Getting Started

Load the usual packages (which should now include huxtable!). Remember to include `warning = FALSE, header = FALSE, message = FALSE` to suppress the loading output.

```
library(tidyverse)
library(huxtable)
```

We will be using the gss dataset for this review. Load the `gssr` package, the `gss_doc` documentation, and the `gss_all` dataframe.

```
library(gssr)
data(gss_doc) # codebook
data(gss_all) # dataset
```

This review exercise will consider how interest in space exploration (the `intspace` variable) differs across religious preference. We will control for confidence in the scientific community (the `consci` variable) which could be a confounder between religion and interest in space exploration. Testing the interaction between religion and confidence in science will let us know if any association we may observe between religion and interest in space exploration varies across the distribution of confidence in science.

Start by creating a new dataframe called `review`, filtering the full gss sample to include only the years we want, and selecting the variables we need.

```
review <- gss_all |>
  filter(year>=2010) |>
  select(year, intspace, consci, relig16, race, hispanic,
         sex, educ, age, id)
```

If you will be pulling the same variables in multiple chunks, it might make sense to store their names in a vector so you don't have to type them all every time.

```
my_variables <- c("year", "intspace", "consci", "relig16", "race",
                  "hispanic", "sex", "educ", "age", "id")
```

If you did this before the previous chunk, you could use the object name `my_variables` in the select function (note that you will get a warning if you don't assert you are using `all_of` the variables):

```
review <- gss_all |>
  filter(year>=2010) |>
  select(all_of(my_variables))
```

Clean Up

The big thing to do before starting analyses is to confirm that all missing values have been coded as NA. Run a summary of the `review` dataframe to make sure NAs have been captured.

```
summary(review)
```

```
##       year         intspace        consci         relig16           race
##  Min.   :2010   Min.   :1.00   Min.   :1.000   Min.   : 1.000   Min.   :1.000
##  1st Qu.:2012   1st Qu.:2.00   1st Qu.:1.000   1st Qu.: 1.000   1st Qu.:1.000
##  Median :2014   Median :2.00   Median :2.000   Median : 1.000   Median :1.000
##  Mean   :2014   Mean   :2.08   Mean   :1.643   Mean   : 2.008   Mean   :1.361
##  3rd Qu.:2016   3rd Qu.:3.00   3rd Qu.:2.000   3rd Qu.: 2.000   3rd Qu.:2.000
##  Max.   :2018   Max.   :3.00   Max.   :3.000   Max.   :13.000   Max.   :3.000
##                 NA's   :6287   NA's   :4188    NA's   :75
##     hispanic           sex             educ            age
##  Min.   : 1.000   Min.   :1.000   Min.   : 0.00   Min.   :18.00
##  1st Qu.: 1.000   1st Qu.:1.000   1st Qu.:12.00   1st Qu.:34.00
##  Median : 1.000   Median :2.000   Median :13.00   Median :48.00
##  Mean   : 1.692   Mean   :1.554   Mean   :13.64   Mean   :48.72
##  3rd Qu.: 1.000   3rd Qu.:2.000   3rd Qu.:16.00   3rd Qu.:62.00
##  Max.   :50.000   Max.   :2.000   Max.   :20.00   Max.   :89.00
##  NA's   :33                       NA's   :20      NA's   :34
##        id
##  Min.   :   1
##  1st Qu.: 589
##  Median :1178
##  Mean   :1201
##  3rd Qu.:1767
##  Max.   :2867
##
```

Recall that you can use the `gss_get_marginals()` function with `gss_doc` to see the labels for specific variables. This is a nice place to use the `my_variables` vector. If you save the output of this function, you will be able to easily refer to it later. I recommend opening the spreadsheet view of `my_codebook` (in the top right pane) after running the chunk below.

```
my_codebook <- gss_get_marginals(varnames = my_variables,
                                 data = gss_doc) |>
  select(id, percent, n, value, label)
```

Let's combine values from the `race` and `hispanic` variables to make a new variable called `racehisp`. The easiest way to do this is to first make a binary variable distinguishing those who are not Hispanic from those who are. The value of 1 for the hispanic variable is for

respondents who are not Hispanic. We can use that in the ifelse function to create our binary variable.

```
review <- review |>
     mutate(anyhispanic = ifelse(hispanic==1, 0, 1))
```

Now we can combine values from this new `anyhispanic` variable and the race variable to create the racehisp categories:

```
review <- review |>
     mutate(racehisp = ifelse(anyhispanic==1, 4, race),
            racehisp = factor(racehisp,
                              labels = c("White",
                                         "Black",
                                         "Other",
                                         "Hispanic")))
```

Sometimes it's easier to create new variables instead of changing the values and labels of existing variables. Here we'll create new variables called `science` (taking the values of `consci`) and `space` (taking the values of `intspace`).

```
review <- review |>
     mutate(science = factor(consci,
                             levels = c(3, 2, 1),
                             labels = c("Hardly any", "Only some",
                                        "A great deal")),
            space = factor(intspace,
                           levels = c(3, 2, 1),
                           labels = c("Not interested",
                                      "Moderately interested",
                                      "Very interested")))
```

We can also collapse existing categories into bigger categories. We'll use the `relig16` variable as an example, creating a new variable called `religion` with broader categories.

```
table(review$relig16)
```

```
##
##    1    2    3    4    5    6    7    9   10   11   12   13
## 6102 3827  199 1007   64   51   60   74   55  244    9    4
```

Here we'll put all the respondents with values of 6-9 in the "Eastern" category, and those who are not Eastern, Protestant, Catholic, Jewish, or None in "Other":

```
review <- review |>
     mutate(religion = ifelse(relig16 == 1, "Protestant",
          ifelse(relig16 == 2, "Catholic",
          ifelse(relig16 == 3, "Jewish",
          ifelse(relig16 == 4, "None",
          ifelse(relig16 %in% 6:9, "Eastern", "Other"))))),
               religion = factor(religion,
                    levels = c("Protestant", "Catholic", "Jewish",
                               "Eastern", "Other", "None")))
```

Three Way Table

For each religious category, we want to know the proportion with each level of confidence in science who are in each category of interest in space. One way to do this is with `group_by()` and `summarize()`. For that approach, we would need binary variables for each of the `space` categories. This might seem tedious, but in the long run it is more efficient since it will allow you to manipulate the variables for other purposes later.

```
review <- review |>
     mutate(space_not_interested =
                    ifelse(space=="Not interested",1,0),
               space_moderately_interested =
                    ifelse(space=="Moderately interested",1,0),
               space_very_interested =
                    ifelse(space=="Very interested",1,0))
```

For each combination of `religion` and `science`, we can now summarize the means of each space binary variable (which represent the proportion of respondents in the related category of space interest):

```
space_summary <- review |>
     group_by(religion, science) |>
     summarize(not_interested =
                    round(mean(space_not_interested, na.rm=TRUE),3),
               moderately_interested =
                    round(mean(space_moderately_interested, na.rm=TRUE),3),
               very_interested =
                    round(mean(space_very_interested, na.rm=TRUE),3))
```

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

```
space_summary
```

```
## # A tibble: 28 x 5
## # Groups:   religion [7]
##    religion   science      not_interested moderately_interested very_interested
##    <fct>      <fct>                 <dbl>                 <dbl>           <dbl>
##  1 Protestant Hardly any            0.596                 0.288           0.116
##  2 Protestant Only some             0.354                 0.493           0.153
##  3 Protestant A great deal          0.207                 0.474           0.319
##  4 Protestant <NA>                  0.342                 0.459           0.2
##  5 Catholic   Hardly any            0.585                 0.264           0.151
##  6 Catholic   Only some             0.346                 0.47            0.184
##  7 Catholic   A great deal          0.22                  0.445           0.335
##  8 Catholic   <NA>                  0.313                 0.449           0.238
##  9 Jewish     Hardly any            0.5                   0.5             0
## 10 Jewish     Only some             0.161                 0.581           0.258
## # ... with 18 more rows
```

Those NAs for `science` and `religion` are annoying. One way to get rid of them is to filter them out. You can do that with an extra line in the chunk above. But we'll redo the whole chunk to compare them, though note it's not necessary to run this twice:

```
space_summary <- review |>
     filter(!is.na(science), !is.na(religion)) |>
     #Keep the observations that are not na for science or religion
     group_by(religion, science) |>
     summarise(not_interested = round(mean(space_not_interested,
                                           na.rm=TRUE),3),
               moderately_interested = round(mean(space_moderately_interested,
                                             na.rm=TRUE),3),
               very_interested = round(mean(space_very_interested,
                                            na.rm=TRUE),3))
```

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

```
space_summary
```

```
## # A tibble: 18 x 5
## # Groups:   religion [6]
##    religion   science      not_interested moderately_interested very_interested
##    <fct>      <fct>                 <dbl>                 <dbl>           <dbl>
##  1 Protestant Hardly any            0.596                 0.288           0.116
##  2 Protestant Only some             0.354                 0.493           0.153
```

```
##  3 Protestant A great deal          0.207                0.474           0.319
##  4 Catholic   Hardly any            0.585                0.264           0.151
##  5 Catholic   Only some             0.346                0.47            0.184
##  6 Catholic   A great deal          0.22                 0.445           0.335
##  7 Jewish     Hardly any            0.5                  0.5             0
##  8 Jewish     Only some             0.161                0.581           0.258
##  9 Jewish     A great deal          0.167                0.633           0.2
## 10 Eastern    Hardly any            0.5                  0               0.5
## 11 Eastern    Only some             0.2                  0.4             0.4
## 12 Eastern    A great deal          0.2                  0.333           0.467
## 13 Other      Hardly any            0.4                  0.6             0
## 14 Other      Only some             0.472                0.321           0.208
## 15 Other      A great deal          0.184                0.469           0.347
## 16 None       Hardly any            0.455                0.242           0.303
## 17 None       Only some             0.352                0.496           0.152
## 18 None       A great deal          0.23                 0.378           0.393
```

You can clean up the column names of this table and wrap it in huxtable before you knit. The new part of the code below is how to create a grouping row to use the "Interest in Space" header across the three levels of that variable. The idea is to first insert a new row, then merge it across columns 3:5, and finally assert that there are two rows (1:2) that should be treated as the table's header.

```
huxtable(space_summary) |>
  insert_row("", "", "Interest in Space", "", "", after = 0) |>
  merge_cells(1, 3:5) |> # in row 1, in columns 3:5
  set_header_rows(1:2, TRUE) |>
  set_contents(2, 1:5, c("Religion", "Confidence in Science",
                         "None", "Moderate", "Very")) |>
  set_align(everywhere, 3:5, "center") |>
  theme_compact()
```

There was a question in class about using conditional formatting to highlight specific cells in a huxtable. The chunk below shows how to set the text of specific cells (or rows or columns) to be bold or italic and how to set the background color of specific cells (or rows or columns). Note that it would be unusual to do all of these adjustments in the same table! They are simply included here as examples.

```
huxtable(space_summary) |>
  insert_row("", "", "Interest in Space", "", "", after = 0) |>
  merge_cells(1, 3:5) |> # in row 1, in columns 3:5
  set_header_rows(1:2, TRUE) |>
  set_contents(2, 1:5, c("Religion", "Confidence in Science",
                         "None", "Moderate", "Very")) |>
```

		Interest in Space		
Religion	**Confidence in Science**	**None**	**Moderate**	**Very**
Protestant	Hardly any	0.596	0.288	0.116
Protestant	Only some	0.354	0.493	0.153
Protestant	A great deal	0.207	0.474	0.319
Catholic	Hardly any	0.585	0.264	0.151
Catholic	Only some	0.346	0.47	0.184
Catholic	A great deal	0.22	0.445	0.335
Jewish	Hardly any	0.5	0.5	0
Jewish	Only some	0.161	0.581	0.258
Jewish	A great deal	0.167	0.633	0.2
Eastern	Hardly any	0.5	0	0.5
Eastern	Only some	0.2	0.4	0.4
Eastern	A great deal	0.2	0.333	0.467
Other	Hardly any	0.4	0.6	0
Other	Only some	0.472	0.321	0.208
Other	A great deal	0.184	0.469	0.347
None	Hardly any	0.455	0.242	0.303
None	Only some	0.352	0.496	0.152
None	A great deal	0.23	0.378	0.393

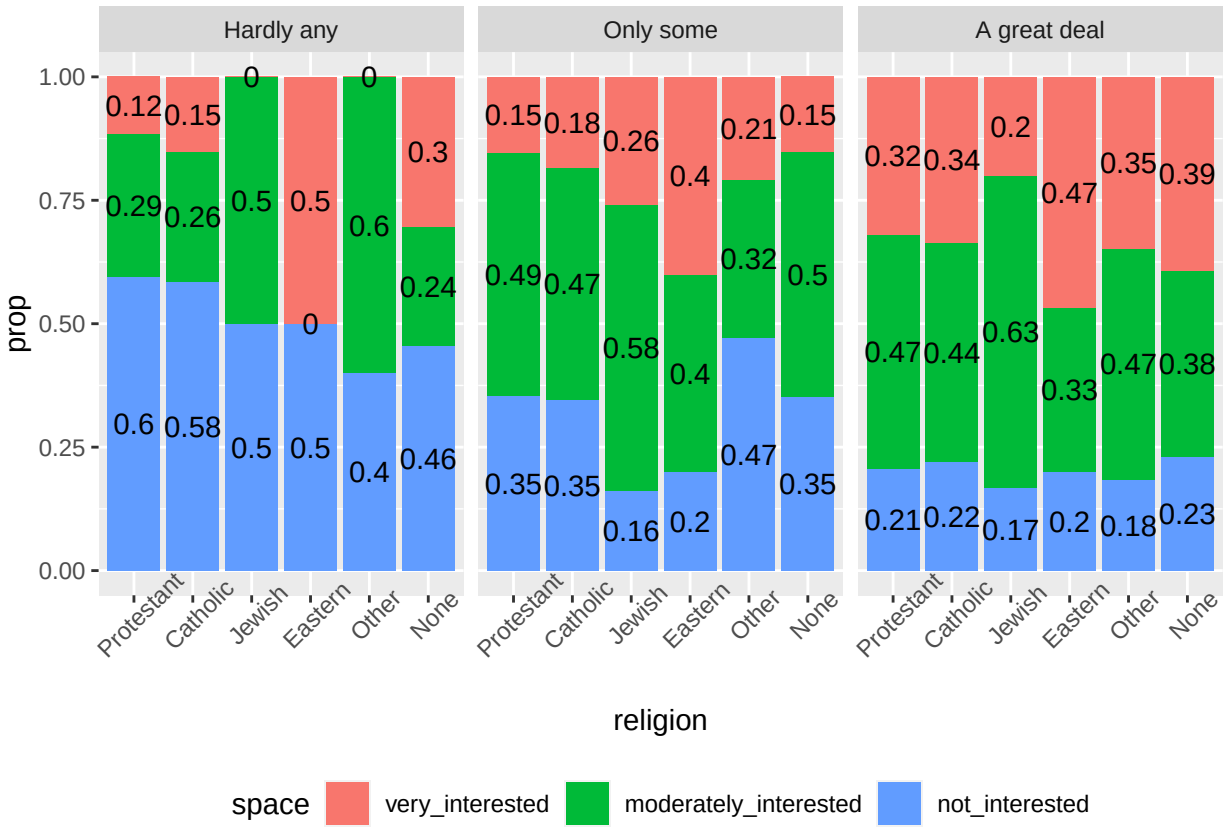
```
  set_align(everywhere, 3:5, "center") |>
  theme_compact() |>
  set_bold(row = 4, col = 3) |> # Bold the cell in row 4 and column 2
  set_italic(row = 5, col = everywhere) |> # Italicize all cells in row 5
  set_background_color(row = -(1:2), col = 5, "grey") # Change the background color to
```

In class on Wednesday, we talked about how to make a plot to visualize the information in the three way table. It's not necessary to do so for your final report! But if you are interested it is possible using all the code in the chunk below. The trick is to use `pivot_longer()` to collapse the three wide columns for the levels of space and make them two long columns - one with the column names and one with the values. Then we can use these columns in our ggplot aesthetic map.

```
space_summary |>
  pivot_longer(names_to = "space", # Make a new column called space
               values_to = "prop", # Make a new column called prop
               3:5) |> # Take the names of the original 3:5 columns and make them the
  mutate(space = factor(space, # reorder the space levels
                        levels = c("very_interested",
                                   "moderately_interested",
```

		Interest in Space		
Religion	**Confidence in Science**	**None**	**Moderate**	**Very**
Protestant	Hardly any	0.596	0.288	0.116
Protestant	Only some	**0.354**	0.493	0.153
Protestant	*A great deal*	*0.207*	*0.474*	*0.319*
Catholic	Hardly any	0.585	0.264	0.151
Catholic	Only some	0.346	0.47	0.184
Catholic	A great deal	0.22	0.445	0.335
Jewish	Hardly any	0.5	0.5	0
Jewish	Only some	0.161	0.581	0.258
Jewish	A great deal	0.167	0.633	0.2
Eastern	Hardly any	0.5	0	0.5
Eastern	Only some	0.2	0.4	0.4
Eastern	A great deal	0.2	0.333	0.467
Other	Hardly any	0.4	0.6	0
Other	Only some	0.472	0.321	0.208
Other	A great deal	0.184	0.469	0.347
None	Hardly any	0.455	0.242	0.303
None	Only some	0.352	0.496	0.152
None	A great deal	0.23	0.378	0.393

```
                                  "not_interested"))) |>
ggplot(aes(x = religion,
           y = prop,
           fill = space)) +
geom_col(position = "fill") +
geom_text(aes(label = round(prop,2)), # Add labels
          position = position_fill(vjust = .5)) +
facet_grid(.~science) + # Separate plot for each level of `science`
theme(legend.position = "bottom", # Move legened to bottom
      axis.text.x = element_text(angle = 45)) # Rotate x axis text
```



Also in class on Wednesday, we talked about a different package called `pollster` that can be useful for three way tables. Install it using the bottom right pane and then load it to use. One big advantage with `pollster` is that you don't have to create binary variables for each level of your dependent variable. The catch with `pollster` is that it requires a weighting variable. The easiest way to deal with that issue is to create a new variable (called `weight` in the example below) that takes the value 1 for every observation. Then you can use that weight in the `pollster` functions.

```
library(pollster)

review <- mutate(review, weight = 1) # Create weight variable with all weights equal to

pollster_3way <- crosstab_3way(review, # data frame name
                               religion, # independent variable
                               space, # dependent variable
                               science, # control variable
                               pct_type = "row", # for row percents
                               weight = weight)

pollster_3way
```

The `pollster` package can also create nice two-way tables using the `crosstab` function. Everything is the same as in the example above except the function is `crosstab` instead of

religion	science	Not interested	Moderately interested	Very interested	n
Protestant	Hardly any	59.6	28.8	11.6	146
Protestant	Only some	35.4	49.3	15.3	913
Protestant	A great deal	20.7	47.4	31.9	667
Catholic	Hardly any	58.5	26.4	15.1	53
Catholic	Only some	34.6	47	18.4	575
Catholic	A great deal	22	44.5	33.5	508
Jewish	Hardly any	50	50	0	2
Jewish	Only some	16.1	58.1	25.8	31
Jewish	A great deal	16.7	63.3	20	30
Eastern	Hardly any	50	0	50	2
Eastern	Only some	20	40	40	20
Eastern	A great deal	20	33.3	46.7	30
Other	Hardly any	40	60	0	5
Other	Only some	47.2	32.1	20.8	53
Other	A great deal	18.4	46.9	34.7	49
None	Hardly any	45.5	24.2	30.3	33
None	Only some	35.2	49.6	15.2	125
None	A great deal	23	37.8	39.3	135

`crosstab_3way` and you don't include the control variable.

```
pollster_2way <- crosstab(review, religion, space,
                          pct_type = "row",
                          weight = weight)

pollster_2way
```

Dealing With NAs In Other Functions

For mean and standard deviation, remove NAs by adding `na.rm = TRUE`:

religion	Not interested	Moderately interested	Very interested	n
Protestant	32.7	46.4	20.9	2.81e+03
Catholic	30.6	44.9	24.5	1.82e+03
Jewish	17.3	57.1	25.5	98
Eastern	20.2	36	43.8	89
Other	34.6	39.2	26.1	153
None	31.1	40.2	28.6	482

```
mean(review$age)
```

```
## [1] NA
```

```
mean(review$age, na.rm = TRUE)
```

```
## [1] 48.72003
```

```
sd(review$educ)
```

```
## [1] NA
```

```
sd(review$educ, na.rm = TRUE)
```

```
## [1] 3.05107
```

For correlation, restrict the estimation to cases with values for both variables by adding `use = "complete"`:

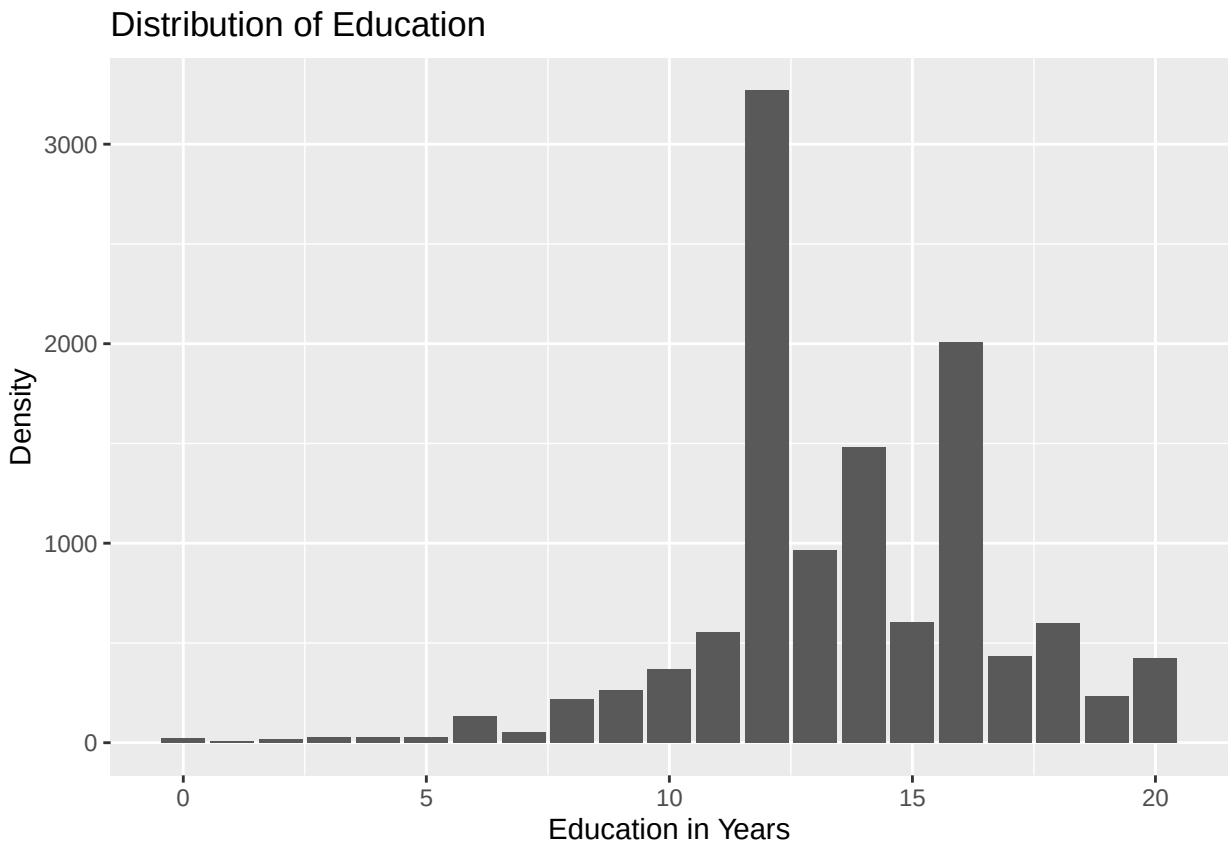
```
cor(review$age, review$educ, use = "complete")
```

```
## [1] -0.0319707
```

For ggplot, R knows to only use complete cases but will warn you that it is doing so. To drop the warning, add `warning = FALSE` to the start of the code chunk:

```
plot <- ggplot(review, aes(x = educ))
plot + geom_bar() +
  labs(x = "Education in Years",
       y = "Density",
       title = "Distribution of Education",
       saubtitle = "General Social Survey, 2010-2018")
```

```
## Don't know how to automatically pick scale for object of type haven_labelled. Default
```



Remember to change the axis labels and add a title to the figure above!

Basic linear models also know to drop NAs. The notes section of the summary informs you how many cases have been deleted from the estimates (in the example below, 6315 observations are deleted due to missingness).

This is new: notice how we are redefining the space factor variable to have a numeric scale in the chunk below. Each of the three factor levels will be assigned a number from 1-3. Since we asserted that the order of levels is "Not at all interested" / "Moderately interested" / "Very interested", now higher scores tell us that respondents are more interested in space. (This is a neat trick, but in general be careful with this approach. It only works if you can assume that the distance between each level is even.)

```
model1 <- lm(as.numeric(space) ~ religion, data = review)
summary(model1)
```

```
##
## Call:
## lm(formula = as.numeric(space) ~ religion, data = review)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.23596 -0.88193  0.06037  0.11807  1.11807
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.88193    0.01384 135.940  < 2e-16 ***
## religionCatholic  0.05769    0.02208   2.613  0.00900 **
## religionJewish    0.19970    0.07544   2.647  0.00814 **
## religionEastern   0.35402    0.07904   4.479 7.65e-06 ***
## religionOther     0.03310    0.06094   0.543  0.58708
## religionNone      0.09317    0.03619   2.574  0.01007 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7341 on 5450 degrees of freedom
##   (6315 observations deleted due to missingness)
## Multiple R-squared:  0.005986,   Adjusted R-squared:  0.005074
## F-statistic: 6.564 on 5 and 5450 DF,  p-value: 4.254e-06
```

By default, the `fitted.values` function will not work if there are NAs in your model. If you have missing values in your model, it is better to use `fitted()` and add `na.action = na.exclude` to your `lm()` code. Now when you run the `fitted()` function any observations not included in your model will have NA as their predicted value.

```
model1 <- lm(as.numeric(space) ~ religion, data = review,
             na.action = na.exclude)

review$predicted_space <- fitted(model1)

summary(review$predicted_space)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.882   1.882   1.882   1.920   1.940   2.236    6315
```

Let's add a model with a control variable and an interactive model.

```
model2 <- lm(as.numeric(space) ~ religion + science,
             data = review,
             na.action = na.exclude)

model3 <- lm(as.numeric(space) ~ religion * science,
             data = review,
             na.action = na.exclude)
```

Use huxreg to combine all three models into your final table:

```
huxreg(model1, model2, model3,
       statistics = c("Number of Observations" = "nobs"),
       coefs = c("(Intercept)" = "(Intercept)",
                 "Religion = Catholic" = "religionCatholic",
                 "Religion = Jewish" = "religionJewish",
                 "Religion = Eastern" = "religionEastern",
                 "Religion = Other" = "religionOther",
                 "Religion = None" = "religionNone",
                 "Science Confidence = Only Some" = "scienceOnly some",
                 "Science Confidence = A Great Deal" = "scienceA great deal",
                 "Catholic X Only Some" = "religionCatholic:scienceOnly some",
                 "Jewish X Only Some" = "religionJewish:scienceOnly some",
                 "Eastern X Only Some" = "religionEastern:scienceOnly some",
                 "Other X Only Some" = "religionOther:scienceOnly some",
                 "None X Only Some" = "religionNone:scienceOnly some",
                 "Catholic X A Great Deal" = "religionCatholic:scienceA great deal",
                 "Jewish X A Great Deal" = "religionJewish:scienceA great deal",
                 "Eastern X A Great Deal" = "religionEastern:scienceA great deal",
                 "Other X A Great Deal" = "religionOther:scienceA great deal",
                 "None X A Great Deal" = "religionNone:scienceA great deal")) |>
  set_caption("Add A Title Here") |>
  theme_compact()
```

Using Markdown For Reports

Hiding Code and Inline Code

Let's start with a case where your output is a single number, like a mean. Imagine you are working on the descriptives part of your project and want to include the mean of age. The place to start is with a regular code chunk with the `mean()` function:

```
mean(review$age, na.rm=TRUE)
```

```
## [1] 48.72003
```

But say you want R to run a code chunk and have only the output - not the code! - show up in your file. Simply add `echo = FALSE` to the first fence:

```
## [1] 48.72003
```

If you want to integrate a single number into your document, you can use inline code. Without opening a full code chunk, just use one backtick to open and close your fence. Then write a sentence as you normally would, and let R Markdown replace your code with the output:

The mean of age is 48.72.

Other Options For Hiding Code

If you want to run the code chunk so you can see the output in your notebook but with neither the code nor the output showing up in your knitted file, use `include = FALSE`.

I recommend starting with `include = FALSE` for your final project, so you can see all your output but then selectively choose what to include and what not to include in your knitted report.

If for some reason you want to show the code but not the output, use `eval = FALSE`.

```
mean(review$age, na.rm=TRUE)
```

R Markdown Tips

Some other things to know about writing in R Markdown...

Use hashtags for headings. One hashtag is for a big heading; additional hashtags shrink the size. For example:

Biggest Heading

Big Heading

Small Heading

Smallest Heading

If you want to italicize text, *wrap it within single asterisks*. If you want to bold text, **wrap it within double asterisks**. And if you want to italicize *and* bold text, ***wrap it within triple asterisks***.

It can sometimes be helpful to highlight original variable names or unusual terms within tickmarks. But note this is similar to the inline code we saw earlier. As long as the word or phrase does not start with a single r, R will not try to run it as code. See the preview file for the difference in what these tickmarks represent:

The mean of `age` is 48.72.

To create an ordered list, leave an empty line and then:

- Start
- Each
- Item
- With
- A
- Dash

To create a numbered list, leave an empty line and then:

1. Start
2. Each
3. Item
4. With
5. A
6. Number and a period

To add a horizontal line rule, include at least three dashes on a single line:

And to add a page break:

This should be the start of a new page!

It's Also The Start Of A New Section

Formatting Summary Tables

We have seen `huxtable()` and `huxreg()` a lot. They are great. Use them.

Here's another example of how to use `huxtable()` in combination with `group_by()` and `summarize()` to make a nice summary table. Let's start with the code for getting means and standard deviations of the `age` and `educ` variables for each `religion` group:

```
summary_table <- review |>
     filter(!is.na(religion)) |>
     group_by(religion) |>
     summarize(mean_age = round(mean(age, na.rm=TRUE),2),
               sd_age = round(sd(age, na.rm=TRUE),2),
               mean_educ = round(mean(educ, na.rm=TRUE),2),
               sd_educ = round(sd(educ, na.rm=TRUE),2))
```

If we huxtable this table, we'll have the religion categories in the rows and the means and standard deviations in the columns:

```
huxtable(summary_table) |>
  insert_row("", "Age", "", "Education", "", after = 0) |>
  merge_cells(1, 2:3) |>
  merge_cells(1, 4:5) |>
  set_contents(2, 1:5, c("Religion", "Mean", "SD", "Mean", "SD")) |>
  set_header_rows(1:2, TRUE) |>
  set_caption("Summary Table: Age and Education by Religion") |>
  set_align(everywhere, 2:5, "center") |>
  theme_compact()
```

Note that `huxtable()` also works well with `t.test()` after asserting that the results should be in tidy format (by wrapping the `t.test()` function in `tidy()`)...

```
tidy(t.test(review$age[review$religion=="Jewish"],
                review$age[review$religion=="Eastern"])) |>
  huxtable() |>
  select(-c(method, parameter)) |> # Drop the method and parameter columns from the tab
  set_caption("T Test: Difference in Mean Age between Jewish and Eastern Respondents") |
  theme_compact()
```

...and `prop.test()`...

```
space_religion <- review |>
     filter(religion=="Protestant" | religion=="None") |>
     select(religion, space_very_interested) |>
     droplevels()

space_religion_table <- table(space_religion$religion,
                              space_religion$space_very_interested)

tidy((prop.test(space_religion_table))) |>
  huxtable() |>
  select(-c(method, parameter)) |>
  set_caption("Proportion Test: Protestant vs No Religion and Very Interested in Space") |>
  theme_compact()
```

...and `chisq.test()`...

```
tidy(chisq.test(review$sex, review$consci)) |>
  huxtable() |>
  select(-c(method, parameter)) |>
  set_caption("Chi-square Test: Sex and Confidence in Science") |>
  theme_compact()
```

...and `fisher.test()`...

```
educ_11_years_only <- review |>
     filter(educ==11) |>
     select(educ, religion, space)

chisq.test(educ_11_years_only$religion, educ_11_years_only$space)$expected
```

```
## Warning in chisq.test(educ_11_years_only$religion, educ_11_years_only$space):
## Chi-squared approximation may be incorrect
```

```
##                            educ_11_years_only$space
## educ_11_years_only$religion Not interested Moderately interested
##                  Protestant     45.9183673            54.5918367
##                  Catholic       30.4897959            36.2489796
##                  Jewish          0.3673469             0.4367347
##                  Eastern         0.3673469             0.4367347
##                  Other           3.3061224             3.9306122
##                  None            9.5510204            11.3551020
```

```
##                              educ_11_years_only$space
## educ_11_years_only$religion Very interested
##                   Protestant      24.4897959
##                   Catholic        16.2612245
##                   Jewish           0.1959184
##                   Eastern          0.1959184
##                   Other            1.7632653
##                   None             5.0938776
```

```
tidy(fisher.test(educ_11_years_only$religion, educ_11_years_only$space,
                   simulate.p.value = TRUE)) |>
  huxtable() |>
  select(-c(method)) |> # Just drop method here; fisher.test output does not include po
  set_caption("Fisher Test: Religion and Interest in Space") |>
  theme_compact()
```

Table 1: Add A Title Here

	(1)	(2)	(3)
(Intercept)	1.882 ***	1.564 ***	1.521 ***
	(0.014)	(0.047)	(0.059)
Religion = Catholic	0.058 **	0.025	0.045
	(0.022)	(0.027)	(0.114)
Religion = Jewish	0.200 **	0.112	-0.021
	(0.075)	(0.092)	(0.508)
Religion = Eastern	0.354 ***	0.267 **	0.479
	(0.079)	(0.101)	(0.508)
Religion = Other	0.033	-0.002	0.079
	(0.061)	(0.071)	(0.324)
Religion = None	0.093 *	0.061	0.328 *
	(0.036)	(0.045)	(0.138)
Science Confidence = Only Some		0.239 ***	0.279 ***
		(0.049)	(0.064)
Science Confidence = A Great Deal		0.535 ***	0.592 ***
		(0.050)	(0.065)
Catholic X Only Some			-0.007
			(0.121)
Jewish X Only Some			0.318
			(0.524)
Eastern X Only Some			-0.079
			(0.533)
Other X Only Some			-0.143
			(0.340)
None X Only Some			-0.327 *
			(0.153)
Catholic X A Great Deal			-0.044
			(0.122)
Jewish X A Great Deal			-0.059
			(0.525)
Eastern X A Great Deal			-0.325
			(0.525)
Other X A Great Deal			-0.029
			(0.341)
None X A Great Deal			-0.277
			(0.153)
Number of Observations	5456	3377	3377

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Table 2: Summary Table: Age and Education by Religion

	Age		Education	
Religion	**Mean**	**SD**	**Mean**	**SD**
Protestant	51	17.9	13.7	2.81
Catholic	47.9	17.1	13.5	3.35
Jewish	56.2	18	16	2.67
Eastern	41.6	16.4	15.2	3.38
Other	38.4	15.4	13.4	2.68
None	41.6	16	13.6	3.07

Table 3: T Test: Difference in Mean Age between Jewish and Eastern Respondents

estimate	**estimate1**	**estimate2**	**statistic**	**p.value**	**conf.low**	**conf.high**	**alternative**
14.7	56.2	41.6	8.33	1.5e-15	11.2	18.1	two.sided

Table 4: Proportion Test: Protestant vs No Religion and Very Interested in Space

estimate1	**estimate2**	**statistic**	**p.value**	**conf.low**	**conf.high**	**alternative**
0.791	0.714	14	0.000186	0.0333	0.122	two.sided

Table 5: Chi-square Test: Sex and Confidence in Science

statistic	**p.value**
60.4	7.81e-14

Table 6: Fisher Test: Religion and Interest in Space

p.value	**alternative**
0.0425	two.sided