dplyr, tidyr, and ggplot2

Intro to the *tidyverse*

Samuel Robinson, Ph.D.

Sep. 15, 2023

Part 1: `dplyr` and `tidyr`

Normal data manipulation in R

```r
#Changes species to factor
plants$Species <- as.factor(plants$Species)
#Changes plant code to factor
plants$Plant.Code <- as.factor(plants$Plant.Code)
#Changes Seed to factor
seeds$Seed <- as.factor(seeds$Seed)
#Changes plant code to factor
seeds$Plant.Code <- as.factor(seeds$Plant.Code)
#Selects Flower, Code, Total.Germ columns
germ <- germ[,c('Flower','Code','Total.Germ')]

#Sets numerics
plants[,c(3:9)] <- as.numeric(unlist(plants[,c(3:9)]))
#Sets Dates
seeds$Collection.Date <- as.Date(seeds$Collection.Date,origin='2012-01-01')
```

Normal data manipulation in R

```
#Changes species to factor
plants$Species <- as.factor(plants$Species)
#Changes plant code to factor
plants$Plant.Code <- as.factor(plants$Plant.Code)
#Changes Seed to factor
seeds$Seed <- as.factor(seeds$Seed)
#Changes plant code to factor
seeds$Plant.Code <- as.factor(seeds$Plant.Code)
#Selects Flower, Code, Total.Germ columns
germ <- germ[,c('Flower','Code','Total.Germ')]

#Sets numerics
plants[,c(3:9)] <- as.numeric(unlist(plants[,c(3:9)]))
#Sets Dates
seeds$Collection.Date <- as.Date(seeds$Collection.Date,origin='2012-01-01')
```

- One line of code per column - lots of typing

Normal data manipulation in R

```
#Changes species to factor
plants$Species <- as.factor(plants$Species)
#Changes plant code to factor
plants$Plant.Code <- as.factor(plants$Plant.Code)
#Changes Seed to factor
seeds$Seed <- as.factor(seeds$Seed)
#Changes plant code to factor
seeds$Plant.Code <- as.factor(seeds$Plant.Code)
#Selects Flower, Code, Total.Germ columns
germ <- germ[,c('Flower','Code','Total.Germ')]

#Sets numerics
plants[,c(3:9)] <- as.numeric(unlist(plants[,c(3:9)]))
#Sets Dates
seeds$Collection.Date <- as.Date(seeds$Collection.Date,origin='2012-01-01')
```

- One line of code per column - lots of typing
- Lots of $$$s

Normal data manipulation in R

```
#Changes species to factor
plants$Species <- as.factor(plants$Species)
#Changes plant code to factor
plants$Plant.Code <- as.factor(plants$Plant.Code)
#Changes Seed to factor
seeds$Seed <- as.factor(seeds$Seed)
#Changes plant code to factor
seeds$Plant.Code <- as.factor(seeds$Plant.Code)
#Selects Flower, Code, Total.Germ columns
germ <- germ[,c('Flower','Code','Total.Germ')]

#Sets numerics
plants[,c(3:9)] <- as.numeric(unlist(plants[,c(3:9)]))
#Sets Dates
seeds$Collection.Date <- as.Date(seeds$Collection.Date,origin='2012-01-01')
```

- One line of code per column - lots of typing
- Lots of $$$s
- Lots of room for errors

Data manipulation using dplyr/tidyr

```r
library(tidyverse)
#Convert factors in plants df
plants <- plants %>% mutate(across(c(Species,Plant.Code)),factor)
#Convert factors in seeds df
seeds <- seeds %>% mutate(across(c(Seed,Plant.Code)),factor)
#Select Flower, Code, and Total.Germ columns in germ df
germ <- germ %>% select(Flower,Code,Total.Germ)

#Change columns 3:9 to numeric
plants <- plants %>% mutate(across(c(3:9)),as.numeric)
#Convert Collection.Date to Date format
seeds <- seeds %>%
  mutate(Collection.Date=as.Date(Collection.Date,origin='2012-01-01'))
```

Data manipulation using dplyr/tidyr

```
library(tidyverse)
#Convert factors in plants df
plants <- plants %>% mutate(across(c(Species,Plant.Code)),factor)
#Convert factors in seeds df
seeds <- seeds %>% mutate(across(c(Seed,Plant.Code)),factor)
#Select Flower, Code, and Total.Germ columns in germ df
germ <- germ %>% select(Flower,Code,Total.Germ)

#Change columns 3:9 to numeric
plants <- plants %>% mutate(across(c(3:9)),as.numeric)
#Convert Collection.Date to Date format
seeds <- seeds %>%
  mutate(Collection.Date=as.Date(Collection.Date,origin='2012-01-01'))
```

- More compact, less typing

Data manipulation using dplyr/tidyr

```
library(tidyverse)
#Convert factors in plants df
plants <- plants %>% mutate(across(c(Species,Plant.Code)),factor)
#Convert factors in seeds df
seeds <- seeds %>% mutate(across(c(Seed,Plant.Code)),factor)
#Select Flower, Code, and Total.Germ columns in germ df
germ <- germ %>% select(Flower,Code,Total.Germ)

#Change columns 3:9 to numeric
plants <- plants %>% mutate(across(c(3:9)),as.numeric)
#Convert Collection.Date to Date format
seeds <- seeds %>%
  mutate(Collection.Date=as.Date(Collection.Date,origin='2012-01-01'))
```

- More compact, less typing
- Easier to read

Data manipulation using dplyr/tidyr

```
library(tidyverse)
#Convert factors in plants df
plants <- plants %>% mutate(across(c(Species,Plant.Code)),factor)
#Convert factors in seeds df
seeds <- seeds %>% mutate(across(c(Seed,Plant.Code)),factor)
#Select Flower, Code, and Total.Germ columns in germ df
germ <- germ %>% select(Flower,Code,Total.Germ)

#Change columns 3:9 to numeric
plants <- plants %>% mutate(across(c(3:9)),as.numeric)
#Convert Collection.Date to Date format
seeds <- seeds %>%
  mutate(Collection.Date=as.Date(Collection.Date,origin='2012-01-01'))
```

- More compact, less typing
- Easier to read
- Faster (matters for large datasets)

Why should I do data manipulation in R?

- Much quicker (once you learn how!)

Why should I do data manipulation in R?

- Much quicker (once you learn how!)
- Can do complex re-arranging and make summary tables very easily

Why should I do data manipulation in R?

- Much quicker (once you learn how!)
- Can do complex re-arranging and make summary tables very easily
- For very large datasets, Excel starts falling apart fairly quickly

Why should I do data manipulation in R?

- Much quicker (once you learn how!)
- Can do complex re-arranging and make summary tables very easily
- For very large datasets, Excel starts falling apart fairly quickly
- *You have a record of exactly what you've done*

Why should I do data manipulation in R?

- Much quicker (once you learn how!)
- Can do complex re-arranging and make summary tables very easily
- For very large datasets, Excel starts falling apart fairly quickly
- *You have a record of exactly what you've done*

Why should I do data manipulation in R?

- Much quicker (once you learn how!)
- Can do complex re-arranging and make summary tables very easily
- For very large datasets, Excel starts falling apart fairly quickly
- *You have a record of exactly what you've done*

Start with small, simple tasks, and work your way up to larger, complicated ones

Things to learn today:

- Basic syntax and table verbs

Things to learn today:

- Basic syntax and table verbs
- Piping

Things to learn today:

- Basic syntax and table verbs
- Piping
- Reshaping

Things to learn today:

- Basic syntax and table verbs
- Piping
- Reshaping
- Grouping

Things to learn today:

- Basic syntax and table verbs
- Piping
- Reshaping
- Grouping
- Exercise!

Basic Syntax

Both `dplyr` and `tidyr` work with `data frames` or `tibbles`

- `data frame`: similar to matrix, but with different data types for each column

Basic Syntax

Both `dplyr` and `tidyr` work with `data frames` or `tibbles`

- `data frame`: similar to matrix, but with different data types for each column
- `tibble`: "compact" data frame, with some annoying features removed

```
head(iris) #Regular data frame
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Basic Syntax

```
as_tibble(iris) #This is usually done automatically
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # i 140 more rows
```

Basic verbs - subsetting

- **select**: returns only columns that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Basic verbs - subsetting

- **select**: returns only columns that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

- Select `Petal.Length`,`Petal.Width`, and `Species` columns

```
irisTemp <- select(iris,Petal.Length,Petal.Width,Species)
```

```
##   Petal.Length Petal.Width Species
## 1          1.4         0.2  setosa
## 2          1.4         0.2  setosa
## 3          1.3         0.2  setosa
## 4          1.5         0.2  setosa
## 5          1.4         0.2  setosa
## 6          1.7         0.4  setosa
```

Basic verbs - subsetting

- Helper functions for **select**: *colon* operator

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

Basic verbs - subsetting

- Helper functions for **select**: *colon* operator

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- Get all columns between `Petal.Length` and `Species`

```
irisTemp <- select(iris,Petal.Length:Species)
```

```
##   Petal.Length Petal.Width Species
## 1          1.4         0.2  setosa
## 2          1.4         0.2  setosa
## 3          1.3         0.2  setosa
```

Basic verbs - subsetting

- Helper functions for **select**: −, and *contains*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

Basic verbs - subsetting

- Helper functions for **select**: -, and *contains*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- -: selects all columns EXCEPT the one(s) specified

```
irisTemp <- select(iris,-Species)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
```

Basic verbs - subsetting

- Helper functions for **select**: -, and *contains*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- -: selects all columns EXCEPT the one(s) specified

```
irisTemp <- select(iris,-Species)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
```

- Some common selection helpers:

Basic verbs - subsetting

- Helper functions for **select**: -, and *contains*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- -: selects all columns EXCEPT the one(s) specified

```
irisTemp <- select(iris,-Species)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
```

- Some common selection helpers:
  - `contains()` contains a string

Basic verbs - subsetting

- Helper functions for **select**: -, and *contains*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- -: selects all columns EXCEPT the one(s) specified

```
irisTemp <- select(iris,-Species)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
```

- Some common selection helpers:
  - `contains()` contains a string
  - `starts_with()` column name starts with a string

Basic verbs - subsetting

- **filter**: returns only rows that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

Basic verbs - subsetting

- **filter**: returns only rows that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

- Chooses rows where `Species` is *versicolor*

```
irisTemp <- filter(iris,Sepal.Length<5,Species=='versicolor')
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          4.9         2.4          3.3           1 versicolor
```

Basic verbs - subsetting

- **filter**: returns only rows that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

- Chooses rows where `Species` is *versicolor*

```
irisTemp <- filter(iris,Sepal.Length<5,Species=='versicolor')
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          4.9         2.4          3.3           1 versicolor
```

- Some common logical operators:

Basic verbs - subsetting

- **filter**: returns only rows that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

- Chooses rows where `Species` is *versicolor*

```
irisTemp <- filter(iris,Sepal.Length<5,Species=='versicolor')
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          4.9         2.4          3.3           1 versicolor
```

- Some common logical operators:
  - == equal to, != not equal to

Basic verbs - subsetting

- **filter**: returns only rows that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

- Chooses rows where `Species` is *versicolor*

```
irisTemp <- filter(iris,Sepal.Length<5,Species=='versicolor')
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          4.9         2.4          3.3           1 versicolor
```

- Some common logical operators:
  - == equal to, != not equal to
  - $<$ greater than, $>$ less than

Basic verbs - subsetting

- **filter**: returns only rows that you want

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

- Chooses rows where `Species` is *versicolor*

```
irisTemp <- filter(iris,Sepal.Length<5,Species=='versicolor')
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          4.9         2.4          3.3           1 versicolor
```

- Some common logical operators:
  - == equal to, != not equal to
  - $<$ greater than, $>$ less than
  - & AND, | OR

Basic verbs - make new variables

- **mutate**: add new columns or alter existing ones

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species P.Width2
## 1          5.1         3.5          1.4         0.2  setosa     0.04
## 2          4.9         3.0          1.4         0.2  setosa     0.04
## 3          4.7         3.2          1.3         0.2  setosa     0.04
```

Basic verbs - make new variables

- **mutate**: add new columns or alter existing ones

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species P.Width2
## 1          5.1         3.5          1.4         0.2  setosa     0.04
## 2          4.9         3.0          1.4         0.2  setosa     0.04
## 3          4.7         3.2          1.3         0.2  setosa     0.04
```

- Squares the `Petal.Width` column

```
irisTemp <- mutate(iris,Petal.Width=Petal.Width^2)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4        0.04  setosa
## 2          4.9         3.0          1.4        0.04  setosa
## 3          4.7         3.2          1.3        0.04  setosa
```

Basic verbs - make new variables

- **across**: uses the function on a number of columns. Must be used *inside* verbs like *mutate*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

Basic verbs - make new variables

- **across**: uses the function on a number of columns. Must be used *inside* verbs like *mutate*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- Squares the `Sepal.Length` and `Petal.Width` columns

```
irisTemp <- mutate(iris, across(c(Sepal.Length,Petal.Width), ~.^2))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1        26.01         3.5          1.4        0.04  setosa
## 2        24.01         3.0          1.4        0.04  setosa
## 3        22.09         3.2          1.3        0.04  setosa
```

Basic verbs - make new variables

- **across**: uses the function on a number of columns. Must be used *inside* verbs like *mutate*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- Squares the `Sepal.Length` and `Petal.Width` columns

```
irisTemp <- mutate(iris, across(c(Sepal.Length,Petal.Width), ~.^2))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1        26.01         3.5          1.4        0.04  setosa
## 2        24.01         3.0          1.4        0.04  setosa
## 3        22.09         3.2          1.3        0.04  setosa
```

  - ~ is called a lambda (similar to a function)

Basic verbs - make new variables

- **across**: uses the function on a number of columns. Must be used *inside* verbs like *mutate*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- Squares the `Sepal.Length` and `Petal.Width` columns

```
irisTemp <- mutate(iris, across(c(Sepal.Length,Petal.Width), ~.^2))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1        26.01         3.5          1.4        0.04  setosa
## 2        24.01         3.0          1.4        0.04  setosa
## 3        22.09         3.2          1.3        0.04  setosa
```

  - ~ is called a lambda (similar to a function)
  - . means "input data from column X", so...

Basic verbs - make new variables

- **across**: uses the function on a number of columns. Must be used *inside* verbs like *mutate*

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- Squares the `Sepal.Length` and `Petal.Width` columns

```
irisTemp <- mutate(iris, across(c(Sepal.Length,Petal.Width), ~.^2))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1        26.01         3.5          1.4        0.04  setosa
## 2        24.01         3.0          1.4        0.04  setosa
## 3        22.09         3.2          1.3        0.04  setosa
```

  - ~ is called a lambda (similar to a function)
  - . means "input data from column X", so...
  - ~.^2 means "square anything in these named column

Basic verbs - make new variables

- **rename** & **transmute**

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

Basic verbs - make new variables

- **rename** & **transmute**

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- Renames `Petal.Length` to PLength

```
irisTemp <- rename(iris, PWidth=Petal.Width, PLength=Petal.Length)
```

```
##   Sepal.Length Sepal.Width PLength PWidth Species
## 1          5.1         3.5     1.4    0.2  setosa
## 2          4.9         3.0     1.4    0.2  setosa
## 3          4.7         3.2     1.3    0.2  setosa
```

Basic verbs - make new variables

- **rename** & **transmute**

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

- Renames `Petal.Length` to PLength

```
irisTemp <- rename(iris, PWidth=Petal.Width, PLength=Petal.Length)
```

```
##   Sepal.Length Sepal.Width PLength PWidth Species
## 1          5.1         3.5     1.4    0.2  setosa
## 2          4.9         3.0     1.4    0.2  setosa
## 3          4.7         3.2     1.3    0.2  setosa
```

- Same as *mutate* but drops other columns

```
irisTemp2 <- transmute(iris, P.Width2=(Petal.Width^2))
```

```
##   P.Width2
## 1     0.04
## 2     0.04
## 3     0.04
```

Mini-summary

- **select**: subset columns

Mini-summary

- **select**: subset columns
- **filter**: subset rows

Mini-summary

- **select**: subset columns
- **filter**: subset rows
- **mutate**: add new columns, or alter existing

Mini-summary

- **select**: subset columns
- **filter**: subset rows
- **mutate**: add new columns, or alter existing
- **transmute**: same as above, but drops other columns

Mini-summary

- **select**: subset columns
- **filter**: subset rows
- **mutate**: add new columns, or alter existing
- **transmute**: same as above, but drops other columns
- **rename**: changes column names

Mini-summary

- **select**: subset columns
- **filter**: subset rows
- **mutate**: add new columns, or alter existing
- **transmute**: same as above, but drops other columns
- **rename**: changes column names
- **across**: applies function across multiple columns

Mini-summary

- **select**: subset columns
- **filter**: subset rows
- **mutate**: add new columns, or alter existing
- **transmute**: same as above, but drops other columns
- **rename**: changes column names
- **across**: applies function across multiple columns
  - Used *inside of* table verbs

First challenge

Using the `iris` dataset (type `data(iris)`):

- Filter only rows with "virginica"

```
##      Species P.Area S.Area
## 1  virginica  15.00  20.79
## 2  virginica   9.69  15.66
## 3  virginica  12.39  21.30
## 4  virginica  10.08  18.27
## 5  virginica  12.76  19.50
## 6  virginica  13.86  22.80
## 7  virginica   7.65  12.25
## 8  virginica  11.34  21.17
## 9  virginica  10.44  16.75
## 10 virginica  15.25  25.92
```

First challenge

Using the `iris` dataset (type `data(iris)`):

- Filter only rows with "virginica"
- Make 2 new "area" columns, which are length $\times$ width of Petals and Sepals

```
##      Species P.Area S.Area
## 1  virginica  15.00  20.79
## 2  virginica   9.69  15.66
## 3  virginica  12.39  21.30
## 4  virginica  10.08  18.27
## 5  virginica  12.76  19.50
## 6  virginica  13.86  22.80
## 7  virginica   7.65  12.25
## 8  virginica  11.34  21.17
## 9  virginica  10.44  16.75
## 10 virginica  15.25  25.92
```

First challenge

Using the `iris` dataset (type `data(iris)`):

- Filter only rows with "virginica"
- Make 2 new "area" columns, which are length $\times$ width of Petals and Sepals
- Get rid of all columns except "Species" + 2 new columns

```
##      Species P.Area S.Area
## 1  virginica  15.00  20.79
## 2  virginica   9.69  15.66
## 3  virginica  12.39  21.30
## 4  virginica  10.08  18.27
## 5  virginica  12.76  19.50
## 6  virginica  13.86  22.80
## 7  virginica   7.65  12.25
## 8  virginica  11.34  21.17
## 9  virginica  10.44  16.75
## 10 virginica  15.25  25.92
```

Piping - %>%

This is where the tidyverse becomes *very* useful

- Takes data from one verb and passes it to the next one.

Piping - %>%

This is where the tidyverse becomes *very* useful

- Takes data from one verb and passes it to the next one.
- Lets you string together complex operations:

```
irisTemp <- select(iris,Sepal.Length,Species) %>% #Selects Sepal.Length & Species
  filter(Sepal.Length>5,Species=='versicolor') %>% #Filters using dataframe from above
  mutate(SLength2=Sepal.Length^2) #Mutates using dataframe from above
```

```
##   Sepal.Length    Species SLength2
## 1          7.0 versicolor    49.00
## 2          6.4 versicolor    40.96
## 3          6.9 versicolor    47.61
## 4          5.5 versicolor    30.25
## 5          6.5 versicolor    42.25
## 6          5.7 versicolor    32.49
```

Reshaping - i.e. “data gymnastics”

- This is very tedious to do in base R and Excel

Reshaping - i.e. “data gymnastics”

- This is very tedious to do in base R and Excel
- Reshaping operations in `tidyr` make this much easier

Reshaping - i.e. “data gymnastics”

- This is very tedious to do in base R and Excel
- Reshaping operations in `tidyr` make this much easier
- Main commands:

Reshaping - i.e. “data gymnastics”

- This is very tedious to do in base R and Excel
- Reshaping operations in `tidyr` make this much easier
- Main commands:

1. `pivot_longer` - gather columns into rows (‘long format’)

Reshaping - i.e. “data gymnastics”

- This is very tedious to do in base R and Excel
- Reshaping operations in `tidyr` make this much easier
- Main commands:

1. `pivot_longer` - gather columns into rows (‘long format’)
2. `pivot_wider` - spread rows into columns (‘wide format’)

Reshaping - *pivot_longer*: columns to rows

```
##   bat weight height wings
## 1   a      1    2.5     2
## 2   b      2    4.0     2
## 3   c      3    5.5     2
```

Some data in **wide** format: data for each "unit" listed in multiple columns

```
##   bat weight height wings
## 1   a      1    2.5     2
## 2   b      2    4.0     2
## 3   c      3    5.5     2
```

The same data in **long** format: data listed in single column, plus and ID column

```
## # A tibble: 9 x 3
##   bat   name   value
##   <chr> <chr>  <dbl>
## 1 a     weight   1
## 2 a     height   2.5
## 3 a     wings    2
## 4 b     weight   2
## 5 b     height   4
## 6 b     wings    2
## 7 c     weight   3
## 8 c     height   5.5
## 9 c     wings    2
```

Reshaping - *pivot_longer*: columns to rows

- Change wide dataframe to long dataframe

```
(longBats <- bats %>% pivot_longer(cols=weight:wings, #Columns to be made into 2
                          names_to='trait', #Name of "naming" column
                          values_to='meas')) #Name of "value" column
```

```
## # A tibble: 9 x 3
##   bat   trait   meas
##   <chr> <chr>  <dbl>
## 1 a     weight   1
## 2 a     height   2.5
## 3 a     wings    2
## 4 b     weight   2
## 5 b     height   4
## 6 b     wings    2
## 7 c     weight   3
## 8 c     height   5.5
## 9 c     wings    2
```

Reshaping - *pivot_wider*: rows to columns

- This is the inverse of *pivot_longer*

```
longBats %>% pivot_wider(names_from=trait, #Names of new columns
                         values_from=meas) #Values to go into new columns
```

```
## # A tibble: 3 x 4
##   bat   weight height wings
##   <chr>  <dbl>  <dbl> <dbl>
## 1 a          1    2.5     2
## 2 b          2    4       2
## 3 c          3    5.5     2
```

```
#Note: this must have unique row identifiers
```

Exercises!

Using the *CO2* dataset:

- Select only *non-chilled* plants from *Quebec*

```
## # A tibble: 7 x 4
##    conc   Qn1   Qn2   Qn3
##   <dbl> <dbl> <dbl> <dbl>
## 1    95  16    13.6  16.2
## 2   175  30.4  27.3  32.4
## 3   250  34.8  37.1  40.3
## 4   350  37.2  41.8  42.1
## 5   500  35.3  40.6  42.9
## 6   675  39.2  41.4  43.9
## 7  1000  39.7  44.3  45.5
```

Exercises!

Using the *CO2* dataset:

- Select only *non-chilled* plants from *Quebec*
- Pipe data frame to next command

```
## # A tibble: 7 x 4
##    conc   Qn1   Qn2   Qn3
##   <dbl> <dbl> <dbl> <dbl>
## 1    95  16    13.6  16.2
## 2   175  30.4  27.3  32.4
## 3   250  34.8  37.1  40.3
## 4   350  37.2  41.8  42.1
## 5   500  35.3  40.6  42.9
## 6   675  39.2  41.4  43.9
## 7  1000  39.7  44.3  45.5
```

Exercises!

Using the *CO2* dataset:

- Select only *non-chilled* plants from *Quebec*
- Pipe data frame to next command
- Change the uptake dataset from long to wide format (each plant should have its own column), with a column at the beginning showing concentration

```
## # A tibble: 7 x 4
##    conc   Qn1   Qn2   Qn3
##   <dbl> <dbl> <dbl> <dbl>
## 1    95  16    13.6  16.2
## 2   175  30.4  27.3  32.4
## 3   250  34.8  37.1  40.3
## 4   350  37.2  41.8  42.1
## 5   500  35.3  40.6  42.9
## 6   675  39.2  41.4  43.9
## 7  1000  39.7  44.3  45.5
```

Exercises!

Using the *CO2* dataset:

- Select only *non-chilled* plants from *Quebec*
- Pipe data frame to next command
- Change the uptake dataset from long to wide format (each plant should have its own column), with a column at the beginning showing concentration
- Hint: *filter* rows and *select* columns you need, then *pivot_wide* to wide format

```
## # A tibble: 7 x 4
##    conc   Qn1   Qn2   Qn3
##   <dbl> <dbl> <dbl> <dbl>
## 1    95  16    13.6  16.2
## 2   175  30.4  27.3  32.4
## 3   250  34.8  37.1  40.3
## 4   350  37.2  41.8  42.1
## 5   500  35.3  40.6  42.9
## 6   675  39.2  41.4  43.9
## 7  1000  39.7  44.3  45.5
```

Grouping

- Often, we want to perform operations only on *groups* within data frames

```
with(iris,tapply(Petal.Width,Species,mean)) #Using tapply
```

```
##     setosa versicolor  virginica
##      0.246      1.326      2.026
```

```
aggregate(Petal.Width~Species,data=iris,mean) #Using aggregate
```

```
##      Species Petal.Width
## 1     setosa       0.246
## 2 versicolor       1.326
## 3  virginica       2.026
```

Grouping

- Often, we want to perform operations only on *groups* within data frames
- For example, what is the average of each species' *Petal.width*?

```
with(iris,tapply(Petal.Width,Species,mean)) #Using tapply
```

```
##     setosa versicolor  virginica
##      0.246      1.326      2.026
```

```
aggregate(Petal.Width~Species,data=iris,mean) #Using aggregate
```

```
##      Species Petal.Width
## 1     setosa       0.246
## 2 versicolor       1.326
## 3  virginica       2.026
```

Grouping

- Often, we want to perform operations only on *groups* within data frames
- For example, what is the average of each species' *Petal.width*?
- This can be done in base R:

```
with(iris,tapply(Petal.Width,Species,mean)) #Using tapply
```

```
##     setosa versicolor  virginica
##      0.246      1.326      2.026
```

```
aggregate(Petal.Width~Species,data=iris,mean) #Using aggregate
```

```
##      Species Petal.Width
## 1     setosa       0.246
## 2 versicolor       1.326
## 3  virginica       2.026
```

Grouping

- How can this be done in `dplyr` and `tidyr`?

```
iris %>% group_by(Species) %>% #Group by species
  summarize(meanPWidth=mean(Petal.Width), #Mean of Petal.Width
            sdPWidth=sd(Petal.Width)) #SD of Petal.Width
```

```
## # A tibble: 3 x 3
##   Species    meanPWidth sdPWidth
##   <fct>           <dbl>    <dbl>
## 1 setosa          0.246    0.105
## 2 versicolor      1.33     0.198
## 3 virginica       2.03     0.275
```

Grouping

- How can this be done in `dplyr` and `tidyr`?

```
iris %>% group_by(Species) %>% #Group by species
  summarize(meanPWidth=mean(Petal.Width), #Mean of Petal.Width
            sdPWidth=sd(Petal.Width)) #SD of Petal.Width
```

```
## # A tibble: 3 x 3
##   Species    meanPWidth sdPWidth
##   <fct>           <dbl>    <dbl>
## 1 setosa          0.246    0.105
## 2 versicolor      1.33     0.198
## 3 virginica       2.03     0.275
```

- Apply *grouping*, then use `summarize` function

Grouping

- How can this be done in `dplyr` and `tidyr`?

```
iris %>% group_by(Species) %>% #Group by species
  summarize(meanPWidth=mean(Petal.Width), #Mean of Petal.Width
            sdPWidth=sd(Petal.Width)) #SD of Petal.Width
```

```
## # A tibble: 3 x 3
##   Species    meanPWidth sdPWidth
##   <fct>           <dbl>    <dbl>
## 1 setosa          0.246    0.105
## 2 versicolor      1.33     0.198
## 3 virginica       2.03     0.275
```

- Apply *grouping*, then use `summarize` function
  - Breaks dataframe into "mini-dataframes" before applying the function

Grouping

- How can this be done in `dplyr` and `tidyr`?

```
iris %>% group_by(Species) %>% #Group by species
  summarize(meanPWidth=mean(Petal.Width), #Mean of Petal.Width
            sdPWidth=sd(Petal.Width)) #SD of Petal.Width
```

```
## # A tibble: 3 x 3
##   Species    meanPWidth sdPWidth
##   <fct>           <dbl>    <dbl>
## 1 setosa          0.246    0.105
## 2 versicolor      1.33     0.198
## 3 virginica       2.03     0.275
```

- Apply *grouping*, then use `summarize` function
  - Breaks dataframe into "mini-dataframes" before applying the function
- Data frame can be fed into other functions after summarizing

Grouping - Examples

```
iris %>% group_by(Species) %>% #Group by species
  summarize(count=n(), #Number of rows
            med=median(Petal.Width), #Median
            iqr=IQR(Petal.Width)) #Inter-quartile range
```

```
## # A tibble: 3 x 4
##   Species    count   med   iqr
##   <fct>      <int> <dbl> <dbl>
## 1 setosa        50   0.2   0.1
## 2 versicolor    50   1.3   0.3
## 3 virginica     50   2     0.5
```

- *n* is empty, because it shows the number of rows of the grouped "mini-dataframe"

Grouping - Examples

- Also useful for applying functions to subsets of data, *without* summarizing

```
iris %>% group_by(Species) %>%
  mutate(ID=1:n()) %>% #Makes ID column, with numbers 1-N
  filter(ID<4) #Selects ID 1-3 from each group
```

```
## # A tibble: 9 x 6
## # Groups:   Species [3]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species       ID
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>      <int>
## 1          5.1         3.5          1.4         0.2 setosa         1
## 2          4.9         3            1.4         0.2 setosa         2
## 3          4.7         3.2          1.3         0.2 setosa         3
## 4          7           3.2          4.7         1.4 versicolor     1
## 5          6.4         3.2          4.5         1.5 versicolor     2
## 6          6.9         3.1          4.9         1.5 versicolor     3
## 7          6.3         3.3          6           2.5 virginica      1
## 8          5.8         2.7          5.1         1.9 virginica      2
## 9          7.1         3            5.9         2.1 virginica      3
```

Grouping

- Another way of doing the same thing

```
iris %>% group_by(Species) %>%
  slice(1:3) #Selects rows 1-3 from each group
```

```
## # A tibble: 9 x 5
## # Groups:   Species [3]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
## 3          4.7         3.2          1.3         0.2 setosa
## 4          7           3.2          4.7         1.4 versicolor
## 5          6.4         3.2          4.5         1.5 versicolor
## 6          6.9         3.1          4.9         1.5 versicolor
## 7          6.3         3.3          6           2.5 virginica
## 8          5.8         2.7          5.1         1.9 virginica
## 9          7.1         3            5.9         2.1 virginica
```

Grouping

- Another way of doing the same thing

```
iris %>% group_by(Species) %>%
  slice(1:3) #Selects rows 1-3 from each group
```

```
## # A tibble: 9 x 5
## # Groups:   Species [3]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
## 3          4.7         3.2          1.3         0.2 setosa
## 4          7           3.2          4.7         1.4 versicolor
## 5          6.4         3.2          4.5         1.5 versicolor
## 6          6.9         3.1          4.9         1.5 versicolor
## 7          6.3         3.3          6           2.5 virginica
## 8          5.8         2.7          5.1         1.9 virginica
## 9          7.1         3            5.9         2.1 virginica
```

- You can use most of the subset and window functions across groups

Exercises!

Using the *InsectSprays* dataset:

- Find the mean and SD of `counts` for each type of `spray`

```
## # A tibble: 2 x 7
##   stat      A     B     C     D     E     F
##   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 mean  14.5  15.3   2.08  4.92  3.5  16.7
## 2 sd     4.72  4.27  1.98  2.50  1.73  6.21
```

Exercises!

Using the *InsectSprays* dataset:

- Find the mean and SD of `counts` for each type of `spray`
- Reshape dataframe so that each `spray` has its own column, with mean and SD in separate rows

```
## # A tibble: 2 x 7
##   stat      A     B     C     D     E     F
##   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 mean  14.5  15.3   2.08  4.92  3.5  16.7
## 2 sd     4.72  4.27  1.98  2.50  1.73  6.21
```

Exercises!

Using the *InsectSprays* dataset:

- Find the mean and SD of `counts` for each type of `spray`
- Reshape dataframe so that each `spray` has its own column, with mean and SD in separate rows
- Hint: get summary stats first, then `pivot_longer` and `pivot_wider`

```
## # A tibble: 2 x 7
##   stat      A     B     C     D     E     F
##   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 mean  14.5  15.3   2.08  4.92  3.5  16.7
## 2 sd     4.72  4.27  1.98  2.50  1.73  6.21
```

Happy data wrangling! Yee-haw!