Keeps column names as they are

dplyr verbs

- First argument is always a table
 - Tibble or data frame
 - Output is a new table
 - Doesn't change input data
 - Must save the output using `<-`
- `new_df <- verb(old_df, ...)`
- OR
- `old_df <- verb(old_df, ...)`

select()

- Selects columns from a data frame
- Input: data and columns to be kept
- Output: data with only the specified columns

```
select(barley, site)
##          site
## 1 University Farm
## 2 Waseca
## 3 Morris
## 4 Crookston
## 5 Grand Rapids
## 6 Duluth
## 7 University Farm
## 8 Waseca
## 9 Morris
## 10 Crookston
## 11 Grand Rapids
## 12 Duluth
## 13 University Farm
## 14 Waseca
## 15 Morris
## 16 Crookston
## 17 Grand Rapids
## 18 Duluth
## 19 University Farm
## 20 Waseca
## 21 Morris
## 22 Crookston
## 23 Grand Rapids
## 24 Duluth
## 25 University Farm
```

filter()

- Choose rows based on values of a variable
- **Input:** data and a logical expression (returns true/false)
 - <, >, >=, <=, ==, !=
- **Output:** data with rows that match the expression

```
filter(barley, site == "Waseca")
## #> #>   yield      variety year site
## #> 1 48.86667 Manchuria 1931 Waseca
## #> 2 55.20000 Glabron 1931 Waseca
## #> 3 47.33333 Svansota 1931 Waseca
## #> 4 50.23333 Velvet 1931 Waseca
## #> 5 63.83330 Trebi 1931 Waseca
## #> 6 58.10000 No. 457 1931 Waseca
## #> 7 65.76670 No. 462 1931 Waseca
## #> 8 48.56666 Peatland 1931 Waseca
## #> 9 46.76667 No. 475 1931 Waseca
## #> 10 58.80000 Wisconsin No. 38 1931 Waseca
## #> 11 33.46667 Manchuria 1932 Waseca
## #> 12 37.73333 Glabron 1932 Waseca
## #> 13 38.50000 Svansota 1932 Waseca
## #> 14 37.40000 Velvet 1932 Waseca
## #> 15 49.23330 Trebi 1932 Waseca
## #> 16 42.20000 No. 457 1932 Waseca
## #> 17 44.70000 No. 462 1932 Waseca
## #> 18 36.03333 Peatland 1932 Waseca
## #> 19 41.26667 No. 475 1932 Waseca
## #> 20 58.16667 Wisconsin No. 38 1932 Waseca
```

Pipe operator %>%

- Combine multiple verbs
- **Syntax:** %>% at the end of the line
- Output of the first line becomes input of next line, etc.
- Say it out loud as “then”

```
barley %>%
  filter(yield > 50) %>%
  select(site, yield)

##      site     yield
## 1 Waseca 55.20000
## 2 Waseca 50.23333
## 3 Waseca 63.83330
## 4 Waseca 58.10000
## 5 Waseca 65.76670
## 6 Waseca 58.80000
## 7 Waseca 58.16667
```

Exercise #1: practice pipes

- Using pipes, subset the `barley` data to include
 - **yield** of individuals higher than 40 and lower than 25

mutate()

- Creates a new column
- **Input:** data and the definition of a new column
- `col_name =`
- **Output:** data with a new column

```
barley %>%
  mutate(yield_kgha = round(yield * 67.25, 0))
```

```
##      yield      variety year      site yield_kgha
## 1  27.00000 Manchuria 1931 University Farm    1816
## 2  48.86667 Manchuria 1931          Waseca    3286
## 3  27.43334 Manchuria 1931          Morris    1845
## 4  39.93333 Manchuria 1931       Crookston    2686
## 5  32.96667 Manchuria 1931     Grand Rapids    2217
## 6  28.96667 Manchuria 1931         Duluth    1948
## 7  43.06666 Glabron 1931 University Farm    2896
## 8  55.20000 Glabron 1931          Waseca    3712
## 9  28.76667 Glabron 1931          Morris    1935
## 10 38.13333 Glabron 1931       Crookston    2564
## 11 29.13333 Glabron 1931     Grand Rapids    1959
## 12 29.66667 Glabron 1931         Duluth    1995
## 13 35.13333 Svansota 1931 University Farm    2363
```

Exercise 2: data frame challenge

- Create a new data frame from the barley data that meets the following criteria:
 - 1 contains only the `site` and `yield` column and a new column called `yield_lb`
 - 2 `yield_lb` contains values that are yield in lb / 1000 sq. ft
- **Hint:** $1 \text{ bu/acre} = 1.38 \text{ lb}/1000 \text{ sq. ft}$

Creating a summary table

- `summarize()`
- **Input:** data and a summary statistic
 - Eg: `mean()`
 - `na.rm = TRUE`
- **Output:** a table with the calculated summary statistic

```
barley %>%
  summarize(mean_yield = mean(yield,
                               na.rm=TRUE))

##   mean_yield
## 1  34.42056
```

Creating a grouped summary table

- **group_by()**
- **Input:** data and a variable
- **Output:** a table with the calculated summary statistic for each unique value in the variable

```
barley %>%
  group_by(site) %>%
  summarize(mean_yield = mean(yield,
                               na.rm=TRUE))

## # A tibble: 6 x 2
##   site      mean_yield
##   <fct>        <dbl>
## 1 Grand Rapids    24.9
## 2 Duluth          28.0
## 3 University Farm 32.7
## 4 Morris          35.4
## 5 Crookston       37.4
## 6 Waseca          48.1
```

Removing missing values

- **is.na()**
 - missing = TRUE
 - not missing = FALSE
- **Input:** a column
- **Output:** logical vector
- Use it as input to **filter()**

```
barley %>%
  filter(!is.na(yield)) %>%
  group_by(variety) %>%
  summarize(mean_yield = mean(round(yield),
    na.rm=TRUE))
```

```
## # A tibble: 10 x 2
##   variety      mean_yield
##   <fct>          <dbl>
## 1 Svansota       30.3 
## 2 No. 462        35.5 
## 3 Manchuria      31.4 
## 4 No. 475        31.8 
## 5 Velvet         32.9 
## 6 Peatland        34.2 
## 7 Glabron        33.3 
## 8 No. 457        35.8 
## 9 Wisconsin No. 38 39.3 
## 10 Trebi          39.6
```

count()

- Count the number of observations
- Input:
 - categorical variable
- sort = TRUE: sorts the results
- Output: a table with a row for each categorical variable and a column called n with counts

```
barley %>%
  count(site)

## # A tibble: 6 x 2
##   site           n
##   <fct>     <int>
## 1 Grand Rapids    20
## 2 Duluth          20
## 3 University Farm 20
## 4 Morris          20
## 5 Crookston       20
## 6 Waseca          20

Same as

barley %>%
  group_by(site) %>%
  summarise(count=n())

## # A tibble: 6 x 2
##   site           count
##   <fct>     <int>
## 1 Grand Rapids    20
## 2 Duluth          20
## 3 University Farm 20
## 4 Morris          20
## 5 Crookston       20
## 6 Waseca          20
```

arrange()

- Order results in ascending order
- **Input:**
 - A variable
 - Use `desc()` to put them in descending order
- **Output:** A table ordered by the values of the input column

```
barley %>%
  group_by(variety) %>%
  arrange(desc(yield))

## # A tibble: 120 x 4
## # Groups:   variety [10]
##   yield variety      year site
##   <dbl> <fct>       <fct> <fct>
## 1 65.8 No. 462    1931 Waseca
## 2 63.8 Trebi     1931 Waseca
## 3 58.8 Wisconsin No. 38 1931 Waseca
## 4 58.2 Wisconsin No. 38 1932 Waseca
## 5 58.1 No. 457    1931 Waseca
## 6 55.2 Glabron    1931 Waseca
## 7 50.2 Velvet     1931 Waseca
## 8 49.9 Wisconsin No. 38 1931 Crookston
## 9 49.2 Trebi     1932 Waseca
## 10 48.9 Manchuria 1931 Waseca
## # ... with 110 more rows
```

Exercise 3

- 1 - Use `group_by()` and `summarize()` to find the `mean()`, `min()`, and `max()` `yield` for each `site`.
- 2 - **Bonus:** What is the yield gap between varieties in each site and location?

Reshaping data with `tidy`

- The shape of your data affects what you can do with it
- **Example:** compare the mean `yield` of each variety adding a new column (High or Low)



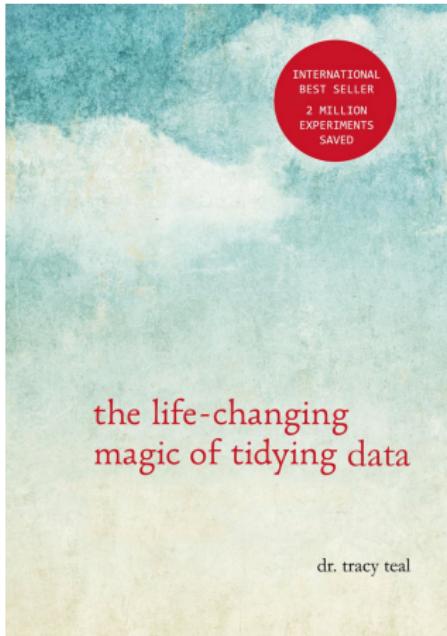
Exercise

- Create a table with columns for **variety** and **mean yield**. Add a logical parameter for mean **yield**, > 35 (High) or < 35 (Low)
- Save to a new object called **barley_nd**

```
barley_nd <- barley %>%  
  select(variety, yield) %>%  
  group_by(variety) %>%  
  summarise(mean_yield = mean(round(yield))) %>%  
  mutate(size = ifelse(mean_yield > 35,  
                      "High", "Low"))  
  
barley_nd  
  
## # A tibble: 10 x 3  
##   variety      mean_yield size  
##   <fct>        <dbl>     <chr>  
## 1 Svansota    30.3      Low  
## 2 No. 462     35.5      High  
## 3 Manchuria   31.4      Low  
## 4 No. 475     31.8      Low  
## 5 Velvet      32.9      Low  
## 6 Peatland    34.2      Low  
## 7 Glabron     33.3      Low  
## 8 No. 457     35.8      High  
## 9 Wisconsin No. 38 39.3      High  
## 10 Trebi      39.6      High
```

Tidy Data

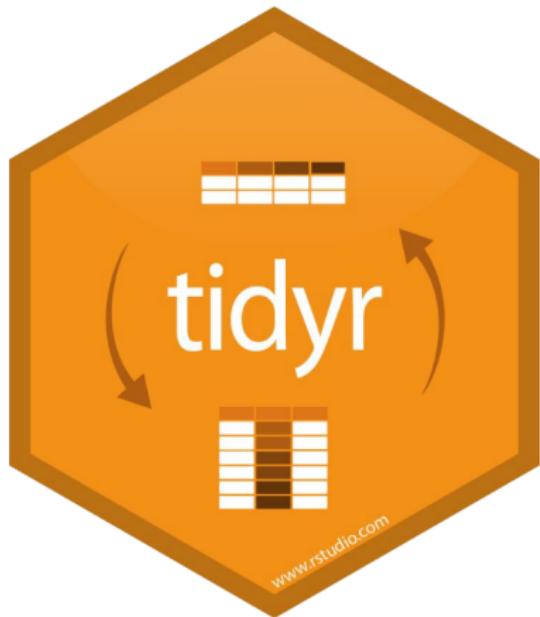
- 1) Each variable has its own column
- 2) Each observation has its own row
- 3) Each value has its own cell
- 4) Each type of observational unit forms a table



Reshaping data with `tidyverse`

- **Spreading:** makes a wider table
 - Unique values in a specified column (key) become variable names

- **Gathering:** makes a longer table
 - Variable names become values in a new column (key)



spread()

- use it when an observation is scattered across multiple rows
- **Input:**
 - **data** (a tibble)
 - **key** column (values become new column names)
 - **value** column (to fill new column variables)
- **Output:** a table with columns for each value of sex

```
barley_spread <- barley_nd %>%
  spread(key = size,
         value = mean_yield)

barley_spread

## # A tibble: 10 x 3
##   variety      High    Low
##   <fct>       <dbl>  <dbl>
## 1 Svansota     NA    30.3
## 2 No. 462      35.5   NA
## 3 Manchuria    NA    31.4
## 4 No. 475      NA    31.8
## 5 Velvet        NA    32.9
## 6 Peatland     NA    34.2
## 7 Glabron      NA    33.3
## 8 No. 457      35.8   NA
## 9 Wisconsin No. 38 39.3   NA
## 10 Trebi       39.6   NA
```

Spread

```
variety  
<fctr>  
Svansota  
No. 462  
Manchuria  
No. 475  
Velvet  
Peatland  
Glabron  
No. 457  
Wisconsin No. 38  
Trebi
```

mean_yield	size
30.33333	Low
35.50000	High
31.41667	Low
31.75000	Low
32.91667	Low
34.25000	Low
33.33333	Low
35.83333	High
39.33333	High
39.58333	High

```
variety  
<fctr>  
Svansota  
No. 462  
Manchuria  
No. 475  
Velvet  
Peatland  
Glabron  
No. 457  
Wisconsin No. 38  
Trebi
```

High	Low
NA	30.33333
35.50000	NA
NA	31.41667
NA	31.75000
NA	32.91667
NA	34.25000
NA	33.33333
35.83333	NA
39.33333	NA
39.58333	NA

gather()

- Use when column names are not names of variables, but values of a variable
- Input:
 - `data` (a tibble)
 - `key` column (created from col names)
 - values column (fill the key variable)
 - A range of columns to gather
- **Output:** a long tibble

```
barley_gather <- barley_spread %>%
  gather(key = size,
         value = mean_yield, 2:3, na.rm=TRUE)

barley_gather

## # A tibble: 10 x 3
##   variety      size mean_yield
##   <fct>       <chr>    <dbl>
## 1 No. 462     High     35.5
## 2 No. 457     High     35.8
## 3 Wisconsin No. 38 High     39.3
## 4 Trebi       High     39.6
## 5 Svansota    Low      30.3
## 6 Manchuria   Low      31.4
## 7 No. 475     Low      31.8
## 8 Velvet      Low      32.9
## 9 Peatland    Low      34.2
## 10 Glabron    Low      33.3
```

Gather

variety	size	High	<dbl>	Low	<dbl>
Svansota		NA		30.33333	
No. 462		35.50000		NA	
Manchuria		NA		31.41667	
No. 475		NA		31.75000	
Velvet		NA		32.91667	
Peatland		NA		34.25000	
Glabron		NA		33.33333	
No. 457		35.83333		NA	
Wisconsin No. 38		39.33333		NA	
Trebi		39.58333		NA	

variety	size	mean_yield	<dbl>
No. 462	High	35.50000	
No. 457	High	35.83333	
Wisconsin No. 38	High	39.33333	
Trebi	High	39.58333	
Svansota	Low	30.33333	
Manchuria	Low	31.41667	
No. 475	Low	31.75000	
Velvet	Low	32.91667	
Peatland	Low	34.25000	
Glabron	Low	33.33333	

write_csv

- Writes a data table to a file
- **Input:** a tibble, a file path
- **Output:** a file at the specified file path

```
write_csv(barley_gather,  
          path = "barley_2.csv")
```

Need help

- E-mail: maxoliveira@wisc.edu
- Data Wrangling Cheat Sheet - [Link](#)
- Thanks to Data Camp for sharing slides