In-Class Problem Set: Scaling Plots with Overdispersed Election Data (R + GitHub *or* Canvas)

Goal. Use overdispersed election data to practice how axis scaling changes what patterns are visible. You will (i) obtain data from the course materials (GitHub *or* Canvas), (ii) build a reproducible workflow, (iii) make the same plot twice (raw vs scaled), (iv) write an interpretation comparing the two, and (v) submit via **GitHub *or* Canvas**.

What to submit (GitHub *or* Canvas).

- A script file: `scripts/lab.R`
- A short write-up: `outputs/writeup.md`
- Two saved figures: `figures/plot_raw.png` and `figures/plot_scaled.png`

If you submit via Canvas, upload the same files listed above as individual files (or as a single zipped folder that preserves the directory structure).

Rules.

- Work inside an **R Project**.
- Use a **sequential, hard-coded workflow** (no user-defined functions).
- Save outputs using code (`ggsave`); do not rely on screenshots.
- If you choose the **GitHub submission option**, Git commands go in the **Terminal tab** (not the R Console).

Submission options

You may submit this assignment using **either** of the following methods:

- **GitHub submission (recommended):** Commit and push your work to your GitHub repository. You will include Git proof (`git status` and `git log`) in your write-up.
- **Canvas submission:** Upload the required files directly to Canvas. You do *not* need to use GitHub if you choose this option.

Both submission methods are graded using the same rubric.

Questions

1. **Get the data (proof required).**
   (a) Choose **one** method:
   - **GitHub option:** Pull the latest version of the course repository to obtain the election dataset (and codebook, if included).
   - **Canvas option:** Download the election dataset (and codebook, if provided) from Canvas and place the dataset file in your project `data/` folder.

   (b) Confirm the dataset file exists at:

   `data/HOUSE_precinct_general.csv`

(c) **Proof (write-up):** In `outputs/writeup.md`, paste:
   - the output of `getwd()` (from inside your R Project), and
   - the output of `list.files("data")` showing the dataset file.

2. **Set up a reproducible workflow (folders + script).**
   (a) Ensure your project contains these folders (create them if missing):
   - `scripts/`
   - `outputs/`
   - `figures/`

   (b) Create a script named `scripts/lab.R`. All code for this problem set must live in this script.

   (c) **Suggested edit (important):** At the top of `scripts/lab.R`, include:
   - a short header comment describing what the script does,
   - `library(...)` calls,
   - `set.seed(123)`.

   (d) **Proof (write-up):** paste the output of `list.files()` from your project root.

3. **Load the election data and build the analysis dataset.**
   (a) Load `data/HOUSE_precinct_general.csv` into an object called `df`.

   (b) Filter the data so it includes only:
   - general election entries (stage = `"GEN"`)
   - major parties only (party_simplified in {`"DEMOCRAT"`,`"REPUBLICAN"`})
   - non-missing county information

   (c) Aggregate to the **county level** and compute:
   - `county_total_votes = DEMOCRAT + REPUBLICAN`
   - `rep_share = REPUBLICAN / (DEMOCRAT + REPUBLICAN)`

   (d) **Pseudo-code (follow, but fill in blanks).** Add something like this to `scripts/lab.R`:

```
# ---- Load + clean + aggregate (pseudo-code) ----

library(____)
library(____)

# 1) Load data
df <- read.csv("data/______.csv")

# 2) Quick inspection (pick at least two)
dim(df)
names(df)
head(df)

# 3) Filter down to the rows we want
df_keep <- df %>%
  filter(stage == "____") %>%
  filter(party_simplified %in% c("____", "____")) %>%
  filter(_____)   # county info is not missing

# 4) County-level aggregation
county_df <- df_keep %>%
  group_by(_____) %>%          # county (and maybe state if needed)
```

```
    summarize(
      dem_votes = sum(votes[party_simplified == "DEMOCRAT"], na.rm = TRUE),
      rep_votes = sum(votes[party_simplified == "REPUBLICAN"], na.rm = TRUE),
      .groups = "drop"
    ) %>%
    mutate(
      county_total_votes = dem_votes + rep_votes,
      rep_share = rep_votes / (dem_votes + rep_votes)
    )

  # 5) Checks (fill in at least two)
  nrow(county_df)
  summary(county_df$county_total_votes)
  summary(county_df$rep_share)
  # Optional: confirm rep_share is between 0 and 1
  # range(_____)
```

(e) **Suggested edit:** Use the codebook (provided with the course materials) to confirm the meaning of `votes`, `party_simplified`, and `county_name`. Cite the codebook filename in your write-up.

(f) **Proof (write-up):** report:
  - number of counties in your aggregated dataset,
  - summary of `county_total_votes`,
  - summary of `rep_share`.

4. **Plot 1: raw scale (required).**
   Create a scatter plot with:
   - x-axis: `county_total_votes`
   - y-axis: `rep_share`
   - point color: `rep_share` (continuous color scale; use this to reflect partisanship)

   Save the figure as:

   `figures/plot_raw.png`

   Suggested edit: Label axes clearly (what is being measured), and include a legend title.
5. **Plot 2: scaled version (required).**
   Make the *same* plot again, but change the scale of the x-axis to address overdispersion. Use one of:
   - log scaling (e.g., log10 x-axis), or
   - another defensible scaling choice discussed in lecture.

   Save the figure as:

   `figures/plot_scaled.png`

   Suggested edit: Make the axis label explicitly indicate the scaling choice (e.g., "log scale").
6. **Interpretation + submission (proof required).**

   (a) In `outputs/writeup.md`, write 8–12 sentences answering:
   - What is mapped to x, y, and color in both plots?
   - What is hard to see on the raw scale but easier to see on the scaled plot?
   - What (if anything) becomes harder to interpret after scaling?
   - If you had to show only one version to a general audience, which would you choose and why?

   (b) **Choose ONE submission method:**

- **GitHub option:** Commit and push your work to GitHub.
- **Canvas option:** Upload `scripts/lab.R`, `outputs/writeup.md`, `figures/plot_raw.png`, and `figures/plot_scaled.png` to Canvas.

(c) **Proof (write-up):**

- If using **GitHub**: paste
  - the output of `git status` after your commit (clean working tree), and
  - the output of `git log -1` (one line is fine).
- If using **Canvas**: paste
  - the output of `list.files("scripts")`,
  - the output of `list.files("outputs")`,
  - the output of `list.files("figures")`,
  - and write one sentence stating you submitted via Canvas.

Optional challenge (one extra)

Create a second scaled plot where you change the scale choice (e.g., compare log10 vs another scaling approach). In 3–5 sentences, explain which scaling choice better supports a clear comparison and why.

Checklist (before you leave)

- `scripts/lab.R` exists and runs top-to-bottom inside an R Project
- `figures/plot_raw.png` exists
- `figures/plot_scaled.png` exists
- `outputs/writeup.md` includes required interpretation and proofs
- Work is either committed and pushed to GitHub *or* uploaded to Canvas