Data Carpentry Workshop
University of Oklahoma Bizzell Library
Collaborative Learning Classroom (LL123)
May 16-17, 2018

Course website: https://oulib-swc.github.io/2018-05-16-ou-dc
This document: Link

Users are expected to follow our ***Code of Conduct***:
https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

--- Scroll to the bottom for FRESH notes --- Scroll to the bottom for FRESH notes ---

Please fill out the pre-class survey located on the course website.

Attendees

name / department or affiliation / twitter

- Matthew Rowe/Biology/non-tweeter
- Theo Acker/Computer Science
- Hawi Burka/ Industrial & Systems Engineering
- Tara Carlisle/OU Libraries/@oudigischolar.com
- Ryan Bond/OU Libraries/@ryan_bond
- David Hille / Biology / @davidchille
- Mehrnoush Nourbakhsh-Rey/ Biology/ non-twitter
- Eric Zemke / OU Libraries
- Steffani Silva/Industrial & Systems Engineering/ non-twitter
- Drew Hutchinson/Journalism, Political Science/@drewatou
- Mehrun Nisa
- Vladimir Garcia / Economics / @vladiroufakis
- David Durica/Biology/ddurica@ou.edu
- Jenna Messick/ Bio Survey & Plant Bio
- Roukayatou Ouedraogo/ Petroleum engineering
- Kyungwon Koh/SLIS/@kyungwonkoh
- Ravi Kumar Manjhi/ Environmental Science
- Joaquim Amorim / Petroleum Engineering

•

Instructors

- Mark Laufersweiler / OU Libraries / @laufers
- Claire Curry / Oklahoma Biological Survey
- Fred Reiss / OU Libraries

Helpers

- Joshua Hatzis/Geography
- Jim Ferguson / OU Supercomputing Center for Education & Research
- Katy Felkner/ Computer Science and Letters

Day One

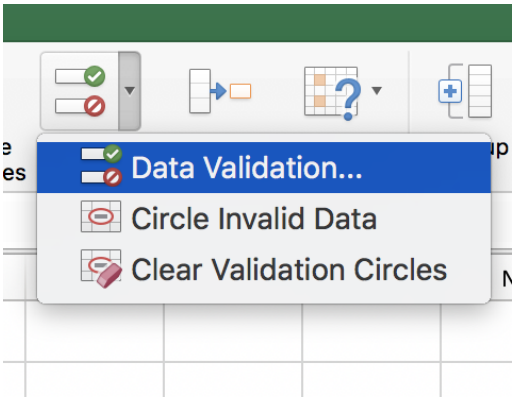
Spreadsheets

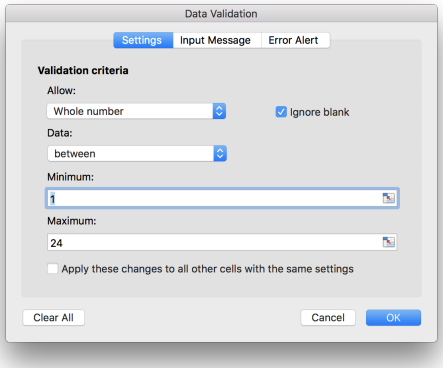
Data set for lesson: https://ndownloader.figshare.com/articles/1314459/versions/9

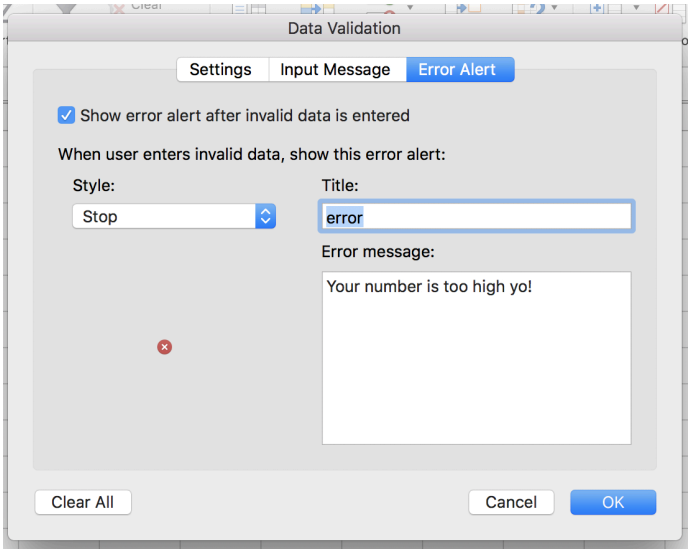
Dates as data - tips and tricks on handling dates
Some spreadsheet software output is not always compatible, so learning how to format data so it is machine readable.
We are not writing code.

When using spreadsheets: Keep notes and data on different sheets.
Formatting spreadsheets properly:
It is better to use underscore or camel case (camelCase) dash line instead of space because computer doesn't like space often times. (dash/hyphen will cause problems with SOME programs but not all, so is better than using a space, but under_score or camelCase preferred).
Variables in columns, each variable gets its own column.Don't put more than 1 variable in a column.
Always save raw data and conduct analyses or edits in another tab or spreadsheet.
In Excel data set for **missing data** the best option is leaving them **blank or put NA** for not available data. **Do not put 0 or any other number** because they may influence the calculation.
Try to make your data set as **clear** as possible for future use by other researchers for instance write "wind_speed" instead of "wind_spd" or "sex" instead of "m/f".
Always make a backup (add the exact date of back up in the file name) from your data before starting any manipulation on your file to organize your data.
To separate the elements of date(year, month and day) in excel we could use the formula "=year(the cell)> enter" and then command c," =month(cell)", "=day(cell)"

Extract the time of filling the data: “now()-today()”. Also we could extract the hour, minute, and second by “=hour(time cell)”, “=minute(time cell),”=second(time cell)”. You can update the time using F9.
If you want to put the date and hour in same cell you can use the format of “yymmddhhmmss”.
Excel>DataTab> “Data Validation” : You can define the variable that is allowed.







Be sure to write a helpful and descriptive error message, as shown above.

Always save your data under comma separated values (CSV). but it may cause problem if you have comma in your data set. Do not save your data under excel spreadsheets (xls, xlsx,…).

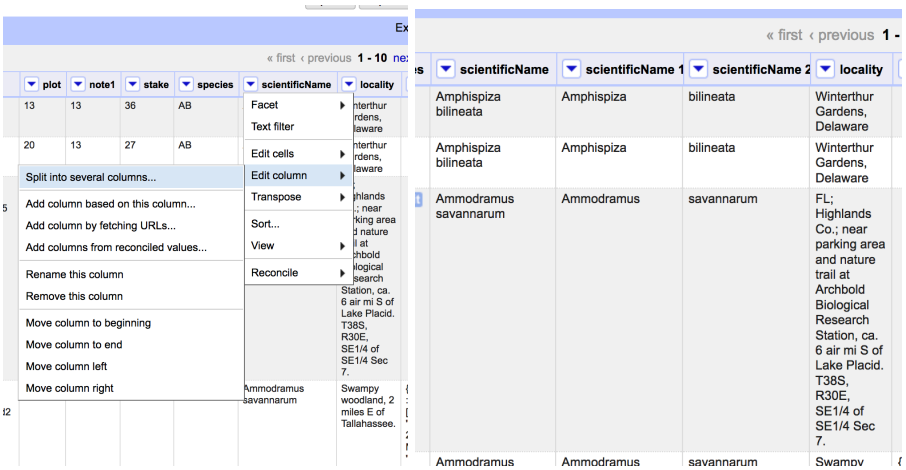
To be able to preserve and clean-up our raw data we could use openRefine software.

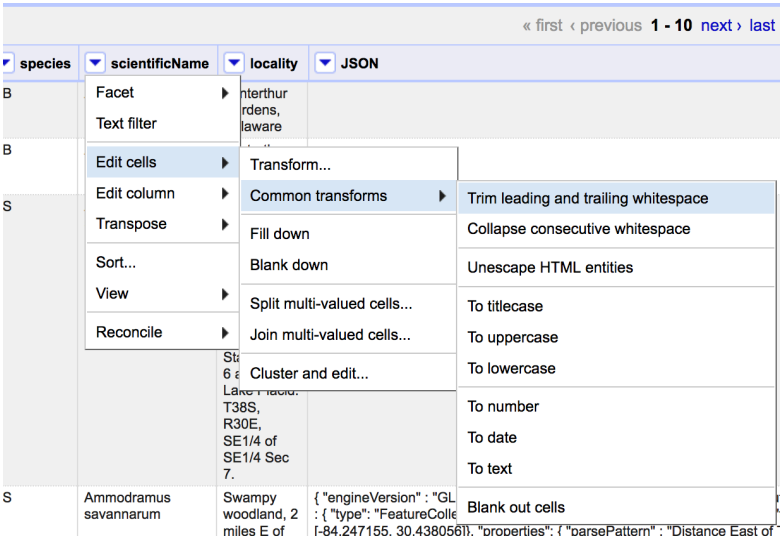
OpenRefine

go on facet>text Facet to see all the group of scientific name and number of each that existed.
Go to "edit cell> common transform> Trim leading and trailing whitespace" to edit the group with the same name together not one by one.
Go to "edit cell> common transform> To titlecase" convert the first letter of each word in the cell to the capital letter.
Go to "edit cell> common transform> To number" to recognize numbers as a number not character. and then we could sort them by open the array under the cell of interest.
Go to " edit column> sprit the column into several columns> choose the separator, the number of column you want sprit to, and choose not remove the column if you want to keep the original column".
Go to "text filter " if you want to filter the rows by a special name or character.
on the "text Facet "tab, if you click on each group you could edit the group instead of clicking on each name and edit those.

Clustering….

Split Columns





R

Notes

Code for download up on Claire's screen:
download.file("https://ndownloader.figshare.com/files/2292169", "data/portal_data_joined.csv")

Get version numbers easily: sessionInfo()

R is case sensitive

Task: try assigning these values and check to see how the variables change

```
mass <- 47.5
age <- 122
mass <- mass *2
age <- age - 20
mass_index <- mass/age

#on with the fun...with FUNCTIONS!

b<-sqrt(4) #get the square root of 4, assign to b
sqrt(b) #get the square root of the value currently assigned to b

round(x=3.14159)  #round off the supplied value
```

```
round(x=3.14159,digits = 2) #same rounding operation, but adds precision to two decimal places
round(digits=2,x=3.14159) #also works

weight_g <- c(50,60,65,82) #create vector with four numeric values

animals <- c("mouse","rat","dog") #create vector with alphanumeric values

length(weight_g) #get number of items in vector

class(weight_g) #determine types of data in vector

str(weight_g) #show values in vector

weight_g <- c(weight_g,90) #add value to vector



weight_g[weight_g > 50]
weight_g[c(TRUE, FALSE)] #logical indices repeat
weight_g[c(TRUE, FALSE, FALSE, TRUE)] #indices repeat
weight_g[weight_g < 30 | weight_g >50]
weight_g[weight_g >= 30 & weight_g == 21]
```

Data fil

```
#We've seen that atomic vectors can be of type character, numeric (or double), integer, and logical. But what happens if we try to mix these types in a single vector?
What will happen in each of these examples? (hint: use class() to check the data type of your objects):

num_char <- c(1, 2, 3, "a")
num_logical <- c(1, 2, 3, TRUE)
char_logical <- c("a", "b", "c", TRUE)
tricky <- c(1, 2, 3, 4)
class(tricky)
#Why do you think it happens?
```

Data file

```
download.file(url = "https://ndownloader.figshare.com/files/2292169",
              "data/portal_data_joined.csv")
```

Challenges

#We've seen that atomic vectors can be of type character, numeric (or double), integer, and logical. But what happens if we try to mix these types in a single vector?
What will happen in each of these examples? (hint: use class() to check the data type of your objects):

```
num_char <- c(1, 2, 3, "a")
num_logical <- c(1, 2, 3, TRUE)
char_logical <- c("a", "b", "c", TRUE)
tricky <- c(1, 2, 3, “4”)
class(tricky)
#Why do you think it happens?
```

Day Two

Attendee Sign in

- Tara Carlisle
- Jenna Messick
- Fred Reiss
- Theo Acker/CS
- Jim Ferguson
- Eric Zemke / OU Libraries
- Joshua Hatzis/Geography & Environmental Sustainability
- Ravi Kumar Manjhi/ Environmental science
- Matthew Rowe/Biology
- Ryan Bond / OU Libraries
- David Hille / Biology
- Steffani Silva
- David Durica/Biology
- Mehrun Nisa/civil engineering
- Joaquim Amorim / Petroleum Engineering
- Hawi Burka Kebede/Industrial & Systems Engineering
- Mike Malahy/ OU libraries
- Vladimir Garcia / Economics
- Mehrnoush Nourbakhsh/ Biology
- Kyungwon Koh/SLIS
- Mert Ajvaz
- Roukayatou Ouedraogo / Petroleum Engineering

R, continued

```
download.file(url = "https://ndownloader.figshare.com/files/2292169",
              "data/portal_data_joined.csv")


## load an external CSV to an R data frame
surveys = read.csv("data/portal_data_joined.csv")

## get summary info about the df
# get first few rows
head(surveys)

# get last few rows
tail(surveys)

# structure of the data (i.e. type of each variable)
```

```
str(surveys)

# numeric vector with dimensions (rows, columns)
dim(surveys)

# number of rows
nrow(surveys)

# number of columns
ncol(surveys)

# get a vector of column names
names(surveys)

# concise summary
summary(surveys)

download.file(url = "https://ndownloader.figshare.com/files/2292169",
              "data/portal_data_joined.csv")

surveys <- read.csv("data/portal_data_joined.csv")
head(surveys, 5)

str(surveys)

dim(surveys)
nrow(surveys)
ncol(surveys)

tail(surveys, 10)

names(surveys)  #gives column names
colnames(surveys) #does same thing as names() here
rownames(surveys) #gives rows, not columns

str(surveys)
summary(surveys)

surveys[1,2] #[row, column]
surveys[,2]
surveys[,6]
surveys[1:3,6]
surveys[1:6,] #equivalent to head(surveys)
head(surveys)
surveys[1] #no commas will get you a data frame and selects a column
```

```
surveys_no_record_id <- surveys[,-1] #- (minus sign/dash) excludes the
record
surveys_exclude_first_row <- surveys[-1,] # excludes first row
surveys[-c(1,7:34786),]

species_id <- surveys["species_id"] #result is a data.frame
species_id_vector <- surveys[, "species_id"] #result is a vector
surveys[["species_id"]]
surveys$species_id


surveys$date <- ymd(paste(surveys$year,
                  surveys$month,
                  surveys$day,
                  sep = "-"))
summary(surveys$date)

head(surveys[is.na(surveys$date), c("year", "month", "day")])

#get dataset for next section
surveys_complete <- surveys %>%
  filter(!is.na(weight),
         !is.na(hindfoot_length),
         !is.na(sex))

species_counts <- surveys_complete %>%
  group_by(species_id) %>%
  tally() %>%
  filter(n >= 50)



surveys_complete <- surveys_complete %>%
  filter(species_id %in% species_counts$species_id)

dim(surveys_complete)
#check that your results are: 30676    13


#create a folder called data_output in working directory

write_csv(surveys_complete,
          path = "data_output/surveys_complete.csv")



surveys_complete <- read.csv("data_output/surveys_complete.csv")
#read.csv if you dont' have tidyverse installed
```

```
library(ggplot2) #only if you haven't loaded tidyverse

#extra
survey_plot <- ggplot(data = surveys_complete,

                      aes(x = weight,
                          y = hindfoot_length))
survey_plot+geom_point(alpha = 0.1)

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

# create a scatterplot with the color based on plot ID with a color scale
# that is a gradient between red and blue
ggplot(data=surveys_complete,
       aes(x=weight,y=hindfoot_length)) +
  geom_point(alpha=0.1,aes(color=plot_id))+
  scale_color_continuous(low="red",high="blue")
```

Ggplot reference sheet:
https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

```
yearly_sex_weight <- surveys_complete %>%
  group_by(year, sex, species_id) %>%
  summarize(avg_weight = mean(weight))

ggplot(data = yearly_sex_weight,
       aes(x = year,
           y = avg_weight,
           color = species_id))+
  geom_line()+
  facet_grid(. ~ sex)
```

Full code from R from Claire's screen

```
1+1
2/2

dput(head(iris))

saveRDS(iris, file = "./data/test.rds")
readRDS(file = "/tmp/iris.rds")
```

```
sessionInfo()

#Set working directory
setwd("/media/Data/Documents/service/software_carpentry/R_data_carp
entry/20180516_DC_ou") #your working directory here

#assign an object

(weight_kg <- 55)

#change value
weight_kg <- 57.5
2*weight_kg
#
weight_lb <- 2.2*weight_kg
#
weight_kg <-100
#reassigning new value to weight
#I got this from stackoverflow
####################
#new section

mass <- 47.5
age <- 122
mass2 <- mass * 2.0
age <- age - 20
mass_index <- mass/age

####Functions
b <- sqrt(4)
sqrt(b)

round(
     digits = 2,
     x = 3.14159)

round ()

weight_g <- c(50, 60, 65, 82)

animals <- c("mouse", "rat", "dog")
length(weight_g)
length(animals)

class(weight_g)
class(animals)

str(weight_g)
```

```
str(animals)

weight_g <- c(weight_g, 90)
weight_g <- c(30, weight_g)


#Challenge

num_char <- c(1, 2, 3, "a")
num_logical <- c(1, 2, 3, TRUE)
char_logical <- c("a", "b", "c", TRUE)
tricky <- c(1, 2, 3, "4")
tricky_fixed <- as.numeric(num_char)
class(as.numeric(tricky))

animals <- c("mouse", "rat", "dog", "cat")
animals[2]
animals[c(3, 2, 2, 2, 2, 2, 4)]

weight_g <- c(21, 34, 39, 54, 55)
weight_g[c(TRUE, FALSE, FALSE, TRUE, FALSE)]
weight_g > 50

weight_g[weight_g > 50]
weight_g[c(TRUE, FALSE)]
weight_g[c(TRUE, FALSE, FALSE, TRUE)]
weight_g[weight_g < 30 | weight_g >50]
weight_g[weight_g >= 30 & weight_g == 21]

animals[animals == "cat" | animals == "rat"]

animals[animals %in% c("rat", "cat", "dog", "duck", "goat")]

heights <- c(2, 4, 4, NA, 6)
class(heights)
mean(heights, na.rm = TRUE)
max(heights, na.rm = TRUE)
median(heights, na.rm = TRUE)

heights[!is.na(heights)]
mean(heights[!is.na(heights)])

na.omit(heights)

heights[complete.cases(heights)]


download.file(url = "https://ndownloader.figshare.com/files/2292169",
```

```
            "data/portal_data_joined.csv")

surveys <- read.csv("data/portal_data_joined.csv")
head(surveys, 5)

str(surveys)

dim(surveys)
nrow(surveys)
ncol(surveys)

tail(surveys, 10)

names(surveys)  #gives column names
colnames(surveys) #does same thing as names() here
rownames(surveys) #gives rows, not columns

str(surveys)
summary(surveys)

surveys[1,2] #[row, column]
surveys[,2]
surveys[,6]
surveys[1:3,6]
surveys[1:6,] #equivalent to head(surveys)
head(surveys)
surveys[1] #no commas will get you a data frame and selects a column

surveys_no_record_id <- surveys[,-1] #- (minus sign/dash) excludes the
record
surveys_exclude_first_row <- surveys[-1,] # excludes first row
surveys[-c(1,7:34786),]

species_id <- surveys["species_id"] #result is a data.frame
species_id_vector <- surveys[, "species_id"] #result is a vector
surveys[["species_id"]]
surveys$species_id

surveys[nrow(surveys),]
class(nrow(surveys))
surveys_200 <- surveys[200,] #challenge answer 1
nrow(surveys)
surveys[34786,]
last_row <- surveys[nrow(surveys),] #2 and 3

surveys[nrow(surveys)/2,]
surveys[-c(7:nrow(surveys)),]
```

```
str(surveys)
sex <- factor(c("male", "female", "female", "male"))
levels(sex)
nlevels(sex)
sex <- factor(sex, levels = c("male", "female"))
levels(sex)
as.character(sex)
year <- factor(c(1999, 1988, 1899))
as.numeric(year) #WRONG!!
as.numeric(as.character(year)) #works
as.numeric(levels(year))[year] #most correct way

plot(surveys$sex)

sex <- surveys$sex
levels(sex)

levels(sex)[1] <- "missing"
levels(sex)[2] <- "female"
levels(sex)[3] <- "male"

levels(sex)

plot(sex)

surveys_demo <- read.csv("data/portal_data_joined.csv",
                         stringsAsFactors = FALSE)
str(surveys_demo)

surveys_demo$plot_type <- factor(surveys$plot_type)

library(lubridate)

my_date <- ymd("2018-05-16")
str(my_date)

surveys$date <- ymd(paste(surveys$year,
                          surveys$month,
                          surveys$day,
                          sep = "-"))
summary(surveys$date)

head(surveys[is.na(surveys$date), c("year", "month", "day")])


library(tidyverse)

surveys <- read_csv("data/portal_data_joined.csv")
```

```
str(surveys)

surveys2 <- select(surveys,
      plot_id,
      species_id,
      weight) #select columns
#base R equivalent
surveys[, c("plot_id",
          "species_id",
          "weight")]

filter(surveys,
      year == 1995) #filter rows
surveys[surveys$year==1995,] #base R equivalent

surveys_sml <- surveys %>%
  filter(weight < 5) %>%
  select (species_id, sex, weight)

weight <- surveys %>%
  filter(!is.na(weight))%>%
  mutate(weight_kg = weight/1000,
        weight_kg2 = weight_kg*2) %>%
  head()

str(weight)

mean(surveys$weight, na.rm = TRUE)

surveys %>%
  filter(!is.na(weight)) %>%
  group_by(species_id, sex)%>%
  summarize(mean_weight = mean(weight),
          sd_weight = sd(weight))
surveys %>%
  group_by(sex) %>%
  tally()

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

str(surveys_gw)

surveys_spread <- surveys_gw %>%
  spread(key = genus,
        value = mean_weight)
```

```
surveys_gathered <- surveys_spread %>%
  gather(key = this_is_our_new_genus_column,
         value = avg_weight,
         Baiomys:Spermophilus)

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

species_counts <- surveys_complete %>%
  group_by(species_id) %>%
  tally() %>%
  filter(n >= 50)

surveys_complete <- surveys_complete %>%
  filter(species_id %in% species_counts$species_id)
dim(surveys_complete)
write_csv(surveys_complete,
          path = "data_output/surveys_complete.csv")



surveys_complete <- read_csv("data_output/surveys_complete.csv")
library(ggplot2) #only if you haven't loaded tidyverse

#extra
survey_plot <- ggplot(data = surveys_complete,
                      aes(x = weight,
                          y = hindfoot_length))
survey_plot+geom_point(alpha = 0.1)









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

ggplot(data = surveys_complete,
```

```
      aes(x = species_id,
          y = weight))+
  geom_jitter(alpha = 0.3,
              color = "tomato")+
  geom_boxplot()

yearly_counts <- surveys_complete %>%
  group_by(year, species_id) %>%
  tally()
#original
ggplot(data = yearly_counts,
      aes(x = year,
          y = n,
          color = species_id)) +
  geom_line()
#extra (experiment with aes and geom)
ggplot(data = yearly_counts,
      aes(x = species_id,
          y = n,
          color = year)) +
  geom_point()

ggplot(data = yearly_counts,
      aes(x = year,
          y = n)) +
  geom_line()+
  facet_wrap(~species_id)

yearly_counts_by_sex <- surveys_complete %>%
  group_by(year, species_id, sex) %>%
  tally()
counts_plot <- ggplot(data = yearly_counts_by_sex,
      aes(x = year,
          y = n,
          color = sex)) +
  geom_line()+
  facet_wrap(~species_id)+
  theme_bw()+
  theme(panel.grid = element_blank())

yearly_sex_weight <- surveys_complete %>%
  group_by(year, sex, species_id) %>%
  summarize(avg_weight = mean(weight))

weight_plot <- ggplot(data = yearly_sex_weight,
      aes(x = year,
          y = avg_weight,
          color = species_id))+
```

```
  geom_line()+
  facet_grid(. ~ sex)

library(gridExtra)
grid.arrange(counts_plot,
             weight_plot,
             ncol = 2,
             widths = c(4,6))

ggsave("figures/my_test_plot.png",
       weight_plot,
       width = 10,
       height = 5)
```

Challenges

Challenge

1. Create a data.frame (surveys_200) containing only the observations from row 200 of the surveys dataset.
2. Notice how nrow() gave you the number of rows in a data.frame? Use that number to pull out just that last row in the data frame. Compare that with what you see as the last row using tail() to make sure it's meeting expectations.
3. Pull out that last row using nrow() instead of the row number. Create a new data frame object (surveys_last) from that last row.
4. Use nrow() to extract the row that is in the middle of the data frame. Store the content of this row in an object named surveys_middle.
5. Combine nrow() with the - notation above to reproduce the behavior of head(surveys) keeping just the first through 6th rows of the surveys dataset.

SQL

As a reminder, we will be using the DB Viewer for SQLite to demonstrate database/SQL functions. If you have not already done so, PLEASE download the application from http://sqlitebrowser.org/

Also, we will be working with the Portal Project data with which we worked during the Excel/OpenRefine and R sessions. Look for the

downloaded content, and have the plots, species, and surveys CSV files on standby, as well as the portal_mammals.sqlite file. If you cannot locate the portal project data, you can re-download it from https://figshare.com/articles/Portal_Project_Teaching_Database/1314459

Wrap up

Before you leave, please take a few minutes to complete the post-training survey at https://www.surveymonkey.com/r/dcpostworkshopassessment?workshop_id=2018-05-16-ou-dc

Last line

Before you leave,