

Stat 331 Midterm

Working with Data in the Tidyverse:

Dataset %>%

filter(is.na(col))

- + This finds all the values where it is na in that specific col
- + Always check to make sure your variable names are spelled correctly

Filter and skim

bakeoff %>%

filter(!is.na(us_season)) %>%

skim()

- + Don't need to put skim(us_season)

Bakers %>%

count(airedus, series) %>%

mutate(proportion_bakers = n/sum(n))

- + Gives you proportion of bakers in each series for the show

Bakers %>%

count(airedus, series) %>%

count(airedus)

- + This will give you number of series that aired in the US and number of series that did not
- + If you didn't add the last count it would just go through each season and tell you which one was aired in the US or anywhere else.

Bakers %>%

count(result == "SB")

- + Gives counts for both not SB and SB

parse_date("17 August 2010", format = "%d %B %Y")

- + If there were dashes add them "%d - %B - %Y"

bakers_by_series %>%

count(baker, sort=TRUE)

- + This will get the baker that appear on the most series.
- + Def need to know sort = TRUE

ggplot(bakeoff, aes(x=episode)) +

geom_bar() +

facet_wrap(~series)

- + Num of had a bar plot for each series and num of shows idk what it was counting exactly

desserts %>%

arrange(desc(uk_airdate))

- + Print by descending uk_airdate or i think (-) before works as well

desserts <- read_csv("desserts.csv",

na = c("", "NA", "N/A"),

col_types = cols(technical = col_number(),

uk_airdate = col_date(format = "%d %B %Y"),

result = col_factor(levels = NULL)

```

    ))
+   I guess it casts result as a factor
desserts_2 <- desserts %>%
  mutate(nut = recode(nut, "filbert" = "hazelnut",
    "no nut" = NA_character_,
    .default = "Other nut" ))
+   Dont forget NA_character_

```

Recode channel as factor: bbc (1) or not (0)

```

ratings <- ratings %>%
  mutate(bbc = recode_factor(channel,
    "Channel 4" = 0,
    .default = 1))

```

Select to look at variables to plot next

```

ratings %>%
  select(series, channel, bbc, viewer_growth)

```

Make a filled bar chart

```

ggplot(ratings, aes(x = series, y = viewer_growth, fill = bbc)) +
  geom_col()

```

Drop 7- and 28-day episode ratings

```

ratings %>%
  select(-ends_with("day"))

```

Move channel to front and drop 7-/28-day episode ratings

```

ratings %>%
  select(channel, everything(), -ends_with("day"))

```

Adapt code to drop 28-day columns; keep 7-day in front

```

viewers_7day <- ratings %>%
  select(viewers_7day_ = ends_with("7day"),
    everything(),
    -ends_with("28day"))

```

```

tidy_ratings <- ratings %>%

```

Gather and convert episode to factor every col but series

```

  gather(key = "episode", value = "viewers_7day", -series,
    factor_key = TRUE, na.rm = TRUE)

```

```

tidy_ratings <- ratings %>%

```

Gather and convert episode to factor

```

gather(key = "episode", value = "viewers_7day", -series,
factor_key = TRUE, na.rm = TRUE) %>%
  # Sort in ascending order by series and episode
  arrange(series, episode) %>%
  # Create new variable using row_number()
  mutate(episode_count = row_number())

```

Plot viewers by episode and series

```

ggplot(tidy_ratings, aes(x = episode_count,
  y = viewers_7day,
  fill = series)) +
  geom_col()

```

Question: **Make a line plot to visualize the 7-day viewers by episode, faceted by series.** Since the x-axis is a factor, you'll use the group aesthetic to plot one line per series

```

week_ratings <- ratings2 %>%
  # Select 7-day viewer ratings
  select(series, ends_with("7day")) %>%
  # Gather 7-day viewers by episode
  gather(episode, viewers_7day, ends_with("7day"), na.rm = TRUE, factor_key = TRUE)

```

Plot 7-day viewers by episode and series

```

ggplot(week_ratings, aes(x = episode,
  y = viewers_7day,
  group = series)) +
  geom_line() +
  facet_wrap(~series)

```

Edit your code for the faceted line plot from the previous exercise to also color the lines by series. Using `guides()`, remove the `color` guide by setting it equal to `FALSE`, and add `theme_minimal()`.

Create week_ratings

```

week_ratings <- ratings2 %>%
  select(series, ends_with("7day")) %>%
  gather(episode, viewers_7day, ends_with("7day"),
    na.rm = TRUE) %>%
  separate(episode, into = "episode", extra = "drop") %>%
  mutate(episode = parse_number(episode))

```

Edit your code to color by series and add a theme

```
ggplot(week_ratings, aes(x = episode, y = viewers_7day,  
                        group = series, color = series)) +  
  geom_line() +  
  facet_wrap(~series) +  
  guides(color = FALSE) +  
  theme_minimal()
```

```
ratings3 <- ratings2 %>%  
  # Unite and change the separator  
  unite(viewers_7day, viewers_millions, viewers_decimal, sep = "") %>%  
  # Adapt to cast viewers as a number  
  mutate(viewers_7day = parse_number(viewers_7day))
```

```
# Print to view  
ratings3
```

Add a line after the %>% to reshape the output of count such that you have 3 columns: series, days_7, and days_28. To do this within `spread()`, set `sep = "_"` as an argument.

```
# Create tidy data with 7- and 28-day viewers  
tidy_ratings_all <- ratings2 %>%  
  gather(episode, viewers, ends_with("day"), na.rm = TRUE) %>%  
  separate(episode, into = c("episode", "days")) %>%  
  mutate(episode = parse_number(episode),  
         days = parse_number(days))
```

```
tidy_ratings_all %>%  
  # Count viewers by series and days  
  count(series, days, wt = viewers) %>%  
  # Adapt to spread counted values  
  spread(days, n, sep = "_")
```

Fill in the blanks in the provided code to filter rows that contain viewer data for each series' premiere and finale episodes- don't forget to undo the grouping when you are done

```
# Fill in blanks to get premiere/finale data
tidy_ratings <- ratings %>%
  gather(episode, viewers, -series, na.rm = TRUE) %>%
  mutate(episode = parse_number(episode)) %>%
  group_by(series) %>%
  filter(episode == 1 | episode == max(episode)) %>%
  ungroup()

-----

# Recode first/last episodes
first_last <- tidy_ratings %>%
  mutate(episode = recode(episode, `1` = "first", .default = "last"))

# Switch the variables mapping x-axis and color
ggplot(first_last, aes(x = episode, y = viewers, color = series)) +
  geom_point() + # keep
  geom_line(aes(group = series)) + # keep
  coord_flip() # keep

-----
```

Fill in the provided code to make a bar chart with one bar per series, where the height of the bar shows the bump in 7-day viewers from the premiere to the finale episodes. Do any series stand out?

```
# Calculate relative increase in viewers
bump_by_series <- first_last %>%
  spread(episode, viewers) %>%
  mutate(bump = (last - first) / first)

# Fill in to make bar chart of bumps by series
ggplot(bump_by_series, aes(x = series, y = bump)) +
  geom_col()+
  scale_y_continuous(labels = scales::percent) # converts to %

-----
```

```
# Create skill variable with 3 levels
```

```
bakers_skill <- bakers %>%  
  mutate(skill = case_when(  
    star_baker > technical_winner ~ "super_star",  
    star_baker < technical_winner ~ "high_tech",  
    TRUE ~ "well_rounded"  
  ))
```

```
# Filter zeroes to examine skill variable
```

```
bakers_skill %>%  
  filter(star_baker == 0 & technical_winner == 0) %>%  
  count(skill)
```

```
# Edit to reverse x-axis order
```

```
ggplot(bakers, aes(x = skill, fill = series_winner)) +  
  geom_bar()  
+   color= won't get it right always especially when want to split up the bars
```

```
# Add a line to extract labeled month
```

```
baker_dates_cast <- baker_dates %>%  
  mutate(last_date_appeared_us = dmy(last_date_appeared_us),  
         last_month_us = month(last_date_appeared_us, label = TRUE))  
Label = TRUE allows "Jan", "Feb"
```

```
# Make bar chart by last month
```

```
ggplot(baker_dates_cast, aes(x=last_month_us)) + geom_bar()
```

```
# Add a line to create whole months on air variable
```

```
baker_time <- baker_time %>%  
  mutate(time_on_air = interval(first_date_appeared_uk, last_date_appeared_uk),  
         weeks_on_air = time_on_air / weeks(1),  
         months_on_air = time_on_air %/% months(1))
```

```
# Add another mutate to replace "THIRD PLACE" with "RUNNER UP" and count
bakers <- bakers %>%
  mutate(position_reached = str_to_upper(position_reached),
         position_reached = str_replace(position_reached, "-", " "),
         position_reached = str_replace(position_reached, "THIRD PLACE", "RUNNER UP"))
```

```
# Count rows
bakers %>%
  count(position_reached)
```

Summarize by country: by_country

```
by_country <- votes_processed %>%
  group_by(country) %>%
  summarize(total = n(),
            percent_yes = mean(vote == 1))
```

Sort in ascending order of percent_yes

```
by_country %>%
  arrange(percent_yes)
```

Now sort in descending order

```
by_country %>%
  arrange(-percent_yes)
```

Load the ggplot2 package

```
library(ggplot2)
```

Create line plot

```
ggplot(by_year, aes(x=year, y=percent_yes)) + geom_line()
```

Change to scatter plot and add smoothing curve

```
ggplot(by_year, aes(year, percent_yes)) + geom_point() + geom_smooth()
```

```
# Start with by_year_country dataset
by_year_country <- votes_processed %>%
  group_by(year, country) %>%
  summarize(total = n(),
             percent_yes = mean(vote == 1))

# Print by_year_country
by_year_country

# Create a filtered version: UK_by_year
UK_by_year <- by_year_country %>%
  filter(country == "United Kingdom" )
# Line plot of percent_yes over time for UK only
ggplot(UK_by_year, aes(x=year,y=percent_yes)) + geom_line()
```

Vector of four countries to examine

```
countries <- c("United States", "United Kingdom",
               "France", "India")
```

Filter by_year_country: filtered_4_countries

```
filtered_4_countries <- by_year_country %>%
  filter(country %in% countries)
```

Line plot of % yes in four countries

```
ggplot(filtered_4_countries, aes(x=year,y=percent_yes,color=country)) +
  geom_line()
```

Line plot of % yes over time faceted by country

```
ggplot(filtered_6_countries, aes(x=year,y=percent_yes)) +
  geom_line() +
  facet_wrap(~ country)
# if we want to scale freely
facet_wrap(~ country , scales = "free_y")
```

```
US_fit <- lm(percent_yes ~ year, data = US_by_year)
library(broom)
```

Call the tidy() function on the US_fit object


```

tidy(US_fit)
# Linear regression of percent_yes by year for US
US_by_year <- by_year_country %>%
  filter(country == "United States")
US_fit <- lm(percent_yes ~ year, US_by_year)

# Fit model for the United Kingdom
UK_by_year <- by_year_country %>%
  filter(country == "United Kingdom")
UK_fit <- lm(percent_yes ~ year, UK_by_year)

# Create US_tidied and UK_tidied
US_tidied <- tidy(US_fit)
UK_tidied <- tidy(UK_fit)
# Combine the two tidied models
bind_rows(US_tidied,UK_tidied)
-----

# All countries are nested besides country
nested <- by_year_country %>%
  nest(-country)

# Print the nested data for Brazil
nested$data[[7]]
-----

# Load tidyr and purrr
library(tidyr)
library(purrr)
# Perform a linear regression on each item in the data column
by_year_country %>%
  nest(-country) %>%
  mutate(model = map(data, ~ lm(percent_yes ~ year, .)))
# Add another mutate that applies tidy() to each model
mutate(tidied = map(model,tidy))
# Add one more step that unnests the tidied column
unnest(tidied)
-----

```

```

# Filter by adjusted p-values
filtered_countries <- country_coefficients %>%
  filter(term == "year") %>%
  mutate(p.adjusted = p.adjust(p.value)) %>%
  filter(p.adjusted < .05)
glimpse(filtered_countries)
# Sort for the countries increasing most quickly
filtered_countries %>%
  arrange(estimate)
-----

# Sort for the countries decreasing most quickly
filtered_countries %>%
  arrange(-estimate)

# Load dplyr package
library(dplyr)

# Print the votes_processed dataset
votes_processed

# Print the descriptions dataset
descriptions
# Join them together based on the "rcid" and "session" columns
votes_joined <- votes_processed %>%
  inner_join(descriptions, by = c("rcid", "session"))
-----

# Filter, then summarize by year: US_co_by_year
US_co_by_year <- votes_joined %>%
  group_by(year) %>%
  filter(co == 1, country == "United States") %>%
  summarize(percent_yes = mean(vote == 1))
# Graph the % of "yes" votes over time
ggplot(US_co_by_year, aes(x=year,y=percent_yes)) +
  geom_line()
-----

library(tidyr)
# Gather the six me/nu/di/hr/co/ec columns
votes_joined %>%
  gather(topic,has_topic,me:ec)
# Perform gather again, then filter
votes_gathered <- votes_joined %>%
  gather(topic,has_topic,me:ec) %>% filter(has_topic == 1)
-----

```

```
# Create vanuatu_by_country_year_topic
vanuatu_by_country_year_topic <- by_country_year_topic %>%
  filter(country == "Vanuatu")

# Plot of percentage "yes" over time, faceted by topic
ggplot(vanuatu_by_country_year_topic, aes(year, percent_yes)) +
  geom_line() +
  facet_wrap(~ topic)
```

Spread Example

| | StudentName | Class | Grade |
|----|-------------|---------|-------|
| 1 | Isla | Math | A- |
| 2 | Isla | Science | B |
| 3 | Isla | History | B+ |
| 4 | Ronan | Math | C |
| 5 | Ronan | Science | A |
| 6 | Ronan | History | A+ |
| 7 | Theodore | Math | D |
| 8 | Theodore | Science | B |
| 9 | Theodore | History | A |
| 10 | Charlotte | Math | B- |
| 11 | Charlotte | Science | B+ |
| 12 | Charlotte | History | A- |
| 13 | Audrey | Math | C+ |
| 14 | Audrey | Science | B |
| 15 | Audrey | History | A+ |

The screenshot shows the R code for the `spread` function with three red callout boxes explaining its arguments:

- name of data set:** Points to `classData` in the `spread(classData, class, grade)` line.
- column that contains values to spread against:** Points to `class` in the same line.
- column you want to spread:** Points to `grade` in the same line.

```
> library(tidyverse)
> ?tidyr
> spread(classData, class, grade)
```

Gather Example

ClassData

Wide Format

| | StudentName | History | Math | Science |
|---|-------------|---------|------|---------|
| 1 | Audrey | A+ | C+ | B |
| 2 | Charlotte | A- | B- | B+ |
| 3 | Isla | B+ | A- | B |
| 4 | Ronan | A+ | C | A |
| 5 | Theodore | A | D | B |

Showing 1 to 5 of 5 entries

Console

Terminal

column name to gather values into

Name of Dataset

```
> longExample <- spread(ClassData, class, grade)
> view(longExample)
> gather(longExample, class, grade, 2:4)
```

column name to gather selected columns into

| | StudentName | class | grade |
|----|-------------|---------|-------|
| 1 | Audrey | History | A+ |
| 2 | Charlotte | History | A- |
| 3 | Isla | History | B+ |
| 4 | Ronan | History | A+ |
| 5 | Theodore | History | A |
| 6 | Audrey | Math | C+ |
| 7 | Charlotte | Math | B- |
| 8 | Isla | Math | A- |
| 9 | Ronan | Math | C |
| 10 | Theodore | Math | D |
| 11 | Audrey | Science | B |
| 12 | Charlotte | Science | B+ |
| 13 | Isla | Science | B |
| 14 | Ronan | Science | A |
| 15 | Theodore | Science | B |