Data wrangling and viualization

Why use tidyverse? 1. Base R can be painful with different data formats 2. Its more intuitive and useful for new learners

For example, often base R forces character to be converted to factors

##dplyr package 1. dplyr package makes tabular data manipulation easier 2. Many functions are compiled in C++ so its faster 3. You can work with data stored in external data bases like SQL so removing limitation of memory as an issue

##tidyr package 1. tidyr allows easier conversion between different data formats e.g., long and wide 2. It makes such conversion easy

Both these packages are available in umbrella package Tidyverse

Data

```
data<- download.file(url="https://ndownloader.figshare.com/files/2292169",
              destfile = "data/combined.csv")
surveys<- read_csv("data/combined.csv")
```

```
## Parsed with column specification:
## cols(
##   record_id = col_double(),
##   month = col_double(),
##   day = col_double(),
##   year = col_double(),
##   plot_id = col_double(),
##   species_id = col_character(),
##   sex = col_character(),
##   hindfoot_length = col_double(),
##   weight = col_double(),
##   genus = col_character(),
##   species = col_character(),
##   taxa = col_character(),
##   plot_type = col_character()
## )
```

```
head(surveys)
```

```
## # A tibble: 6 x 13
##   record_id month   day  year plot_id species_id sex   hindfoot_length weight
##       <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl>
## 1         1     7    16  1977       2 NL         M                  32     NA
## 2        72     8    19  1977       2 NL         M                  31     NA
```

```
## 3       224     9    13  1977       2 NL         <NA>               NA     NA
## 4       266    10    16  1977       2 NL         <NA>               NA     NA
## 5       349    11    12  1977       2 NL         <NA>               NA     NA
## 6       363    11    12  1977       2 NL         <NA>               NA     NA
## # ... with 4 more variables: genus <chr>, species <chr>, taxa <chr>,
## #   plot_type <chr>
```

Packages

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

Packages

```
str(surveys)
```

```
## tibble [34,786 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ record_id      : num [1:34786] 1 72 224 266 349 363 435 506 588 661 ...
##  $ month          : num [1:34786] 7 8 9 10 11 11 12 1 2 3 ...
##  $ day            : num [1:34786] 16 19 13 16 12 12 10 8 18 11 ...
##  $ year           : num [1:34786] 1977 1977 1977 1977 1977 ...
##  $ plot_id        : num [1:34786] 2 2 2 2 2 2 2 2 2 2 ...
##  $ species_id     : chr [1:34786] "NL" "NL" "NL" "NL" ...
##  $ sex            : chr [1:34786] "M" "M" NA NA ...
##  $ hindfoot_length: num [1:34786] 32 31 NA NA NA NA NA NA NA NA ...
##  $ weight         : num [1:34786] NA NA NA NA NA NA NA NA 218 NA ...
##  $ genus          : chr [1:34786] "Neotoma" "Neotoma" "Neotoma" "Neotoma" ...
##  $ species        : chr [1:34786] "albigula" "albigula" "albigula" "albigula" ...
##  $ taxa           : chr [1:34786] "Rodent" "Rodent" "Rodent" "Rodent" ...
##  $ plot_type      : chr [1:34786] "Control" "Control" "Control" "Control" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   record_id = col_double(),
##   ..   month = col_double(),
##   ..   day = col_double(),
##   ..   year = col_double(),
##   ..   plot_id = col_double(),
##   ..   species_id = col_character(),
##   ..   sex = col_character(),
##   ..   hindfoot_length = col_double(),
##   ..   weight = col_double(),
##   ..   genus = col_character(),
##   ..   species = col_character(),
##   ..   taxa = col_character(),
##   ..   plot_type = col_character()
##   .. )
```

```
#View(surveys)
```

Tibble

Tibble is like a data frame, characters are not coerced into columns

Functions we will use

1. Select: subset columns
2. Filer: subset rows based on condition
3. mutate: create new columns by using info from other column
4. group_by() and summarize(): create summary statistics on grouped data
5. arrange(): sort results
6. count(): count discrete values

Select specific columns

```
select(surveys, plot_id,species_id,weight)
```

```
## # A tibble: 34,786 x 3
##    plot_id species_id weight
##      <dbl> <chr>       <dbl>
##  1       2 NL             NA
##  2       2 NL             NA
##  3       2 NL             NA
##  4       2 NL             NA
##  5       2 NL             NA
##  6       2 NL             NA
##  7       2 NL             NA
##  8       2 NL             NA
##  9       2 NL            218
## 10       2 NL             NA
## # ... with 34,776 more rows
```

Select all columns but other than specific columns

```
select(surveys, -record_id, -species_id)
```

```
## # A tibble: 34,786 x 11
##    month   day  year plot_id sex   hindfoot_length weight genus species taxa
##    <dbl> <dbl> <dbl>   <dbl> <chr>           <dbl>  <dbl> <chr> <chr>   <chr>
##  1     7    16  1977       2 M                  32     NA Neot~ albigu~ Rode~
##  2     8    19  1977       2 M                  31     NA Neot~ albigu~ Rode~
##  3     9    13  1977       2 <NA>               NA     NA Neot~ albigu~ Rode~
##  4    10    16  1977       2 <NA>               NA     NA Neot~ albigu~ Rode~
##  5    11    12  1977       2 <NA>               NA     NA Neot~ albigu~ Rode~
##  6    11    12  1977       2 <NA>               NA     NA Neot~ albigu~ Rode~
##  7    12    10  1977       2 <NA>               NA     NA Neot~ albigu~ Rode~
```

```
##  8     1     8  1978       2 <NA>              NA     NA Neot~ albigu~ Rode~
##  9     2    18  1978       2 M                 NA    218 Neot~ albigu~ Rode~
## 10     3    11  1978       2 <NA>              NA     NA Neot~ albigu~ Rode~
## # ... with 34,776 more rows, and 1 more variable: plot_type <chr>
```

Select rows based on specific condition

```
filter(surveys, year==1995)
```

```
## # A tibble: 1,180 x 13
##    record_id month   day  year plot_id species_id sex   hindfoot_length weight
##        <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>      <chr>           <dbl>  <dbl>
##  1     22314     6     7  1995       2 NL         M                  34     NA
##  2     22728     9    23  1995       2 NL         F                  32    165
##  3     22899    10    28  1995       2 NL         F                  32    171
##  4     23032    12     2  1995       2 NL         F                  33     NA
##  5     22003     1    11  1995       2 DM         M                  37     41
##  6     22042     2     4  1995       2 DM         F                  36     45
##  7     22044     2     4  1995       2 DM         M                  37     46
##  8     22105     3     4  1995       2 DM         F                  37     49
##  9     22109     3     4  1995       2 DM         M                  37     46
## 10     22168     4     1  1995       2 DM         M                  36     48
## # ... with 1,170 more rows, and 4 more variables: genus <chr>, species <chr>,
## #   taxa <chr>, plot_type <chr>
```

Doing multiple steps

```
#select weight > 5g
surveys2<-filter(surveys,weight<5)
# now slect only few columns
surveys2_sml<- select(surveys2,species_id,sex,weight )
```

We can do the same by nesting functions

```
#select weight > 5g
surveys2_sml<-select(filter(surveys, weight<5), species_id,sex,weight  )
```

Doing multiple steps at a time

#mac cmd+shift+m #windows ctrl+shift+m

```
#select weight > 5g
surveys2<-surveys %>% filter(weight<5) %>%
  select(species_id,sex,weight )
```

Challenge - 5 mins

Using pipes, subset the surveys data to include animals collected before 1995 and retain only the columns year, sex, and weight.

Split-combine-apply

1. Split the data into groups of categorical variables (e.g., group_by)
2. Apply some analysis to each group (e.g., summarize, gives one row values)
3. Combine the results

Lets compute weight by sex

```
surveys_wt_sex<- surveys %>% group_by(sex) %>%
  summarise(mean_weight= mean(weight,na.rm=TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
surveys_wt_sex
```

```
## # A tibble: 3 x 2
##   sex   mean_weight
##   <chr>       <dbl>
## 1 F            42.2
## 2 M            43.0
## 3 <NA>         64.7
```

Grouping by multiple columns

```
surveys_wt_sex_sp<- surveys %>% group_by(sex, species_id) %>%
  summarise(mean_weight= mean(weight,na.rm=TRUE))
```

```
## `summarise()` regrouping output by 'sex' (override with `.groups` argument)
```

```
surveys_wt_sex_sp
```

```
## # A tibble: 92 x 3
## # Groups:   sex [3]
##    sex   species_id mean_weight
##    <chr> <chr>            <dbl>
##  1 F     BA                9.16
##  2 F     DM               41.6
##  3 F     DO               48.5
##  4 F     DS              118.
```

```
##  5 F     NL              154.
##  6 F     OL               31.1
##  7 F     OT               24.8
##  8 F     OX               21
##  9 F     PB               30.2
## 10 F     PE               22.8
## # ... with 82 more rows
```

```
# NaN beacuse the math was done on NA columns
```

Lets remove NAs first

```
surveys_wt_sex_sp_na<-surveys %>%
  filter(!is.na(weight)) %>%
  group_by(species_id,sex) %>%
    summarize(mean_weight = mean(weight)) %>%
  print(n=15)
```

```
## `summarise()` regrouping output by 'species_id' (override with `.groups` argument)
```

```
## # A tibble: 64 x 3
## # Groups:   species_id [25]
##    species_id sex   mean_weight
##    <chr>      <chr>       <dbl>
##  1 BA         F            9.16
##  2 BA         M            7.36
##  3 DM         F           41.6
##  4 DM         M           44.4
##  5 DM         <NA>        38.3
##  6 DO         F           48.5
##  7 DO         M           49.1
##  8 DO         <NA>        50.7
##  9 DS         F          118.
## 10 DS         M          122.
## 11 DS         <NA>       120
## 12 NL         F          154.
## 13 NL         M          166.
## 14 NL         <NA>       168.
## 15 OL         F           31.1
## # ... with 49 more rows
```

We can add another column of min weight

```
surveys_wt_sex_sp_na<-surveys %>%
  filter(!is.na(weight)) %>%
  group_by(species_id,sex) %>%
    summarize(mean_weight = mean(weight), min_weight=min(weight)) %>%
  arrange(min_weight)
```

```
## `summarise()` regrouping output by 'species_id' (override with `.groups` argument)
```

Count

```
surveys %>%
    count(sex, sort=TRUE)
```

```
## # A tibble: 3 x 2
##   sex       n
##   <chr> <int>
## 1 M     17348
## 2 F     15690
## 3 <NA>   1748
```

```
# arrange based on the count
surveys %>%
  count(sex, species) %>%
  arrange(species, desc(n))
```

```
## # A tibble: 81 x 3
##    sex   species            n
##    <chr> <chr>          <int>
##  1 F     albigula         675
##  2 M     albigula         502
##  3 <NA>  albigula          75
##  4 <NA>  audubonii          75
##  5 F     baileyi         1646
##  6 M     baileyi         1216
##  7 <NA>  baileyi           29
##  8 <NA>  bilineata        303
##  9 <NA>  brunneicapillus   50
## 10 <NA>  chlorurus         39
## # ... with 71 more rows
```

Challenge- 10 mins breakout room

1. How many animals were caught in each plot_type surveyed?
2. Use group_by() and summarize() to find the mean, min, and max hindfoot length for each species (using species_id). Also add the number of observations (hint: see ?n).
3. What was the heaviest animal measured in each year? Return the columns year, genus, species_id, and weight.

Reshaping with gather and spread (Pivot_wider and longer now)

Rules of data: 1. Each variable has its own column 2. Each observation has its own row 3. Each value must have its own cell 4. Each type of observational unit forms a table

In our surveys data each row is one observational unit. Sometimes we want to compare various columns (variable) for each observational unit

Spreading- spreads the data so that rows becomes columns gather- does the reverse

spread() takes three principal arguments: 1. the data 2. the key column variable whose values will become new column names. 3. the value column variable whose values will fill the new column variables.

Lets find mean wt of each genus in each plot over the entire survey period per plot id

```
surveys_gw<- surveys %>%
  filter(!is.na(weight)) %>%
  group_by(plot_id,genus) %>%
  summarize(mean_wt= mean(weight))
```

```
## `summarise()` regrouping output by 'plot_id' (override with `.groups` argument)
```

```
head(surveys_gw)
```

```
## # A tibble: 6 x 3
## # Groups:   plot_id [1]
##   plot_id genus       mean_wt
##     <dbl> <chr>         <dbl>
## 1       1 Baiomys        7
## 2       1 Chaetodipus   22.2
## 3       1 Dipodomys     60.2
## 4       1 Neotoma      156.
## 5       1 Onychomys     27.7
## 6       1 Perognathus    9.62
```

Now take genus and make them columns

```
surveys_Spread<- surveys_gw %>%
  spread(key=genus, value = mean_wt)
head(surveys_Spread)
```

```
## # A tibble: 6 x 11
## # Groups:   plot_id [6]
##   plot_id Baiomys Chaetodipus Dipodomys Neotoma Onychomys Perognathus
##     <dbl>   <dbl>       <dbl>     <dbl>   <dbl>     <dbl>       <dbl>
## 1       1    7           22.2      60.2    156.      27.7        9.62
## 2       2    6           25.1      55.7    169.      26.9        6.95
## 3       3    8.61        24.6      52.0    158.      26.0        7.51
## 4       4   NA           23.0      57.5    164.      28.1        7.82
## 5       5    7.75        18.0      51.1    190.      27.0        8.66
## 6       6   NA           24.9      58.6    180.      25.9        7.81
## # ... with 4 more variables: Peromyscus <dbl>, Reithrodontomys <dbl>,
## #   Sigmodon <dbl>, Spermophilus <dbl>
```

Lets fill the missing values with 0

```
surveys_spread2<-surveys_gw %>%
  spread(key=genus, value=mean_wt, fill = 0) %>%
  head()
```

Gather is opposite of spread

1. the data
2. the key column variable we wish to create from column names.
3. the values column variable we wish to create and fill with values associated with the key.
4. the names of the columns we use to fill the key variable (or to drop).

```
surveys_gather<- surveys_spread2%>%
  gather(key=genus, value=mean_wt, -plot_id) %>%
  head()
surveys_gather
```

```
## # A tibble: 6 x 3
## # Groups:   plot_id [6]
##   plot_id genus   mean_wt
##     <dbl> <chr>     <dbl>
## 1       1 Baiomys    7
## 2       2 Baiomys    6
## 3       3 Baiomys    8.61
## 4       4 Baiomys    0
## 5       5 Baiomys    7.75
## 6       6 Baiomys    0
```

We can only select few columns to become rows

```
surveys_spread2 %>%
  gather(key = "genus", value = "mean_wt", Baiomys:Spermophilus) %>%
  head()
```

```
## # A tibble: 6 x 3
## # Groups:   plot_id [6]
##   plot_id genus   mean_wt
##     <dbl> <chr>     <dbl>
## 1       1 Baiomys    7
## 2       2 Baiomys    6
## 3       3 Baiomys    8.61
## 4       4 Baiomys    0
## 5       5 Baiomys    7.75
## 6       6 Baiomys    0
```

Challenge # 15 mins

1. Spread the surveys data frame with year as columns, plot_id as rows, and the number of genera per plot as the values. You will need to summarize before reshaping, and use the function n_distinct() to get the number of unique genera within a particular chunk of data. It's a powerful function! See ?n_distinct for more.
2. Now take that data frame and gather() it again, so each row is a unique plot_id by year combination.
3. The surveys data set has two measurement columns: hindfoot_length and weight. This makes it difficult to do things like look at the relationship between mean values of each measurement per year in different plot types. Let's walk through a common solution for this type of problem. First, use gather() to create a dataset where we have a key column called measurement and a value column that takes on the value of either hindfoot_length or weight. Hint: You'll need to specify which columns are being gathered.
4. With this new data set, calculate the average of each measurement in each year for each different plot_type. Then spread() them into a data set with a column for hindfoot_length and weight. Hint: You only need to specify the key and value columns for spread()

Exporting data

```
surveys_complete <- surveys %>%
  filter(!is.na(weight),           # remove missing weight
         !is.na(hindfoot_length),  # remove missing hindfoot_length
         !is.na(sex))                # remove missing sex

## Extract the most common species_id
species_counts <- surveys_complete %>%
    count(species_id) %>%
    filter(n >= 50)

## Only keep the most common species
surveys_complete <- surveys_complete %>%
  filter(species_id %in% species_counts$species_id)

#To make sure that everyone has the same data set, check that surveys_complete has 30463 rows and 13 co

write_csv(surveys_complete, path = "data/surveys_complete.csv")
```

Visualizing data

Objectives 1. Produce scatter plots, boxplots, and time series plots using ggplot. 2. Set universal plot settings. 3. Describe what faceting is and apply faceting in ggplot. 4. Modify the aesthetics of an existing ggplot plot (including axis labels and color). 5. Build complex and customized plots from data in a data frame.

Plotting with ggplot

1. Helps creates publication quality graphs

2. more intuitive
3. step by step addition of layers
4. likes the data in long format

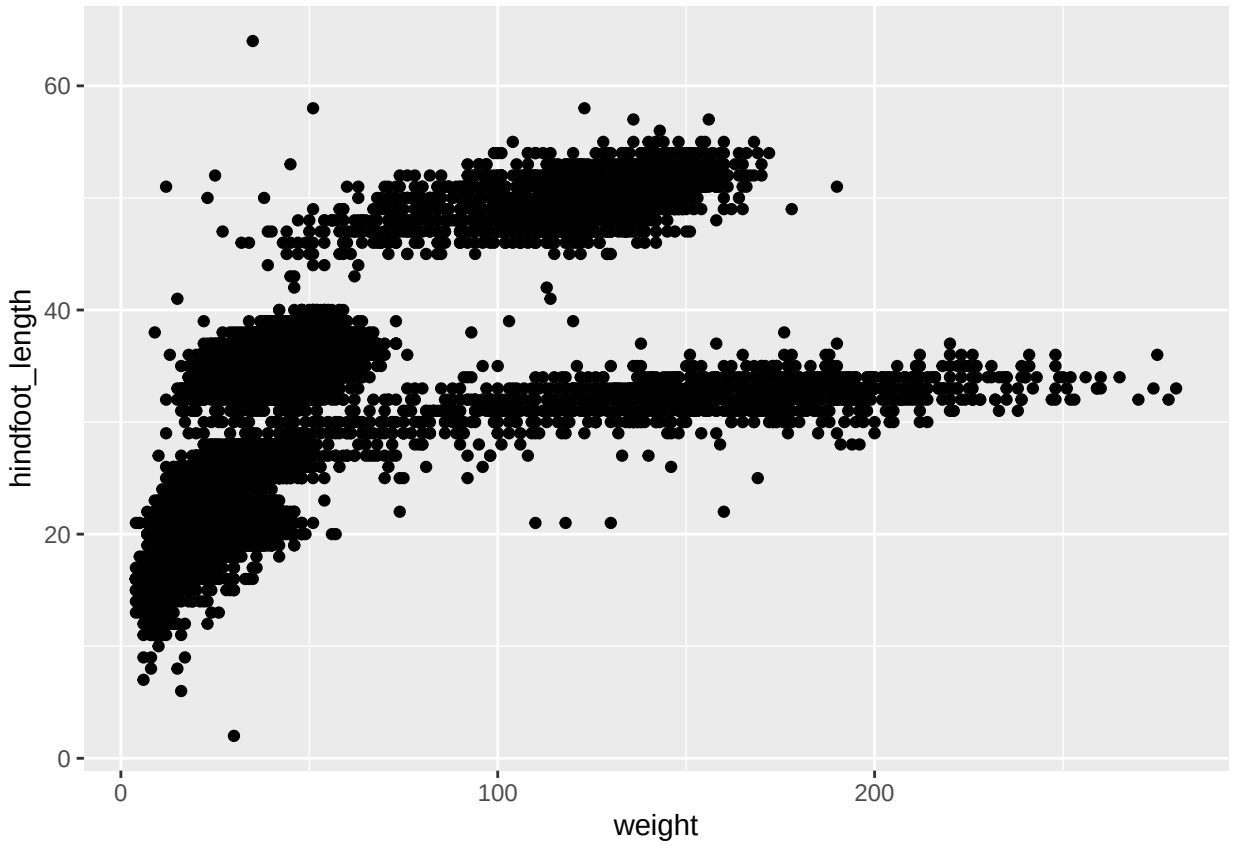
ggplot(data = , mapping = aes()) + ()

Scatterplot

```
surveys_complete<-read_csv("data/surveys_complete.csv")
```

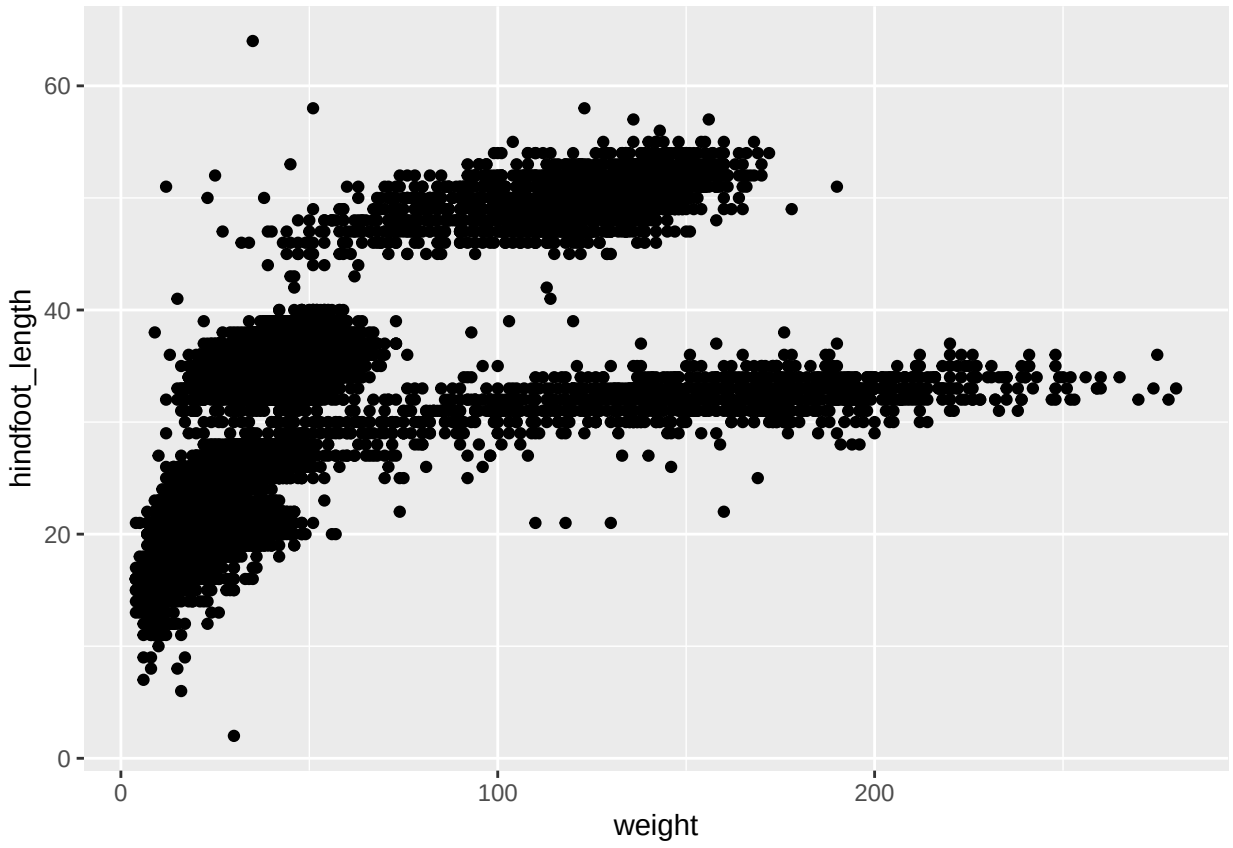
```
## Parsed with column specification:
## cols(
##   record_id = col_double(),
##   month = col_double(),
##   day = col_double(),
##   year = col_double(),
##   plot_id = col_double(),
##   species_id = col_character(),
##   sex = col_character(),
##   hindfoot_length = col_double(),
##   weight = col_double(),
##   genus = col_character(),
##   species = col_character(),
##   taxa = col_character(),
##   plot_type = col_character()
## )
```

```
ggplot(data = surveys_complete, aes(x = weight, y = hindfoot_length)) +
  geom_point()
```



'+' allows you to assign a name to plot and customize it with as many layers

```
surveys_plot <- ggplot(data = surveys_complete,
                       mapping = aes(x = weight, y = hindfoot_length))
# Draw the plot
surveys_plot +
    geom_point()
```



Custom setting

Anything you add on ggplot function The + sign used to add layers must be placed at the end of each line containing a layer.

Lets modify the plots

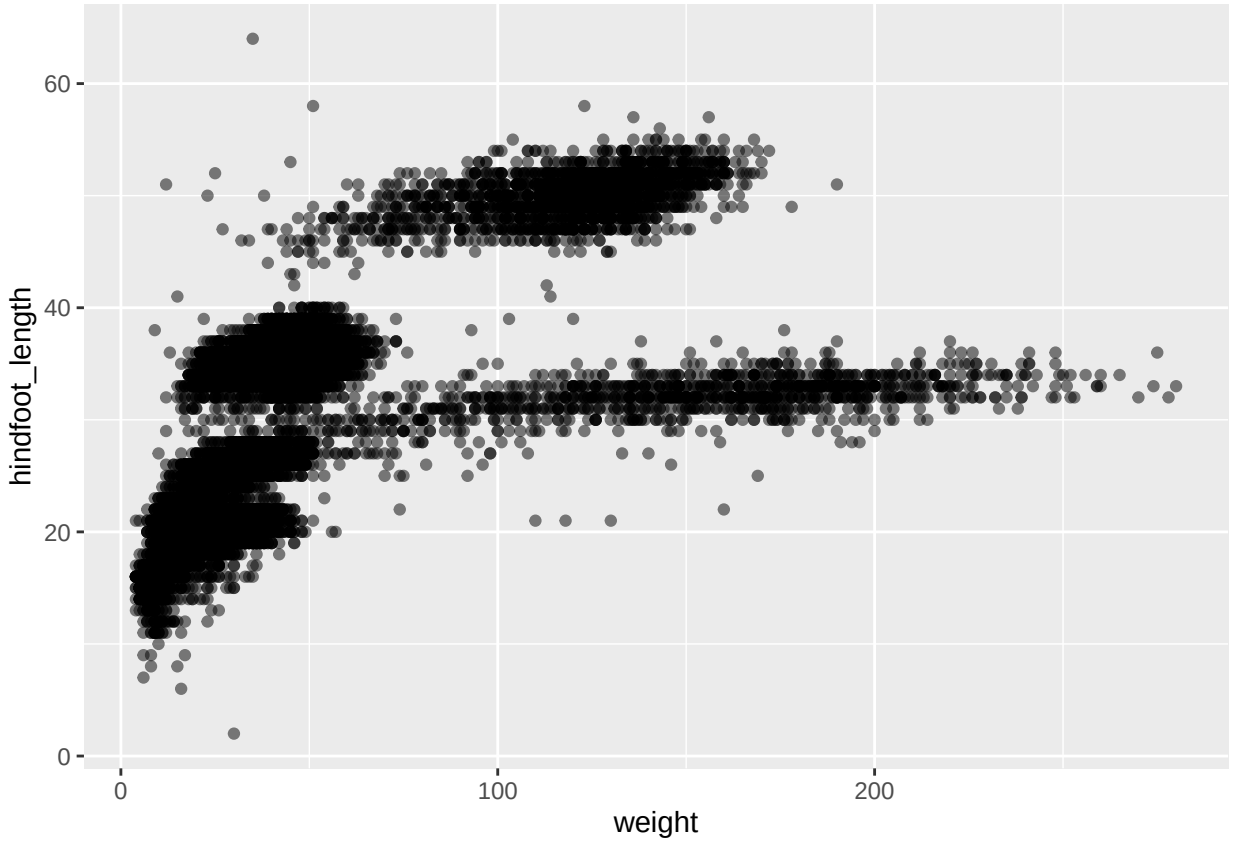
Reduce opacity

```
surveys_complete<-read_csv("data/surveys_complete.csv")
```

```
## Parsed with column specification:
## cols(
##   record_id = col_double(),
##   month = col_double(),
##   day = col_double(),
##   year = col_double(),
##   plot_id = col_double(),
##   species_id = col_character(),
##   sex = col_character(),
##   hindfoot_length = col_double(),
##   weight = col_double(),
##   genus = col_character(),
```

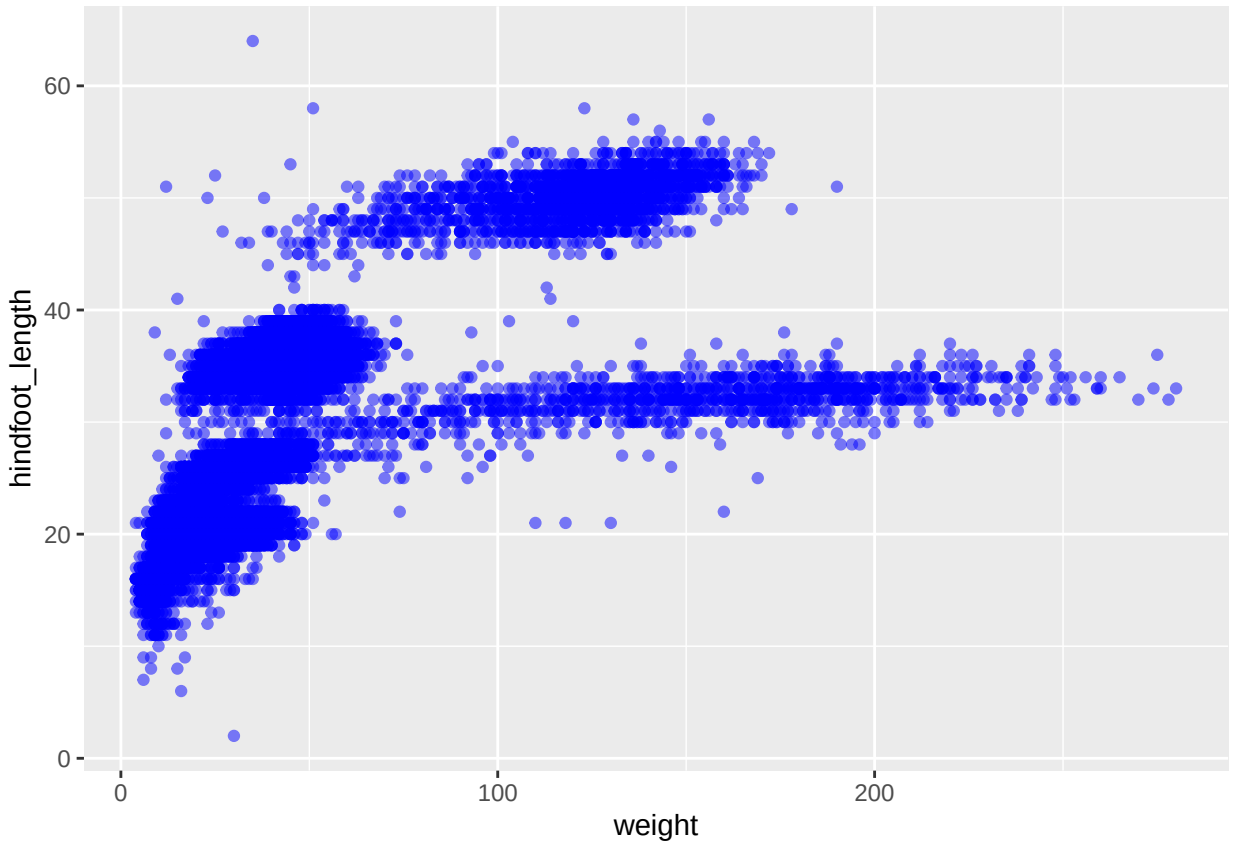
```
##   species = col_character(),
##   taxa = col_character(),
##   plot_type = col_character()
## )
```

```
ggplot(data = surveys_complete, aes(x = weight, y = hindfoot_length) ) +
  geom_point(alpha=0.5)
```



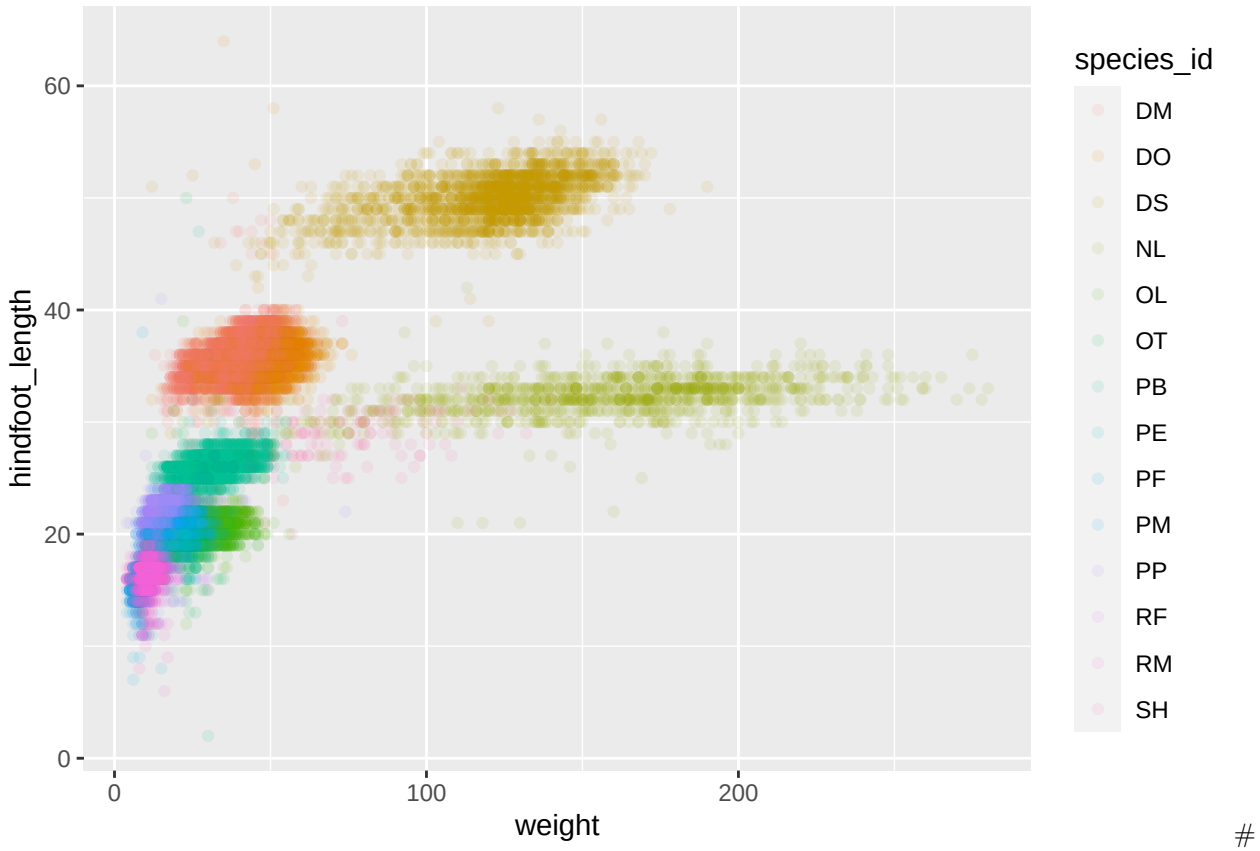
Lets change color of points

```
ggplot(data = surveys_complete, aes(x = weight, y = hindfoot_length) ) +
  geom_point(alpha=0.5,color='blue')
```



Lets plot so that each species has a diff color

```
ggplot(data = surveys_complete, aes(x = weight, y = hindfoot_length) ) +
  geom_point(aes(color=species_id), alpha=0.1)
```

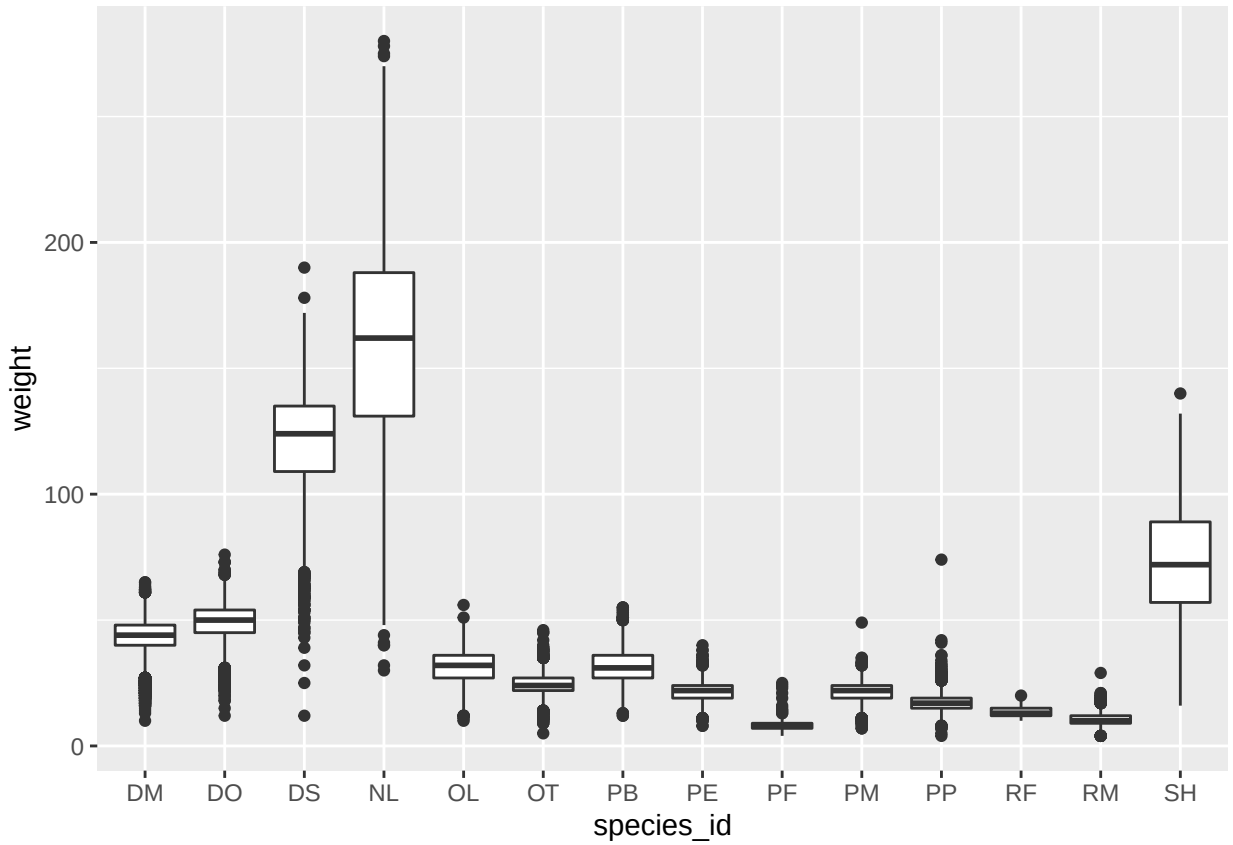


#

Challenge Use what you just learned to create a scatter plot of weight over species_id with the plot types showing in different colors. Is this a good way to show this type of data?

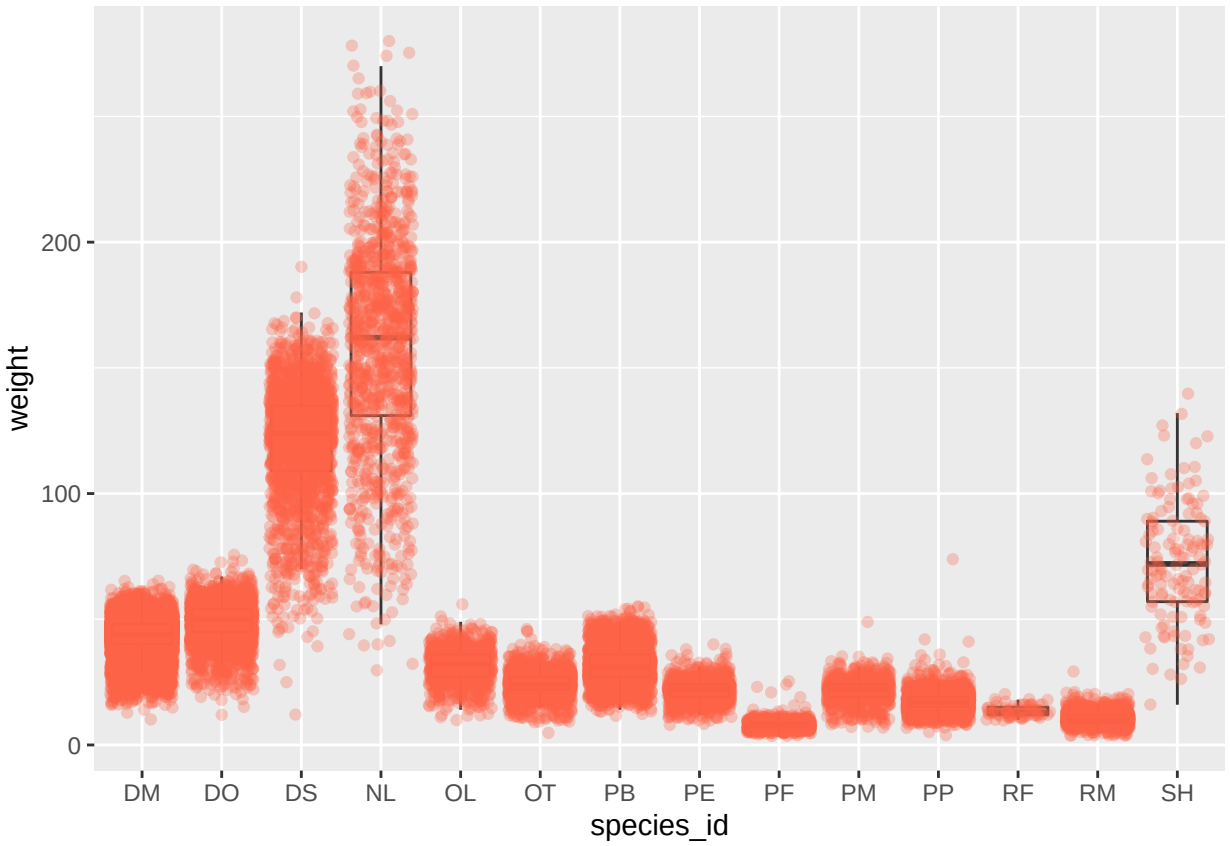
Boxplot

```
ggplot(data = surveys_complete, mapping = aes(x = species_id, y = weight)) +
    geom_boxplot()
```



Lets plot data over box plot

```
ggplot(data = surveys_complete, mapping = aes(x = species_id, y = weight)) +
    geom_boxplot(alpha = 0) +
    geom_jitter(alpha = 0.3, color = "tomato")
```



Challege

Boxplots are useful summaries, but hide the shape of the distribution. For example, if there is a bimodal distribution, it would not be observed with a boxplot. An alternative to the boxplot is the violin plot (sometimes known as a beanplot), where the shape (of the density of points) is drawn.

1. Replace the box plot with a violin plot; see geom_violin().

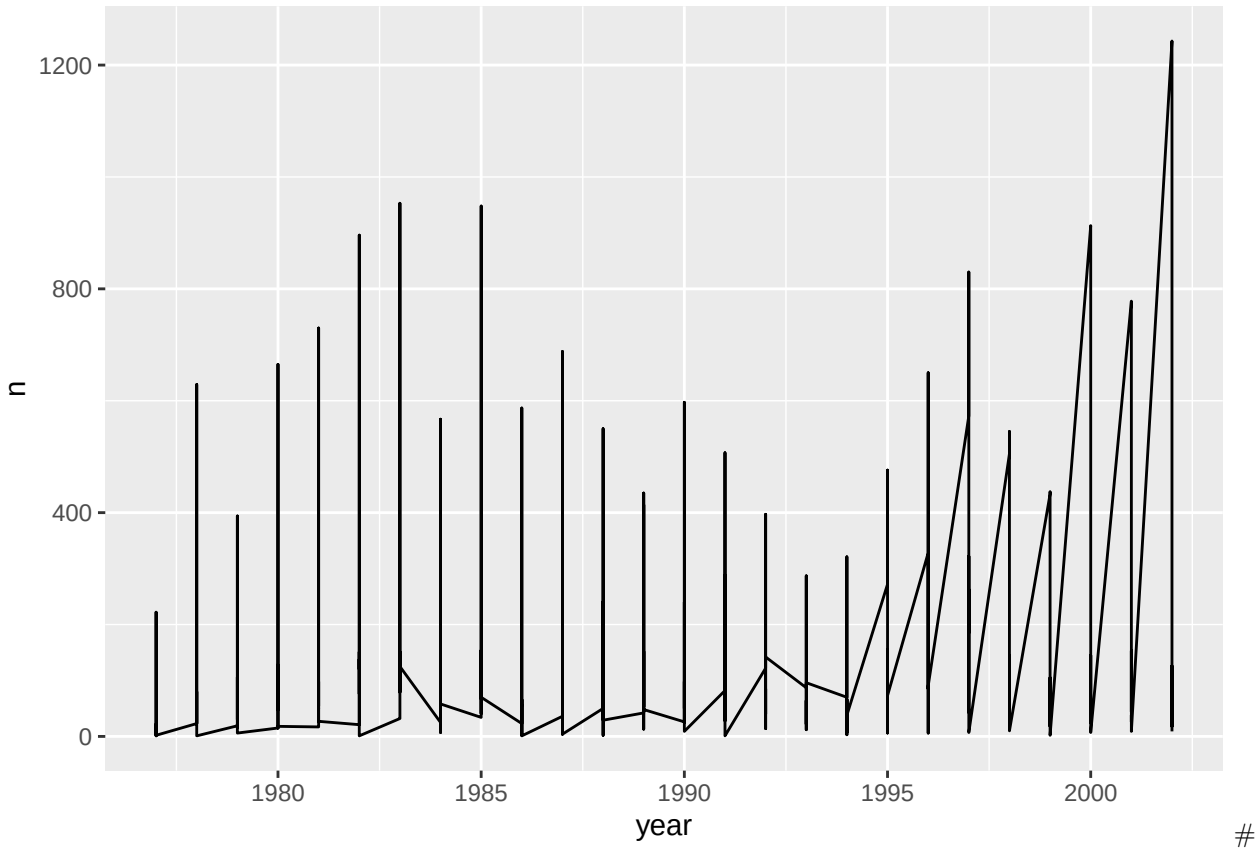
In many types of data, it is important to consider the scale of the observations. 2. Represent weight on the log10 scale; see scale_y_log10()

Plotting time series data

Lets look at counts of every genus every year

```
yearly_counts <- surveys_complete %>%
  count(year, genus)
```

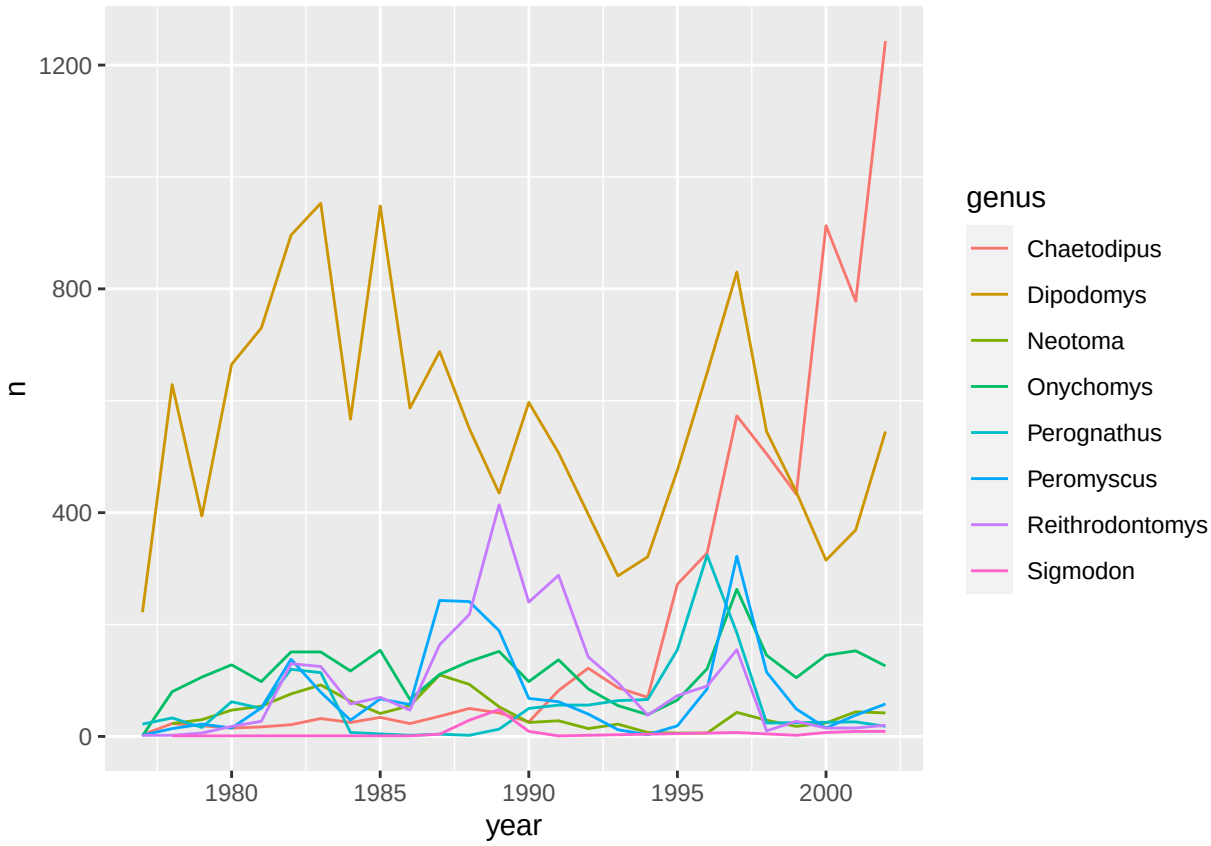
```
ggplot(data = yearly_counts, aes(x = year, y = n)) +
     geom_line()
```



#

Lets have one line for each genus

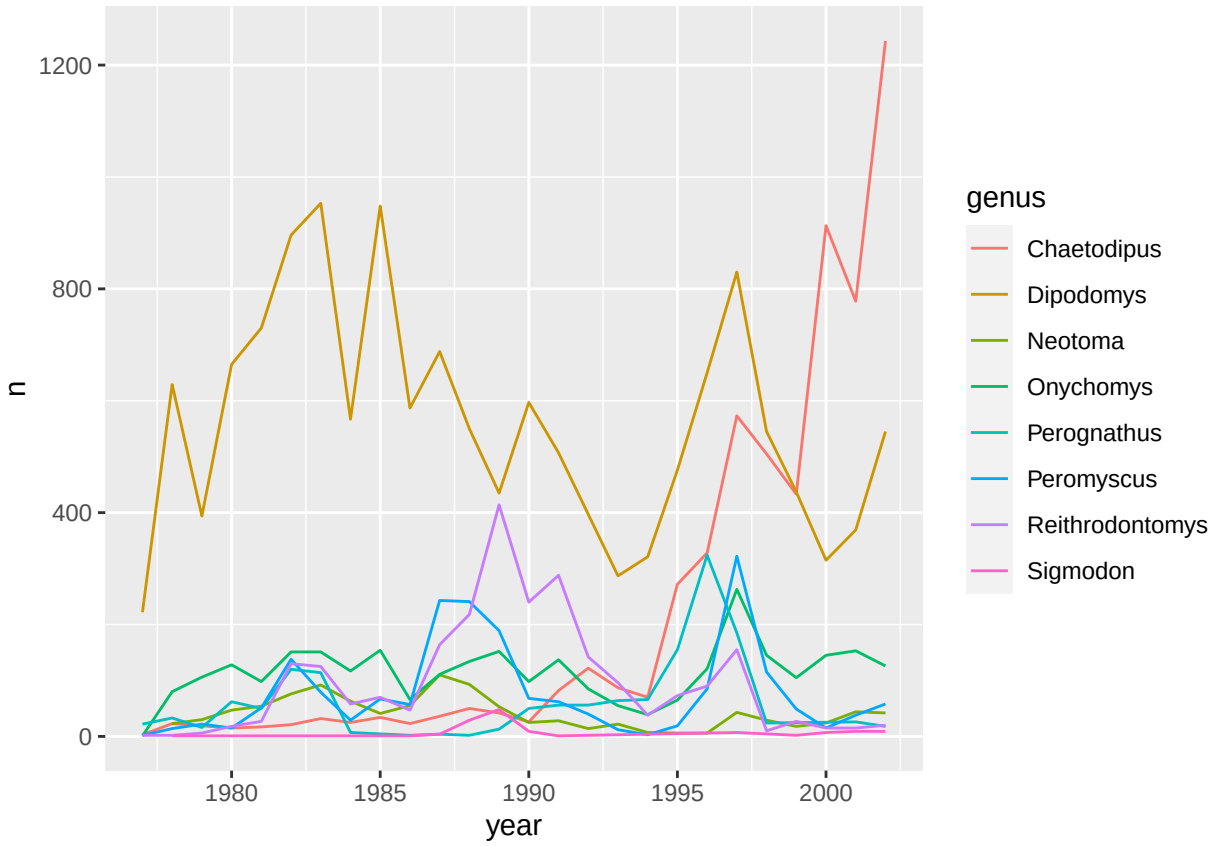
```
ggplot(data = yearly_counts, aes(x = year, y = n)) +
     geom_line(aes(color=genus))
```



Integrating pipe operator with ggplot

Make life easy

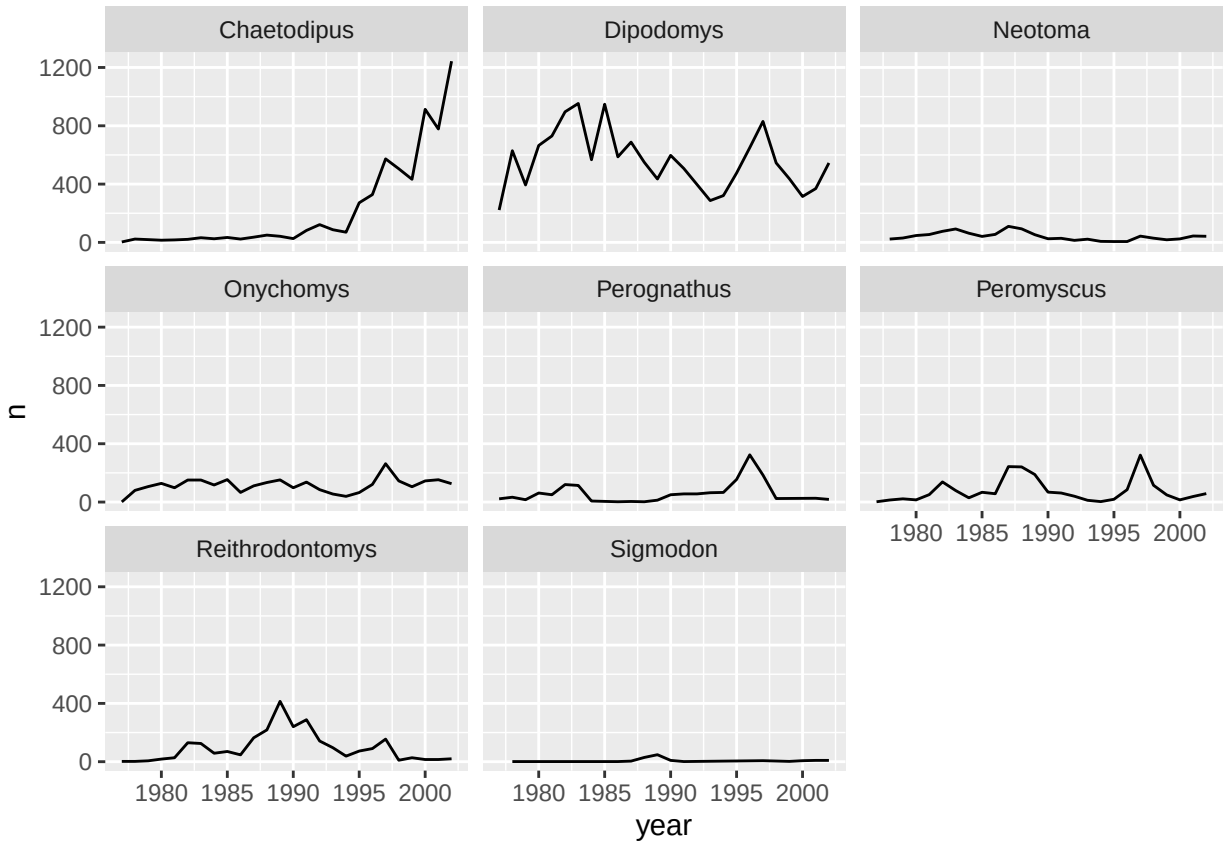
```
yearly_counts %>%
    ggplot(mapping = aes(x = year, y = n, color = genus)) +
    geom_line()
```



Faceting

Create mini plots

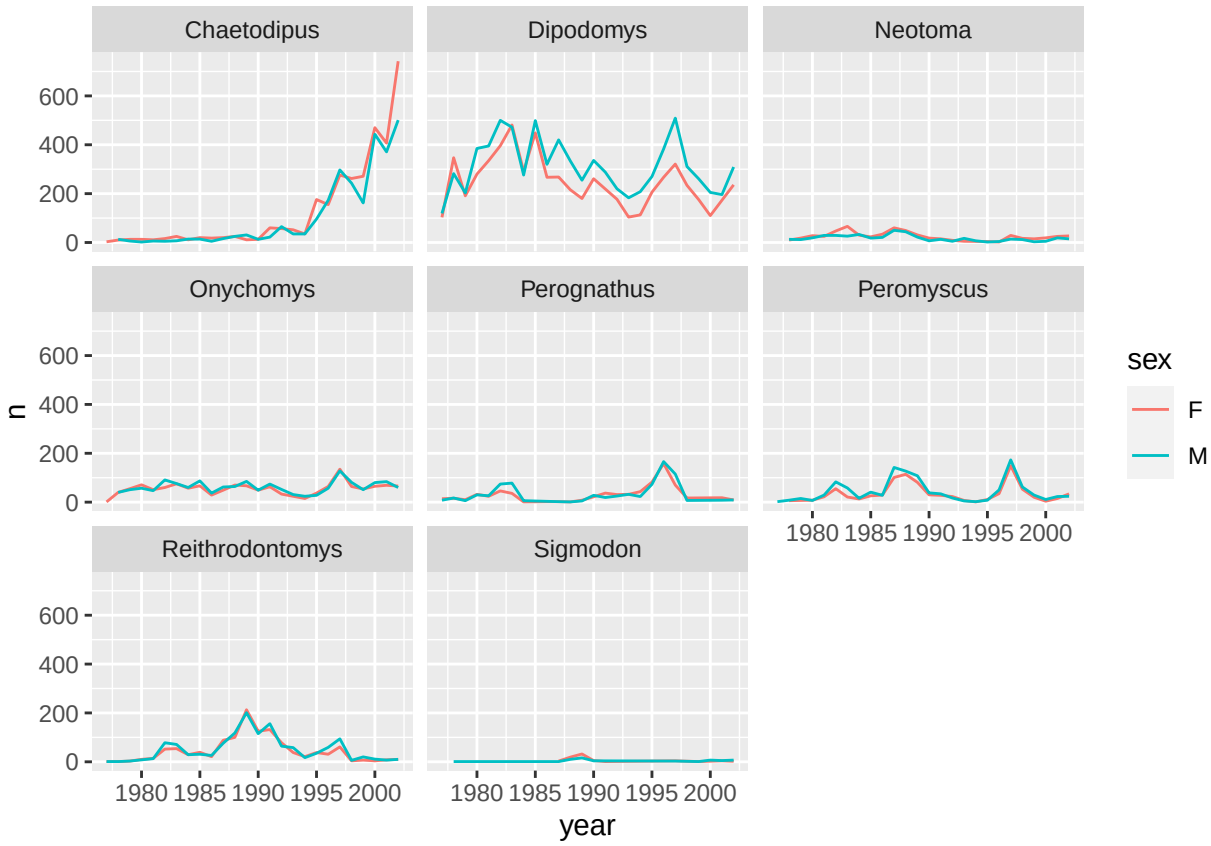
```
ggplot(data = yearly_counts, aes(x = year, y = n)) +
    geom_line() +
    facet_wrap(facets = vars(genus))
```



#

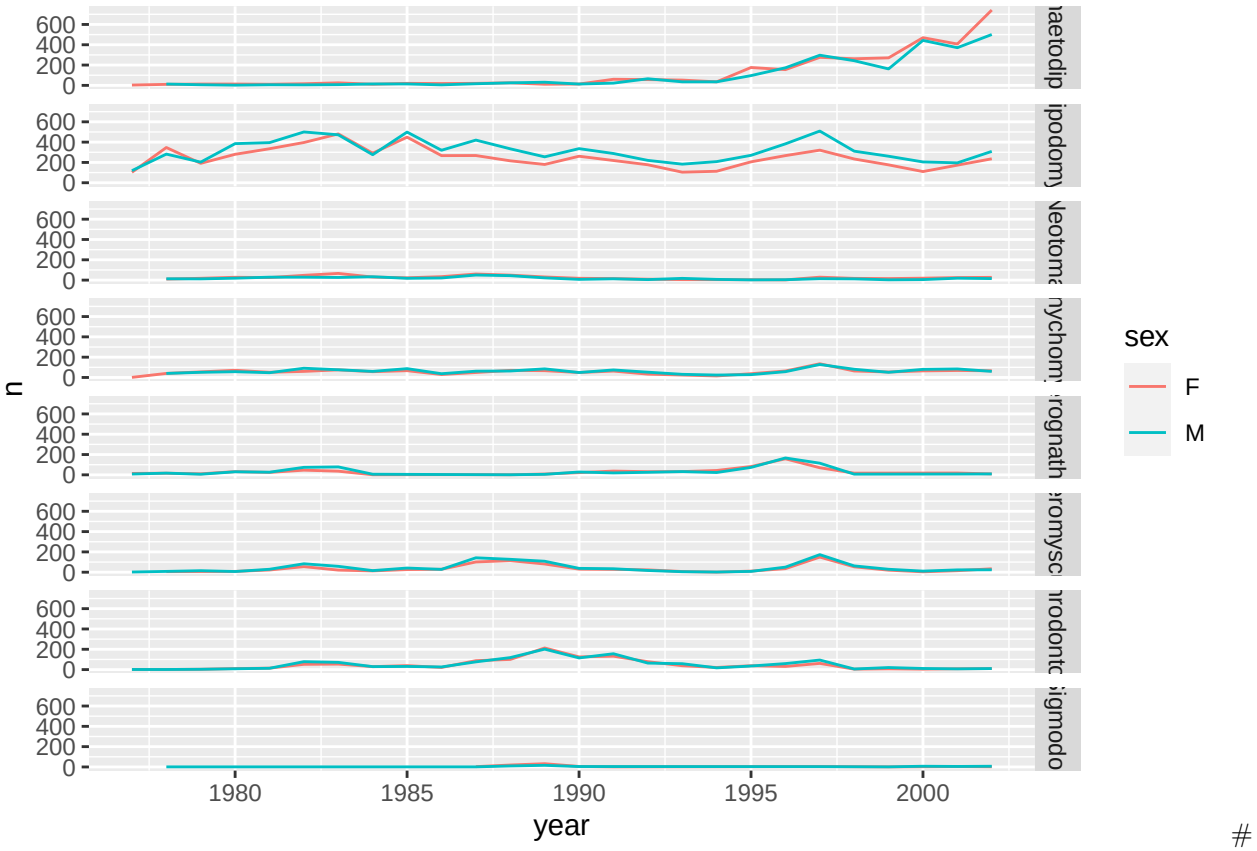
Now color with sex

```
 surveys_complete %>%
  count(year, genus, sex) %>%
ggplot( aes(x = year, y = n,color=sex)) +
    geom_line() +
    facet_wrap(facets = vars(genus))
```



Use facet_grid for arranging only in rows

```
 surveys_complete %>%
  count(year, genus, sex) %>%
ggplot( aes(x = year, y = n,color=sex)) +
    geom_line() +
    facet_grid(facets = vars(genus))
```

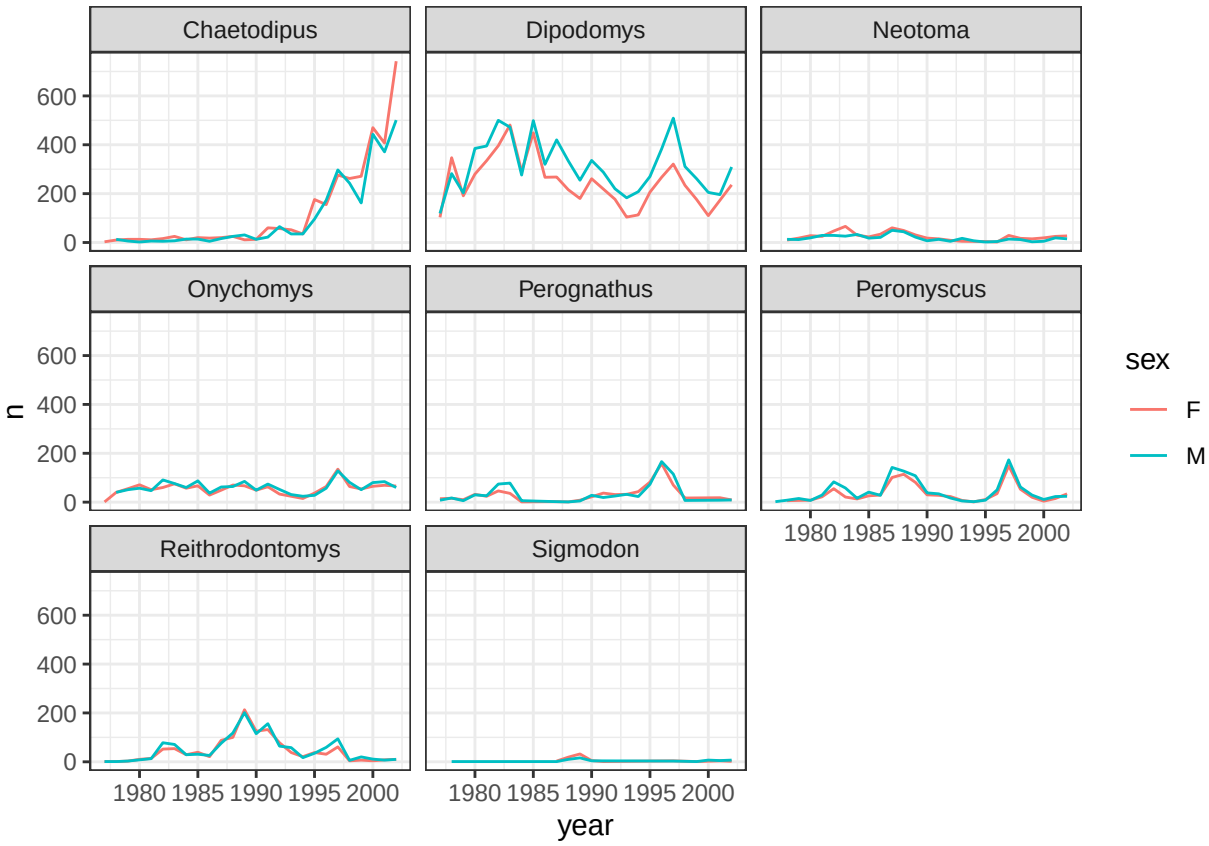


#

ggplot themes

Publication quality plots

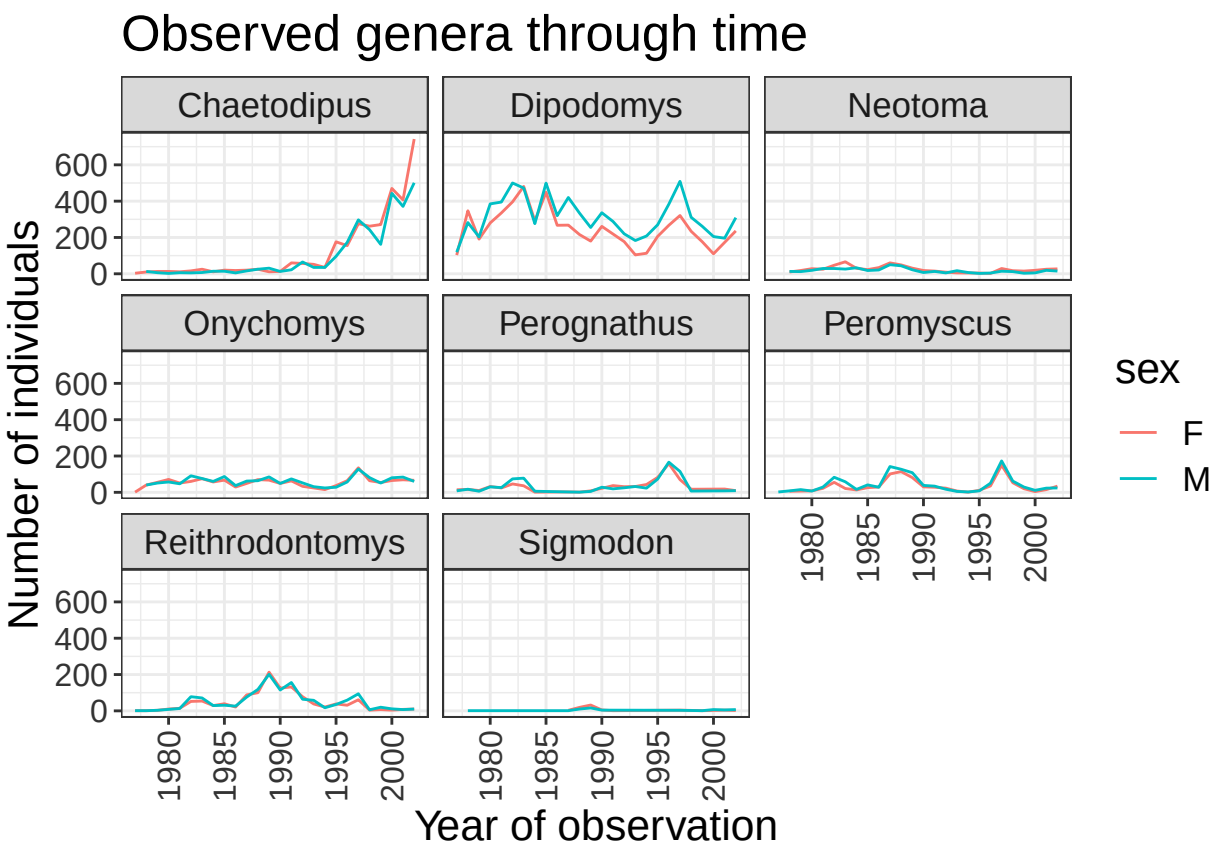
```
surveys_complete %>%
  count(year, genus, sex) %>%
 ggplot( mapping = aes(x = year, y = n, color = sex)) +
     geom_line() +
     facet_wrap(vars(genus)) +
     theme_bw()
```



#

Challenge- 15 mins Use what you just learned to create a plot that depicts how the average weight of each species changes through the years. Use ggplot cheat sheet and improve your plot, share some ideas with your group https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

```
surveys_complete %>%
  count(year, genus, sex) %>%
ggplot(mapping = aes(x = year, y = n, color = sex)) +
    geom_line() +
    facet_wrap(vars(genus)) +
    labs(title = "Observed genera through time",
        x = "Year of observation",
        y = "Number of individuals") +
    theme_bw() +
       theme(axis.text.x = element_text(colour = "grey20", size = 12, angle = 90,hjust = 0.5, vjust = 0
```

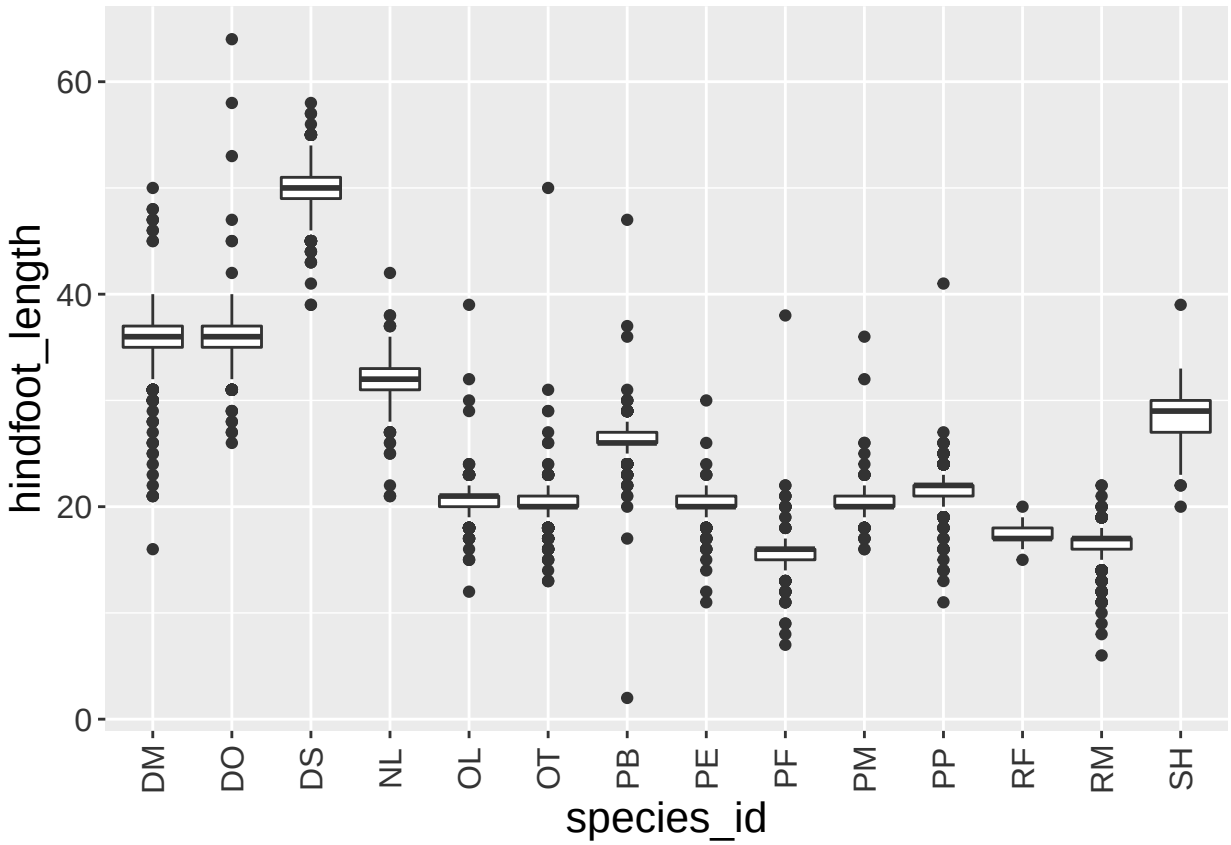


#

Save your themes

```
grey_theme <- theme(axis.text.x = element_text(colour="grey20", size = 12,
                                               angle = 90, hjust = 0.5,
                                               vjust = 0.5),
                    axis.text.y = element_text(colour = "grey20", size = 12),
                    text=element_text(size = 16))

ggplot(surveys_complete, aes(x = species_id, y = hindfoot_length)) +
    geom_boxplot() +
    grey_theme
```



Challenge -15 mins

With all of this information in hand, please take another five minutes to either improve one of the plots generated in this exercise or create a beautiful graph of your own. Use the RStudio ggplot2 cheat sheet for inspiration.

Here are some ideas:

See if you can change the thickness of the lines. Can you find a way to change the name of the legend? What about its labels? Try using a different color palette (see http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/).

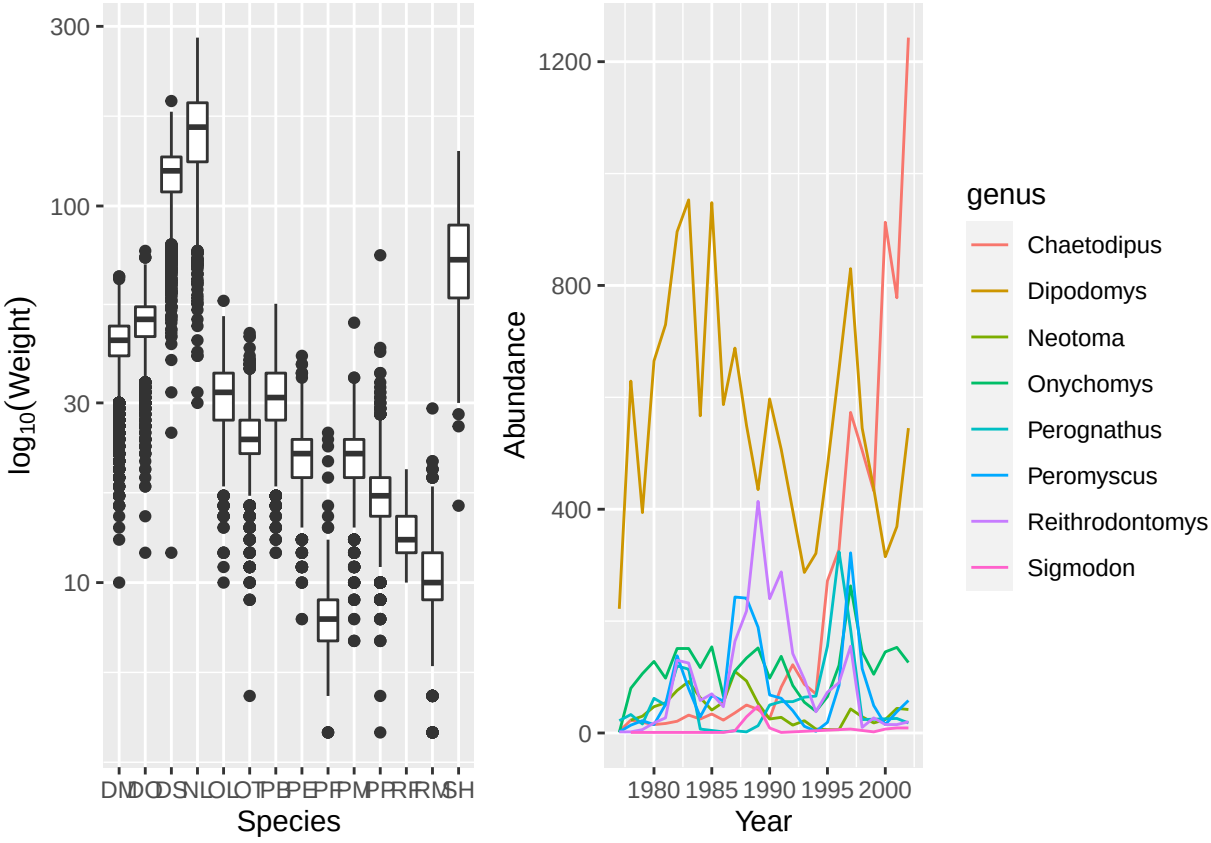
Exporting plots

```
library(gridExtra)

spp_weight_boxplot <- ggplot(data = surveys_complete,
                             aes(x = species_id, y = weight)) +
  geom_boxplot() +
  labs(x = "Species",
       y = expression(log[10](Weight))) +
  scale_y_log10() +
  labs()
```

```
spp_count_plot <- ggplot(data = yearly_counts,
                         aes(x = year, y = n, color = genus)) +
  geom_line() +
  labs(x = "Year", y = "Abundance")

comb_plot<-grid.arrange(spp_weight_boxplot, spp_count_plot, ncol = 2, widths = c(4, 6))
```




```
  ggsave("combo_plot.png",comb_plot,dpi=300)
```

```
## Saving 6.5 x 4.5 in image
```