Exercise 2: Tidy Data

Eric Kelchlin

March 2019

# Load the required packages

You can use requre() or library() to load a package.

require(knitr) # to knit our documents  
require(tidyverse) # to wrangle data  
require(readxl) # to import excel sheets  
require(kableExtra) # for nice tables  
require(here)

## Task 1.1 - Import and view the data

The easiest way to read-in Excel data is to use the read\_excel() function from the readxl package which is a happy member of the tidyverse family of packages.

Ibis <- read\_excel(here("Ibis\_Data.xlsx"))

What if we had a spreadsheet with a couple lines that we wanted to skip or perhaps a specific range of cells that we only wanted to read-in? Use the help on the read\_excel() function to see if it’s possible. Do this by simply clicking *F1* while your cursur is on the function name.

# View the Data

glimpse(Ibis)

## Observations: 6  
## Variables: 8  
## $ Site <chr> "Carson Lake", "Stillwater Res", "Tule Lake", "Mallard...  
## $ Lat <dbl> -118.7313, -118.6387, -118.7262, -118.5092, -118.5364,...  
## $ Long <dbl> 39.31216, 39.46690, 39.48407, 39.58643, 39.59010, 39.6...  
## $ Habitat <chr> "Hardstem Bullrush", "Hardstem Bullrush", "Alkali Bull...  
## $ `1972` <dbl> 0, 1000, 1300, 500, 20, 200  
## $ `1973` <dbl> 500, 400, 800, 900, 200, 0  
## $ `1974` <dbl> 0, 800, 1400, 2000, 6000, 1200  
## $ `1975` <dbl> 788, 900, 1500, 0, 5000, 300

names(Ibis)

## [1] "Site" "Lat" "Long" "Habitat" "1972" "1973" "1974"   
## [8] "1975"

head(Ibis, n = 5)

## # A tibble: 5 x 8  
## Site Lat Long Habitat `1972` `1973` `1974` `1975`  
## <chr> <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 Carson Lake -119. 39.3 Hardstem Bullrush 0 500 0 788  
## 2 Stillwater Res -119. 39.5 Hardstem Bullrush 1000 400 800 900  
## 3 Tule Lake -119. 39.5 Alkali Bullrush 1300 800 1400 1500  
## 4 Mallard Lake -119. 39.6 Hardstem Bullrush 500 900 2000 0  
## 5 Big Pool -119. 39.6 Alkali Bullrush 20 200 6000 5000

View(Ibis) # note the capital V

# Remove the eval = FALSE and run code chunk  
  
kable(Ibis, caption = "Table 1: Ibis Data in a Wide Format") %>%  
 kable\_styling(full\_width = FALSE, font\_size = 16)

# Task 1.2 - Convert the table from a wide to a long format

Use the pipe **%>%** symbol **(Ctrl + Shft + M)** to simplify coding.

### gather() function template

new object <- data %>% gather(key = , value = , columns to gather, other arguments)

# Remove the eval = FALSE and run code chunk  
  
Ibis2 <- Ibis %>%   
 gather(key = MyYear, value = MyCount, `1972`:`1975`, convert = TRUE)  
  
# What does the convert = TRUE argument do?  
  
# Knit the table in a nice format. Notice how you can nest functions with the pipe.  
  
Ibis2 %>%   
 head(n=10) %>%   
 kable(caption="Table 2: Ibis Data in a Long Format") %>%  
 kable\_styling(full\_width = T, bootstrap\_options = c("hover", "condensed"))

Table 2: Ibis Data in a Long Format

Site

Lat

Long

Habitat

MyYear

MyCount

Carson Lake

-118.7313

39.31216

Hardstem Bullrush

1972

0

Stillwater Res

-118.6387

39.46690

Hardstem Bullrush

1972

1000

Tule Lake

-118.7262

39.48407

Alkali Bullrush

1972

1300

Mallard Lake

-118.5092

39.58643

Hardstem Bullrush

1972

500

Big Pool

-118.5364

39.59010

Alkali Bullrush

1972

20

Little Pool

-118.5172

39.61178

Alkali Bullrush

1972

200

Carson Lake

-118.7313

39.31216

Hardstem Bullrush

1973

500

Stillwater Res

-118.6387

39.46690

Hardstem Bullrush

1973

400

Tule Lake

-118.7262

39.48407

Alkali Bullrush

1973

800

Mallard Lake

-118.5092

39.58643

Hardstem Bullrush

1973

900

## Task 1.3 - There are two other ways to identify the the fields to gather in this example. Can you figure this out? Tip: one way is to identify what not to gather and the other identifies the columns by location.

Ibis3 <- Ibis %>%   
 gather(key = MyYear, value = MyCount, -(1:4), convert = TRUE) # or use names of fields  
  
Ibis4 <- Ibis %>%   
 gather(key = MyYear, value = MyCount, 5:8, convert = TRUE)

## Task 1.3 - The spread() function restructures the long format to a wide format. Use this function to return our gathered dataset back into the orignal format. This one is easier than the gather function.

### spread() template

newobject <- data %>% spread(key = column to pivot, value = column with the values)

IbisWide <- Ibis2 %>% spread(MyYear, MyCount)

## Task 1.5 - Export the Data. Run the line below to easily export the data as a .csv file. Find the write\_? function from the readr package to export the data as a comma delimited (" “) .txt file.

write\_csv(Ibis2, here("IbisClean.csv"))  
  
write\_delim(Ibis2, here("IbisClean.txt"))

## Task 1.6 - Independent Exercise

1. Use the read\_csv() function to import the *Sage Grouse Data.csv* dataset. Assign the data into a new object called *Sage*
2. Examine the data. How many Observations and variables do we have? What are the variable names? What data type are the 4 date fields and what should they be?
3. We have multiple observational units combined in one table. How many do you think we have?
4. Some variables are in wide format (i.e., observations), can you pick them out?

Sage <- read\_csv(here("Sage Grouse Data.csv"))

## Parsed with column specification:  
## cols(  
## .default = col\_character(),  
## FID = col\_double(),  
## Observatio = col\_double(),  
## LATITUDE = col\_double(),  
## LONGITUDE = col\_double()  
## )

## See spec(...) for full column specifications.

head(Sage, n = 5)

## # A tibble: 5 x 24  
## FID Observatio Common\_Nam Scientific Species\_Of OBSERVAT\_3 TRS COUNTY  
## <dbl> <dbl> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 0 2 Greater S~ Centrocoe~ Concern Crow Cons~ T4S ~ Bigho~  
## 2 1 3 Greater S~ Centrocoe~ Concern Crow Cons~ T2S ~ Yello~  
## 3 2 2 Greater S~ Centrocoe~ Concern Crow Cons~ T3S ~ Yello~  
## 4 3 2 Greater S~ Centrocoe~ Concern Crow Cons~ T4S ~ Yello~  
## 5 4 4 Greater S~ Centrocoe~ Concern Crow Cons~ T3S ~ Bigho~  
## # ... with 16 more variables: LOCATION\_C <chr>, LATITUDE <dbl>,  
## # LONGITUDE <dbl>, Field\_Date1 <chr>, Field\_Date2 <chr>,  
## # Field\_Date3 <chr>, Field\_Date4 <chr>, FieldResults1 <chr>,  
## # FieldResults2 <chr>, FieldResults3 <chr>, FeildResults4 <chr>,  
## # Kingdom <chr>, Phylum <chr>, Class <chr>, Family <chr>, Order\_ <chr>

#glimpse(Sage)

# Task 1.7 - Class Exercise

Question: What is the maximum count of male and female sage grouse per lek?

To answer this question, we’ll need to use the select(), group\_by() and summarise()functions from the **dplyr** package. This is a great introduction of what we will be learning more of tomorrow.

## Steps:

1. Select only the variables you need using the select() function.
2. Gather the *FieldResults* fields into a long format using the gather() function. Use the na.rm = TRUE argument to remove the cells with missing values.
3. Seperate the field with the data using the seperate() function
4. Use the group\_by() and summarise()functions to calculate the maximum number of sage grouse per lek. Note: summarise() is finiky to missing values, that’s why we need to filter them out first.

Sage1 <- Sage %>%   
 select(LOCATION\_C, FieldResults1:FeildResults4) %>%   
 gather(key = Results, value = SexCount, -LOCATION\_C, na.rm = TRUE) %>%   
 separate(SexCount, c("No\_Males", "MSex", "No\_Females", "FSex"), sep = " ") %>%   
 select(LOCATION\_C, No\_Males, No\_Females)  
  
Sage2 <- Sage1 %>%   
 group\_by(LOCATION\_C) %>%   
 summarise(max\_count\_male = max(No\_Males), max\_count\_female = max(No\_Females))