EV Power - Lab 4 Project Report

Example Solution 1

Part 0: libraries

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0
──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.1     ✔ stringr   1.5.2
✔ ggplot2   4.0.0     ✔ tibble    3.3.0
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.1.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts()
──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(sf)
```

```
Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(rnaturalearth)
library(lubridate)
library(stringr)
```

Part 1: Defining Research Question

Chosen Question: How has the share of renewable energy changed from 2021-2023 across states?

Part 2: Data Preparation and Cleaning

```
# Open each data file and rename them:
# Adding a new year column because eventually when I combine all these
different years into one dataset, I am able to recognize in each row which
year the observation belongs to.
renew_2021 <- read_csv("~/stat133/ev-power-leyna-liu/data/renew-use-2021.csv")
```

```
|>
    mutate(year = 2021)
```

```
Rows: 260 Columns: 3
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (3): State, Energy_Source, Renewable_Use_2021

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
renew_2022 <- read_csv("~/stat133/ev-power-leyna-liu/data/renew-use-2022.csv")
|>
    mutate(year = 2022)
```

```
Rows: 260 Columns: 3
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (3): State, Energy_Source, Renewable_Use_2022

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
renew_2023 <- read_csv("~/stat133/ev-power-leyna-liu/data/renew-use-2023.csv")
|>
    mutate(year = 2023)
```

```
Rows: 260 Columns: 3
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (3): State, Energy_Source, Renewable_Use_2023

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
total_2021 <- read_csv("~/stat133/ev-power-leyna-liu/data/total-use-2021.csv")
|>
    mutate(year = 2021)
```

```
Rows: 5 Columns: 53
── Column specification
────────────────────────────────────────────────────────
Delimiter: ","
chr  (1): Energy_Source
dbl (52): AK, AL, AR, AZ, CA, CO, CT, DC, DE, FL, GA, HI, IA, ID, IL, IN,
KS...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
total_2022 <- read_csv("~/stat133/ev-power-leyna-liu/data/total-use-2022.csv")
|>
    mutate(year = 2022)
```

```
Rows: 5 Columns: 53
── Column specification
────────────────────────────────────────────────────────
Delimiter: ","
chr  (1): Energy_Source
dbl (52): AK, AL, AR, AZ, CA, CO, CT, DC, DE, FL, GA, HI, IA, ID, IL, IN,
KS...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
total_2023 <- read_csv("~/stat133/ev-power-leyna-liu/data/total-use-2023.csv")
|>
    mutate(year = 2023)
```

```
Rows: 5 Columns: 53
── Column specification
────────────────────────────────────────────────────────
Delimiter: ","
chr  (1): Energy_Source
dbl (52): AK, AL, AR, AZ, CA, CO, CT, DC, DE, FL, GA, HI, IA, ID, IL, IN,
KS...
```

```
ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
# Now, I am going to Standardize the column names, or basically just make sure
all the column names are the same/consistent throughout all datasets
# So, I am going to make all column names lower case through all datasets
# names(dataframe) -> outputs the names of all columns
# tolower(names(dataframe)) -> makes all the column names lowercase
# names(dataframe) <- tolower(names(dataframe)): then assigning the lower case
column names back to the original column names to save it back into the
dataframe
names(renew_2021) <- tolower(names(renew_2021))
names(renew_2022) <- tolower(names(renew_2022))
names(renew_2023) <- tolower(names(renew_2023))
names(total_2021) <- tolower(names(total_2021))
names(total_2022) <- tolower(names(total_2022))
names(total_2023) <- tolower(names(total_2023))
```

```
# Now that the column names are in lowercase, I will make sure all state names
are in lower case too (in the State column), to make sure state names are
consistent across all files
renew_2021 <- renew_2021 |>
    mutate(state = tolower(state))
renew_2022 <- renew_2022 |>
    mutate(state = tolower(state))
renew_2023 <- renew_2023 |>
    mutate(state = tolower(state))

# Since, in the total_2021, total_2022, total_2023 datasets the state names
are across the columns I can't use the tolower() function across all columns
that would be way too tedious, so I will deal with them in Part 3, where I can
use pivots.
```

```
# I noticed how in the total_2021, total_2022, total_2023 datasets, the
Energy_Source column's cells have different spellings for each energy source
(ex. Coal vs. ), so to make it all the same I will use strings.

# Here, the str_replace() follows this format:
# str_replace(dataset or column you want to change, what you want to change in
that column or dataset, and what you want to change that word into)
# In: "Natural Gas.*" -> the . is a wildcard meaning that it will output any
character followed by gas, and the * means any amount, so together it means
any amount of characters after gas
total_2021 <- total_2021 |>
```

```
    mutate(
        energy_source = str_replace(energy_source, "Coal", "coal"),
        energy_source = str_replace(energy_source, "Natural Gas.*", "natural
gas"),
        energy_source = str_replace(energy_source, "Petroleum.*",
"petroleum"),
        energy_source = str_replace(energy_source, "nuclear", "nuclear"),
        energy_source = str_replace(energy_source, "total_renewable_energy",
"total_renewable_energy")
    )
# Now, I will repeat this process with total_2022 and total_2023
total_2022 <- total_2022 |>
    mutate(
        energy_source = str_replace(energy_source, "coal.*", "coal"),
        energy_source = str_replace(energy_source, "Natural-Gas", "natural
gas"),
        energy_source = str_replace(energy_source, "petroleum.*",
"petroleum"),
        energy_source = str_replace(energy_source, "Nuclear.*", "nuclear"),
        energy_source = str_replace(energy_source, "total_renewable.*",
"total_renewable_energy")
    )

total_2023 <- total_2023 |>
    mutate(
        energy_source = str_replace(energy_source, "coal.*", "coal"),
        energy_source = str_replace(energy_source, "Natural.*", "natural
gas"),
        energy_source = str_replace(energy_source, "petroleum.*",
"petroleum"),
        energy_source = str_replace(energy_source, "nuclear.*", "nuclear"),
        energy_source = str_replace(energy_source, "total renewable.*",
"total_renewable_energy")
    )
```

Part 3: Joining / Pivoting Datasets for Analysis

```
# In order to combine the different years (in total dataset) into one dataset,
I first need to pivot the total datasets so that the columns are not years


# Now, I will combine all the yearly files into single tables. The new year
column I created in Part 2 with each year listed in each row will come in
handy now.
```

Part 4: Mapping Visualization