Lab 03: Boots the House Down, Mama

PSTAT 100, Summer Session A 2025 with Ethan P. Marzban

MEMBER 1 (NetID 1) MEMBER 2 (NetID 2)
MEMBER 3 (NetID 3)

July 1, 2025

Required Packages

```
library(ottr)        # for checking test cases (i.e. autograding)
library(pander)      # for nicer-looking formatting of dataframe outputs
library(tidyverse)   # for graphs, data wrangling, etc.
```

Logistical Details

Logistical Details

- This lab is due by **11:59pm on Wednesday, July 2, 2025**.
- Collaboration is allowed, and encouraged!
  - If you work in groups, list ALL of your group members' names and NetIDs (not Perm Numbers) in the appropriate spaces in the YAML header above.
  - Please delete any "MEMBER X" lines in the YAML header that are not needed.
  - No more than 3 people in a group, please.
- Ensure your Lab properly renders to a `.pdf`; non-`.pdf` submissions will not be graded and will receive a score of 0.
- Ensure all test cases pass (test cases that have passed will display a message stating `"All tests passed!"`)

Lab Overview and Objectives

Welcome to another PSTAT 100 Lab! In this lab, we will cover the following:

- Exploring missingness in a dataset

- Performing PCA in `R`
- Producing and interpreting **PCA Plots** and screeplots

A Review of PCA

In **Principal Components Analysis** (PCA), we seek to project a high-dimensional (i.e. multivariable) $(n \times p)$ dataset $\mathbf{X}$ onto a subspace spanned by the vectors along which the projected data has maximal variance. These directional vectors are called the **principal components** (with the elements being called **loadings**), and are found to be the eigenvectors of the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ (assuming mean-centered data); the variance of the data projected onto the k^{th} principal component is found to be the corresponding eigenvalue λ_k. We saw that there are three main ways to obtain these principal components:

- using the SVD of $\mathbf{X}$:
- using the EVD of $\mathbf{X}^\mathsf{T}\mathbf{X}$
- using the `prcomp()` function in `R`

Given principal components $\{\vec{\boldsymbol{v}}_1, \cdots, \vec{\boldsymbol{v}}_p\}$, we can obtain a d-dimensional low-rank reconstruction of $\mathbf{X}$ by:

$$\mathbf{X}_{\text{red}} = \mathbf{X}\mathbf{V}_d\mathbf{V}_d^\mathsf{T}$$

where $\mathbf{V}_d := \begin{bmatrix} \vec{\boldsymbol{v}}_1 & \cdots & \vec{\boldsymbol{v}}_d \end{bmatrix}$ is the matrix whose columns is comprised of the first d principal components.

When the data is projected along the k^{th} principal component $\vec{\boldsymbol{v}}_k$, the resulting data has variance λ_k. Hence, the proportion of total variance contributed by the k^{th} principal component is

$$s_k := \frac{\lambda_k}{\sum_{k=1}^p \lambda_k} = \frac{\sigma_k^2}{\sum_{k=1}^p \sigma_k^2}$$

where σ_k denotes the k^{th} singular value of $\mathbf{X}$. A **screeplot** plots s_k against k, and can be used to identify a "good" value for d. Specifically, we look for an "elbow" in the screeplot, indiciating a point at which further principal components contribute negligibly to the total variance.

Introduction to the Dataset

In the US political system, Congress is divided into the **House of Representatives** and the **Senate**. Members of the House serve for two years, whereas members of the Senate serve for six. (If you're curious to learn more, the Official White House Page explains a bit further.) The dataset we will be exploring today contains the roll-call votes of the 118th House of Representatives from the first 5 months of 2024. To make things more manageable, I've provided an abridged version of the dataset that includes only results of votes in which the outcome was to either pass or fail to pass a particular piece of legislation (as opposed to, for example, votes designed to agree upon an amendment, votes on a motion to recommit, etc.) The dataset has been sourced from the Clerk of the United States House of Representatives, is stored in the `votes.csv` file located in the `data/` subfolder.

! Question 1

Read in the `votes.csv` file, and assign this to a data frame called `votes`.
Solution:

```
## replace this line with your code
votes <- read.csv("data/votes.csv")
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q1.R")})
```

```
All tests passed!
```

Data Preprocessing

There are a few bits of preprocessing we need to do. First, note that votes have been encoded verbally, as words - we'd like to convert everything to numbers. As such, let's start off by replacing all instances of a "positive" vote (either "Yea" or "Aye) with 1, and anything else ("Nea", "Not Voted", etc.) as 0.

! Question 2

In the `Votes` volumn of the `votes` dataframe, replace all instances of "Yea" or "Aye" with `1` and all other values with `0`. As a hint: look up the `ifelse()` function.

Solution:

```
## replace this line with your code
votes$Vote <- ifelse(votes$Vote %in% c("Yea", "Aye"), 1, 0)
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q2.R")})
```

```
All tests passed!
```

Ultimately, we'd like to compare votes across bills. As it stands, the format of the `votes` dataframe isn't particularly conducive to that. Let's fix that using one of the dataframe transformations we discussed last week!

! Question 3

Perform an appropriate transformation on the `votes` dataframe to create a separate column for each of the bills present in the dataset; assign this to a variable called `votes_mod`. It's up to you to figure out which transformation will achieve this!

Solution:

```
## replace this line with your code
votes_mod <- votes %>% pivot_wider(
  names_from = Bill,
  values_from = Vote
)
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q3.R")})
```

```
All tests passed!
```

We also have to contend with missing values present in this dataset. Though there typically isn't a one-size-fits-all approach for dealing with missing values, we'll discuss some strategies for handlind missingness later in this course. For now, let's get a sense of how much of the data is missing.

Tip

The function `complete.cases()` returns a logical vector, indicating which cases (rows) are complete (contain no missing values)

! Question 4

What proportion of rows in the dataframe contain missing values? Store your answer in a variable called `prop_missing`.

Solution:

```
## replace this line with your code
prop_missing <- 1 - (sum(complete.cases(votes_mod)) / nrow(votes_mod) )
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q4.R")})
```

```
All tests passed!
```

Unfortunately, the `prcomp()` function is not well-equipped to handle missing values. Thankfully, the proportion of missing values in our dataset (as you saw in Question 4 above) is quite small; therefore, to be able to proceed with this lab, we'll go ahead and remove all rows containing missing values.

Caution

Again, to stress, dropping missing values is not always recommended; we're just doing that for the purposes of this lab.

Change the `eval` option from `FALSE` to `TRUE` in the following code chunk, so a new variable called `votes_mod_complete` (containing only complete cases of the `votes_mod` dataframe) is created.

```
votes_mod_complete <- votes_mod[complete.cases(votes_mod), ]
```

PCA

It's time to start our dimension reduction! Because we have such limited time on this lab, we'll restrict ourselves to one-dimensional considerations - that is, we'll only consider projecting onto a one-dimensional subspace. In the final question, we'll examine how many dimensions are necessary to capture maximal information about the dataset as possible.

Question 5

First create a matrix `X` that contains only the numerical portion of the `voter_mod_complete` dataframe. Then, using the `prcomp()` function, create a matrix `X1` that stores the matrix resulting from projecting `X` onto the subspace spanned by the first principal component. **Important:** make sure to *standardize* your matrix - that is, not just mean-center but also rescale each column to have unit variance. **Hint:** look up the help file for `prcomp()` - there may be an argument built into the function that helps you do this...

Solution:

```
## replace this line with your code
X <- votes_mod_complete[, -c(1:4)] %>% as.matrix() %>% scale()
PCA_X <- prcomp(X, scale. = TRUE)
PC1 <- PCA_X$rotation[, 1]
X1 <- X %*% PC1
```

Answer Check:

```
# DO NOT EDIT THIS LINE
invisible({check("tests/q5.R")})
```

```
All tests passed!
```

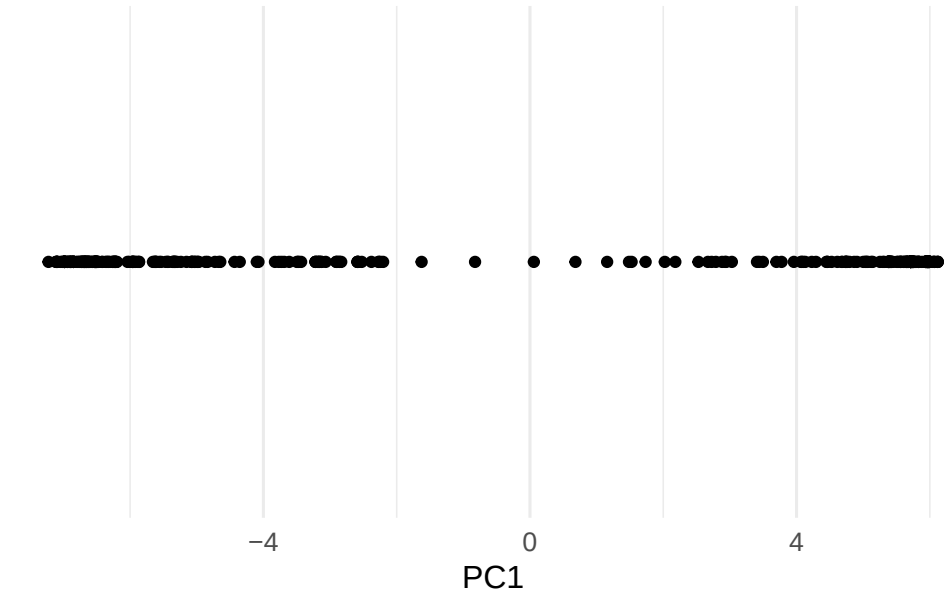
Because the original dataset has over 40 variables, it's impossible to visualize all observations at once. However, we *can* visualize our one-dimensionally-projected data!

! Question 6

Create a dotplot (i.e. a scatterplot where all points lie along a horizontal line) of the `X1` values you created in Question 5 above.

Solution:

```
## replace this line with your code
data.frame(X1) %>% ggplot(aes(x = X1, y = rep(0, length(X1)))) +
  geom_point() + xlab("PC1") + ylab("") +
  theme_minimal(base_size = 12) + ggtitle("Projection onto First PC") +
  theme(axis.text.y = element_blank(),
        panel.grid.major.y = element_blank(),
        panel.grid.minor.y = element_blank())
```



Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Recall that we use the term **score** to refer to the data post-projection; so, the plot we just created is a visualization of the first score. Interpreting this plot is absolutely crucial! First, it is important to note that each score is *not* a variable on its own, but rather a *combination* of several variables. (By definition, they are the combinations of variables that capture *maximal variance*.)

So, for instance, notice that our plot from Question 6 displays roughly two clusters of points. Because these clusters are relatively distinct (i.e. far away from one another), we see that the first PC (i.e. the axis on this plot) is doing a good job of *clustering* our data. That is, the PC captures a *lot* of variability in the data.

Now, on occasion, it may turn out that one or more of the PCs align closely with one or more variables present in the dataset. Spoiler alert: the voter dataset we've been exploring is one of them. Specifically, it's plausible to surmize that **congresspersons vote along party lines**. If that is the case, the clusters induced by the first PC should correspond closely with the party of the respective congressperson. Let's test this theory.

! Question 7

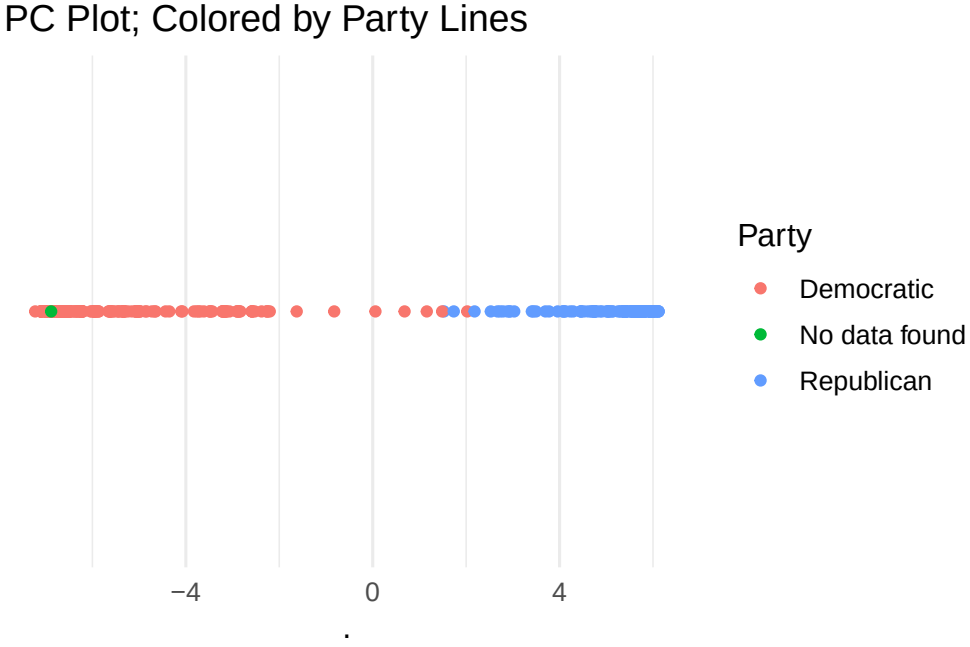
Assuming you've been careful about keeping track of the dimensions of the various dataframes we've created, the order of the rows in the `X1` matrix should be in the same order as the rows in the original `votes_mod_complete` dataframe. As such, we can column-bind the party information from the `votes_mod_complete` dataframe onto the `X1` matrix.

Copy-paste the following code into the blank code chunk below, and fill in the blanks to perform the above-column-bind operation. Then, pipe the result into a call to `ggplot()` to again produce a PC plot identical to the one you producd in Question 6 above, now coloring points by the Party of the member they represent. Does the division of points along the first PC seem to correspond well to party lines?

```
X1 %>% data.frame() %>%
  mutate(
    <fill this in>
  ) %>%
  <Your call to ggplot should go here>
```

Solution:

```
## replace this line with your code
X1 %>% data.frame() %>%
  mutate(
    Party = votes_mod_complete$Party
  ) %>%
  ggplot(aes(x = ., y = rep(0, nrow(X1)))) +
  geom_point(aes(colour = Party)) +
  theme_minimal(base_size = 12) +
  ggtitle("PC Plot; Colored by Party Lines") +
  labs(aes(colour = "Party")) +
  theme(axis.text.y = element_blank(),
        panel.grid.major.y = element_blank(),
        panel.grid.minor.y = element_blank()) + ylab("")
```



Answer Check:

There is no autograder for this question; your TA will manually check that your answers are correct.

As a spoiler for a future lab, one use of PCA is to help us navigate **missing data**.

! Question 8

In your plot for Question 7, you should note that one individual does not have a party recorded. Based on your plot, what party do you think this individual belongs to?

Solution:
Replace this line with your answer.

The green point is well within the cluster of salmon-colored points, meaning it is likely that this individual is a Democrat.

Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.

Finally, let's address the question of how many dimensions should be used in PCA. Recall that this is often answered by way of a **screeplot**, which plots the proportion of total variance captured by each PC.

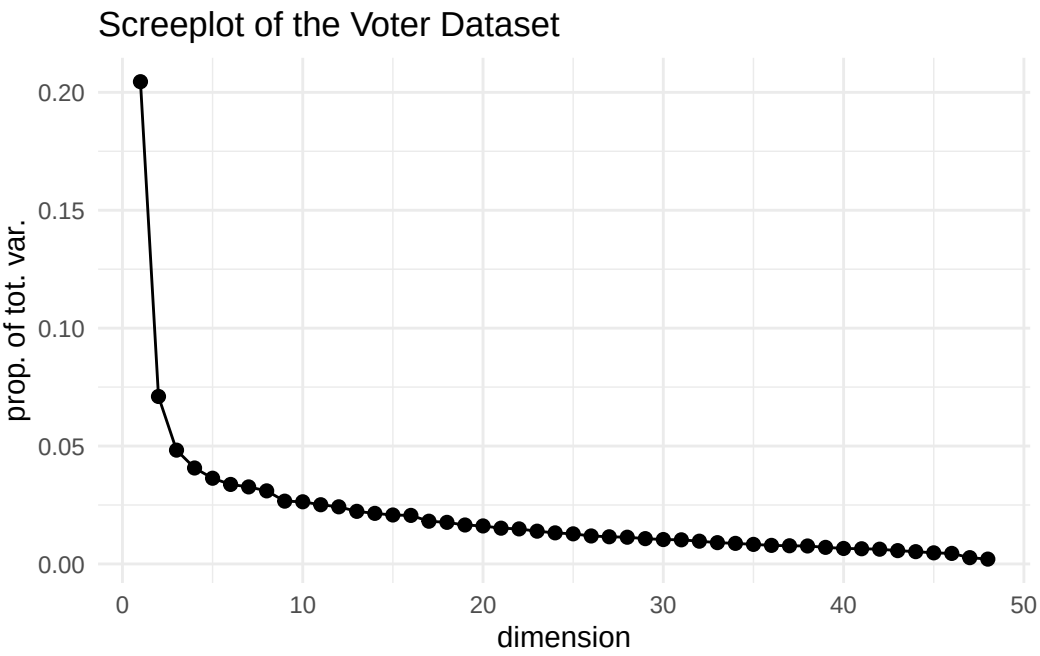
! Question 9

Using the `prcomp()` function, generate a screeplot of the `votes_mod_complete` dataframe. Use this to determine the optimal number of dimensions to include when performing PCA.

Solution:

```
## replace this line with your code
sk_PCA <- PCA_X$sdev / sum(PCA_X$sdev)
sk_SVD <- sqrt(svd(scale(X))$d^2 / (nrow(X) - 1)) / ## just to show how to use SVD
  sum(sqrt(svd(scale(X))$d^2 / (nrow(X) - 1)))

data.frame(k = 1:ncol(X), sk_PCA) %>% ggplot(aes(x = k, y = sk_PCA)) +
  geom_point(size = 2) + geom_line() +
  theme_minimal() + xlab("dimension") + ylab("prop. of tot. var.") +
  ggtitle("Screeplot of the Voter Dataset")
```



Answer Check:
There is no autograder for this question; your TA will manually check that your answers are correct.
The first three PCs capture around 95% of the total variance, so around three dimensions should be sufficient. Arguably, however, any answer between 3 and 10 (or even 3 and 20) could be considered valid.

Submission Details

Congrats on finishing this PSTAT 100 lab! Please carry out the following steps:

Submission Details

1) Check that all of your tables, plots, and code outputs are rendering correctly in your final `.pdf`.
2) Check that you passed all of the test cases (on questions that have autograders). You'll know that you passed all tests for a particular problem when you get the message "All tests passed!".
3) Submit **ONLY** your `.pdf` to Gradescope. Make sure to match pages to your questions - we'll be lenient on the first few labs, but after a while failure to match pages will result in point penalties.