

# Introduction to data wrangling and tidy data

Stephanie J. Spielman

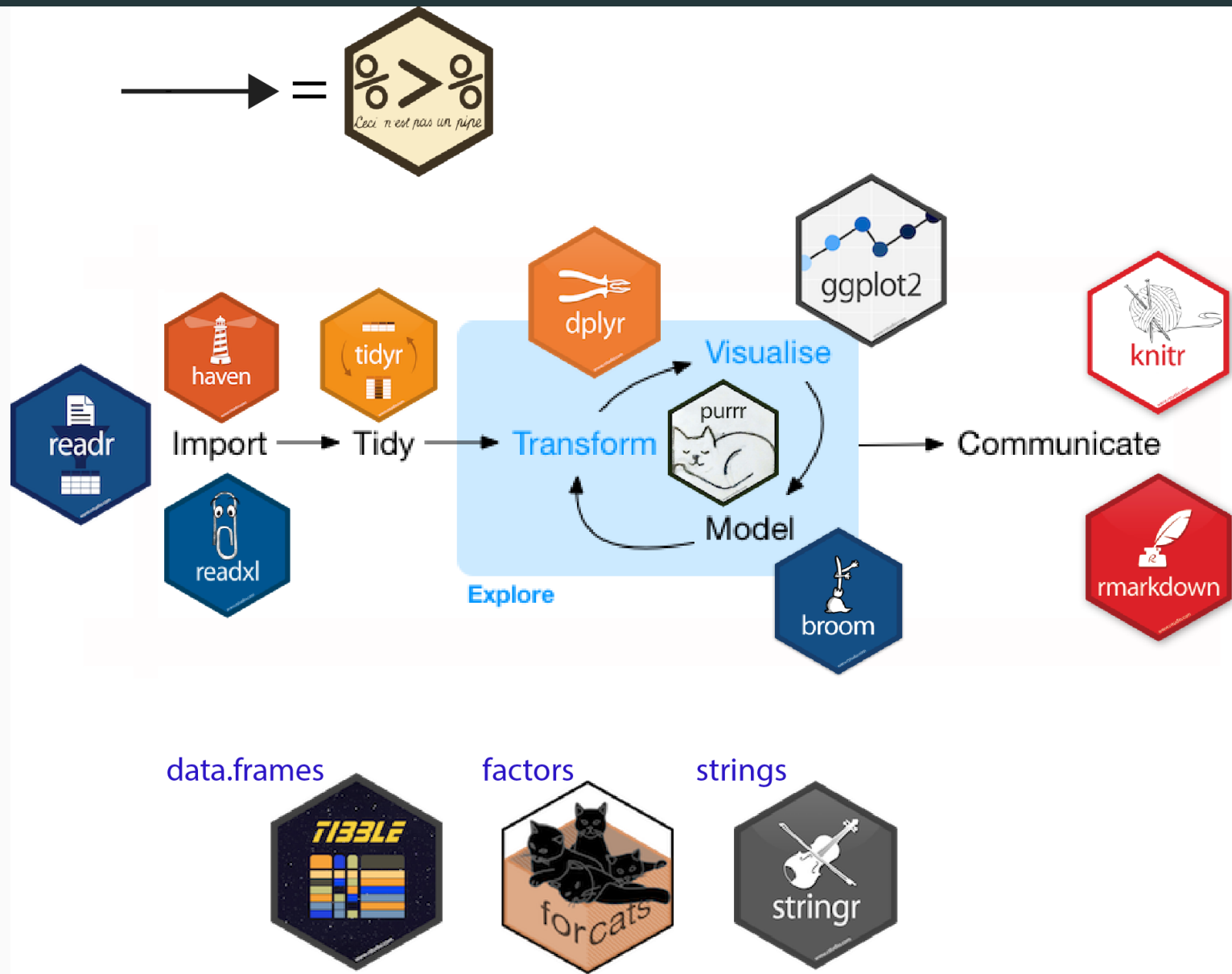
CB2R Data Science Workshop, Summer 2020



# Enter the tidyverse



# Enter the tidyverse



# We will be ~~learning~~ starting to learn...

- `dplyr` ("dee-plier") for manipulating ("wrangling") data
- `tidyr` ("tidy-are") for cleaning and *tidying* data
- `ggplot2` ("gee-gee-plot") for data visualization
- `tibble` ("tibble") for making data frames more enjoyable to work with (incidentally)
- `readr` ("read-are") and `readxl` ("read-excel") for reading and writing data
- `rmarkdown` ("are-markdown") for producing professional documents

# The philosophy of tidy data

country	year	cases	population
Afghanistan	1999	1745	19987071
Afghanistan	2000	1666	20593360
Brazil	1999	31737	172006362
Brazil	2000	80488	174504898
China	1999	211258	1272915272
China	2000	216766	128042583

variables

country	year	cases	population
Afghanistan	1999	1745	19987071
Afghanistan	2000	1666	20593360
Brazil	1999	31737	172006362
Brazil	2000	80488	174504898
China	1999	211258	1272915272
China	2000	216766	128042583

observations

country	year	cases	population
Afghanistan	99	745	987071
Afghanistan	00	666	593360
Brazil	99	737	006362
Brazil	00	488	504898
China	99	211258	915272
China	00	216766	42583

values

# "Messy" vs tidy data

## messy

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

## tidy

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

# Which version is tidy?

First, identify what the *variables* are in the dataset (the *data types* can help us!!)

```
## # A tibble: 5 x 3
##   seal oxygen_use_nonfeeding oxygen_use_feeding
##   <dbl>          <dbl>          <dbl>
## 1     1           42.2           71
## 2     2           51.7          77.3
## 3     3           82.8          78.6
## 4     4           66.5          96.1
## 5     5           81.9         107.
```



```
## # A tibble: 10 x 3
##   seal experiment_type oxygen_use
##   <dbl> <chr>          <dbl>
## 1     1 1 nonfeeding      42.2
## 2     1 1 feeding         71
## 3     2 2 nonfeeding      51.7
## 4     2 2 feeding         77.3
## 5     3 3 nonfeeding      82.8
## 6     3 3 feeding         78.6
## 7     4 4 nonfeeding      66.5
## 8     4 4 feeding         96.1
## 9     5 5 nonfeeding      81.9
## 10    5 5 feeding        107.
```



# Data wrangling with dplyr

I do not know all of the CHEATSHEET (let alone the whole package!). But, I know about the cheatsheet.

## Data Transformation with dplyr : : CHEAT SHEET

dplyr functions work with pipes and expect tidy data. In tidy data:

Each variable is in its own column. Each observation, or case, is in its own row.  $x \%>\% f() \rightarrow y$  becomes  $f(x, y)$

### Summarise Cases

These apply summary functions to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

**summary function**  
`summarise(data, ...)`  
Compute table of summaries.  
`summarise(mtcars, mpg = mean(mpg))`  
`count(x, ...)` Count number of rows in each group defined by the variables in ... Also tally.  
`count(mtcars, Species)`

**VARIATIONS**  
`summarise_at()` Apply funs to every column.  
`summarise_at(1)` Apply funs to specific columns.  
`summarise_if()` Apply funs to all cols of one type.

### Group Cases

Use `group_by()` to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

`mtcars %>% group_by(cyl) %>% summarise(avg = mean(mpg))`  
`group_by(data, ..., add = FALSE)`  
Returns copy of table grouped by ...  
`giris %>% group_by(giris, Species)`  
`ungroup(x, ...)`  
Returns ungrouped copy of table.  
`ungroup(giris)`

### Manipulate Cases

**EXTRACT CASES**  
Row functions return a subset of rows as a new table.

`filter(data, ...)` Extract rows that meet logical criteria. `filter(mtcars, Sepal.Length > 7)`  
`distinct(data, ...)` Keep all = FALSE Remove rows with duplicate values.  
`distinct(mtcars, Species)`  
`sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, n = parent.frame())` Randomly select fraction of rows.  
`sample_frac(mtcars, 0.5, replace = TRUE)`  
`sample_n(tbl, size, replace = FALSE, weight = NULL, n = parent.frame())` Randomly select size rows.  
`sample_n(mtcars, 10, replace = FALSE)`  
`slice(data, ...)` Select rows by position.  
`slice(mtcars, 1:10)`  
`top_n(x, n, wt)` Select and order top n entries (by group if grouped data).  
`top_n(mtcars, 5, Sepal.Length)`

**Logical and boolean operators to use with filter()**  
`<` `<=` `is.na()` `%in%` `|` `&` `xor()`  
See **These logic and comparison** for help.

**ARRANGE CASES**  
`arrange(data, ...)` Order rows by values of a column or columns (low to high), use with `desc()` to order from high to low.  
`arrange(mtcars, mpg)`  
`arrange(mtcars, desc(mpg))`

**ADD CASES**  
`add_row(data, ..., before = NULL, after = NULL)`  
Add one or more rows to a table.  
`add_row(mtcars, mpg = 1, weight = 1)`

### Manipulate Variables

**EXTRACT VARIABLES**  
Column functions return a set of columns as a new vector or table.

`pull(data, var = 1)` Extract column values as a vector. Choose by name or index.  
`pull(mtcars, Sepal.Length)`  
`select(data, ...)` Extract columns as a table. Also `select_if()`.  
`select(mtcars, Sepal.Length, Species)`  
Use these helpers with `select()`, e.g. `select(mtcars, starts_with("Sepal"))`  
`contains(match)` num. range(prefix, range) e.g. `mpg > 10`  
`ends_with(match)` one. off. ... e.g. `Species`  
`matches(match)` starts\_with(match)

**MAKE NEW VARIABLES**  
These apply vectorized functions to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

**vectorized function**  
`mutate(data, ...)` Compute new column(s).  
`mutate(mtcars, gpm = 1/mpg)`  
`transmute(data, ...)` Compute new column(s), drop others.  
`transmute(mtcars, gpm = 1/mpg)`

`mutate_at(tbl, cols, funs, ...)` Apply funs to every column. Use with `funs()`. Also `mutate_if()`.  
`mutate_at(mtcars, funs(log10, log2))`  
`mutate_if(mtcars, is.numeric, funs(log10))`  
`mutate_at(tbl, cols, funs, ...)` Apply funs to specific columns. Use with `funs()`.  
`mutate_if(mtcars, is.numeric, funs(log10))`

`add_column(data, ..., before = NULL, after = NULL)`  
Add new column(s). Also `add_count()`.  
`add_count(mtcars, new = 1:10)`  
`add_tally()` Add column(mtcars, new = 1:10)  
`rename(data, ...)` Rename columns.  
`rename(mtcars, Length = Sepal.Length)`

### Vector Functions

**TO USE WITH MUTATE()**  
`mutate()` and `transmute()` apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

**vectorized function**

**OFFSETS**  
`dplyr::lag()` - Offset elements by 1  
`dplyr::lead()` - Offset elements by -1

**CUMULATIVE AGGREGATES**  
`dplyr::cumall()` - Cumulative all  
`dplyr::cumany()` - Cumulative any  
`dplyr::cummax()` - Cumulative max  
`dplyr::cummean()` - Cumulative mean  
`dplyr::cummin()` - Cumulative min  
`dplyr::cumprod()` - Cumulative prod  
`dplyr::cumsum()` - Cumulative sum

**RANKINGS**  
`dplyr::cume_dist()` - Proportion of all values <= dplyr::rank() - rank with ties = min, no gaps  
`dplyr::min_rank()` - rank with ties = min  
`dplyr::ntile()` - bins into n bins  
`dplyr::percent_rank()` - min, rank scaled to [0,1]  
`dplyr::row_number()` - rank with ties = "first"

**MATH**  
`+`, `-`, `*`, `/`, `%/%`, `%/%` - arithmetic ops  
`log()`, `log2()`, `log10()` - logs  
`<`, `<=`, `>`, `>=`, `==` - logical comparisons  
`dplyr::between()` - x = left & x = right  
`dplyr::near()` - safe == for floating point numbers

**MISC**  
`dplyr::case_when()` - multi-case if, else  
`dplyr::coalesce()` - first non-NA values by element across a set of vectors  
`dplyr::if_else()` - element-wise if/else  
`dplyr::na_if()` - replace specific values with NA  
`dplyr::na_rm()` - element-wise NA  
`dplyr::reorder()` - vectorized switch  
`dplyr::recode_factor()` - vectorized switch for factors

`also_has_rownames()`, `remove_rownames()`

### Summary Functions

**TO USE WITH SUMMARISE()**  
`summarise()` applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

**summary function**

**COUNTS**  
`dplyr::n()` - number of values/rows  
`dplyr::n_distinct()` - # of unique  
`sum(is.na())` - # of non NAs

**LOCATION**  
`mean()` - mean, also `mean(is.na())`  
`median()` - median

**LOGICALS**  
`mean()` - Proportion of TRUE's  
`sum()` - # of TRUE's

**POSITION/ORDER**  
`dplyr::first()` - first value  
`dplyr::last()` - last value  
`dplyr::nth()` - value in nth location of vector

**RANK**  
`quantile()` - nth quantile  
`min()` - minimum value  
`max()` - maximum value

**SPREAD**  
`IQR()` - Inter-Quartile Range  
`mad()` - median absolute deviation  
`sd()` - standard deviation  
`var()` - variance

### Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

`rownames_to_column()`  
`rownames(mtcars) %>% to_column()`  
`rownames_to_column(mtcars, var = "C")`  
`column_to_rownames()`  
`rownames(mtcars) %>% from_column()`  
`column_to_rownames(mtcars, var = "C")`

### Combine Tables

**COMBINE VARIABLES**  
`x` `y` `=` `x %>% bind_cols(y)`  
`bind_cols(x, y)` Returns tables placed side by side as a single table. BE SURE THAT ROWS ALIGN.

**COMBINE CASES**  
`x` `y` `=` `x %>% bind_rows(y)`  
`bind_rows(x, y)` Returns tables one on top of the other as a single table. Set id to a column name to add a column of the original table names (as pictured).

`bind_rows(x, y, id = NULL)`  
`intersect(x, y, ...)` Rows that appear in both x and y.

`setdiff(x, y, ...)` Rows that appear in x but not y.

`union(x, y, ...)` Rows that appear in x or y. (Duplicates removed; union\_all) retains duplicates.

`left_join(x, y, by = NULL, copy = FALSE, suffixes = c("x", "y"))`  
Join matching values from y to x.

`right_join(x, y, by = NULL, copy = FALSE, suffixes = c("x", "y"))`  
Join matching values from x to y.

`inner_join(x, y, by = NULL, copy = FALSE, suffixes = c("x", "y"))`  
Join data. Retain only rows with matches.

`full_join(x, y, by = NULL, copy = FALSE, suffixes = c("x", "y"))`  
Join data. Retain all values, all rows.

`use_semi_join()` to test whether two data sets contain the exact same rows (in any order).

`filter_join(x, y, by = NULL, ...)`  
Use a "Filtering Join" to filter one table against the rows of another.

`semi_join(x, y, by = NULL, ...)`  
Return rows of x that have a match in y. USEFUL TO SEE WHAT WILL BE JOINED.

`anti_join(x, y, by = NULL, ...)`  
Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.



RStudio is a trademark of RStudio, Inc. • CC BY-SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browser/getting/package = "CtyR", "RStudio" • dplyr 0.7.0 • 1000x 3.2.0 • Updated: 2017-03



RStudio is a trademark of RStudio, Inc. • CC BY-SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browser/getting/package = "CtyR", "RStudio" • dplyr 0.7.0 • 1000x 3.2.0 • Updated: 2017-03

# Data wrangling with dplyr

## The greatest hits of `dplyr` verbs

### This lesson

- `glimpse()` as an alternative to `str()` or `head()`
- `select()` subsets **columns**
- `filter()` subsets **rows**
- `mutate()` adds new columns
- `rename()` changes the names of columns
- `arrange()` changes the order of rows

### Future lessons

- `group_by()` and `ungroup()`
- `summarize()`
- `tally()`
- `count()`

### Time-pending lessons

- `bind_rows()`
- `bind_cols()`
- `left_join()` and friends

# The great and powerful **pipe**



# Examples of piping, and formatting goals

```
## tired  
log(5)
```

```
## [1] 1.609438
```

```
## Wired  
5 %>% log()
```

```
## [1] 1.609438
```

```
## ON FIRE  
5 %>%  
  log()
```

```
## [1] 1.609438
```

```
## KILN-LEVEL FIRE  
5 %>%  
  log(10)
```

```
## [1] 0.69897
```

# Cold tolerance experiment on grass species

```
head(plant)
```

```
## # A tibble: 6 x 5
##   plant_id origin treatment  co2_conc co2_rate
##   <ord>      <fct>   <fct>      <dbl>   <dbl>
## 1 Qn1       Quebec nonchilled    95      16
## 2 Qn1       Quebec nonchilled   175     30.4
## 3 Qn1       Quebec nonchilled   250     34.8
## 4 Qn1       Quebec nonchilled   350     37.2
## 5 Qn1       Quebec nonchilled   500     35.3
## 6 Qn1       Quebec nonchilled   675     39.2
```

```
nrow(plant)
```

```
## [1] 84
```

```
ncol(plant)
```

```
## [1] 5
```

# Loading libraries

```
## LOAD THE DPLYR LIBRARY  
library(dplyr) # must load for each R session
```

```
## your first dplyr function!  
glimpse(plant)
```

```
## Rows: 84  
## Columns: 5  
## $ plant_id   <ord> Qn1, Qn1, Qn1, Qn1, Qn1, Qn1, Qn1, Qn2, Qn2, Qn2, Qn2, Qn2,...  
## $ origin     <fct> Quebec, Quebec, Quebec, Quebec, Quebec, Quebec, Quebec, Que...  
## $ treatment  <fct> nonchilled, nonchilled, nonchilled, nonchilled, nonchilled,...  
## $ co2_conc   <dbl> 95, 175, 250, 350, 500, 675, 1000, 95, 175, 250, 350, 500, ...  
## $ co2_rate   <dbl> 16.0, 30.4, 34.8, 37.2, 35.3, 39.2, 39.7, 13.6, 27.3, 37.1,...
```

# First, ALWAYS LOOK AT YOUR DATA

```
levels(plant$plant_id)
```

```
## [1] "Qn1" "Qn2" "Qn3" "Qc1" "Qc3" "Qc2" "Mn3" "Mn2" "Mn1" "Mc2" "Mc3" "Mc1"
```

```
levels(plant$origin)
```

```
## [1] "Quebec" "Mississippi"
```

```
levels(plant$treatment)
```

```
## [1] "nonchilled" "chilled"
```

# Subsetting columns with select

```
names(plant)
```

```
## [1] "plant_id" "origin" "treatment" "co2_conc" "co2_rate"
```

Select `plant_id` and `origin`

```
plant %>%  
  select(plant_id, origin)
```

```
## # A tibble: 84 x 2  
##   plant_id origin  
##   <ord>    <fct>  
## 1 Qn1      Quebec  
## 2 Qn1      Quebec  
## 3 Qn1      Quebec  
## 4 Qn1      Quebec  
## 5 Qn1      Quebec  
## 6 Qn1      Quebec  
## 7 Qn1      Quebec  
## 8 Qn2      Quebec  
## 9 Qn2      Quebec  
## 10 Qn2     Quebec
```

Select all columns *except* `treatment`

```
plant %>%  
  select(-treatment)
```

```
## # A tibble: 84 x 4  
##   plant_id origin co2_conc co2_rate  
##   <ord>    <fct>    <dbl>    <dbl>  
## 1 Qn1      Quebec      95      16  
## 2 Qn1      Quebec     175     30.4  
## 3 Qn1      Quebec     250     34.8  
## 4 Qn1      Quebec     350     37.2  
## 5 Qn1      Quebec     500     35.3  
## 6 Qn1      Quebec     675     39.2  
## 7 Qn1      Quebec    1000     39.7  
## 8 Qn2      Quebec      95     13.6  
## 9 Qn2      Quebec     175     27.3  
## 10 Qn2     Quebec     250     37.1
```



# The power of the pipe

```
plant %>%  
  select(plant_id, origin)
```

```
## # A tibble: 84 x 2  
##   plant_id origin  
##   <ord>    <fct>  
## 1 Qn1      Quebec  
## 2 Qn1      Quebec  
## 3 Qn1      Quebec  
## 4 Qn1      Quebec  
## 5 Qn1      Quebec  
## 6 Qn1      Quebec  
## 7 Qn1      Quebec  
## 8 Qn2      Quebec  
## 9 Qn2      Quebec  
## 10 Qn2     Quebec  
## # ... with 74 more rows
```

```
plant %>%  
  select(plant_id, origin) %>%  
  head()
```

```
## # A tibble: 6 x 2  
##   plant_id origin  
##   <ord>    <fct>  
## 1 Qn1      Quebec  
## 2 Qn1      Quebec  
## 3 Qn1      Quebec  
## 4 Qn1      Quebec  
## 5 Qn1      Quebec  
## 6 Qn1      Quebec
```

```
plant %>%  
  select(plant_id, origin) %>%  
  head(2)
```

```
## # A tibble: 2 x 2  
##   plant_id origin  
##   <ord>    <fct>  
## 1 Qn1      Quebec
```

# Subsetting rows with filter

```
names(plant)
```

```
## [1] "plant_id" "origin" "treatment" "co2_conc" "co2_rate"
```

## Only plants from Mississippi

```
plant %>%  
  filter(origin == "Mississippi")
```

```
## # A tibble: 42 x 5  
##   plant_id origin      treatment co2_conc co2_rate  
##   <ord>    <fct>      <fct>      <dbl>    <dbl>  
## 1 Mn1      Mississippi nonchilled      95     10.6  
## 2 Mn1      Mississippi nonchilled     175     19.2  
## 3 Mn1      Mississippi nonchilled     250     26.2  
## 4 Mn1      Mississippi nonchilled     350      30  
## 5 Mn1      Mississippi nonchilled     500     30.9  
## 6 Mn1      Mississippi nonchilled     675     32.4  
## 7 Mn1      Mississippi nonchilled    1000     35.5  
## 8 Mn2      Mississippi nonchilled      95      12  
## 9 Mn2      Mississippi nonchilled     175      22  
## 10 Mn2     Mississippi nonchilled     250     30.6  
## # ... with 32 more rows
```

# Subsetting rows with filter

```
names(plant)
```

```
## [1] "plant_id" "origin" "treatment" "co2_conc" "co2_rate"
```

Only plants with rate of co2 uptake of at least 30

```
plant %>%  
  filter(co2_rate ≥ 30)
```

```
## # A tibble: 41 x 5  
##   plant_id origin treatment co2_conc co2_rate  
##   <ord>    <fct> <fct>      <dbl>   <dbl>  
## 1 Qn1      Quebec nonchilled    175    30.4  
## 2 Qn1      Quebec nonchilled    250    34.8  
## 3 Qn1      Quebec nonchilled    350    37.2  
## 4 Qn1      Quebec nonchilled    500    35.3  
## 5 Qn1      Quebec nonchilled    675    39.2  
## 6 Qn1      Quebec nonchilled   1000    39.7  
## 7 Qn2      Quebec nonchilled    250    37.1  
## 8 Qn2      Quebec nonchilled    350    41.8  
## 9 Qn2      Quebec nonchilled    500    40.6  
## 10 Qn2     Quebec nonchilled    675    41.4  
## # ... with 31 more rows
```

# Subsetting rows with filter

```
names(plant)
```

```
## [1] "plant_id" "origin" "treatment" "co2_conc" "co2_rate"
```

Only Mississippi plants with rate of at least 30

```
plant %>%  
  # Use a , for "and" (&) in filter()  
  filter(co2_rate ≥ 30, origin = "Mississippi")
```

```
## # A tibble: 9 x 5
```

```
##   plant_id origin      treatment co2_conc co2_rate  
##   <ord>      <fct>      <fct>      <dbl>   <dbl>  
## 1 Mn1      Mississippi nonchilled    350     30  
## 2 Mn1      Mississippi nonchilled    500    30.9  
## 3 Mn1      Mississippi nonchilled    675    32.4  
## 4 Mn1      Mississippi nonchilled   1000    35.5  
## 5 Mn2      Mississippi nonchilled    250    30.6  
## 6 Mn2      Mississippi nonchilled    350    31.8  
## 7 Mn2      Mississippi nonchilled    500    32.4  
## 8 Mn2      Mississippi nonchilled    675    31.1  
## 9 Mn2      Mississippi nonchilled   1000    31.5
```

# Subsetting rows with filter

Only plants exposed to a co2\_conc of 500 or 675

```
plant %>%  
  filter(co2_conc = 500, co2_conc = 675)
```

```
## # A tibble: 0 x 5  
## # ... with 5 variables: plant_id <ord>, origin <fct>, treatment <fct>,  
## #   co2_conc <dbl>, co2_rate <dbl>
```

```
plant %>%  
  ## Need to use the OR operator |  
  filter(co2_conc = 500 | co2_conc = 675)
```

```
## # A tibble: 24 x 5  
##   plant_id origin treatment co2_conc co2_rate  
##   <ord>      <fct>  <fct>      <dbl>    <dbl>  
## 1 Qn1      Quebec nonchilled    500     35.3  
## 2 Qn1      Quebec nonchilled    675     39.2  
## 3 Qn2      Quebec nonchilled    500     40.6  
## 4 Qn2      Quebec nonchilled    675     41.4  
## 5 Qn3      Quebec nonchilled    500     42.9  
## 6 Qn3      Quebec nonchilled    675     43.9  
## 7 Qc1      Quebec chilled      500     32.5  
## 8 Qc1      Quebec chilled      675     35.4
```

# Introducing the %in% operator

```
"Stephanie" %in% c("Stephanie", "Spielman")  
## [1] TRUE  
  
10 %in% 15:20  
## [1] FALSE
```

Only plants exposed to a co2\_conc of 500 or 675

```
plant %>%  
  filter(co2_conc %in% c(675, 500)) %>%  
  head(5) # to fit on slide
```

```
## # A tibble: 5 x 5  
##   plant_id origin treatment  co2_conc co2_rate  
##   <ord>      <fct>   <fct>      <dbl>    <dbl>  
## 1 Qn1      Quebec nonchilled    500     35.3  
## 2 Qn1      Quebec nonchilled    675     39.2  
## 3 Qn2      Quebec nonchilled    500     40.6  
## 4 Qn2      Quebec nonchilled    675     41.4  
## 5 Qn3      Quebec nonchilled    500     42.9
```

# Combining dplyr verbs

Only Mississippi plants with rate of at least 30, and remove origin column

```
plant %>%  
  filter(co2_rate ≥ 30, origin = "Mississippi") %>%  
  select(-origin) %>%  
  head() ## to fit on slides!
```

```
## # A tibble: 6 x 4  
##   plant_id treatment  co2_conc co2_rate  
##   <ord>      <fct>      <dbl>   <dbl>  
## 1 Mn1      nonchilled    350     30  
## 2 Mn1      nonchilled    500    30.9  
## 3 Mn1      nonchilled    675    32.4  
## 4 Mn1      nonchilled   1000    35.5  
## 5 Mn2      nonchilled    250    30.6  
## 6 Mn2      nonchilled    350    31.8
```

```
# we can SAVE the output !!  
plant %>%  
  filter(co2_rate ≥ 30, origin = "Mississippi") %>%  
  select(-origin) → miss_plant_over30
```

# And don't forget to COMMENT!!

```
plant %>%  
  # Keep only plants with update ≥ 30 and who are from mississippi  
  filter(co2_rate ≥ 30, origin = "Mississippi") %>%  
  # Remove the origin column  
  select(-origin) → miss_plant_over30
```



# A note about namespaces

- `filter()` and `select()` are functions in the `dplyr` package (library)
- `select()` is ALSO a function in another library called `MASS`
  - Package that accompanies book "Modern Applied Statistics with S"
- `filter()` is ALSO a function in another library called `stats`
  - Package that comes with R
- **How does R know which `filter()` or `select()` you are using?**

# A note about namespaces

- All variables, functions, definitions of any kind are part of a **namespace**
- R searches the *most recently loaded* namespace first
  - If you get a weird error, you probably didn't load the library
- You can always *explicitly* use functions as `namespace::function()`
  - `dplyr::filter()`
  - `dplyr::select()`

# Creating new columns with mutate()

```
names(plant)
```

```
## [1] "plant_id" "origin" "treatment" "co2_conc" "co2_rate"
```

Currently, `co2_conc` is in units mL/L. What if we want it in dL/L?

```
plant %>%  
  mutate(co2_conc_dL = co2_conc / 10)
```

```
## # A tibble: 84 x 6
```

```
##   plant_id origin treatment  co2_conc co2_rate co2_conc_dL  
##   <ord>    <fct>  <fct>      <dbl>   <dbl>    <dbl>  
## 1 Qn1      Quebec nonchilled    95     16      9.5  
## 2 Qn1      Quebec nonchilled   175    30.4    17.5  
## 3 Qn1      Quebec nonchilled   250    34.8     25  
## 4 Qn1      Quebec nonchilled   350    37.2     35  
## 5 Qn1      Quebec nonchilled   500    35.3     50  
## 6 Qn1      Quebec nonchilled   675    39.2    67.5  
## 7 Qn1      Quebec nonchilled  1000    39.7    100  
## 8 Qn2      Quebec nonchilled    95     13.6     9.5  
## 9 Qn2      Quebec nonchilled   175    27.3    17.5  
## 10 Qn2     Quebec nonchilled   250    37.1     25  
## # ... with 74 more rows
```

```
plant %>%  
  ## This will OVERWRITE the existing column - be careful!!  
  mutate(co2_conc = co2_conc / 10) %>%  
  head(2)
```

```
## # A tibble: 2 x 5  
##   plant_id origin treatment  co2_conc co2_rate  
##   <ord>     <fct>  <fct>      <dbl>   <dbl>  
## 1 Qn1      Quebec nonchilled    9.5     16  
## 2 Qn1      Quebec nonchilled   17.5    30.4
```

# Renaming columns with rename()

- Syntax: `rename(NEW = OLD)`. Very easy to mess up! That's ok!

```
plant %>%  
  rename(concentration = co2_conc) %>%  
  head(2)
```

```
## # A tibble: 2 x 5  
##   plant_id origin treatment concentration co2_rate  
##   <ord>      <fct>  <fct>          <dbl>      <dbl>  
## 1 Qn1      Quebec nonchilled          95         16  
## 2 Qn1      Quebec nonchilled         175        30.4
```

```
plant %>%  
  rename(co2_conc = concentration)
```

```
## Error: Can't rename columns that don't exist.  
## [31mx [39m Column `concentration` doesn't exist.
```

# Arranging rows with arrange()

```
small_plant <- plant %>% select(co2_rate, co2_conc, origin)
```

Arrange in order of `co2_rate`

```
small_plant %>%  
  arrange(co2_rate)
```

```
## # A tibble: 84 x 3  
##   co2_rate co2_conc origin  
##   <dbl>    <dbl> <fct>  
## 1      7.7      95 Mississippi  
## 2      9.3      95 Quebec  
## 3     10.5      95 Mississippi  
## 4     10.6      95 Mississippi  
## 5     10.6      95 Mississippi  
## 6     11.3      95 Mississippi  
## 7     11.4     175 Mississippi  
## 8      12      95 Mississippi  
## 9     12.3     250 Mississippi  
## 10     12.5     500 Mississippi  
## # ... with 74 more rows
```

Arrange in **descending** order of `co2_rate`

```
small_plant %>%  
  arrange(desc(co2_rate))
```

```
## # A tibble: 84 x 3  
##   co2_rate co2_conc origin  
##   <dbl>    <dbl> <fct>  
## 1     45.5    1000 Quebec  
## 2     44.3    1000 Quebec  
## 3     43.9     675 Quebec  
## 4     42.9     500 Quebec  
## 5     42.4    1000 Quebec  
## 6     42.1     350 Quebec  
## 7     41.8     350 Quebec  
## 8     41.4     675 Quebec  
## 9     41.4    1000 Quebec  
## 10     40.6     500 Quebec  
## # ... with 74 more rows
```

# A mini wrangle!

```
head(plant)
```

```
## # A tibble: 6 x 5
##   plant_id origin treatment  co2_conc co2_rate
##   <ord>      <fct>  <fct>      <dbl>   <dbl>
## 1 Qn1       Quebec nonchilled    95      16
## 2 Qn1       Quebec nonchilled   175     30.4
## 3 Qn1       Quebec nonchilled   250     34.8
## 4 Qn1       Quebec nonchilled   350     37.2
## 5 Qn1       Quebec nonchilled   500     35.3
## 6 Qn1       Quebec nonchilled   675     39.2
```

Find the ratio of concentration/rate for all the plants from Quebec exposed to a "chilled" environment

- First, make a plan - how would you do this **by hand**?
- Second, WRITE YOUR PLAN DOWN
- Third, go *line by line* to build up your plan

# Conc/rate ratio for chilled Quebec plants

```
plant
```

```
## # A tibble: 84 x 5
##   plant_id origin treatment  co2_conc co2_rate
##   <ord>      <fct>  <fct>      <dbl>    <dbl>
## 1 Qn1       Quebec nonchilled      95      16
## 2 Qn1       Quebec nonchilled     175     30.4
## 3 Qn1       Quebec nonchilled     250     34.8
## 4 Qn1       Quebec nonchilled     350     37.2
## 5 Qn1       Quebec nonchilled     500     35.3
## 6 Qn1       Quebec nonchilled     675     39.2
## 7 Qn1       Quebec nonchilled    1000     39.7
## 8 Qn2       Quebec nonchilled      95     13.6
## 9 Qn2       Quebec nonchilled     175     27.3
## 10 Qn2      Quebec nonchilled     250     37.1
## # ... with 74 more rows
```



# Conc/rate ratio for chilled Quebec plants

```
plant %>%  
  filter(origin == "Quebec")
```

```
## # A tibble: 42 x 5  
##   plant_id origin treatment  co2_conc co2_rate  
##   <ord>      <fct>  <fct>      <dbl>    <dbl>  
## 1 Qn1      Quebec nonchilled      95      16  
## 2 Qn1      Quebec nonchilled     175     30.4  
## 3 Qn1      Quebec nonchilled     250     34.8  
## 4 Qn1      Quebec nonchilled     350     37.2  
## 5 Qn1      Quebec nonchilled     500     35.3  
## 6 Qn1      Quebec nonchilled     675     39.2  
## 7 Qn1      Quebec nonchilled    1000     39.7  
## 8 Qn2      Quebec nonchilled      95     13.6  
## 9 Qn2      Quebec nonchilled     175     27.3  
## 10 Qn2     Quebec nonchilled     250     37.1  
## # ... with 32 more rows
```

# Conc/rate ratio for chilled Quebec plants

```
plant %>%  
  filter(origin = "Quebec") %>%  
  filter(treatment = "chilled")
```

```
## # A tibble: 21 x 5  
##   plant_id origin treatment co2_conc co2_rate  
##   <ord>      <fct>  <fct>      <dbl>    <dbl>  
## 1 Qc1       Quebec chilled        95     14.2  
## 2 Qc1       Quebec chilled       175     24.1  
## 3 Qc1       Quebec chilled       250     30.3  
## 4 Qc1       Quebec chilled       350     34.6  
## 5 Qc1       Quebec chilled       500     32.5  
## 6 Qc1       Quebec chilled       675     35.4  
## 7 Qc1       Quebec chilled      1000     38.7  
## 8 Qc2       Quebec chilled        95        9.3  
## 9 Qc2       Quebec chilled       175     27.3  
## 10 Qc2      Quebec chilled       250        35  
## # ... with 11 more rows
```

# Conc/rate ratio for chilled Quebec plants

```
plant %>%  
  filter(origin = "Quebec") %>%  
  filter(treatment = "chilled") %>%  
  mutate(ratio = co2_conc / co2_rate)
```

```
## # A tibble: 21 x 6
```

```
##   plant_id origin treatment co2_conc co2_rate ratio  
##   <ord>    <fct>  <fct>      <dbl>    <dbl> <dbl>  
## 1 Qc1      Quebec chilled      95     14.2  6.69  
## 2 Qc1      Quebec chilled     175     24.1  7.26  
## 3 Qc1      Quebec chilled     250     30.3  8.25  
## 4 Qc1      Quebec chilled     350     34.6 10.1  
## 5 Qc1      Quebec chilled     500     32.5 15.4  
## 6 Qc1      Quebec chilled     675     35.4 19.1  
## 7 Qc1      Quebec chilled    1000     38.7 25.8  
## 8 Qc2      Quebec chilled      95       9.3 10.2  
## 9 Qc2      Quebec chilled     175     27.3  6.41  
## 10 Qc2     Quebec chilled     250     35     7.14  
## # ... with 11 more rows
```

# Why you have to go step-by-step

```
plant %>%  
  filter(Origin = "Quebec") %>%  
  filter(treatment = "chill") %>%  
  mutate(ratio = conc_co2 / rate)
```

```
## Error: Problem with `filter()` input `..1`.  
## x object 'Origin' not found  
## i Input `..1` is `Origin = "Quebec"`.
```

```
plant %>%  
  filter(origin = "Quebec") %>%  
  filter(treatment = "chill") %>%  
  mutate(ratio = conc_co2 / co2_rate)
```

```
## Error: Problem with `mutate()` input `ratio`.  
## x object 'conc_co2' not found  
## i Input `ratio` is `conc_co2/co2_rate`.
```