Week Two, Class One

ML

1/18/2022

Set up

(Note: Need a space between the hashtags and the text for headers to knit properly)

Load the packages you might have used to complete Assignment One

```
library(tidyverse)
library(scales)
library(RColorBrewer)
library(ggthemes)
```

Note the additional options added to the first line of the code chunk above. They suppress any warnings, errors, and messages accompanying whatever code is run. (Loading the tidyverse packages creates a lot of output, for example.) You'll still that output in this notebook but it won't be in the knitted file.

There are other options to suppress or modify how chunks knit:

- If you want the knitted file to run the chunk and show the output but not the code, use `echo = FALSE`. That is helpful for final versions of papers or reports.
- If you want the knitted file to show the code but not run the output, use `eval = FALSE`. That is helpful at early stages of a project when there are chunks you want to see in the notebook but that create problems when knitting.
- If you want the knitted file to show neither the code nor run the output, use `include = FALSE`. That is helpful for early versions of papers or reports when you are unsure what output will be in a final version.

In the chunk below, I want to run the chunk but not show the code in the knitted file. And I want the knitted file to suppress any warnings, errors, and messages.

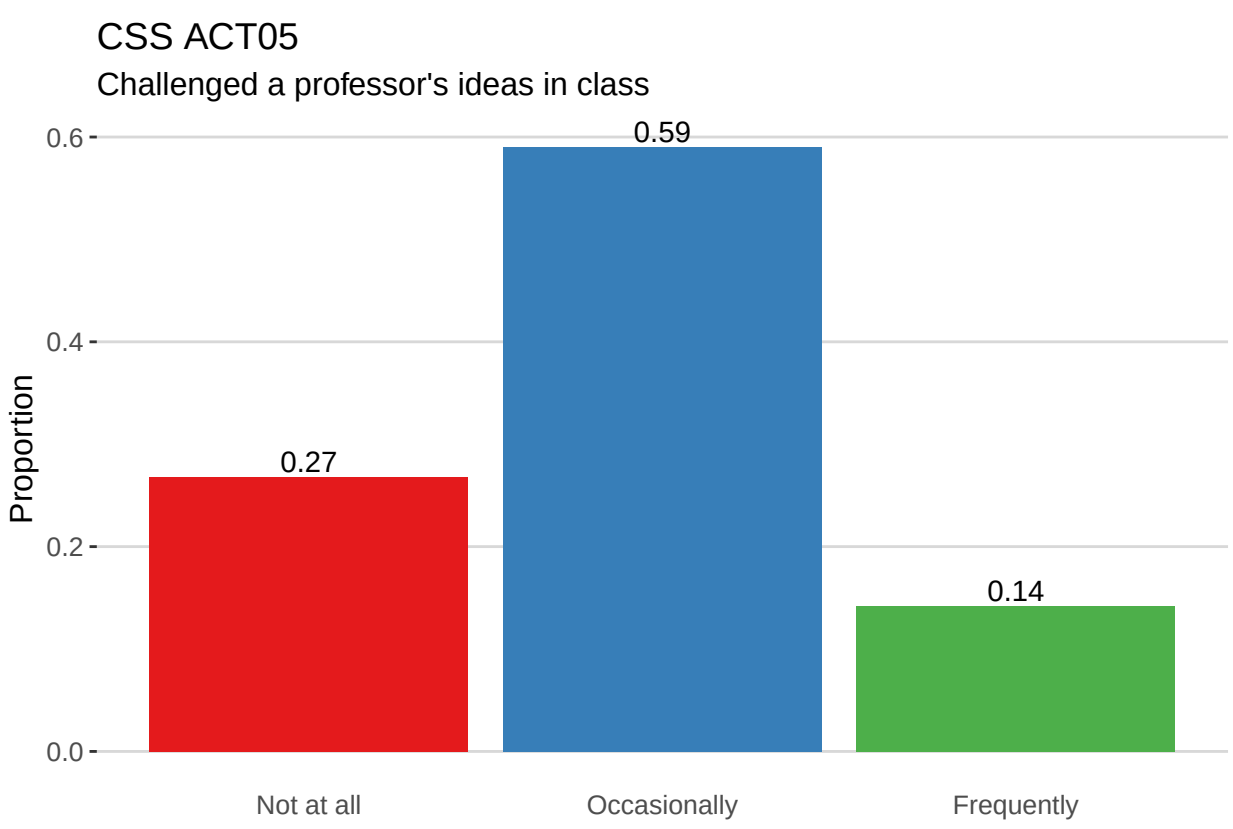
Assignment One Debrief

The data frames we are using today are slightly different from the ones you used on your assignment. What looks different?

Part One - Plots

```r
act05_plot_dodge <- css_summary |>
  filter(Name == "ACT05") |>
  mutate(Summary = factor(Summary,
                          levels = c("ACT05.notatall",
                                     "ACT05.occasionally",
                                     "ACT05.frequently"),
                          labels = c("Not at all",
                                     "Occasionally",
                                     "Frequently"))) |>
  ggplot(aes(x = Summary, y = wtdprop, fill = Summary)) +
  geom_col() +
  geom_text(aes(label = round(wtdprop, 2)),
            position = position_dodge(width = .9),
            vjust = -.25) +
  labs(x = "", y = "Proportion", fill = "",
       title = "CSS ACT05", # I like the survey and variable names in the title
       subtitle = "Challenged a professor's ideas in class") + # And the description in the subtitle
  theme_hc() +
  theme(axis.ticks.x = element_blank(),
        legend.position = "none") +
  scale_fill_brewer(palette = "Set1")

act05_plot_dodge
```

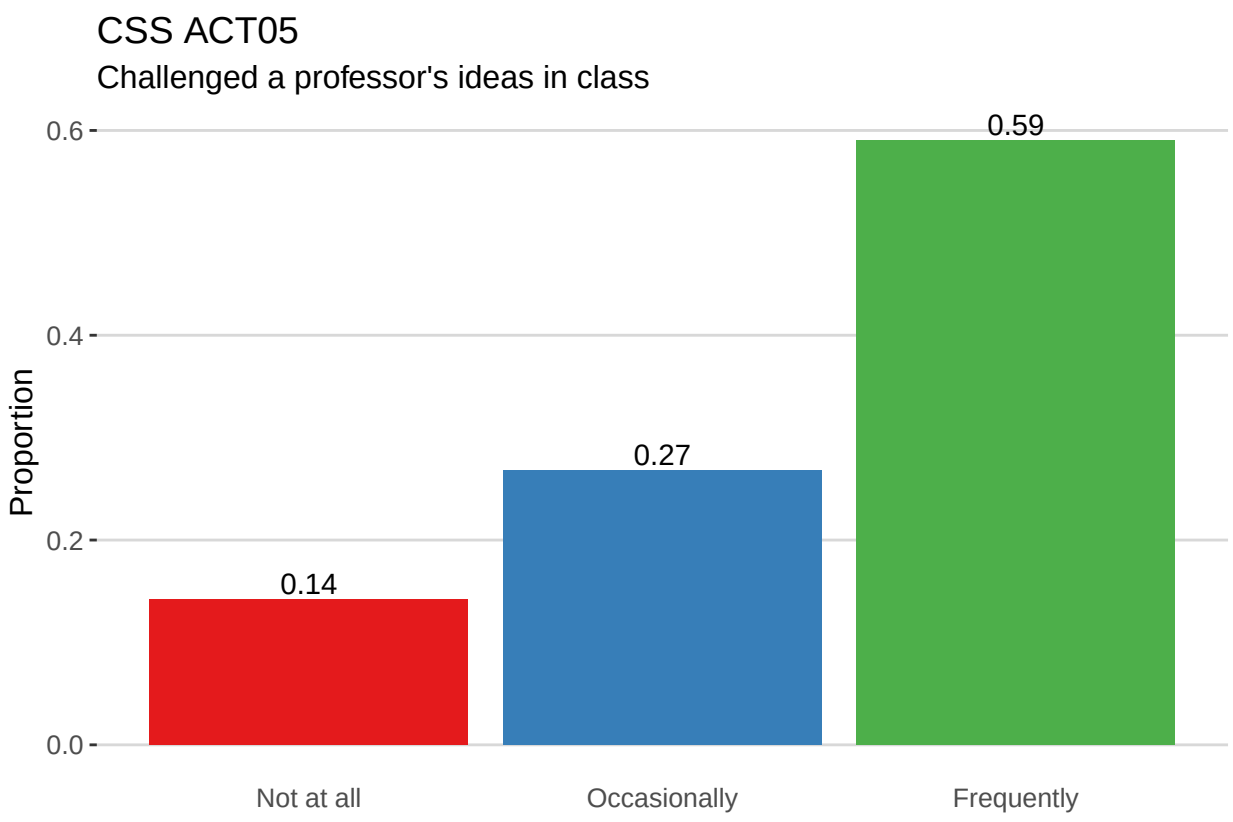


Any ways to simplify?

With the new variables in the data frame, we can replace Summary with Label.

```
act05_plot_dodge <- css_summary |>
  filter(Name == "ACT05") |>
  mutate(Label = factor(Label, # use the Label variable here
                        labels = c("Not at all",
                                   "Occasionally",
                                   "Frequently"))) |>
  ggplot(aes(x = Label, y = wtdprop, fill = Label)) + # use the Label variable here
  geom_col() +
  geom_text(aes(label = round(wtdprop, 2)),
            position = position_dodge(width = .9),
            vjust = -.25) +
  labs(x = "", y = "Proportion", fill = "",
       title = "CSS ACT05", # I like the survey and variable names in the title
       subtitle = "Challenged a professor's ideas in class") + # And the description in the subtitle
  theme_hc() +
  theme(axis.ticks.x = element_blank(),
        legend.position = "none") +
  scale_fill_brewer(palette = "Set1")

act05_plot_dodge
```

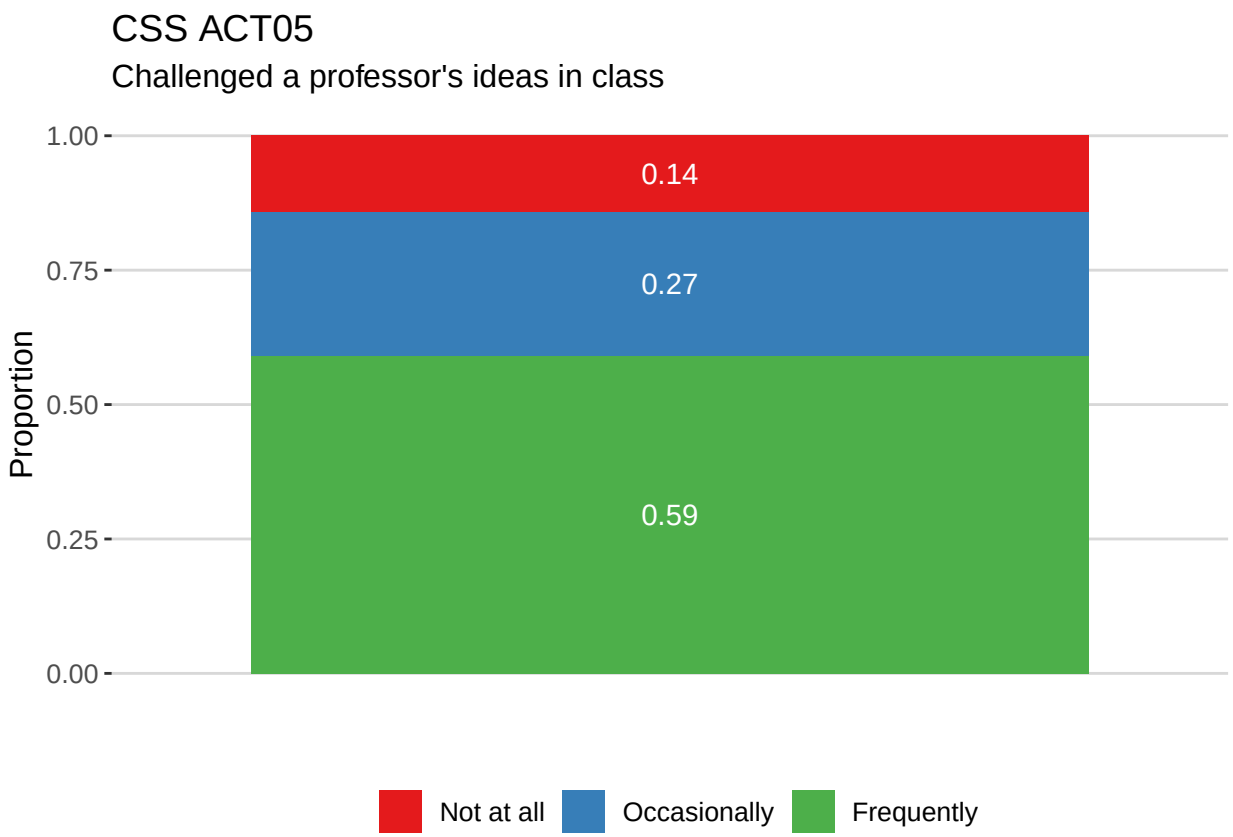


Here's the example code for a stacked column plot:

```
act05_plot_stack <- css_summary |>
  filter(Name == "ACT05") |>
  mutate(Label = factor(Label,
                        labels = c("Not at all",
```

```
                                    "Occasionally",
                                    "Frequently"))) |>
  ggplot(aes(x = Name, y = wtdprop, fill = Label)) +
  geom_col(position = "stack") +
  geom_text(aes(label = round(wtdprop, 2)),
            position = position_stack(vjust = .5),
            color = "white") + # An example of how to change the color of the geom_text label
  labs(x = "", y = "Proportion", fill = "",
       title = "CSS ACT05", # I like the survey and variable names in the title
       subtitle = "Challenged a professor's ideas in class") + # And the description in the subtitle + 
  theme_hc() +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank(), # Remove the Name from the x axis
        legend.position = "bottom") + # Should include a legend on this plot
  scale_fill_brewer(palette = "Set1")

act05_plot_stack
```



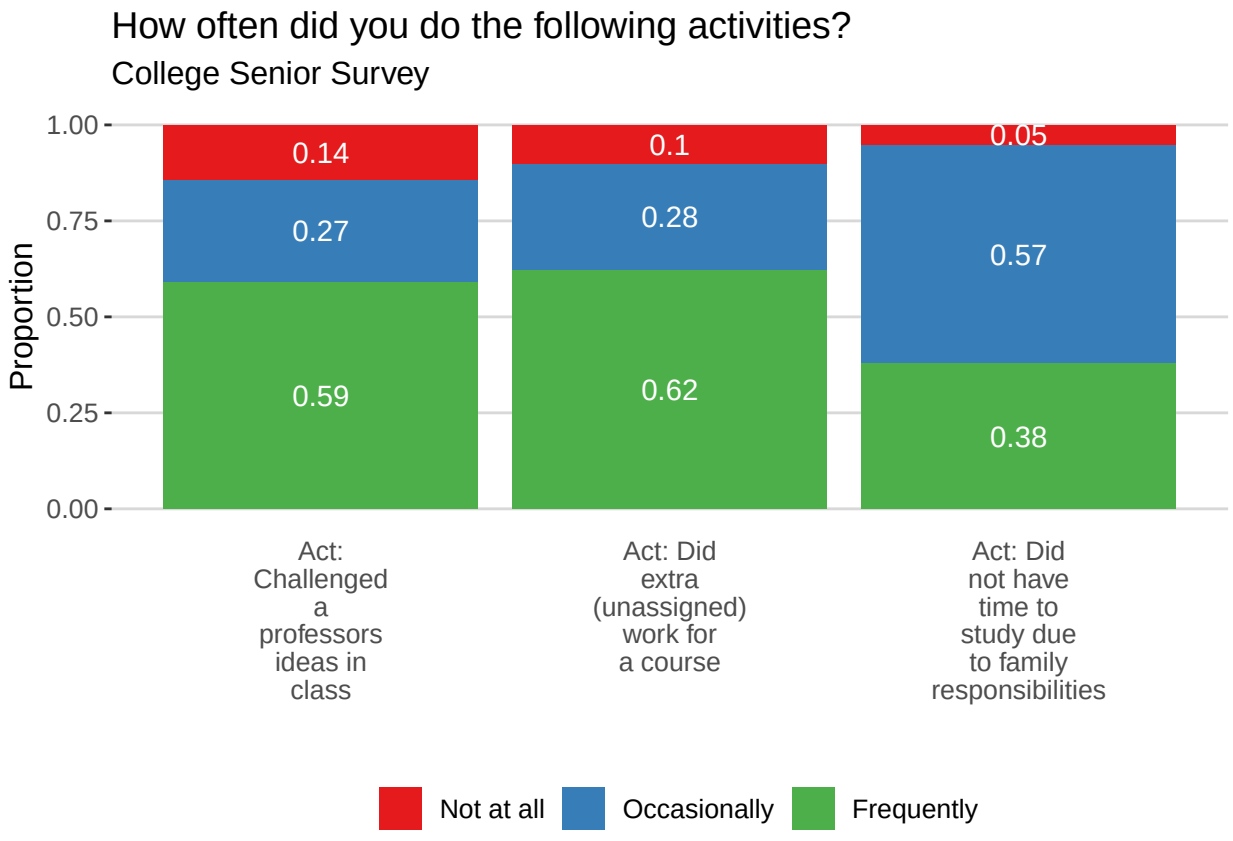
Exercise: How could you have one figure with multiple variables? Try ACT05, ACT11, and ACT12 from the CSS.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
act05_plot_stack_multiple <- css_summary |>
  filter(Name %in% c("ACT05", "ACT11", "ACT12")) |> # Filter for all the variable
```

```
  mutate(Label = factor(Label,
                        labels = c("Not at all",
                                   "Occasionally",
                                   "Frequently"))) |>
  ggplot(aes(x = Description, y = wtdprop, fill = Label)) + # Use Description as the x variable
  geom_col(position = "stack") +
  geom_text(aes(label = round(wtdprop, 2)),
            position = position_stack(vjust = .5),
            color = "white") +
  labs(x = "", y = "Proportion", fill = "",
       title = "How often did you do the following activities?",
       subtitle = css_summary$Survey_Name) +
  theme_hc() +
  theme(axis.ticks.x = element_blank(),
        #axis.text.x = element_blank(), # You need the x axis text to identify the separate plots
        legend.position = "bottom") + # Should include a legend on this plot
  scale_fill_brewer(palette = "Set1") +
  scale_x_discrete(labels = label_wrap(10))

act05_plot_stack_multiple
```

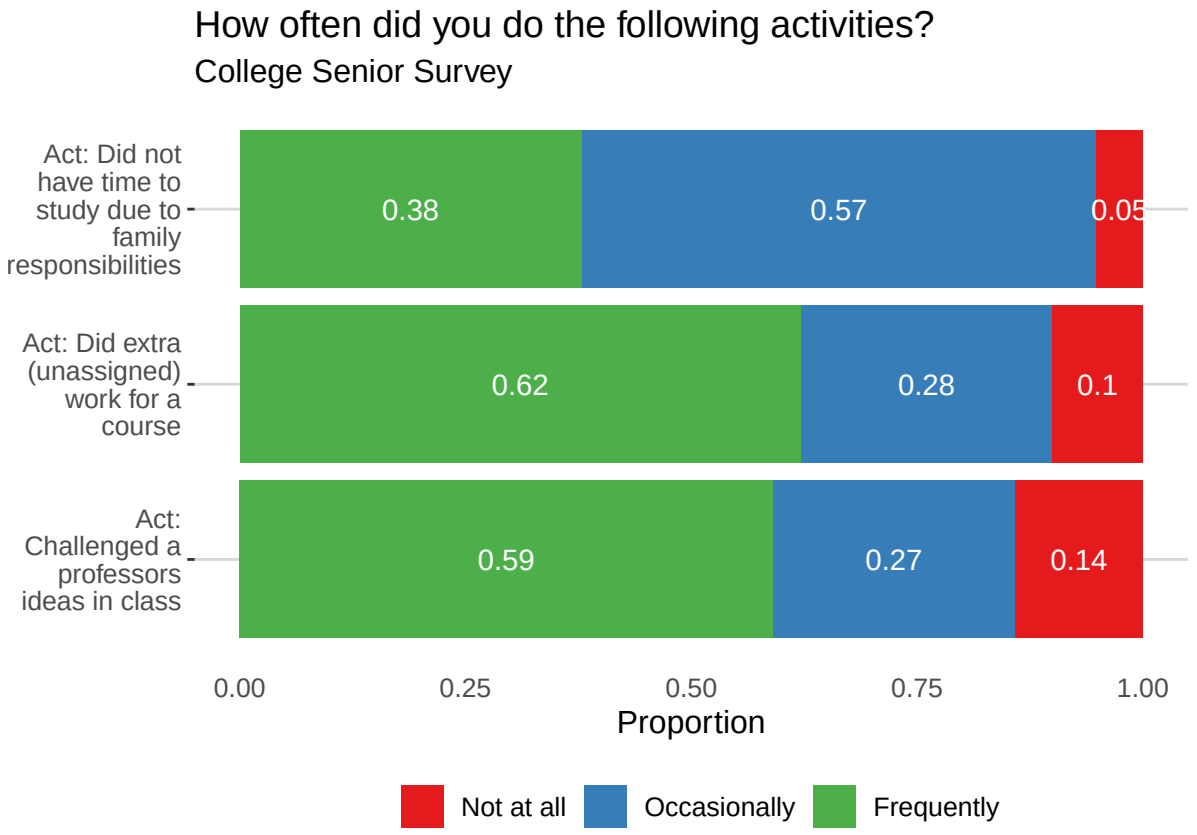


That looks good. One possible change to consider would be using horizontal columns rather than vertical columns. You can switch them with the `coord_flip()` function.

```
act05_plot_stack_multiple_horizontal <- css_summary |>
  filter(Name %in% c("ACT05", "ACT11", "ACT12")) |> # Filter for all the variable
  mutate(Label = factor(Label,
```

```
                          labels = c("Not at all",
                                     "Occasionally",
                                     "Frequently"))) |>
  ggplot(aes(x = Description, y = wtdprop, fill = Label)) + # Use Description as the x variable
  geom_col(position = "stack") +
  geom_text(aes(label = round(wtdprop, 2)),
            position = position_stack(vjust = .5),
            color = "white") +
  labs(x = "", y = "Proportion", fill = "",
       title = "How often did you do the following activities?",
       subtitle = css_summary$Survey_Name) +
  theme_hc() +
  theme(axis.ticks.x = element_blank(),
        #axis.text.x = element_blank(), # You need the x axis text to identify the separate plots
        legend.position = "bottom") + # Should include a legend on this plot
  scale_fill_brewer(palette = "Set1") +
  scale_x_discrete(labels = label_wrap(15)) + # Max characters per line can be higher
  coord_flip() # to switch the x and y axes

act05_plot_stack_multiple_horizontal
```



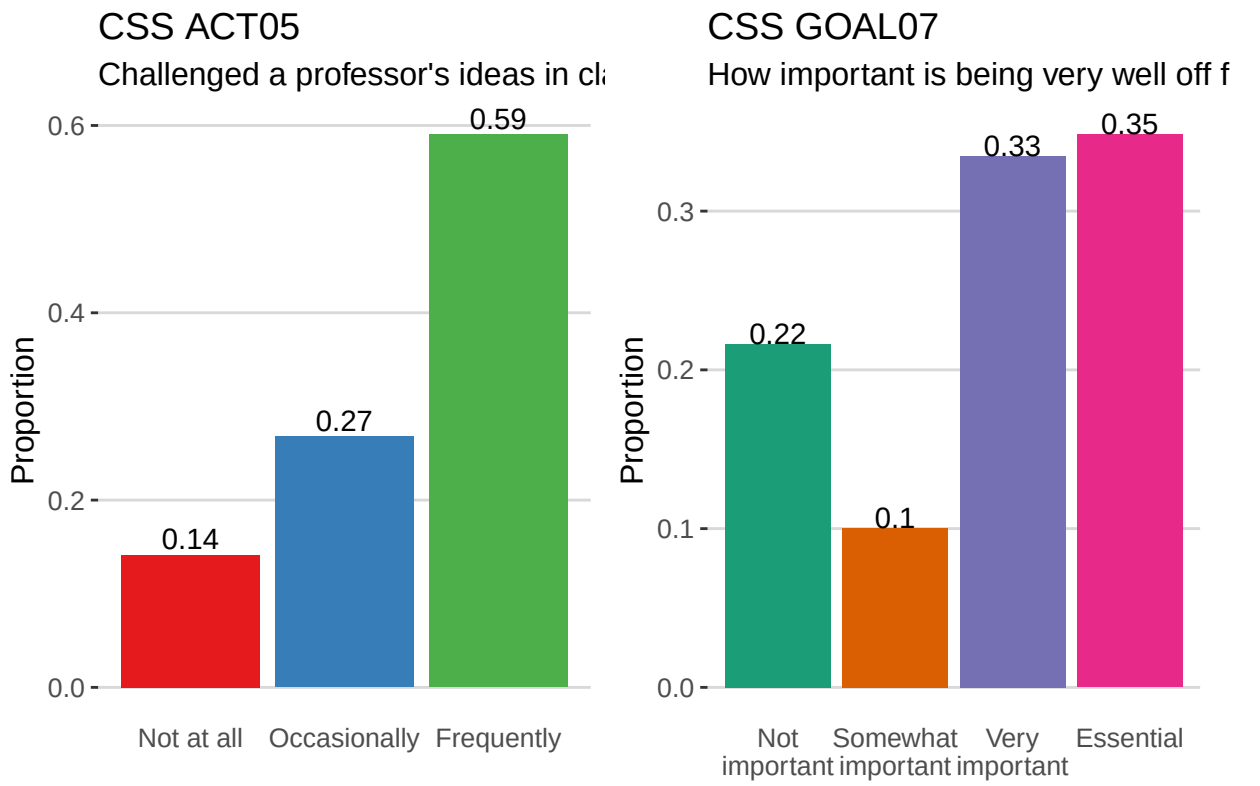
What about combining plots that don't share the same X axis? The `patchwork` package is helpful for that. Let's make another plot for this example. We'll use the `GOAL07` question.

```
goal07_plot_dodge <- css_summary |>
  filter(Name == "GOAL07") |>
```

```
  mutate(Label = factor(Label, # use the Label variable here
                        labels = c("Not important",
                                   "Somewhat important",
                                   "Very important",
                                   "Essential"))) |>
  ggplot(aes(x = Label, y = wtdprop, fill = Label)) + # use the Label variable here
  geom_col() +
  geom_text(aes(label = round(wtdprop, 2)),
            position = position_dodge(width = .9),
            vjust = 0) +
  labs(x = "", y = "Proportion", fill = "",
       title = "CSS GOAL07",
       subtitle = "How important is being very well off financially?") +
  theme_hc() +
  theme(axis.ticks.x = element_blank(),
        legend.position = "none") +
  scale_fill_brewer(palette = "Dark2") +
  scale_x_discrete(labels = label_wrap(10))
```

To position plots side by side, separate their saved object names with a plus

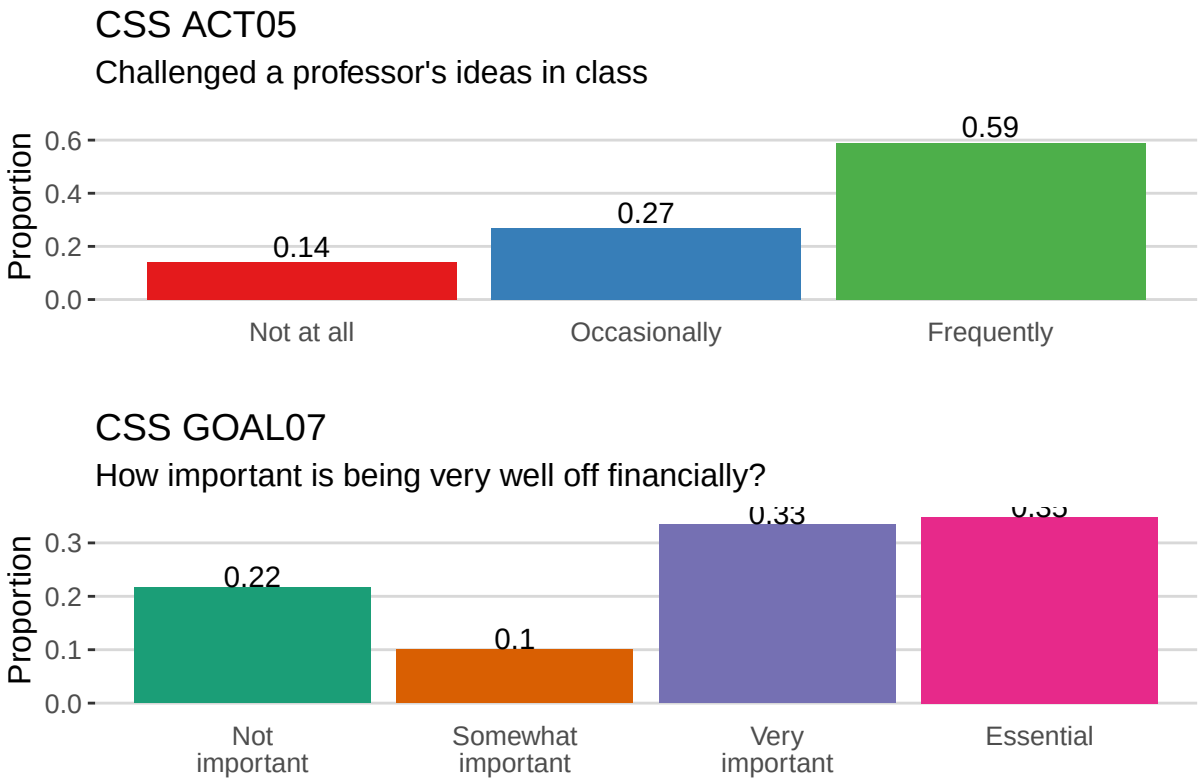
```
act05_plot_dodge + goal07_plot_dodge
```



Watch scales. Can position above and below by replacing the plus sign with a slash.

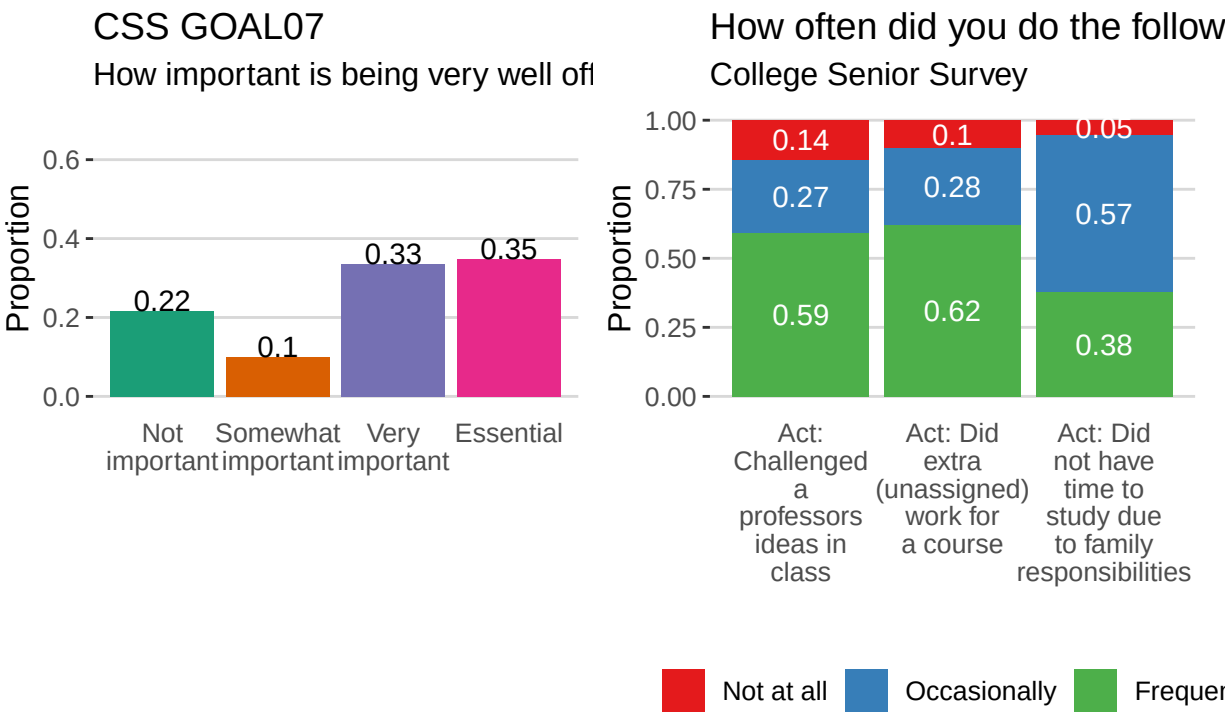
```
act05_plot_dodge <- act05_plot_dodge + ylim(c(0,.7))
goal_07_plot_dodge <- goal07_plot_dodge + ylim(c(0,.7))

act05_plot_dodge / goal07_plot_dodge
```



Can get complicated quickly. The takeaway is to think about what you want *before* you create your plots so patching them together goes smoothly.

```
(goal_07_plot_dodge + act05_plot_stack_multiple) +
  plot_annotation(
  title = "Can include a common title",
  subtitle = "And a common subtitle"
)
```



Part Two - Grouped Summaries

Here's an example from the TFS showing how to combine multiple means and combine multiple responses.

```
weighted_mean_summary <- tfs_by_college |>
  group_by(type) |>
  summarise(lessthan51 = weighted.mean(DISTHOME.miles10less_mean +
                                         DISTHOME.miles11to50_mean,
                                       w = n_responses),
            miles51to100 = weighted.mean(DISTHOME.miles51to100_mean,
                                         w = n_responses),
            morethan100 = weighted.mean(DISTHOME.miles101to500_mean +
                                          DISTHOME.miles501plus_mean,
                                        w = n_responses)) |>
  mutate(across(where(is.numeric), round, 3)) # trick for rounding all numeric values

weighted_mean_summary
```

```
## # A tibble: 2 x 4
##   type               lessthan51 miles51to100 morethan100
##   <chr>                   <dbl>        <dbl>       <dbl>
## 1 Private non-profit      0.273        0.143       0.584
## 2 Public                  0.3          0.186       0.515
```

When you are working in your notebook, this is fine. But when you knit (for a paper report, or presentation), it can be good to make the tables nicer. The `kable` package is helpful for that.

Table 1: Proportion Of Students Traveling Each Distance From Home To Campus, By College Type

Institution Type	50 miles or less	51 to 100 miles	More than 100 miles
Private non-profit	0.273	0.143	0.584
Public	0.300	0.186	0.515

```
#install.packages("kableExtra") # hashtag this line after installing
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
weighted_mean_summary |>
  kbl(booktabs = TRUE, # Adds default styling
      align = "lccc", # To align each column
      col.names = c("Institution Type", "50 miles or less", "51 to 100 miles",
                    "More than 100 miles"),
      caption = "Proportion Of Students Traveling Each Distance From Home To Campus, By College Type")
```

An example with a few more options...

```
weighted_mean_summary_tier <- tfs_by_college |>
  group_by(type, tier_name) |> # Grouping by multiple variables
  summarise(lessthan51 = weighted.mean(DISTHOME.miles10less_mean +
                                         DISTHOME.miles11to50_mean,
                                       w = n_responses),
            miles51to100 = weighted.mean(DISTHOME.miles51to100_mean,
                                         w = n_responses),
            morethan100 = weighted.mean(DISTHOME.miles101to500_mean +
                                        DISTHOME.miles501plus_mean,
                                      w = n_responses)) |>
  mutate(across(where(is.numeric), round, 3)) |>
  mutate(tier_name = ifelse(str_detect(tier_name, "Other"), # full name was too long for table
                            "Other elite schools",
                            tier_name)) |>
  kbl(booktabs = TRUE, # Adds default styling
      align = "llccc", # To align each column
      col.names = c("Institution Type",
                    "Institution Tier",
                    "50 miles or less",
                    "51 to 100 miles",
                    "More than 100 miles"),
      caption = "Proportion Of Students Traveling Each Distance From Home To Campus, By College Type An
  pack_rows("Private non-profit", 1, 4) |>  # To group rows with a label
  pack_rows("Public", 5, 7) |> # Avoids `type` values repeating in each row
  column_spec(1, color = "white") |>  # `type` column still exists; change font so values don't appear
  add_header_above(c(" " = 2, "Distance From Home To Campus" = 3)) # Header for column groups
```

Table 2: Proportion Of Students Traveling Each Distance From Home To Campus, By College Type And Tier

		Distance From Home To Campus		
Institution Type	Institution Tier	50 miles or less	51 to 100 miles	More than 100 miles
Private non-profit				
	Highly selective private	0.264	0.138	0.598
	Ivy Plus	0.076	0.045	0.879
	Other elite schools	0.151	0.076	0.773
	Selective private	0.326	0.172	0.501
Public				
	Highly selective public	0.263	0.181	0.555
	Other elite schools	0.261	0.192	0.547
	Selective public	0.316	0.186	0.498

```
## `summarise()` has grouped output by 'type'. You can override using the `.groups` argument.
```

```
weighted_mean_summary_tier
```

The online documentation for Kable has many more details:

For HTML tables: https://haozhu233.github.io/kableExtra/awesome_table_in_html.html For PDF tables: https://rdrr.io/cran/kableExtra/f/inst/doc/awesome_table_in_pdf.pdf

Working With Strings

This morning we saw lots of functions for working with strings. Let's use the `css_summary` file to practice. What are some of the issues with this file that could use some data cleaning?

The first row of the dataframe has some things to fix. As a starter, can you use string functions to change *professors* to *professor's* in the `Description` variable for `ACT05`?

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
css_summary <- css_summary |>
  mutate(Description = ifelse(Description == "ACT05",
                              str_replace(Description, # where to find the string
                                   "professors", # what string to find
                                   "professor's"),
                              Description)) # how to replace match
```

What else can we do with the `Description` variable? It could be good to get rid of the text before the colon. Any ideas for how to do that?

REPLACE THIS LINE WITH YOUR CODE CHUNK

Does every question have a colon? Use `str_count` to check.

```
css_summary2 <- css_summary |>
  mutate(find_colon = str_count(Description, ":"))
```

The default is to put new variables after the last column. Use `relocate()` to move columns. The default location for `relocate()` is the first column, but you can adjust that with the `.before =` or `.after =` options.

```
css_summary2 <- css_summary2 |>
  relocate(find_colon, .after = Description)
```

Check a table of the `find_colon` variable to see how bad the problem is.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
table(css_summary2$find_colon)
```

```
##
##   0   1
##  36 804
```

So we have some rows that require additional care. What are our options?

One possibility is to break the `Description` values into two pieces: a string before a colon and a string after (if there is a colon). Then we can use those two pieces to repair the existing values. We can do that with the `separate()` function:

```
css_summary2 <- css_summary2 |>
  separate(Description, # variable to separate
           c("before_colon", "after_colon"), # the two new columns to create
           sep = "[:]", # where to separate
           remove = FALSE) # don't remove the variable you are separating
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 36 rows [110,
## 111, 112, 113, 416, 417, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440,
## 441, 442, 443, ...].
```

Our operations above left some whitespace at the start of the strings in the `after_colon` variable. Trim it.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
css_summary2 <- css_summary2 |>
  mutate(after_colon = str_trim(after_colon, side = c("left")))
```

Now try to change the existing Description values that have a colon to only take the text after the colon.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
css_summary2 <- css_summary2 |>
  mutate(Description = ifelse(find_colon==1, # if this is true...
                              after_colon, # do this...
                              Description)) # if not true do this
```

Now how do our `Description` values look? Anything else we need to do?

Does `ACT26` need any help?

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
css_summary2 <- css_summary2 |>
  mutate(Description = str_to_sentence(Description))
```

You can also change case with:

- str_to_lower (for strings like this)
- str_to upper (FOR STRINGS LIKE THIS)
- str_to_title (For Strings Like This, Capitalizing Each First Letter)

Redoing the RACE variables

The survey has a different question for each race/ethnicity value (`RACE1` to `RACE9`). It would be helpful to combine all the possible values into one variable to be able to make the kinds of plots we saw earlier. There are various ways to do that. Here's one.

Make a new dataframe called `race_variables` (based on `css_summary2`) with only the rows that include "RACE" in the `Name` variable and "yes" in the `Summary` variable.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
race_variables <- css_summary2 |>
  filter(str_detect(Name, "RACE"),
         str_detect(Summary, "yes"))
```

The `RACEGROUP` row is unnecessary. Exclude it by using `str_detect()`.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
race_variables <- race_variables |>
  filter(!str_detect(Name, "GROUP"))
```

The `Description` variable has info that we want to use as the values of the `Label` variable. How can we make that happen? And change the value of the `Name` variable to `RACE_ETHNICITY` so it better fits the values.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
race_variables <- race_variables |>
  mutate(Label = Description) |>
  mutate(Name = "RACE_ETHNICITY")
```

Almost done. Remove all the rows with `RACE` in the `Name` column from `css_summary2`.

REPLACE THIS LINE WITH YOUR CODE CHUNK

```
css_summary2 <- css_summary2 |>
  filter(!str_detect(Name, "RACE"))
```

Then use `bind_rows()` to add the `race_variables` dataframe to the end of `css_summary2`.

```
css_summary2 <- bind_rows(css_summary2, race_variables)
```