# Intro to *data wrangling*

## dplyr & tidyr workshop

Samuel Robinson, Ph.D.

November 5, 2020

# 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')
```

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

# Data manipulation using dplyr/tidyr

```r
library(tidyverse)
#Convert factors in plants df
plants <- plants %>% mutate_at(vars(Species,Plant.Code),factor)
#Convert factors in seeds df
seeds <- seeds %>% mutate_at(vars(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_at(vars(3:9),funs(as.numeric))
#Convert Collection.Date to Date format
seeds <- mutate(Collection.Date=as.Date(Collection.Date,origin='2012-01-01'))
```

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

# Things to learn today:

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



"TEACH ME."
-STEPHEN STRANGE

# 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

```r
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
## # ... with 140 more rows
```

# Basic verbs - subsetting

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

```
head(iris,3)
```

```
##   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
```

```
#Select Petal.Length,Petal.Width,Species
irisTemp <- select(iris,Petal.Length,Petal.Width,Species)
head(irisTemp,3)
```

```
##   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**: *colon* operator

```
irisTemp <- select(iris,Petal.Length:Species)
head(irisTemp,3) #All columns between Petal.Length and Species
```

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

```
irisTemp2 <- select(iris,Petal.Length,Petal.Width,Species)
head(irisTemp2,3) #This is the same thing
```

```
##   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*

```
irisTemp <- select(iris,-Species)
head(irisTemp,3) #Selects all columns EXCEPT 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
```

```
irisTemp2 <- select(iris,contains('Petal'))
head(irisTemp2,3) #Selects columns with names containing 'Petal'
```

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

# Basic verbs - subsetting

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

```
head(iris,3)
```

```
##   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
```

```
irisTemp <- filter(iris,Sepal.Length<5,Species=='versicolor')
head(irisTemp,3) #Chooses rows matching logical criteria
```

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

# Basic verbs - make new variables

- **mutate**: add variables or alter existing ones

```
head(iris,3)
```

```
##   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
```

```
irisTemp <- mutate(iris,P.Width2=Petal.Width^2)
head(irisTemp,3) #Squares Petal.Width
```

```
##   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

```
head(irisTemp,3)
```

```
##   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
```

```
irisTemp <- mutate(iris,P.Width2=(Petal.Width^2)*2)
head(irisTemp,3) #Alters variable in place
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species P.Width2
## 1          5.1         3.5          1.4         0.2  setosa     0.08
## 2          4.9         3.0          1.4         0.2  setosa     0.08
## 3          4.7         3.2          1.3         0.2  setosa     0.08
```

# Basic verbs - make new variables

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

```
head(iris,3)
```

```
##   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
```

```
# "~" is called a lambda (similar to a function)
# "." means "data from column", so...
# "~.^2" means "square anything in this column"
irisTemp <- mutate(iris, across(c(Sepal.Length,Petal.Width), ~.^2))
head(irisTemp,3)
```

```
##   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

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

```
irisTemp <- rename(iris, PWidth=Petal.Width, PLength=Petal.Length)
head(irisTemp,3) #Renames columns
```

```
##   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
```

```
irisTemp2 <- transmute(iris, P.Width2=(Petal.Width^2))
head(irisTemp2,3) #Same as mutate, but drops other columns
```

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

# Exercises!

Using the `iris` dataset:

- Filter only rows with "virginica"
- Make 2 new "area" columns, which are length × 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 - %>%

- Takes data from one verb and passes it to the next one
- Allows you to 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
head(irisTemp)
```

```
##   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 operations in `tidyr` make this much easier
- Four main commands:

1. `pivot_longer` - gather columns into rows ('long format')
2. `pivot_wider` - spread rows into columns ('wide format')
3. `unite` - unite many columns into one (similar to `paste`)
4. `separate` - separates one column into many (similar to `strsplit`)

# Reshaping - *pivot_longer*: columns to rows

- Some data in a "wide" format

```
#Some data to work with
(bats <- data.frame(bat=letters[1:3], weight = seq(1,3,1),
                    height=seq(2.5,5.5,length=3), wings=rep(2,3)))
```

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

# Reshaping - *unite*: many columns into one

- Useful when combined with other reshaping functions

```
irisTemp <- iris %>% unite(newCol,Sepal.Length:Petal.Width,sep='_')
head(irisTemp,10)
```

```
##            newCol Species
## 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
```

# Reshaping - *separate*: one column into many

```
irisTemp %>% separate(newCol,c('SLength','SWidth','PLength','PWidth'),sep='_') %>%
  head(10)
```

```
##    SLength SWidth PLength PWidth Species
## 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
```

# Reshaping - combinations of reshaping functions

Say we wanted lengths and widths in separate columns, split by Petal & Sepal

```
irisTemp <- iris %>% unite(col=sepals,contains('Sepal'),sep='_') %>%
  unite(col=petals,contains('Petal'),sep='_')
  head(irisTemp,10)
```

```
##    sepals  petals Species
## 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
```

# Reshaping - combinations of reshaping functions

- Now that measurements are *united*, we *pivot_longer* and then *separate* them

```
irisTemp %>% pivot_longer(cols=sepals:petals,names_to='Type',values_to='Measurement') %>%
  separate(Measurement,c('Length','Width'),sep='_',convert=T) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##    Species Type   Length Width
##    <fct>   <chr>   <dbl> <dbl>
##  1 setosa  sepals    5.1   3.5
##  2 setosa  petals    1.4   0.2
##  3 setosa  sepals    4.9   3
##  4 setosa  petals    1.4   0.2
##  5 setosa  sepals    4.7   3.2
##  6 setosa  petals    1.3   0.2
##  7 setosa  sepals    4.6   3.1
##  8 setosa  petals    1.5   0.2
##  9 setosa  sepals    5     3.6
## 10 setosa  petals    1.4   0.2
```

# 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
- For example, what is the average of each species' *Petal.width*?

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

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

```r
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/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 0.100
## 2 versicolor    50   1.3 0.3
## 3 virginica     50   2   0.500
```

- *n* is empty, because it uses the length of the subsetted data frame

# 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

```r
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 each type of spray type
- 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
```

# Final remarks

- dplyr & tidyr interface well with other parts of the tidyverse



```r
library(ggplot2)
#Code for dplyr begins here
CO2 %>% filter(conc==1000) %>%
  group_by(Type,Treatment) %>%
  summarize(meanUp=mean(uptake),
            maxUp=max(uptake),
            minUp=min(uptake)) %>%
  #Code for ggplot begins here
  ggplot(aes(x=Type,col=Treatment))+
  geom_pointrange(aes(y=meanUp,
                      ymax=maxUp,
                      ymin=minUp))+
  labs(x='Area',y='Uptake at 1000ppm')+
  scale_colour_manual(values=c('red','blue'))
```
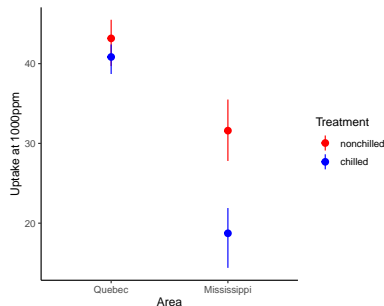
# Final remarks

- dplyr & tidyr can pass data frames to and from other functions: use '.' operator

```
co2mod <- CO2 %>%
  filter(Type=='Quebec') %>%
  #Code for nls begins here
  nls(uptake~SSasymp(conc,A,B,C),
      start=list(A=30,B=-15,C=-5),data=.)

data.frame(conc=seq(50,1000,20)) %>%
  predict(co2mod,newdata=.) %>%
  data.frame(conc=seq(50,1000,20),predUp=.) %>%
  #Code for ggplot begins here
  ggplot(aes(conc,predUp))+
  geom_line()+
  geom_point(data=filter(CO2,Type=='Quebec'),
             aes(conc,uptake))+
  labs(x='CO2 Concentration',y='Uptake')
```