Final Project

Savannah Hammerton

2022-12-11

```
# Load packages
library(tidyverse)
library(here)
library(naniar)
library(gt)
library(broom)

# Set ggplot2 theme for whole script
ggplot2::theme_set(ggplot2::theme_test())

# Set paths
here::here()
```

```
## [1] "/Users/savannahhammerton/Desktop/GitHub/EPID7500_Final_Project"
```

Data

The data used for this project is the "Bee Colonies" TidyTuesday data. The GitHub repository with the data itself, instructions on how to load the data into R, and explanations of the data can be found at https://github.com/rfordatascience/tidytuesday/tree/master/data/2022/2022-01-11. In order to preserve the data files I used for this project, I have downloaded the data via GitHub in another script and saved them as .rds files in this project.

The data originally comes from the USDA, and contains information on the number of bee colonies in various states during specific quarters (specified three month periods during a specified year). The data is split into two files: a `colony.csv` file (containing basic colony information) and a `stressor.csv` file (containing information on the specific stressors colonies experienced). In the `colony.csv` file, there is data on the total number of colonies, the maximum number of colonies, the number of colonies lost, the percent of total colonies lost, the number of colonies added, the number of colonies renovated, and the percent of colonies renovated. In the `stressor.csv` file, there is information of the types of stress experienced by colonies, and the percent of colonies affected by that stressor during that quarter (this allows for multiple stressors in the same quarter). Both files have three identifying variables: year, months (or, quarter), and state. Since both files have these three variables, they can be used to join the two data sets into one for easier exploration and analysis.

Data import and basic info

```
# Load data
colony <- readr::read_rds(here::here("data/colony.rds"))
stressor <- readr::read_rds(here::here("data/stressor.rds"))

# Check out the data
dplyr::glimpse(colony)
```

```
## Rows: 1,222
## Columns: 10
## $ year            <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
## $ months          <chr> "January-March", "January-March", "January-March", "Ja~
## $ state           <chr> "Alabama", "Arizona", "Arkansas", "California", "Color~
## $ colony_n        <dbl> 7000, 35000, 13000, 1440000, 3500, 3900, 305000, 10400~
## $ colony_max      <dbl> 7000, 35000, 14000, 1690000, 12500, 3900, 315000, 1050~
## $ colony_lost     <dbl> 1800, 4600, 1500, 255000, 1500, 870, 42000, 14500, 380~
## $ colony_lost_pct <dbl> 26, 13, 11, 15, 12, 22, 13, 14, 4, 4, 40, 22, 18, 23, ~
## $ colony_added    <dbl> 2800, 3400, 1200, 250000, 200, 290, 54000, 47000, 3400~
## $ colony_reno     <dbl> 250, 2100, 90, 124000, 140, NA, 25000, 9500, 760, 8000~
## $ colony_reno_pct <dbl> 4, 6, 1, 7, 1, NA, 8, 9, 7, 9, 4, 1, 2, 1, NA, 13, NA,~
```

```r
dplyr::glimpse(stressor)
```

```
## Rows: 7,332
## Columns: 5
## $ year       <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,~
## $ months     <chr> "January-March", "January-March", "January-March", "January~
## $ state      <chr> "Alabama", "Alabama", "Alabama", "Alabama", "Alabama", "Ala~
## $ stressor   <chr> "Varroa mites", "Other pests/parasites", "Disesases", "Pest~
## $ stress_pct <dbl> 10.0, 5.4, NA, 2.2, 9.1, 9.4, 26.9, 20.5, 0.1, NA, 1.8, 3.1~
```

The `colony.rds` data set has 10 variables and 1,222 observations/records, while the `stressor.rds` data set has 5 variables and 7,332 observations/records. Both data sets have both character and numeric variables. All numeric variables are integers excepting the `stress_pct` variable in the `stressor.rds` data set. This matches the information in the data dictionary supplied on the GitHub page linked above.

Data processing

I'm going to join the colony and stressor data sets into one, matching on `year`, `months`, and `state`. now so I have one final data set to work with. I don't really want any data about stressors without any colony data, so I'm going to keep all the rows in the colony dataset, using `dplyr::left_join()`. While I am manipulating the data, I am also going to create a more specific time variable. This will allow me to group on time overall later. To do this, I will create a variable who's first portion is the pasted year from the year variable, followed either by a .0, .25. .5, or .75. Then if I want to plot totally over time, everything should be in the correct order.

Since there are so many states, it is also going to be difficult to visualize anything by geographical location unless I aggregate the states into regions. I will do this by using the `state.x77` data set and `state.region` vector already loaded into R (an explanation and example can be found here: https://gexijin.github.io/learnR/the-state-dataset.html#reading-in-and-manipulating-data). I will then join the regions to the final data set by matching the state names.

```r
# Create state names and regions data set
regions <-
  cbind(data.frame(rownames(state.x77)), state.region) |>
  data.frame() |>
  dplyr::rename(state = rownames.state.x77.,
                region = state.region)

# Join data sets on year, months, and state, keeping all the rows in colony
# Also join in state regions, matching on state
# Create new time variable
savethebees <-
  dplyr::left_join(colony, stressor,
                   by = c("year", "months", "state")) |> #colony and stressor
  dplyr::left_join(regions,
                   by = "state") |> #join state regions
  dplyr::mutate(quarter = as.numeric( #create quarter variable based on months
    dplyr::case_when(
      grepl("January", months) ~ .0,
      grepl("April", months) ~ .25,
      grepl("July", months) ~ .5,
      grepl("October", months) ~ .75
    )
  ),
  time = year + quarter) |> #add year and quarter for one time variable
  dplyr::select(!quarter) #don't need quarter variable anymore, remove
# Check out the new data set
dplyr::glimpse(savethebees)
```

```
## Rows: 7,332
## Columns: 14
## $ year            <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
## $ months          <chr> "January-March", "January-March", "January-March", "Ja~
## $ state           <chr> "Alabama", "Alabama", "Alabama", "Alabama", "Alabama",~
## $ colony_n        <dbl> 7000, 7000, 7000, 7000, 7000, 7000, 35000, 35000, 3500~
## $ colony_max      <dbl> 7000, 7000, 7000, 7000, 7000, 7000, 35000, 35000, 3500~
## $ colony_lost     <dbl> 1800, 1800, 1800, 1800, 1800, 1800, 4600, 4600, 4600, ~
## $ colony_lost_pct <dbl> 26, 26, 26, 26, 26, 26, 13, 13, 13, 13, 13, 13, 11, 11~
## $ colony_added    <dbl> 2800, 2800, 2800, 2800, 2800, 2800, 3400, 3400, 3400, ~
## $ colony_reno     <dbl> 250, 250, 250, 250, 250, 250, 2100, 2100, 2100, 2100, ~
## $ colony_reno_pct <dbl> 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, ~
## $ stressor        <chr> "Varroa mites", "Other pests/parasites", "Disesases", ~
## $ stress_pct      <dbl> 10.0, 5.4, NA, 2.2, 9.1, 9.4, 26.9, 20.5, 0.1, NA, 1.8~
## $ region          <fct> South, South, South, South, South, South, West, West, ~
## $ time            <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
```

Since I now have the same number of rows in my final dataset that I did in the stressor dataset (which had exactly six times the number of rows in the colony data set), I can conclude that there are six stressors observed, and the data collectors included all of the six stressors for each year/location combination. I want to break those stressors into indicator/dummy variables so I can explore them a little more in depth. I will use `tidyr::pivot_wider()`, pulling names from the stressors themselves, and values from the percentage of colonies affected by the stressors during that time period in that location. When I do that, I will also rename the two stressors that have spaces in the names, and use `janitor::clean_names()` to make sure every variable is lower case.

```
# Pivot stressor data wider and rename variables with spaces
savethebees_wide <-
  savethebees |>
  dplyr::mutate(stressor = ifelse(stressor == "Disesases", "Diseases", stressor)) |>
  tidyr::pivot_wider(names_from = stressor, values_from = stress_pct) |>
  dplyr::rename(varroa_mites = `Varroa mites`,
                other_pests_parasites = `Other pests/parasites`) |>
  janitor::clean_names()

# Check out new data
dplyr::glimpse(savethebees_wide)
```

```
## Rows: 1,222
## Columns: 18
## $ year                   <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
## $ months                 <chr> "January-March", "January-March", "January-March~
## $ state                  <chr> "Alabama", "Arizona", "Arkansas", "California", ~
## $ colony_n               <dbl> 7000, 35000, 13000, 1440000, 3500, 3900, 305000,~
## $ colony_max             <dbl> 7000, 35000, 14000, 1690000, 12500, 3900, 315000~
## $ colony_lost            <dbl> 1800, 4600, 1500, 255000, 1500, 870, 42000, 1450~
## $ colony_lost_pct        <dbl> 26, 13, 11, 15, 12, 22, 13, 14, 4, 4, 40, 22, 18~
## $ colony_added           <dbl> 2800, 3400, 1200, 250000, 200, 290, 54000, 47000~
## $ colony_reno            <dbl> 250, 2100, 90, 124000, 140, NA, 25000, 9500, 760~
## $ colony_reno_pct        <dbl> 4, 6, 1, 7, 1, NA, 8, 9, 7, 9, 4, 1, 2, 1, NA, 1~
## $ region                 <fct> South, West, South, West, West, Northeast, South~
## $ time                   <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
## $ varroa_mites           <dbl> 10.0, 26.9, 17.6, 24.7, 14.6, 2.5, 22.3, 6.2, 38~
## $ other_pests_parasites <dbl> 5.4, 20.5, 11.4, 7.2, 0.9, 1.4, 13.5, 4.9, 37.7,~
## $ diseases               <dbl> NA, 0.1, 1.5, 3.0, 1.8, NA, 0.8, 3.3, 1.6, 12.5,~
## $ pesticides             <dbl> 2.2, NA, 3.4, 7.5, 0.6, NA, 8.9, 2.6, NA, 4.8, 0~
## $ other                  <dbl> 9.1, 1.8, 1.0, 6.5, 2.6, 21.2, 5.1, 4.8, 2.0, 8.~
## $ unknown                <dbl> 9.4, 3.1, 1.0, 2.8, 5.9, 2.4, 4.4, 10.5, NA, 4.9~
```

I also want to create functions to rename variables for plots and tables. This will allow me to keep the variable names easy to type when I'm just exploring and running analyses, but make things easier to understand when presenting results.

```r
# Create function to rename variables in long (initial) dataset
rename_long <- function(data) {
  data |>
    dplyr::rename(Year = year,
                  Quarter = months,
                  State = state,
                  Region = region,
                  Time = time,
                  `Number of colonies` = colony_n,
                  `Maximum colonies` = colony_max,
                  `Colonies lost` = colony_lost,
                  `Percentage of total colonies lost` = colony_lost_pct,
                  `Colonies added` = colony_added,
                  `Colonies renovated` = colony_reno,
                  `Percent of colonies renoavated` = colony_reno_pct,
                  `Stress type` = stressor,
                  `Percent of colonies affected by stressor in quarter` =
                    stress_pct)
}

# Test the new function
savethebees |>
  rename_long() |>
  dplyr::glimpse()
```

```
## Rows: 7,332
## Columns: 14
## $ Year                                                  <dbl> 2015, 2015, 2015~
## $ Quarter                                               <chr> "January-March",~
## $ State                                                 <chr> "Alabama", "Alab~
## $ `Number of colonies`                                  <dbl> 7000, 7000, 7000~
## $ ‘Maximum colonies‘                                     <dbl> 7000, 7000, 7000~
## $ ‘Colonies lost‘                                        <dbl> 1800, 1800, 1800~
## $ ‘Percentage of total colonies lost‘                    <dbl> 26, 26, 26, 26, ~
## $ ‘Colonies added‘                                       <dbl> 2800, 2800, 2800~
## $ ‘Colonies renovated‘                                   <dbl> 250, 250, 250, 2~
## $ ‘Percent of colonies renoavated‘                       <dbl> 4, 4, 4, 4, 4, 4~
## $ ‘Stress type‘                                          <chr> "Varroa mites", ~
## $ ‘Percent of colonies affected by stressor in quarter‘ <dbl> 10.0, 5.4, NA, 2~
## $ Region                                                 <fct> South, South, So~
## $ Time                                                   <dbl> 2015, 2015, 2015~
```

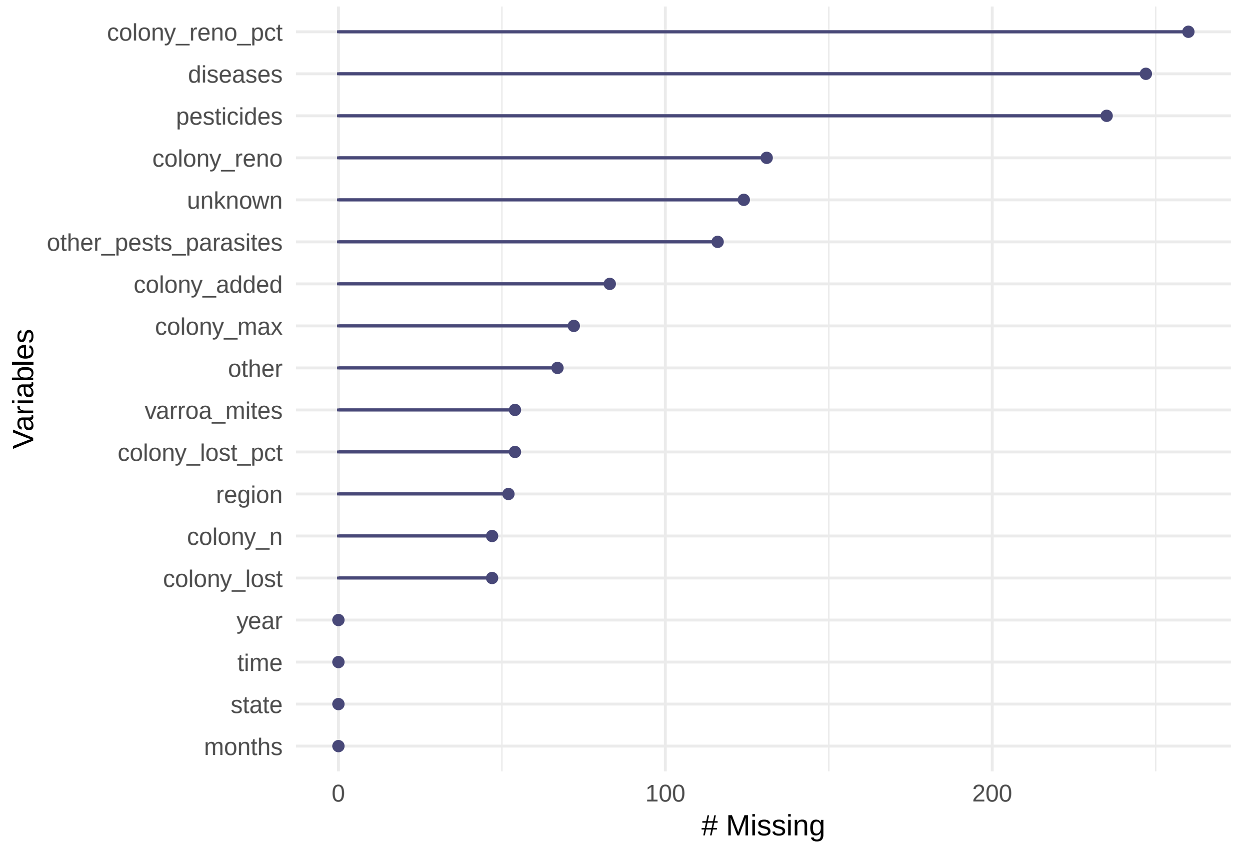
```
# Create function to rename variables in qide dataset
rename_wide <- function(data) {
    data |>
    dplyr::rename(Year = year,
                  Quarter = months,
                  State = state,
                  Region = region,
                  Time = time,
                  `Number of colonies` = colony_n,
                  `Maximum colonies` = colony_max,
                  `Colonies lost` = colony_lost,
                  `Percentage of total colonies lost` = colony_lost_pct,
                  `Colonies added` = colony_added,
                  `Colonies renovated` = colony_reno,
                  `Percent of colonies renoavated` = colony_reno_pct,
                  `Varroa mites` = varroa_mites,
                  `Other pests and parasites` = other_pests_parasites,
                  `Diseases` = diseases,
                  `Pesticides` = pesticides,
                  `Other` = other,
                  `Unknown` = unknown)

}

# Test the new function
savethebees_wide |>
  rename_wide() |>
  dplyr::glimpse()
```

```
## Rows: 1,222
## Columns: 18
## $ Year                                <dbl> 2015, 2015, 2015, 2015, 2015, 2015~
## $ Quarter                             <chr> "January-March", "January-March", ~
## $ State                               <chr> "Alabama", "Arizona", "Arkansas", ~
## $ ‘Number of colonies‘                <dbl> 7000, 35000, 13000, 1440000, 3500,~
## $ ‘Maximum colonies‘                  <dbl> 7000, 35000, 14000, 1690000, 12500~
## $ ‘Colonies lost‘                     <dbl> 1800, 4600, 1500, 255000, 1500, 87~
## $ ‘Percentage of total colonies lost‘ <dbl> 26, 13, 11, 15, 12, 22, 13, 14, 4,~
## $ ‘Colonies added‘                    <dbl> 2800, 3400, 1200, 250000, 200, 290~
## $ ‘Colonies renovated‘                <dbl> 250, 2100, 90, 124000, 140, NA, 25~
## $ ‘Percent of colonies renoavated‘    <dbl> 4, 6, 1, 7, 1, NA, 8, 9, 7, 9, 4, ~
## $ Region                              <fct> South, West, South, West, West, No~
## $ Time                                <dbl> 2015, 2015, 2015, 2015, 2015, 2015~
## $ ‘Varroa mites‘                      <dbl> 10.0, 26.9, 17.6, 24.7, 14.6, 2.5,~
## $ `Other pests and parasites`         <dbl> 5.4, 20.5, 11.4, 7.2, 0.9, 1.4, 13~
## $ Diseases                            <dbl> NA, 0.1, 1.5, 3.0, 1.8, NA, 0.8, 3~
## $ Pesticides                          <dbl> 2.2, NA, 3.4, 7.5, 0.6, NA, 8.9, 2~
## $ Other                               <dbl> 9.1, 1.8, 1.0, 6.5, 2.6, 21.2, 5.1~
## $ Unknown                             <dbl> 9.4, 3.1, 1.0, 2.8, 5.9, 2.4, 4.4,~
```

The data set is now back to the number of rows in the initial colony data set, which makes sense as the stressor variables were what were making the dataset longer. I'm now going to check for missingness using `naniar::gg_miss_var()`, which will show me the number of data points missing for each variable.

```
# Check for missingness
naniar::gg_miss_var(savethebees_wide)
```

```
## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.
```



The variable with the most missing data points is `colony_reno_pct`, which is the percentage of colonies renovated. I'm going to check what the range is of that variable, and see if it's possible that the NA's actually just mean it should be zero. While I'm at it, I'm going to do this for all the variables since all but the identifier variables have some missing values. While region is an identifying variable and does have missing data, I expected this as the data dictionary states that there are "United States" and "Other States" options, which will not have matched with any regions, so I will leave those be.

```
# See the minimum values for each variable
savethebees_wide |>
  rename_wide() |>
  dplyr::select(!c(Year, Quarter, State, Time, Region)) |>
  dplyr::summarise(dplyr::across(.cols = everything(),
                                 .fns = min, na.rm = TRUE)) |>
  t() |>
  as.data.frame() |>
  tibble::rownames_to_column(var = "Variable") |>
  dplyr::rename("Minimium" = V1) |>
  gt::gt()
```

Variable	Minimium
Number of colonies	1300.0
Maximum colonies	1700.0
Colonies lost	20.0
Percentage of total colonies lost	1.0
Colonies added	10.0
Colonies renovated	10.0
Percent of colonies renoavated	1.0
Varroa mites	0.1
Other pests and parasites	0.1
Diseases	0.1
Pesticides	0.1
Other	0.1
Unknown	0.1

Since not a single variable contains the value zero, I am going to function under the assumption that NAs just mean something like "zero observed." This is a risky/bold assumption, but I think it will allow me to visualize things more easily.

For each data set, I will select everything but the character variables, and then mutate across all those remaining variables, replacing NAs with zeros.

```
# Replace NAs with zeros
savethebees <-
  savethebees |>
  dplyr::mutate(dplyr::across(!c(state, months, year, stressor, time, region),
                              ~ tidyr::replace_na(.x, 0)))

# Replace NAs with zeros
savethebees_wide <-
  savethebees_wide |>
  dplyr::mutate(dplyr::across(!c(year, months, state, time, region),
                               ~ tidyr::replace_na(.x, 0)))

# Recheck minimums
savethebees_wide |>
  rename_wide() |>
  dplyr::summarise(dplyr::across(!c(Year, Quarter, State, Time, Region),
                                 .fns = min, na.rm = TRUE)) |>
  t() |>
  as.data.frame() |>
  tibble::rownames_to_column(var = "Variable") |>
  dplyr::rename("Minimium" = V1) |>
  gt::gt()
```

Variable	Minimium
Number of colonies	0
Maximum colonies	0
Colonies lost	0
Percentage of total colonies lost	0
Colonies added	0
Colonies renovated	0
Percent of colonies renoavated	0
Varroa mites	0
Other pests and parasites	0
Diseases	0
Pesticides	0
Other	0
Unknown	0

Now I'm going to check the final data sets one last time to make sure everything looks like it should.

```
# Check the long data set
dplyr::glimpse(savethebees)
```

```
## Rows: 7,332
## Columns: 14
## $ year           <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
## $ months         <chr> "January-March", "January-March", "January-March", "Ja~
## $ state          <chr> "Alabama", "Alabama", "Alabama", "Alabama", "Alabama",~
## $ colony_n       <dbl> 7000, 7000, 7000, 7000, 7000, 7000, 35000, 35000, 3500~
## $ colony_max     <dbl> 7000, 7000, 7000, 7000, 7000, 7000, 35000, 35000, 3500~
## $ colony_lost    <dbl> 1800, 1800, 1800, 1800, 1800, 1800, 4600, 4600, 4600, ~
## $ colony_lost_pct <dbl> 26, 26, 26, 26, 26, 26, 13, 13, 13, 13, 13, 13, 11, 11~
## $ colony_added   <dbl> 2800, 2800, 2800, 2800, 2800, 2800, 3400, 3400, 3400, ~
## $ colony_reno    <dbl> 250, 250, 250, 250, 250, 250, 2100, 2100, 2100, 2100, ~
## $ colony_reno_pct <dbl> 4, 4, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, ~
## $ stressor       <chr> "Varroa mites", "Other pests/parasites", "Disesases", ~
## $ stress_pct     <dbl> 10.0, 5.4, 0.0, 2.2, 9.1, 9.4, 26.9, 20.5, 0.1, 0.0, 1~
## $ region         <fct> South, South, South, South, South, South, West, West, ~
## $ time           <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
```

```
# Check the wide data set
dplyr::glimpse(savethebees_wide)
```

```
## Rows: 1,222
## Columns: 18
## $ year                 <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
## $ months               <chr> "January-March", "January-March", "January-March~
## $ state                <chr> "Alabama", "Arizona", "Arkansas", "California", ~
## $ colony_n             <dbl> 7000, 35000, 13000, 1440000, 3500, 3900, 305000,~
## $ colony_max           <dbl> 7000, 35000, 14000, 1690000, 12500, 3900, 315000~
## $ colony_lost          <dbl> 1800, 4600, 1500, 255000, 1500, 870, 42000, 1450~
## $ colony_lost_pct      <dbl> 26, 13, 11, 15, 12, 22, 13, 14, 4, 4, 40, 22, 18~
## $ colony_added         <dbl> 2800, 3400, 1200, 250000, 200, 290, 54000, 47000~
## $ colony_reno          <dbl> 250, 2100, 90, 124000, 140, 0, 25000, 9500, 760,~
## $ colony_reno_pct      <dbl> 4, 6, 1, 7, 1, 0, 8, 9, 7, 9, 4, 1, 2, 1, 0, 13,~
## $ region               <fct> South, West, South, West, West, Northeast, South~
## $ time                  <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, ~
## $ varroa_mites          <dbl> 10.0, 26.9, 17.6, 24.7, 14.6, 2.5, 22.3, 6.2, 38~
## $ other_pests_parasites <dbl> 5.4, 20.5, 11.4, 7.2, 0.9, 1.4, 13.5, 4.9, 37.7,~
## $ diseases              <dbl> 0.0, 0.1, 1.5, 3.0, 1.8, 0.0, 0.8, 3.3, 1.6, 12.~
## $ pesticides            <dbl> 2.2, 0.0, 3.4, 7.5, 0.6, 0.0, 8.9, 2.6, 0.0, 4.8~
## $ other                 <dbl> 9.1, 1.8, 1.0, 6.5, 2.6, 21.2, 5.1, 4.8, 2.0, 8.~
## $ unknown               <dbl> 9.4, 3.1, 1.0, 2.8, 5.9, 2.4, 4.4, 10.5, 0.0, 4.~
```

Everything looks good now, so I can move on to the visualizations and analyses.

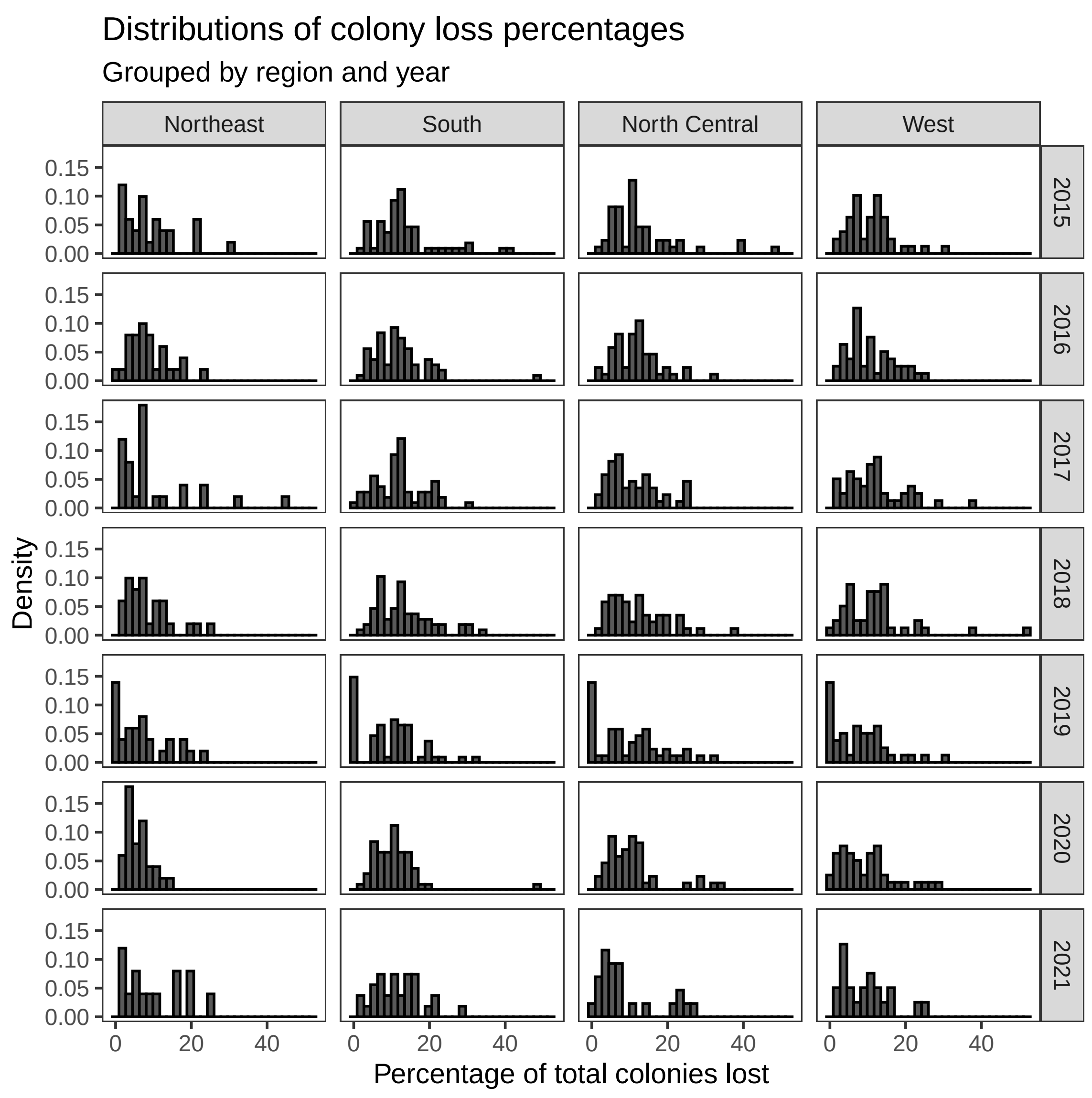
Visualizations

I mostly want to see how the colonies have been faring, and what has seemed to harm them. To visualize this, I'll make two plot grids, both of which will be grouped by region (50 states will just be too much to visualize at once) - in both grids, the columns will be the regions.

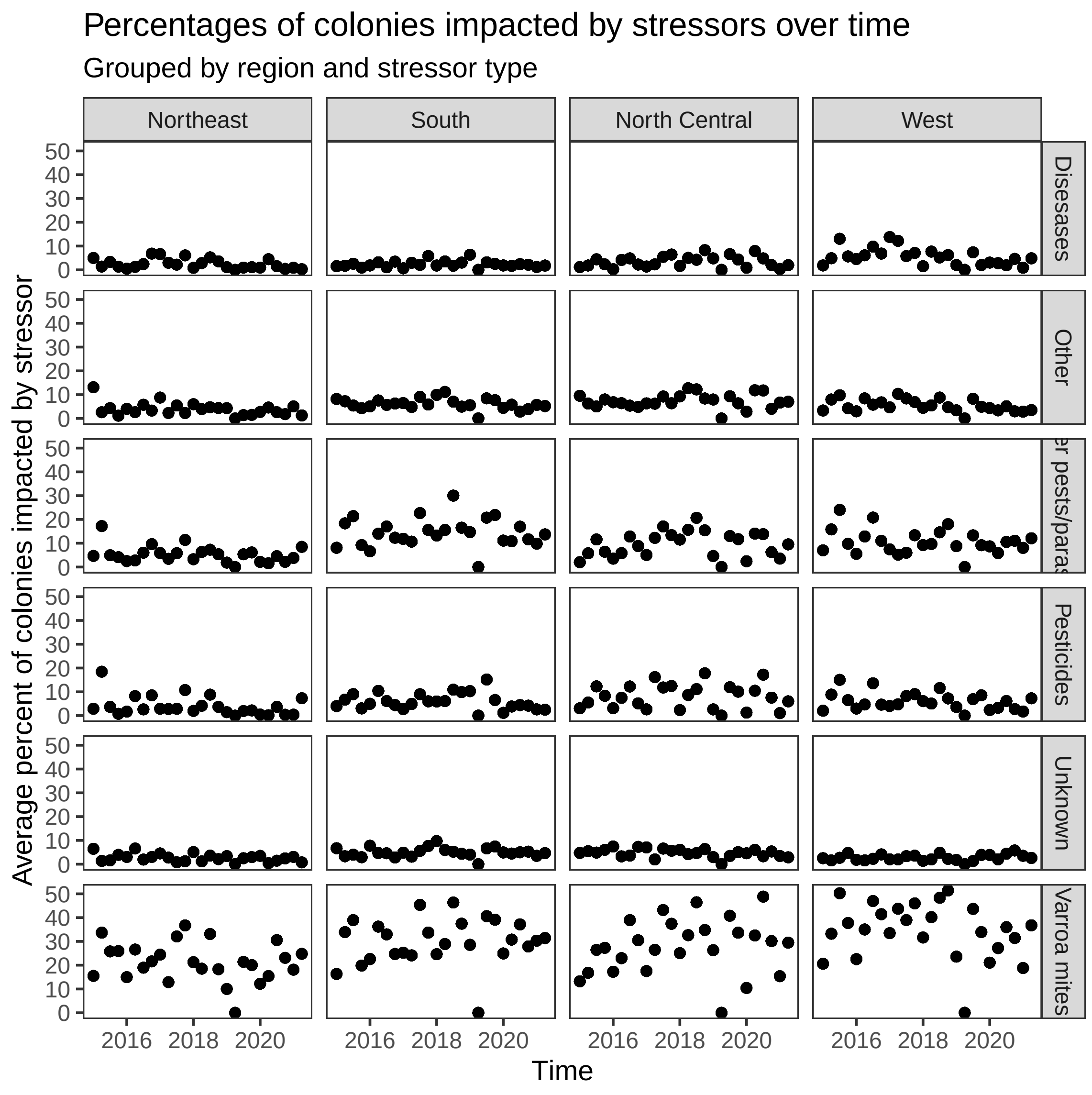
The first plot grid will be histograms of the percentage of total colonies lost. The rows of this grid will be the year, the x-axis will be the percentage of total colonies lost, and the y-axis will be the density.

The second plot will show the percentages of colonies impacted by stressors over time. Here, the grid rows will be the different stressor types, the x-axis will be time (the time variable we created that includes quarters), and the y-axis will be the average percent of colonies impacted by that stressor (we need to take the mean so we don't end up plotting points for each individual state).

```r
savethebees |>
  rename_long() |>
  dplyr::filter(Region != "NA") |>
  ggplot2::ggplot(aes(x = `Percentage of total colonies lost`)) +
  ggplot2::geom_histogram(aes(y = ..density..),
                          color = "black") +
  ggplot2::facet_grid(Year ~ Region, scales = "fixed") +
  ggplot2::labs(title =
        "Distributions of colony loss percentages",
        subtitle = "Grouped by region and year",
        y = "Density")
```



```
savethebees |>
  rename_long() |>
  group_by(Region, Time, `Stress type`) |>
  dplyr::filter(Region != "NA") |>
  mutate(`Average percent of colonies impacted by stressor` =
           mean(`Percent of colonies affected by stressor in quarter`)) |>
  ggplot2::ggplot(aes(x = Time,
                      y = `Average percent of colonies impacted by stressor`)) +
  ggplot2::geom_point() +
  facet_grid(`Stress type` ~ Region) +
  ggplot2::labs(title =
                  "Percentages of colonies impacted by stressors over time",
                subtitle = "Grouped by region and stressor type")
```



Analysis

Based on what I saw in my plots, I'm interested to see if the percentage of colonies lost can be explained by some of the other variables I explored. Specifically, I'll look at year, region, months/quarter, and the stressors diseases, varroa mites, pesticides, and other pests/parasites. Based on the plots above and just not really knowing what they are, I'm going to leave out the "other" and "unknown" stressors. I'll put all of these into a linear regression model, and use the number of colonies as weights. Performing this analysis will allow me to see if any of the variables stand out as having an impact on the loss of bee colonies.

```
# Run multiple linear regression
mymodel <-
  lm(formula = colony_lost_pct ~ year + region + months +
     diseases + varroa_mites + pesticides + other_pests_parasites,
     weights = colony_n,
    data = savethebees_wide)

# View model summary
```

```r
summary(mymodel)
```

```
##
## Call:
## lm(formula = colony_lost_pct ~ year + region + months + diseases +
##     varroa_mites + pesticides + other_pests_parasites, data = savethebees_wide,
##     weights = colony_n)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -4038.4  -503.4   -33.2   496.2  6206.4
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             599.08163  137.51991   4.356 1.44e-05 ***
## year                     -0.29465    0.06815  -4.323 1.68e-05 ***
## regionSouth               0.99658    0.76965   1.295    0.196
## regionNorth Central       0.26664    0.76593   0.348    0.728
## regionWest               -0.38893    0.75355  -0.516    0.606
## monthsJanuary-March       5.21162    0.36861  14.139  < 2e-16 ***
## monthsJuly-September      3.55385    0.38086   9.331  < 2e-16 ***
## monthsOctober-December    3.42769    0.37402   9.165  < 2e-16 ***
## diseases                  0.01097    0.02955   0.371    0.711
## varroa_mites              0.06958    0.01176   5.915 4.40e-09 ***
## pesticides                0.09576    0.01915   5.001 6.64e-07 ***
## other_pests_parasites    -0.02301    0.01804  -1.275    0.202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1078 on 1113 degrees of freedom
##   (52 observations deleted due to missingness)
## Multiple R-squared:  0.2325, Adjusted R-squared:  0.2249
## F-statistic: 30.65 on 11 and 1113 DF,  p-value: < 2.2e-16
```

```r
# View model output/estimates
mymodel |>
  broom::tidy() |>
  knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	599.0816334	137.5199093	4.3563266	0.0000144
year	-0.2946456	0.0681548	-4.3231841	0.0000168
regionSouth	0.9965765	0.7696527	1.2948393	0.1956442
regionNorth Central	0.2666419	0.7659342	0.3481264	0.7278111
regionWest	-0.3889297	0.7535523	-0.5161283	0.6058673
monthsJanuary-March	5.2116232	0.3686053	14.1387618	0.0000000
monthsJuly-September	3.5538520	0.3808573	9.3311914	0.0000000
monthsOctober-December	3.4276927	0.3740159	9.1645640	0.0000000
diseases	0.0109694	0.0295520	0.3711891	0.7105673
varroa_mites	0.0695850	0.0117633	5.9154172	0.0000000
pesticides	0.0957597	0.0191497	5.0005740	0.0000007
other_pests_parasites	-0.0230063	0.0180391	-1.2753555	0.2024496

Based on the results of the linear regression, the variables/predictors that have statistically significant relationships ($\alpha < 0.05$) with the percent of colonies lost, while controlling for all the other predictors, are year, all of the quarters, varroa mites, and pesticides. All three of the quarters listed appear to be associated with increased colony loss. This makes sense as the reference quarter is April-June, and springtime is prime bee season. The year estimate being negative indicates decreased colony loss as years increase. Finally, varroa mites and pesticides are both positively associated with colony loss, indicating that as verroa mites and pesticides increase, colony loss also increases.

It is also important to note that linear regression was not the most appropriate model to use for this kind of data. However, for the sake of staying within what we covered in class, it seemed the best option for this project. Also, as this is all aggregated data, we cannot draw any conclusions about individual colonies.

Session Info

```
sessioninfo::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.2.1 (2022-06-23)
##  os       macOS Big Sur ... 10.16
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/New_York
##  date     2022-12-11
##  pandoc   2.19.2 @ /Applications/RStudio.app/Contents/MacOS/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -------------------------------------------------------------------
##  package       * version date (UTC) lib source
##  assertthat      0.2.1   2019-03-21 [1] CRAN (R 4.2.0)
##  backports       1.4.1   2021-12-13 [1] CRAN (R 4.2.0)
##  broom         * 1.0.1   2022-08-29 [1] CRAN (R 4.2.0)
##  cellranger      1.1.0   2016-07-27 [1] CRAN (R 4.2.0)
##  cli             3.4.1   2022-09-23 [1] CRAN (R 4.2.0)
##  colorspace      2.0-3   2022-02-21 [1] CRAN (R 4.2.0)
##  crayon          1.5.2   2022-09-29 [1] CRAN (R 4.2.0)
##  DBI             1.1.3   2022-06-18 [1] CRAN (R 4.2.0)
##  dbplyr          2.2.1   2022-06-27 [1] CRAN (R 4.2.0)
##  digest          0.6.30  2022-10-18 [1] CRAN (R 4.2.0)
##  dplyr         * 1.0.10  2022-09-01 [1] CRAN (R 4.2.0)
##  ellipsis        0.3.2   2021-04-29 [1] CRAN (R 4.2.0)
##  evaluate        0.16    2022-08-09 [1] CRAN (R 4.2.0)
##  fansi           1.0.3   2022-03-24 [1] CRAN (R 4.2.0)
##  farver          2.1.1   2022-07-06 [1] CRAN (R 4.2.0)
##  fastmap         1.1.0   2021-01-25 [1] CRAN (R 4.2.0)
##  forcats       * 0.5.2   2022-08-19 [1] CRAN (R 4.2.0)
##  fs              1.5.2   2021-12-08 [1] CRAN (R 4.2.0)
##  gargle          1.2.1   2022-09-08 [1] CRAN (R 4.2.0)
##  generics        0.1.3   2022-07-05 [1] CRAN (R 4.2.0)
##  ggplot2       * 3.3.6   2022-05-03 [1] CRAN (R 4.2.0)
##  glue            1.6.2   2022-02-24 [1] CRAN (R 4.2.0)
##  googledrive     2.0.0   2021-07-08 [1] CRAN (R 4.2.0)
##  googlesheets4    1.0.1   2022-08-13 [1] CRAN (R 4.2.0)
##  gt             * 0.7.0   2022-08-25 [1] CRAN (R 4.2.0)
##  gtable           0.3.1   2022-09-01 [1] CRAN (R 4.2.0)
##  haven            2.5.1   2022-08-22 [1] CRAN (R 4.2.0)
##  here           * 1.0.1   2020-12-13 [1] CRAN (R 4.2.0)
##  highr            0.9     2021-04-16 [1] CRAN (R 4.2.0)
##  hms              1.1.2   2022-08-19 [1] CRAN (R 4.2.0)
##  htmltools        0.5.3   2022-07-18 [1] CRAN (R 4.2.0)
##  httr             1.4.4   2022-08-17 [1] CRAN (R 4.2.0)
##  janitor          2.1.0   2021-01-05 [1] CRAN (R 4.2.0)
##  jsonlite         1.8.3   2022-10-21 [1] CRAN (R 4.2.0)
##  knitr            1.40    2022-08-24 [1] CRAN (R 4.2.0)
##  labeling         0.4.2   2020-10-20 [1] CRAN (R 4.2.0)
##  lifecycle        1.0.3   2022-10-07 [1] CRAN (R 4.2.0)
##  lubridate        1.8.0   2021-10-07 [1] CRAN (R 4.2.0)
##  magrittr         2.0.3   2022-03-30 [1] CRAN (R 4.2.0)
##  modelr           0.1.9   2022-08-19 [1] CRAN (R 4.2.0)
##  munsell          0.5.0   2018-06-12 [1] CRAN (R 4.2.0)
##  naniar         * 0.6.1   2021-05-14 [1] CRAN (R 4.2.0)
##  pillar           1.8.1   2022-08-19 [1] CRAN (R 4.2.0)
##  pkgconfig        2.0.3   2019-09-22 [1] CRAN (R 4.2.0)
##  purrr          * 0.3.4   2020-04-17 [1] CRAN (R 4.2.0)
##  R6               2.5.1   2021-08-19 [1] CRAN (R 4.2.0)
##  readr          * 2.1.2   2022-01-30 [1] CRAN (R 4.2.0)
##  readxl           1.4.1   2022-08-17 [1] CRAN (R 4.2.0)
##  reprex           2.0.2   2022-08-17 [1] CRAN (R 4.2.0)
##  rlang            1.0.6   2022-09-24 [1] CRAN (R 4.2.0)
##  rmarkdown        2.16    2022-08-24 [1] CRAN (R 4.2.0)
##  rprojroot        2.0.3   2022-04-02 [1] CRAN (R 4.2.0)
##  rstudioapi       0.14    2022-08-22 [1] CRAN (R 4.2.0)
##  rvest            1.0.3   2022-08-19 [1] CRAN (R 4.2.0)
##  scales           1.2.1   2022-08-20 [1] CRAN (R 4.2.0)
##  sessioninfo      1.2.2   2021-12-06 [1] CRAN (R 4.2.0)
##  snakecase        0.11.0  2019-05-25 [1] CRAN (R 4.2.0)
##  stringi          1.7.8   2022-07-11 [1] CRAN (R 4.2.0)
##  stringr        * 1.4.1   2022-08-20 [1] CRAN (R 4.2.0)
##  tibble         * 3.1.8   2022-07-22 [1] CRAN (R 4.2.0)
##  tidyr          * 1.2.1   2022-09-08 [1] CRAN (R 4.2.0)
##  tidyselect       1.1.2   2022-02-21 [1] CRAN (R 4.2.0)
##  tidyverse      * 1.3.2   2022-07-18 [1] CRAN (R 4.2.0)
##  tzdb             0.3.0   2022-03-28 [1] CRAN (R 4.2.0)
##  utf8             1.2.2   2021-07-24 [1] CRAN (R 4.2.0)
##  vctrs            0.5.0   2022-10-22 [1] CRAN (R 4.2.0)
##  visdat           0.5.3   2019-02-15 [1] CRAN (R 4.2.0)
##  withr            2.5.0   2022-03-03 [1] CRAN (R 4.2.0)
##  xfun             0.33    2022-09-12 [1] CRAN (R 4.2.0)
##  xml2             1.3.3   2021-11-30 [1] CRAN (R 4.2.0)
##  yaml             2.3.5   2022-02-21 [1] CRAN (R 4.2.0)
##
##  [1] /Library/Frameworks/R.framework/Versions/4.2/Resources/library
##
## ------------------------------------------------------------------------------
```