STA130 W3

Monday, October 28, 2019 11:39 PM

Tidy data

1. Each variable must have its **own column**.
2. Each observation must have its **own row**.
3. Each value must have its **own cell**.

A general rule of thumb:

It is easier to describe functional relationships **between variables** (e.g., z is a linear combination of x and y, density is the ratio of weight to volume) than **between rows**.
It is easier to make **comparisons** between groups of **observations** (e.g., average of group A vs. average of group B) than between **groups of columns**.

Not tidy:

Colour	N
Brown	97
Green / blue / gray	9
Other	23

```
brown_eyes            <- rep("Brown", times = 97);
green_blue_gray_eyes  <- rep("Green/blue/gray", times = 9)
other_colour_eyes     <- rep("Other", times = 23)
eye_colour <- c(brown_eyes, green_blue_gray_eyes, other_colour_eyes)
eye_data <- data_frame(ID = 1:129, eye_colour = eye_colour)
```

*to create a data frame

This is tidy!	**Summary**
head(eye_data, n=5) ## # A tibble: 5 x 2 ## ID eye_colour ## <int> <chr> ## 1 1 Brown ## 2 2 Brown ## 3 3 Brown ## 4 4 Brown ## 5 5 Brown	eye_data %>% group_by(eye_colour) %>% summarise(n=n()) ## # A tibble: 3 x 2 ## eye_colour n ## <chr> <int> ## 1 Brown 97 ## 2 Green/blue/gray 9 ## 3 Other 23

Data wrangling allows us to transform data frames to make them more useful to answer interesting questions.

The ***ggplot*** library implements a **grammar of graphics**.
Similarly the ***dplyr*** library presents a **grammar for data wrangling**.

The **pipe operator** (%>%) is used to perform **an action on a dataframe**.
Glimpse the college_recent_grads data frame:
college_recent_grads %>% glimpse()

Select variables/columns using select():

Focus only on a few **variables** in the frame: major, major_category, # of male graduates (men), # of female graduates (women), and the median salary (median)

We use the **select()** function from **dplyr** to extract a dataframe with only these **variables**

```
college_recent_grads %>%
  select(major, major_category, men, women, median)
```

Select observations/rows using filter():

Extract only **observations** degrees in Computer Science and Mathematics

college_recent_grads %>% **filter**(major_category == "Computers & Mathematics")	*Only keep rows where it is TRUE **!=** : filtered out C & M

Combining select() and filter():

Extract certain subsets of the data, and save the new data frame as an R object by giving it a name. *give the new data frame a name to save it

```
CS_Math_grads <- college_recent_grads %>%
  select(major, major_category, men, women, median) %>%
  filter(major_category == "Computers & Mathematics")
```

We can use **Logicals** to write conditions on the variables in filter() to extract only the observations/rows **where the condition is true**.

Create new variables using mutate():

For each of the majors where the median earnings is at least $60,000, what percentage of the graduates are women?

college_recent_grads %>% filter(median >= 60000) %>% select(major, men, women) %>% **mutate**(total = men + women, pct_female = round((women / total)*100, 2))	total: variable for total # of graduates pct_female: % of female graduates

*calculate the percentage of women grads for each program

*will have new variables and new column for each variable

Sort observations based on new or existing variables using arrange():

```
college_recent_grads %>%
   filter(median >= 60000) %>%
   select(major, men, women) %>%
   mutate(total = men + women,
        pct_female = round((women / total)*100, 2)) %>%
  arrange(pct_female) Sort values in a column, from smallest to largest
```

If sorting words, it will be in alphabetical order

`arrange(desc(pct_female))` sorts in descending order

Create new variables from existing variables using mutate() and ifelse():

Create a categorical variable to identify majors with approximately equal numbers of male and female graduates: majors with between 45% and 55% female graduates.

Use **ifelse()** in a mutate() statement.

The format:

`ifelse(test condition (logical), yes, no)`

```
percent <- c(40, 47, 55, 58);
gender_balance <- ifelse(percent >= 45 & percent <= 55, yes = "YES",
"NO");
data_frame(percent, gender_balance)
## # A tibble: 4 x 2
##   percent gender_balance
##     <dbl> <chr>
## 1      40 NO
## 2      47 YES
## 3      55 YES
## 4      58 NO
my_college_dat <- college_recent_grads %>%
   select(major, men, women, median) %>%
   mutate(total = men + women,
         pct_female = round((women / total)*100, 2),
         gender_balanced = ifelse(pct_female >= 45 & pct_female <=
55, yes="Yes", no="No"))
```

Rename variables using rename():

Underscores are preferred over periods in variable names (but both work)

We can use **rename()** to change the name of gender.balanced to gender_balanced.

rename([**NEW** VARIABLE NAME] = [**OLD** VARIABLE NAME])

```
my_college_dat <- my_college_dat %>%
   rename(median salary = median)
glimpse(my_college_dat)
```

`head(n=3)` only show the first 3 observation

Missing values (coded as NA) in R:

The $ notation is used to refer to **a specific variable** in a data frame

```
college_recent_grads$women
```

is.na() to create a vector indicating TRUE where there is an NA and FALSE otherwise

```
is.na(college_recent_grads$women)
```

Removing NAs from calculations:

1. Using **na.rm=TRUE** within the mean function removes the NA observations from the mean calculation

```
college_recent_grads %>%  summarise(femgrad_mean = mean(women, na.rm=TRUE), N=n())
```

2. Filtering just the observations that are **not NA** before doing the calculations

```
college_recent_grads %>%
 filter(!is.na(women)) %>%
 summarise(femgrad_mean = mean(women), N=n())
```

Modify the gender_balanced variable with **three categories** (labelled "Mostly Men", "Mostly Women", and "Balanced") and compare means:

```
my_college_dat <- my_college_dat %>%
  mutate(gender_balanced = ifelse(pct_female < 45,
                                 yes="Mostly Men", no=gender_balanced),
             gender_balanced = ifelse(pct_female > 55,
                                 yes="Mostly Women", no=gender_balanced),
            gender_balanced = ifelse(pct_female >= 45 & pct_female <= 55,
                                 yes="Balanced", no=gender_balanced))
```