Data Wrangling

Henry Sun, Maxine Dobbs, and Miranda Yang

We obtained the data from Healthy Minds Network project, and it was shared with us as a folder of CSV and Excel files in Dropbox after we requested the dataset through the provided Google Form on the official website of Healthy Minds Network. We downloaded the folder consisting of the entire set of files onto Google Drive on 10/8/25.

Full citation: Healthy Minds Network (2007-2025). Healthy Minds Study among Colleges and Universities, 2007-2025 datasets. Healthy Minds Network, University of Michigan, University of California Los Angeles, Boston University, and Wayne State University. https://healthymindsnetwork.org/research/data-for-researchers.

Note that, before the code block loading in all_years_organized.RData, all blocks of code are set not to run automatically. Running them takes a very long time and only needs to be done once; given that all_years_organized.RData has already been created and stored in this project, running these blocks of code is unnecessary. It also requires adding all files from the Health Minds study into the initial_data subfolder manually. (See README.md) Once this is done, you can run code blocks one by one with data_wrangling.qmd opened in RStudio.

As a first step, the code below reads CSV files containing survey data from the Healthy Minds Network and creates R data frames from them.

```
# Code in this block was written by Henry

# Read all CSV files
year_2007_2013 <-
  read.csv("../initial_data/HMS_2007-2013_Aggregate.csv")
year_2013_2014 <-
  read.csv("../initial_data/HMS_2013-2014_PUBLIC.csv")
year_2016_2017 <-
  read.csv("../initial_data/HMS_2016-2017_PUBLIC.csv")
year_2017_2018 <-
  read.csv("../initial_data/HMS_2017-2018_PUBLIC_instchars.csv")
year_2018_2019 <-
  read.csv("../initial_data/HMS_2018-2019_PUBLIC_instchars.csv")
```

```
year_2019_2020 <-
  read.csv("../initial_data/HMS_2019-2020_PUBLIC_instchars.csv")
year_2020_2021 <-
  read.csv("../initial_data/HMS_2020-2021_PUBLIC_instchars.csv")
year_2021_2022 <-
  read.csv("../initial_data/HMS_2021-2022_PUBLIC_instchars.csv")
year_2022_2023 <-
  read.csv("../initial_data/HMS_2022-2023_PUBLIC_instchars.csv")
year_2023_2024 <-
  read.csv("../initial_data/HMS_2023-2024_PUBLIC_instchars.csv")
year_2024_2025 <-
  read.csv("../initial_data/HMS_2024-2025_PUBLIC_instchars.csv")

# Read all XLSX files
year_2014_2015 <-
  read.xlsx("../initial_data/HMS_2014-2015_PUBLIC.xlsx", 1)
year_2015_2016 <-
  read.xlsx("../initial_data/HMS_2015-2016_PUBLIC.xlsx", 1)
```

After the data frames are pulled, the code below saves them into the RData file "all_years_untidy.RData", where they will be much quicker to retrieve from the hard drive.

```
# Code in this block was written by Henry

# Save data pulled from the files for later
save(
  year_2007_2013,
  year_2013_2014,
  year_2014_2015,
  year_2015_2016,
  year_2016_2017,
  year_2017_2018,
  year_2018_2019,
  year_2019_2020,
  year_2020_2021,
  year_2021_2022,
  year_2022_2023,
  year_2023_2024,
  year_2024_2025,
  file = "../partially_wrangled_data/all_years_untidy.RData")
```

This code loads the data frames of the form `year_20XX_20XX` from "all_years_untidy.RData".

```
# Code in this block was written by Henry

# Load the files
load(file = "../partially_wrangled_data/all_years_untidy.RData")
```

For each year, we create a new data frame of the form `syear_20XX_20XX` that only includes variables we need. Furthermore, in the case of `survey_year`, we make sure it is always recorded as a number both for consistency's sake and so future visualizations can order data points over time.

```
# Code in this block was written by Henry

# Select from the data the variables that we chose for the project
variables <- c("inst_geo", "adjust_aca_1", "deprawsc",
               "anx_score", "dx_dep", "dx_anx", "dx_ax", "GEOGRAPHY")
degs <- c("deg_ass", "deg_bach", "deg_mast", "deg_jd", "deg_md", "deg_phd")
degrees <- c("degree_ass", "degree_bach", "degree_ma",
             "degree_jd", "degree_md", "degree_phd")

# Some of the years don't have all the variables we need
syear_2007_2013 <- year_2007_2013 |>
  select(any_of(c(variables, degs)), survey_year)

syear_2013_2014 <- year_2013_2014 |>
  select(any_of(c(variables, degs))) |>
  mutate(survey_year = 2014)

syear_2014_2015 <- year_2014_2015 |>
  select(any_of(c(variables, degs))) |>
  mutate(survey_year = 2015)

syear_2015_2016 <- year_2015_2016 |>
  select(any_of(c(variables, degrees))) |>
  mutate(survey_year = 2016)

syear_2016_2017 <- year_2016_2017 |>
  select(any_of(c(variables, degs))) |>
  mutate(survey_year = 2017)

syear_2017_2018 <- year_2017_2018 |>
```

```
  select(any_of(c(variables, degs))) |>
  mutate(survey_year = 2018)

syear_2018_2019 <- year_2018_2019 |>
  select(any_of(c(variables, degs))) |>
  mutate(survey_year = 2019)

syear_2019_2020 <- year_2019_2020 |>
  select(any_of(c(variables, degs))) |>
  mutate(survey_year = 2020)

syear_2020_2021 <- year_2020_2021 |>
  select(any_of(c(variables, degs))) |>
  mutate(survey_year = 2021)

syear_2021_2022 <- year_2021_2022 |>
  select(any_of(c(variables, degrees))) |>
  mutate(survey_year = 2022)

syear_2022_2023 <- year_2022_2023 |>
  select(any_of(c(variables, degrees))) |>
  mutate(survey_year = 2023)

syear_2023_2024 <- year_2023_2024 |>
  select(any_of(c(variables, degrees))) |>
  mutate(survey_year = 2024)

syear_2024_2025 <- year_2024_2025 |>
  select(any_of(c(variables, degrees))) |>
  mutate(survey_year = 2025)
```

New data frames of the form `fyear_20XX_20XX` are created from those of the form `syear_20XX_20XX` from the last code block. Values meaning the same thing under a column are recorded differently by academic year. The code below mutates variables such that:

- anxiety diagnosis is recorded under `dx_anx` with "0" for false and "1" for true;
- the variable `degree_type` records pursuing an associate's degree as "1" and a bachelor's degree as "2";
- if not recorded numerically, `adjust_aca_1` is changed such that "very easy" to "very difficult" is replaced with "1" to "5" respectively.

This step almost creates total consistency in the data among all data frames for all years. (Later on in Miranda's code, it is still required that NAs are turned into 0 for `dx_dep` and `dx_anx`.)

```
# Code in this block was written by Henry (except as otherwise noted)

fyear_2007_2013 <- syear_2007_2013 |>
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
  mutate(dx_dep = if_else(dx_dep != "", 1, 0)) |>
  mutate(dx_anx = if_else(dx_ax != "", 1, 0)) |>
  select(-dx_ax, -all_of(degs))

fyear_2013_2014 <- syear_2013_2014 |>
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
  rename(dx_anx = dx_ax) |>
  select(-all_of(degs))

fyear_2014_2015 <- syear_2014_2015 |>
```

```
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
  rename(dx_anx = dx_ax) |>
  select(-all_of(degs))

fyear_2015_2016 <- syear_2015_2016 |>
  mutate(degree_type = case_when(
    degree_phd == 1 ~ 6,
    degree_md == 1 ~ 5,
    degree_jd == 1 ~ 4,
    degree_ma == 1 ~ 3,
    degree_bach == 1 ~ 2,
    degree_ass == 1 ~ 1
  )) |>
  select(-all_of(degrees))

fyear_2016_2017 <- syear_2016_2017 |>
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
  mutate(adjust_aca_1 = case_when(
    adjust_aca_1 == "very easy" ~ 1,
    adjust_aca_1 == "easy" ~ 2,
    adjust_aca_1 == "somewhat easy" ~ 3,
    adjust_aca_1 == "somewhat difficult" ~ 4,
    adjust_aca_1 == "difficult" ~ 5,
    adjust_aca_1 == "very difficult" ~ 6
  )) |>
  mutate(dx_anx = if_else(dx_anx == "", 0, 1)) |>
  select(-all_of(degs))
```

```r
fyear_2017_2018 <- syear_2017_2018 |>
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
  select(-all_of(degs))

fyear_2018_2019 <- syear_2018_2019 |>
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
  select(-all_of(degs))

fyear_2019_2020 <- syear_2019_2020 |>
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
  select(-all_of(degs))

fyear_2020_2021 <- syear_2020_2021 |>
  mutate(degree_type = case_when(
    deg_phd == 1 ~ 6,
    deg_md == 1 ~ 5,
    deg_jd == 1 ~ 4,
    deg_mast == 1 ~ 3,
    deg_bach == 1 ~ 2,
    deg_ass == 1 ~ 1
  )) |>
```

```
  mutate(adjust_aca_1 = case_when(
    adjust_aca_1 == "Very easy" ~ 1,
    adjust_aca_1 == "Easy" ~ 2,
    adjust_aca_1 == "Somewhat easy" ~ 3,
    adjust_aca_1 == "Somewhat difficult" ~ 4,
    adjust_aca_1 == "Difficult" ~ 5,
    adjust_aca_1 == "Very Difficult" ~ 6
  )) |>
  mutate(dx_anx = if_else(dx_anx == "", 0, 1)) |>
  select(-all_of(degs)) |>
  # Replaces GEOGRAPHY with inst_geo and converts to numeric
  # This specific code is by Maxine
  mutate(inst_geo = case_when(
    GEOGRAPHY == "1" ~ 1,
    GEOGRAPHY == "2" ~ 2,
    GEOGRAPHY == "3" ~ 3,
    GEOGRAPHY == "4" ~ 4,
    GEOGRAPHY == "5" ~ 5,
    GEOGRAPHY == "6" ~ 6,
    GEOGRAPHY == "7" ~ 7,
    GEOGRAPHY == "8" ~ 8,
    GEOGRAPHY == "9" ~ 9,
    .default = NA
  )) |>
  select(-GEOGRAPHY)

fyear_2021_2022 <- syear_2021_2022 |>
  mutate(degree_type = case_when(
    degree_phd == 1 ~ 6,
    degree_md == 1 ~ 5,
    degree_jd == 1 ~ 4,
    degree_ma == 1 ~ 3,
    degree_bach == 1 ~ 2,
    degree_ass == 1 ~ 1
  )) |>
  select(-all_of(degrees))

fyear_2022_2023 <- syear_2022_2023 |>
  mutate(degree_type = case_when(
    degree_phd == 1 ~ 6,
    degree_md == 1 ~ 5,
    degree_jd == 1 ~ 4,
```

```
    degree_ma == 1 ~ 3,
    degree_bach == 1 ~ 2,
    degree_ass == 1 ~ 1
  )) |>
  select(-all_of(degrees))

fyear_2023_2024 <- syear_2023_2024 |>
  mutate(degree_type = case_when(
    degree_phd == 1 ~ 6,
    degree_md == 1 ~ 5,
    degree_jd == 1 ~ 4,
    degree_ma == 1 ~ 3,
    degree_bach == 1 ~ 2,
    degree_ass == 1 ~ 1
  )) |>
  select(-all_of(degrees))

fyear_2024_2025 <- syear_2024_2025 |>
  mutate(degree_type = case_when(
    degree_phd == 1 ~ 6,
    degree_md == 1 ~ 5,
    degree_jd == 1 ~ 4,
    degree_ma == 1 ~ 3,
    degree_bach == 1 ~ 2,
    degree_ass == 1 ~ 1
  )) |>
  select(-all_of(degrees))
```

With data frames for all years made consistent with each other, they are all joined together into the `allyears` data frame, which is saved into an RData file.

```
# Code in this block was written by Henry

# Join all years of data
allyears <- fyear_2007_2013 |>
  full_join(fyear_2013_2014) |>
  full_join(fyear_2014_2015) |>
  full_join(fyear_2015_2016) |>
  full_join(fyear_2016_2017) |>
  full_join(fyear_2017_2018) |>
  full_join(fyear_2018_2019) |>
  full_join(fyear_2019_2020) |>
```

```
  full_join(fyear_2020_2021) |>
  full_join(fyear_2021_2022) |>
  full_join(fyear_2022_2023) |>
  full_join(fyear_2023_2024) |>
  full_join(fyear_2024_2025)
save(allyears,
     file = "../partially_wrangled_data/all_years_organized.RData")
```

Loads back the `allyears` data structure from "all_years_organized.RData". This is the first code that runs for rendering a pdf and in general, as it is more efficient to store `allyears` once it is created and then retrieve it, rather than run the time-consuming code above more than once.

```
# Code in this block was written by Henry

load(file = "../partially_wrangled_data/all_years_organized.RData")
```

The following code creates six summary tables where academic adjustment scores (`adjust_aca_1`) are averaged among all people with specified anxiety scores (`anx_score`) or depression scores (`deprawsc`), with three tables for each. The tables are further separated for each institution type (`degree_type`) by degree it confers: associate's degree, bachelor's degree, and total. The data is also filtered such that students with NA values for anxiety or depression scores and academic adjustment scores are not included. Lastly, the six data frames are joined into three based on degree, which is then stored in the corresponding RDS files.

Note that the `final_wrangled_data` subfolder that contains these RDS files, as well as all future RDS files saved, is itself located in the `shiny_app` subfolder. This was required for us to publish our Shiny application.

```
# Code in this block was written by Maxine

# Gets the average academic adjustment by anxiety score for all students who
# - are not recorded as NA for academic adjustment
# - are not recorded as NA for anxiety

anx_aca_total <- allyears |>
  drop_na(adjust_aca_1, anx_score) |>
  group_by(anx_score) |> # anx_score meaning anxiety score
  summarize(
    "Avg Academic Adjustment" = mean(adjust_aca_1)
  ) |>
```

```
  mutate("Mental Condition" = "Anxiety") |>
  mutate("Degree" = "All")


# Gets the average academic adjustment by depression score for all
# students who
# - are not recorded as NA for academic adjustment
# - are not recorded as NA for depression

dep_aca_total <- allyears |>
  drop_na(adjust_aca_1, deprawsc) |>
  group_by(deprawsc) |> # deprawsc meaning depression score
  summarize(
    "Avg Academic Adjustment" = mean(adjust_aca_1)
  ) |>
  mutate("Mental Condition" = "Depression") |>
  mutate("Degree" = "All")

# Gets the average academic adjustment by anxiety score among students
# pursuing associate's degrees
anx_aca_assoc <- allyears |>
  drop_na(adjust_aca_1, anx_score) |>
  filter(degree_type == 1) |> # indicates associate's degree
  group_by(anx_score) |>
  summarize(
    "Avg Academic Adjustment" = mean(adjust_aca_1)
  ) |>
  mutate("Mental Condition" = "Anxiety") |>
  mutate("Degree" = "Associate's")

# Gets the average academic adjustment by depression score among students
# pursuing associate's degrees
dep_aca_assoc <- allyears |>
  drop_na(adjust_aca_1, deprawsc) |>
  filter(degree_type == 1) |> # indicates associate's degree
  group_by(deprawsc) |>
  summarize(
    "Avg Academic Adjustment" = mean(adjust_aca_1)
  ) |>
  mutate("Mental Condition" = "Depression") |>
  mutate("Degree" = "Associate's")
```

```r
# Gets the average academic adjustment by anxiety score among students
# pursuing bachelor's degrees
anx_aca_bach <- allyears |>
  drop_na(adjust_aca_1, anx_score) |>
  filter(degree_type == 2) |> # indicates bachelor's degree
  group_by(anx_score) |>
  summarize(
    "Avg Academic Adjustment" = mean(adjust_aca_1)
  ) |>
  mutate("Mental Condition" = "Anxiety") |>
  mutate("Degree" = "Bachelor's")

# Gets the average academic adjustment by depression score among students
# pursuing bachelor's degrees
dep_aca_bach <- allyears |>
  drop_na(adjust_aca_1, deprawsc) |>
  filter(degree_type == 2) |> # indicates bachelors's degree
  group_by(deprawsc) |>
  summarize(
    "Avg Academic Adjustment" = mean(adjust_aca_1)
  ) |>
  mutate("Mental Condition" = "Depression") |>
  mutate("Degree" = "Bachelor's")

avg_adjust_aca_undergrad <- anx_aca_total |>
  full_join(
    dep_aca_total,
    by = join_by("Avg Academic Adjustment", "Mental Condition", "Degree")
  )

avg_adjust_aca_assoc <- anx_aca_assoc |>
  full_join(
    dep_aca_assoc,
    by = join_by("Avg Academic Adjustment", "Mental Condition", "Degree")
  )

avg_adjust_aca_bach <- anx_aca_bach |>
  full_join(
    dep_aca_bach,
    by = join_by("Avg Academic Adjustment", "Mental Condition", "Degree")
  )
```

```
saveRDS(avg_adjust_aca_undergrad,
        "../shiny_app/final_wrangled_data/avg_adjust_aca_undergrad.rds")
saveRDS(avg_adjust_aca_assoc,
        "../shiny_app/final_wrangled_data/avg_adjust_aca_assoc.rds")
saveRDS(avg_adjust_aca_bach,
        "../shiny_app/final_wrangled_data/avg_adjust_aca_bach.rds")
```

A new variable, `adjust_easy`, is created in a new data frame, `modif_allyears`. It takes the value of 1 if `adjust_aca_1` is between 1-3 and a value of 0 if it is between 4-6. If `adjust_aca_1` is NA, `adjust_easy` will also be NA.

Since there is an issue with the proportion of anxiety and depression being 100% in certain years, we will turn NAs into 0 in columns `dx_anx` and `dx_dep` for all years and make the proportion be the percent of anxiety or depression diagnosis out of total responses and non-responses.

```
# Code in this block was written by Miranda

# add column adjust_easy
modif_allyears <- allyears |>
  mutate(adjust_easy = case_when(
    # adjust_easy = 1 if adjust_aca_1 is 1-3
    adjust_aca_1 >= 1 & adjust_aca_1 <= 3 ~ 1,
    # adjust_easy = 0 if adjust_aca_1 is 4-5
    adjust_aca_1 >= 4 & adjust_aca_1 <= 6 ~ 0,
    # else (if NA), keep as NA
    TRUE ~ NA
    ))

# dx_anx, dx_dep = 0 if NA
modif_allyears$dx_anx[is.na(modif_allyears$dx_anx)] <- 0
modif_allyears$dx_dep[is.na(modif_allyears$dx_dep)] <- 0
```

Then, we created a dataframe of the proportion of students who answered `adjust_easy` = 1 out of the total number of students who responded to the question on `adjust_aca_1` by year. NAs were dropped for `adjust_easy` (people who did not answer the question).

We also created a dataframe for the proportion of `adjust_easy` by both year and region. For the data frame on region, we filter for years greater or equal to 2018 since, in the years before that, there were a lot of missing data for regions.

```
# Code in this block was written by Miranda (except one line as noted)

# proportion of adjust_easy by year
prop_easy_yr <- modif_allyears |>
  #drop NAs in adjust easy
  drop_na(adjust_easy) |>
  filter(degree_type == 1 | degree_type == 2) |>
  # group by year
  group_by(survey_year, degree_type) |>
  # divide the counts of easy by total counts of adjust_easy
  # sum works bc easy is counted as 1 while difficult is 0
  summarize(prop_easy = sum(adjust_easy)/n())
```

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

```
# proportion of adjust_easy by region and year
prop_easy_geo <- modif_allyears |>
  #drop NAs in adjust easy and inst_geo
  drop_na(adjust_easy, inst_geo) |>
  filter(degree_type == 1 | degree_type == 2) |>
  # Removes years before geographic data was collected
  filter(survey_year >= 2018) |> # (added by Maxine)
  # group by region
  group_by(survey_year, inst_geo) |>
  summarize(prop_easy = sum(adjust_easy)/n())
```

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

Next, we created a data frame for the proportion of anxiety diagnosis, `dx_anx`, out of the total number of responses and non-responses by year and a separate data frame for by year and region.

We also added the count of `dx_anx` = 1 and the total count (`dx_anx` = 1 and `dx_anx` = 0 after turning NAs into 0s) as separate columns.

```
# Code in this block was written by Miranda (except one line as noted)

# anxiety diagnosis by year
prop_anx_yr <- modif_allyears |>
  drop_na(dx_anx) |>
  filter(degree_type == 1 | degree_type == 2) |>
  group_by(survey_year, degree_type) |>
  summarize(prop_anx = sum(dx_anx)/n(),
            anx = sum(dx_anx),
            total = n())
```

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

```
# anxiety diagnosis by year and region
prop_anx_geo <- modif_allyears |>
  drop_na(dx_anx, inst_geo) |>
  filter(degree_type == 1 | degree_type == 2) |>
  # Removes years before geographic data was collected (from Maxine)
  filter(survey_year >= 2018) |> # (added by Maxine)
  group_by(survey_year, inst_geo) |>
  summarize(prop_anx = sum(dx_anx)/n(),
            anx =  sum(dx_anx), total = n())
```

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

We created a similar data frame for the proportion of depression diagnosis, `dx_dep`, out of the total number of responses and non-responses by year and a separate data frame for by year and region.

We also added the count of `dx_dep` = 1 and the total count (`dx_dep` = 1 and `dx_dep` = 0 after turning NAs into 0s) as separate columns.

```
# Code in this block was written by Miranda (except one line as noted)

# depression diagnosis by year
prop_dep_yr <- modif_allyears |>
  drop_na(dx_dep) |>
  filter(degree_type == 1 | degree_type == 2) |>
  group_by(survey_year, degree_type) |>
  summarize(prop_dep = sum(dx_dep)/n(),
            dep =  sum(dx_dep),
            total = n())
```

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

```
# depression diagnosis by year and region
prop_dep_geo <- modif_allyears |>
  drop_na(dx_dep, inst_geo) |>
  filter(degree_type == 1 | degree_type == 2) |>
  # Removes years before geographic data was collected
  filter(survey_year >= 2018) |> # (added by Maxine)
  group_by(survey_year, inst_geo) |>
  summarize(prop_dep = sum(dx_dep)/n(),
            dep =  sum(dx_dep), total = n())
```

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

The code that follows joins the `prop_easy_yr`, `prop_anx_yr`, and `prop_dep_yr` data frames into one and the `prop_easy_geo`, `prop_anx_geo`, and `prop_dep_geo` data frames into one. This is important for later wrangling and when we build visualizations from the wrangled data.

```
# Code in this block was written by Maxine

# Adds labels for each type of explanatory variable
prop_easy_yr <- prop_easy_yr |>
  mutate(var = "Ease")
prop_anx_yr <- prop_anx_yr |>
  mutate(var = "Anxiety")
prop_dep_yr <- prop_dep_yr |>
  mutate(var = "Depression")

# Adds labels for each type of explantory variable
prop_easy_geo <- prop_easy_geo |>
  mutate(var = "Ease")
prop_anx_geo <- prop_anx_geo |>
  mutate(var = "Anxiety")
prop_dep_geo <- prop_dep_geo |>
  mutate(var = "Depression")

# Joins "proportion by year" data sets together
prop_yr_all <- prop_easy_yr |>
  full_join(prop_anx_yr, by = join_by(survey_year, degree_type, var)) |>
  full_join(prop_dep_yr, by = join_by(survey_year, degree_type, var, total))

# Joins "proportion by year and region" data sets together
prop_geo_all <- prop_easy_geo |>
  full_join(prop_anx_geo, by = join_by(survey_year, inst_geo, var)) |>
  full_join(prop_dep_geo, by = join_by(survey_year, inst_geo, var, total))
```

We only save `prop_yr_all` into its own RDS file, which is used for our bar graph in our Shiny app.

```
# Code in this block was written by Henry

saveRDS(prop_yr_all,
        file = "../shiny_app/final_wrangled_data/prop_yr_all.rds")
```

The following code reads in the shapefile that consists of all US states and transforms it so that it has the right projection (and doesn't create an issue with an inconsistent datum). We group states together to form 9 different regions based on the `inst_geo` variable in the HMS codebook. We had to join together the dataframe consisting of the polygon shapes for the states in the shapefile and our previously-made dataframe that had the proportion of `dx_anx`, `dx_dep`, and `adjust_easy` by region so that we can plot the region shapes and fill them in by the proportion value of the selected variable.

```
# Code in this block was written by Miranda

# read in shapefile
states_sf <- read_sf("../shiny_app/cb_2018_us_state_500k.shp") %>%
  # solves sf layer has inconsistent datum issue
  sf::st_transform('+proj=longlat +datum=WGS84')

# create dataframe for region 1 shape
region1 <- states_sf %>%
  # only include states named in region 1
  dplyr::filter(NAME %in% c("Connecticut", "Maine", "Massachusetts",
                            "New Hampshire", "Rhode Island", "Vermont")) %>%
  # group the states together into 1 single shape
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "1") |>
  # rename
  rename(geometry = "(geometry = sf::st_union(geometry))")

# create dataframe for region 2 shape
region2 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("New Jersey", "New York", "Pennsylvania")) %>%
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "2")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")

# create dataframe for region 3 shape
region3 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("Illinois", "Indiana", "Michigan",
                            "Ohio", "Wisconsin")) %>%
```

```
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "3")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")

# create dataframe for region 4 shape
region4 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("Iowa", "Kansas", "Minnesota",
                            "Missouri", "Nebraska",
                            "North Dakota", "South Dakota")) %>%
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "4")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")

# create dataframe for region 5 shape
region5 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("Delaware", "District of Columbia", "Florida",
                            "Georgia", "Maryland",
                            "North Carolina", "South Carolina",
                            "Virginia", "West Virginia")) %>%
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "5")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")

# create dataframe for region 6 shape
region6 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("Alabama", "Kentucky", "Mississippi",
                            "Tennessee")) %>%
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "6")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")
```

```r
# create dataframe for region 7 shape
region7 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("Arkansas", "Louisiana", "Oklahoma",
                            "Texas")) %>%
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "7")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")

# create dataframe for region 8 shape
region8 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("Arizona", "Colorado", "Idaho",
                            "Montana", "Nevada", "New Mexico",
                            "Utah", "Wyoming")) %>%
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup() |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "8")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")

# create dataframe for region 9 shape
region9 <- states_sf %>%
  # only include relevant states
  dplyr::filter(NAME %in% c("Alaska", "California", "Hawaii",
                            "Oregon", "Washington")) %>%
  dplyr::summarize((geometry = sf::st_union(geometry))) %>%
  ungroup()  |>
  # add inst_geo var so that we can join by this var later
  mutate(inst_geo = "9")|>
  rename(geometry = "(geometry = sf::st_union(geometry))")

# combine together all the region dataframes
allregions <- region1 |>
  rbind(region2)|>
  rbind(region3)|>
  rbind(region4)|>
  rbind(region5)|>
  rbind(region6)|>
  rbind(region7)|>
```

```
  rbind(region8)|>
  rbind(region9)

# convert inst_geo to numeric to match inst_geo type in
# prop_easy_geo, prop_anx_geo, and prop_dep_geo
allregions$inst_geo = as.numeric(allregions$inst_geo)
region1$inst_geo = as.numeric(region1$inst_geo)
region2$inst_geo = as.numeric(region2$inst_geo)
region3$inst_geo = as.numeric(region3$inst_geo)
region4$inst_geo = as.numeric(region4$inst_geo)
region5$inst_geo = as.numeric(region5$inst_geo)
region6$inst_geo = as.numeric(region6$inst_geo)
region7$inst_geo = as.numeric(region7$inst_geo)
region8$inst_geo = as.numeric(region8$inst_geo)
region9$inst_geo = as.numeric(region9$inst_geo)

# combine all regions' shape with proportion of conditions
combinedgeo <- prop_geo_all |>
  inner_join(allregions, by = "inst_geo")

# combine geometry with proportions, by region and condition

# For ease of adjustment
combinedr1_ease <- prop_easy_geo |>
  inner_join(region1, by = "inst_geo") |>
  filter(var == "Ease")

combinedr2_ease <- prop_easy_geo |>
  inner_join(region2, by = "inst_geo")|>
  filter(var == "Ease")

combinedr3_ease <- prop_easy_geo |>
  inner_join(region3, by = "inst_geo")|>
  filter(var == "Ease")

combinedr4_ease <- prop_easy_geo |>
  inner_join(region4, by = "inst_geo")|>
  filter(var == "Ease")

combinedr5_ease <- prop_easy_geo |>
  inner_join(region5, by = "inst_geo")|>
  filter(var == "Ease")
```

```r
combinedr6_ease <- prop_easy_geo |>
  inner_join(region6, by = "inst_geo")|>
  filter(var == "Ease")

combinedr7_ease <- prop_easy_geo |>
  inner_join(region7, by = "inst_geo")|>
  filter(var == "Ease")

combinedr8_ease <- prop_easy_geo |>
  inner_join(region8, by = "inst_geo")|>
  filter(var == "Ease")

combinedr9_ease <- prop_easy_geo |>
  inner_join(region9, by = "inst_geo")|>
  filter(var == "Ease")

# For anxiety
combinedr1_anx <- prop_anx_geo |>
  inner_join(region1, by = "inst_geo") |>
  filter(var == "Anxiety")

combinedr2_anx <- prop_anx_geo |>
  inner_join(region2, by = "inst_geo")|>
  filter(var == "Anxiety")

combinedr3_anx <- prop_anx_geo |>
  inner_join(region3, by = "inst_geo")|>
  filter(var == "Anxiety")

combinedr4_anx <- prop_anx_geo |>
  inner_join(region4, by = "inst_geo")|>
  filter(var == "Anxiety")

combinedr5_anx <- prop_anx_geo |>
  inner_join(region5, by = "inst_geo")|>
  filter(var == "Anxiety")

combinedr6_anx <- prop_anx_geo |>
  inner_join(region6, by = "inst_geo")|>
  filter(var == "Anxiety")

combinedr7_anx <- prop_anx_geo |>
```

```
  inner_join(region7, by = "inst_geo")|>
  filter(var == "Anxiety")

combinedr8_anx <- prop_anx_geo |>
  inner_join(region8, by = "inst_geo")|>
  filter(var == "Anxiety")

combinedr9_anx <- prop_anx_geo |>
  inner_join(region9, by = "inst_geo")|>
  filter(var == "Anxiety")

# For depression

combinedr1_dep <- prop_dep_geo |>
  inner_join(region1, by = "inst_geo") |>
  filter(var == "Depression")

combinedr2_dep <- prop_dep_geo |>
  inner_join(region2, by = "inst_geo")|>
  filter(var == "Depression")

combinedr3_dep <- prop_dep_geo |>
  inner_join(region3, by = "inst_geo")|>
  filter(var == "Depression")

combinedr4_dep <- prop_dep_geo |>
  inner_join(region4, by = "inst_geo")|>
  filter(var == "Depression")

combinedr5_dep <- prop_dep_geo |>
  inner_join(region5, by = "inst_geo")|>
  filter(var == "Depression")

combinedr6_dep <- prop_dep_geo |>
  inner_join(region6, by = "inst_geo")|>
  filter(var == "Depression")

combinedr7_dep <- prop_dep_geo |>
  inner_join(region7, by = "inst_geo")|>
  filter(var == "Depression")

combinedr8_dep <- prop_dep_geo |>
```

```
  inner_join(region8, by = "inst_geo")|>
  filter(var == "Depression")

combinedr9_dep <- prop_dep_geo |>
  inner_join(region9, by = "inst_geo")|>
  filter(var == "Depression")

# save as RDS files
saveRDS(combinedgeo,
        file = "../shiny_app/final_wrangled_data/combinedgeo.rds")
saveRDS(combinedr1_ease,
        file = "../shiny_app/final_wrangled_data/combinedr1_ease.rds")
saveRDS(combinedr2_ease,
        file = "../shiny_app/final_wrangled_data/combinedr2_ease.rds")
saveRDS(combinedr3_ease,
        file = "../shiny_app/final_wrangled_data/combinedr3_ease.rds")
saveRDS(combinedr4_ease,
        file = "../shiny_app/final_wrangled_data/combinedr4_ease.rds")
saveRDS(combinedr5_ease,
        file = "../shiny_app/final_wrangled_data/combinedr5_ease.rds")
saveRDS(combinedr6_ease,
        file = "../shiny_app/final_wrangled_data/combinedr6_ease.rds")
saveRDS(combinedr7_ease,
        file = "../shiny_app/final_wrangled_data/combinedr7_ease.rds")
saveRDS(combinedr8_ease,
        file = "../shiny_app/final_wrangled_data/combinedr8_ease.rds")
saveRDS(combinedr9_ease,
        file = "../shiny_app/final_wrangled_data/combinedr9_ease.rds")
saveRDS(combinedr1_anx,
        file = "../shiny_app/final_wrangled_data/combinedr1_anx.rds")
saveRDS(combinedr2_anx,
        file = "../shiny_app/final_wrangled_data/combinedr2_anx.rds")
saveRDS(combinedr3_anx,
        file = "../shiny_app/final_wrangled_data/combinedr3_anx.rds")
saveRDS(combinedr4_anx,
        file = "../shiny_app/final_wrangled_data/combinedr4_anx.rds")
saveRDS(combinedr5_anx,
        file = "../shiny_app/final_wrangled_data/combinedr5_anx.rds")
saveRDS(combinedr6_anx,
        file = "../shiny_app/final_wrangled_data/combinedr6_anx.rds")
saveRDS(combinedr7_anx,
        file = "../shiny_app/final_wrangled_data/combinedr7_anx.rds")
```

```
saveRDS(combinedr8_anx,
        file = "../shiny_app/final_wrangled_data/combinedr8_anx.rds")
saveRDS(combinedr9_anx,
        file = "../shiny_app/final_wrangled_data/combinedr9_anx.rds")
saveRDS(combinedr1_dep,
        file = "../shiny_app/final_wrangled_data/combinedr1_dep.rds")
saveRDS(combinedr2_dep,
        file = "../shiny_app/final_wrangled_data/combinedr2_dep.rds")
saveRDS(combinedr3_dep,
        file = "../shiny_app/final_wrangled_data/combinedr3_dep.rds")
saveRDS(combinedr4_dep,
        file = "../shiny_app/final_wrangled_data/combinedr4_dep.rds")
saveRDS(combinedr5_dep,
        file = "../shiny_app/final_wrangled_data/combinedr5_dep.rds")
saveRDS(combinedr6_dep,
        file = "../shiny_app/final_wrangled_data/combinedr6_dep.rds")
saveRDS(combinedr7_dep,
        file = "../shiny_app/final_wrangled_data/combinedr7_dep.rds")
saveRDS(combinedr8_dep,
        file = "../shiny_app/final_wrangled_data/combinedr8_dep.rds")
saveRDS(combinedr9_dep,
        file = "../shiny_app/final_wrangled_data/combinedr9_dep.rds")
```