```
from client.api.notebook import Notebook
ok = Notebook("hw12.ipynb")
```

Homework 12: Principal Component Analysis

In lecture we discussed how PCA can be used for dimensionality reduction. Specifically, given a high dimensional dataset, PCA allows us to:

1. Understand the rank of the data. If k principal components capture almost all of the variance, then the data is roughly rank k.
2. Create 2D scatterplots of the data. Such plots are a rank 2 representation of our data, and allow us to visually identify clusters of similar observations.

A solid geometric understanding of PCA will help you understand why PCA is able to do these two things. In this homework, we'll build that geometric intuition and look at PCA on two datasets: one where PCA works poorly, and the other where it works pretty well.

Due Date

This assignment is due **Monday, August 9th at 11:59 PM PDT**.

Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the homework, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** in the cell below.

Collaborators: ...

Score Breakdown

Question	Points
Question 1a	1
Question 1b	1
Question 1c	1
Question 1d	1
Question 1e	1
Question 2a	2
Question 2b	1
Question 2c	1
Question 2d	3
Question 2e	2
Question 3a	1

Question	Points
Question 3b	1
Question 3c	1
Question 3d	2
Question 3e	2
Question 3f	2
Question 3g	1
Question 3h	2
Question 3i	2
Total	28

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import plotly.express as px

# Note: If you're having problems with the 3d scatter plots, uncomment the two lin
#      number that is at least 4.1.1.
import plotly
plotly.__version__

     '4.4.1'
```

Question 1: PCA on 3D Data

In question 1, our goal is to see visually how PCA is simply the process of rotating the coordinate axes of our data.

The code below reads in a 3D dataset. We have named the DataFrame `surfboard` because the data resembles a surfboard when plotted in 3D space.

```
surfboard = pd.read_csv("data3d.csv")
surfboard.head(5)
```

	x	y	z
0	0.005605	2.298191	1.746604
1	-1.093255	2.457522	0.170309
2	0.060946	0.473669	-0.003543
3	-1.761945	2.151108	3.132426
4	1.950637	-0.194469	-2.101949

The cell below will allow you to view the data as a 3D scatterplot. Rotate the data around and zoom in and out using your trackpad or the controls at the top right of the figure.

You should see that the data is an ellipsoid that looks roughly like a surfboard or a [hashbrown patty](). That is, it is pretty long in one direction, pretty wide in another direction, and relatively thin along its third dimension. We can think of these as the "length", "width", and "thickness" of the surfboard data.

Observe that the surfboard is not aligned with the x/y/z axes.

If you get an error that your browser does not support webgl, you may need to restart your kernel and/or browser.

```
fig = px.scatter_3d(surfboard, x='x', y='y', z='z', range_x = [-10, 10], range_y =
fig.show()
```

To give the figure a little more visual pop, the following cell does the same plot, but also assigns a pre-determined color value (that we've arbitrarily chosen) to each point. These colors do not mean anything important, they're simply there as a visual aid.

You might find it useful to use `colorize_surfboard_data` later in this assignment.

```
def colorize_surfboard_data(df):
    colors = pd.read_csv("surfboard_colors.csv", header = None).values
    df_copy = df.copy()
    df_copy.insert(loc = 3, column = "color", value = colors)
    return df_copy

fig = px.scatter_3d(colorize_surfboard_data(surfboard), x='x', y='y', z='z', range_
fig.show()
```

Question 1a

Now that we've understood the data, let's work on understanding what PCA will do when applied to this data.

To properly perform PCA, we will first need to "center" the data so that the mean of each feature is 0.

Compute the columnwise mean of `surfboard` in the cell below, and store the result in `surfboard_mean`. You can choose to make `surfboard_mean` a numpy array or a series, whichever is more convenient for you. Regardless of what data type you use, `surfboard_mean` should have 3 means, 1 for each attribute, with the x coordinate first, then y, then z.

Then, subtract `surfboard_mean` from `surfboard`, and save the result in `surfboard_centered`. The order of the columns in `surfboard_centered` should be x, then y, then z

```
surfboard_mean = np.mean(surfboard, axis = 0)
surfboard_centered = surfboard - surfboard_mean
```

```
ok.grade("q1a");
```

```
    /content/tests/q1a.py: All tests passed!
```

✓

Question 1b

As you may recall from lecture, PCA is a specific application of the singular value decomposition (SVD) for matrices. If we have a data matrix X, we can decompose it into U, Σ and V^T such that $X = U\Sigma V^T$.

In the following cell, use the `np.linalg.svd` function to compute the SVD of `surfboard_centered`. Store the U, Σ, and V^T matrices in `u`, `s`, and `vt` respectively. This is one line of simple code, exactly like what we saw in lecture.

Hint: Set the `full_matrices` argument of `np.linalg.svd` to `False`.

[ ] ↳ *3 cells hidden*

✓

Question 1c: Total Variance

[ ] ↳ *7 cells hidden*

✓

Question 1d: Explained Variance and Scree Plots

[ ] ↳ *6 cells hidden*

✓

Question 1e: V as a Rotation Matrix

In lecture, we saw that the first column of XV contained the first principal component values for each observation, the second column of XV contained the second principal component values for each observation, and so forth.

Let's give this matrix a name: $P = XV$ is sometimes known as the "principal component matrix".

Compute the P matrix for the surfboard dataset and store it in the variable `surfboard_pcs`.

[ ] ↳ *4 cells hidden*

Visualizing the Principal Component Matrix

[ ] ↳ 4 cells hidden

Question 1 Summary

Above, we saw that the principal component matrix P is simply the original data rotated in space so that it appears axis-aligned.

Whenever we do a 2D scatter plot of only the first 2 columns of P, we are simply looking at the data from "above", i.e. so that the 3rd (or higher) PC is invisible to us.

Question 2

[ ] ↳ 6 cells hidden

Question 2a

Using PCA, we can try to visualize student performance on ALL questions simultaneously. In the cell below, create a **DataFrame** called `mid1_1st_2_pcs` that has 992 rows and 2 columns, where the first column is named `pc1` and represents the first principal component, and the second column is named `pc2` and represents the second principal component. The columns of your dataframe should be named `pc1` and `pc2`.

Reminder: make sure to center your data first!

```
mid_mean = np.mean(mid1_grades, axis = 0)
mid1_grades_centered = mid1_grades - mid_mean
u_2a, s_2a, vt_2a = np.linalg.svd(mid1_grades_centered, full_matrices = False)
mid1_1st_2_pcs = pd.DataFrame(data= (u_2a*s_2a)[:, 0:2], columns=['pc1', 'pc2'])
mid1_1st_2_pcs
```

	pc1	pc2
0	1.710289	-3.488682
1	1.984097	-1.912580
2	-3.866735	-0.610573
3	-4.045114	-3.764690

```
ok.grade("q2a");
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-63-f74810b3110e> in <module>()
----> 1 ok.grade("q2a");

                         2 frames
/usr/local/lib/python3.7/dist-packages/okgrade/grader.py in
parse_ok_test(path)
     24
     25     # Do not support point values other than 1
---> 26     assert test_spec.get('points', 1) == 1
     27
     28     test_suite = test_spec['suites'][0]

AssertionError:
```

SEARCH STACK OVERFLOW

Question 2b

In the cell below, we create a 2d scatterplot of the first two principal components of the data. Observe that the plot appears to have some sort of structure: The data shows diagonal bands. We will not explore the reasons for these diagonal bands, but we leave this as an optional exercise (Q4) at the very end of the homework if you're curious.

As with the surfboard data, we have assigned arbitrary colors to each student as a visual aid. There is no special meaning to the colors.

[ ] ↳ *5 cells hidden*

Question 2c

While the plot above captures around 39% of the variance, it's hard to interpret. One approach is to do something similar to what we did during lecture where we investigated how much each column of our data contributes to each principal component.

The function defined in the cell below plots and labels the rows of V^T.

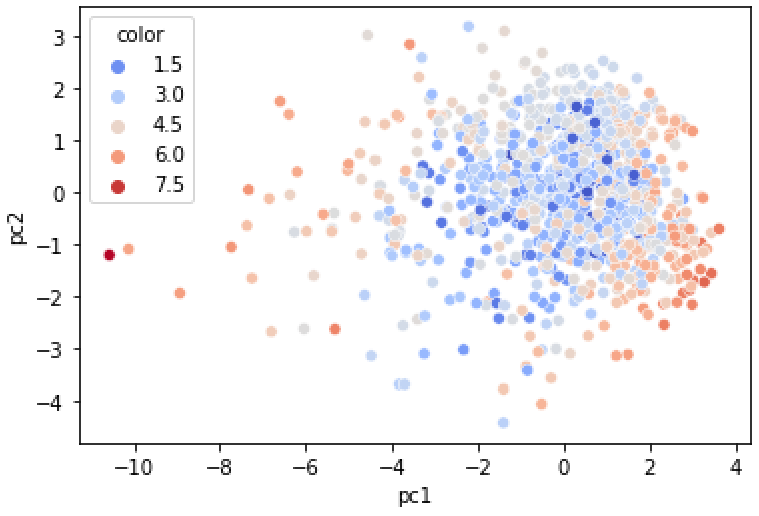
[ ] ↳ *8 cells hidden*

Question 2d

Create a 2D scatterplot of the first two principal components of `mid1_grades_centered_scaled`. Use `colorize_midterm_data` to add a `color` column to `mid1_1st_2_pcs`. Your code will be very similar to the code from problems 2a and 2b. Your result should look like this

```
u, s, vt = np.linalg.svd(mid1_grades_centered_scaled, full_matrices = False)
mid1_1st_2_pcs = pd.DataFrame(data= (u*s)[:, 0:2], columns=['pc1', 'pc2'])
mid1_1st_2_pcs
sns.scatterplot(data = colorize_midterm_data(mid1_1st_2_pcs), x = "pc1", y = "pc2"
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feef9fab310>
```



This scatterplot is quite different. The diagonal banding we saw before is gone.

By looking at the colors, we can also get some sense of how our centering process altered the locations of each student in the 2D PCA scatterplot. Observe that the blue students are still mostly close to each other, and the red points are still largely at the margins.

This plot shows relatively little structure and has no clusters. There does not appear to be much to learn here.

Question 2e

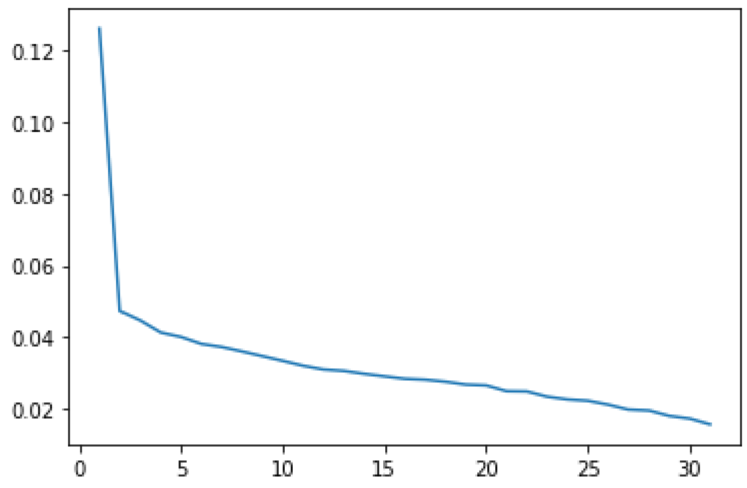
If you compute the fraction of the variance captured by this 2D scatter plot, you'll see it's only 17%, roughly 12% by the 1st PC, and roughly 5% by the 2nd PC. **In the cell below, create a scree plot showing the fraction of the variance explained by each principal componant using the data from 2d.**

Informally, we can say that our midterm scores matrix has a high rank. More formally, we can say that 2 principal components only capture a small fraction of the variance, and thus the data are not particularly amenable to 2D PCA scatterplotting.

```
np.arange(1,32)
s**2
```

```
    array([3875.61169069, 1455.78958463, 1374.50910338, 1271.68994694,
           1233.46703893, 1173.82711571, 1147.37946944, 1109.25589923,
           1069.31465589, 1028.82724284,  987.02221368,  954.72321536,
            942.71717973,  917.83437925,  895.44879036,  874.56966371,
            866.86775905,  849.51493868,  824.97267927,  818.10364192,
            768.78481026,  766.43193331,  721.36816435,  697.74009786,
            686.48011435,  653.24797644,  610.78137101,  602.35649615,
            556.18037697,  533.13863063,  484.04381997])
```

```
sns.lineplot(x = np.arange(1,32), y= ((s**2)/992)/(np.sum(s**2)/(992)))
plt.show()
```



Question 3

PCA really shines on data where you have reason to believe that the data is relatively low in rank.

In this final question of the homework, we'll look at how states voted in presidential elections between 1972 and 2016. **Our ultimate goal in question 3 is to show how 2D PCA scatterplots can allow us to identify clusters in a high dimensional dataset.** For this example, that means finding groups of states that vote similarly by plotting their 1st and 2nd principal components.

```
df = pd.read_csv("presidential_elections.csv")
df.head(5)
```

	State	1789	1792	1796	1800 †	Unnamed: 5	1804	1808	1812	1816	1820	1824
0	Alabama	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR	Jacks
1	Alaska	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	Arizona	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	Arkansas	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	California	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

```
df
```

	State	1789	1792	1796	1800 †	Unnamed: 5	1804	1808	1812	1816	1820
0	Alabama	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
1	Alaska	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Arizona	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Arkansas	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	California	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	Colorado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	Connecticut	GW	GW	F	F	NaN	F	F	F	F	DR
7	Delaware	GW	GW	F	F	NaN	F	F	F	F	DR
8	D.C.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	Florida	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	Georgia	GW	GW	DR	DR	NaN	DR	DR	DR	DR	DR
11	Hawaii	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	Idaho	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	Illinois	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
14	Indiana	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR	DR
15	Iowa	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	Kansas	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	Kentucky	NaN	GW	DR	DR	NaN	DR	DR	DR	DR	DR
18	Louisiana	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR	DR	DR
19	Maine	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
20	Maryland	GW	GW	F	SP	NaN	DR	DR	DR	DR	DR

The data in this table is pretty messy, so let's create a clean version. The clean table should contain exactly 51 rows (corresponding to the 50 states plus Washington DC) and 13 columns (one for each of the election years from 1972 to 2020). The index of this dataframe should be the state name.

Note: In your personal projects, it is sometimes more convenient to manually do your data cleaning using Excel or Google Sheets. The downside of doing this is that you have no record of what you did, and if you have to redownload the data, you have to redo the manual data cleaning process.

```
df_clean = (
        df.iloc[:, -15:]
        .drop(['Unnamed: 60'], axis = 1)
        .rename(columns = {"2000 ‡": "2000", "2016 ‡": "2016", "State.1": "State"}
        .drop([51])
        .set_index("State")
)
df_clean.head(5)
```

	1972	1976	1980	1984	1988	1992	1996	2000	2004	2008	2012	2016
State												
Alabama	R	D	R	R	R	R	R	R	R	R	R	R
Alaska	R	R	R	R	R	R	R	R	R	R	R	R
Arizona	R	R	R	R	R	R	D	R	R	R	R	R
Arkansas	R	D	R	R	R	D	D	R	R	R	R	R
California	R	R	R	R	R	D	D	D	D	D	D	D

Question 3a

What does each row in `df_clean` represent?

45	Vermont	NaN	GW	F	F	NaN	DR	DR	DR	DR	DR

```
df
```

	State	1789	1792	1796	1800 †	Unnamed: 5	1804	1808	1812	1816	1820
0	Alabama	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
1	Alaska	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Arizona	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Arkansas	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	California	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	Colorado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	Connecticut	GW	GW	F	F	NaN	F	F	F	F	DR
7	Delaware	GW	GW	F	F	NaN	F	F	F	F	DR
8	D.C.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	Florida	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	Georgia	GW	GW	DR	DR	NaN	DR	DR	DR	DR	DR
11	Hawaii	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	Idaho	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	Illinois	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
14	Indiana	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR	DR
15	Iowa	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	Kansas	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	Kentucky	NaN	GW	DR	DR	NaN	DR	DR	DR	DR	DR
18	Louisiana	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR	DR	DR
19	Maine	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
20	Maryland	GW	GW	F	SP	NaN	DR	DR	DR	DR	DR
21	Massachusetts	GW	GW	F	F	NaN	DR	F	F	F	DR
22	Michigan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
23	Minnesota	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	Mississippi	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
25	Missouri	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	DR
26	Montana	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	Nebraska	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
28	Nevada	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
29	New Hampshire	GW	GW	F	F	NaN	DR	F	F	DR	DR
30	New Jersey	GW	GW	F	F	NaN	DR	DR	F	DR	DR

31	New Mexico	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	New York	NaN	GW	F	DR	NaN	DR	DR	F	DR	DR
33	North Carolina	NaN	GW	DR	DR	NaN	DR	DR	DR	DR	DR
34	North Dakota	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Type your answer here, replacing this text.

35	Oklahoma	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Question 3b

To perform PCA, we need to convert our data into numerical values. To do this, replace all of the "D" characters with the number 0, and all of the "R" characters with the number 1.

Hint: Use `df.replace`.

[ ] ↳ *2 cells hidden*

Question 3c

Now center the data so that the mean of each column is 0 and scale the data so that the variance of each column is 1. Store your result in `df_standardized`.

[ ] ↳ *2 cells hidden*

Question 3d

We now have our data in a nice and tidy centered and scaled format, phew. We are now ready to do PCA.

Create a new dataframe `first_2_pcs` that contains exactly the first two columns of the principal components matrix. The first column should be labeled `pc1` and the second column should be labeled `pc2`. Store your result in `first_2_pcs`.

[ ] ↳ *2 cells hidden*

Question 3e

The cell below plots the 1st and 2nd principal components of our 50 states + Washington DC.

[ ] ↳ *3 cells hidden*

Question 3f

To label our points, the best option is to use the `plotly` library that we used earlier in this homework. `plotly` is an incredibly powerful plotting library that will automatically add axis labels, and will also provide controls so that you can zoom and pan around to look at the data.

One important skill as a user of modern tools is using existing documentation and examples to get the plot you want.

Using the example given on this page as a guide, we can create a scatter plot of the **jittered data** from 3e.

Question 3g

We can also look at the contributions of each year's elections results on the values for our principal components. Below, we use the `plot_pc` function from question 3c to plot the 1st row of V^T in the cell below.

Here by "1st row" we mean the row that is used to generate `pc1`, and by "2nd row" we mean the row that is used to generate `pc2`.

Note: If you want to adjust the size of your plot for a single figure, you can set `plt.figure(figsize=(x, y))` before creating the figure.

Note: If you get an error when running this cell, make sure you are properly assigning the `vt_q3` variable in Question 3d.

[ ] ↳ *3 cells hidden*

Question 3h

Using your plots from question 3g as well as the original table, give a description of what it means to have a relatively large positive value for `pc1` (right side of the 2D scatter plot), and what it means to have a relatively large positive value for `pc2` (top side of the 2D scatter plot).

In other words, what is generally true about a state with relatively large positive value for `pc1`? For a large positive value for `pc2`?

Note: `pc2` is pretty hard to interpret, and the staff doesn't really have a consensus on what it means either. We'll be nice when grading.

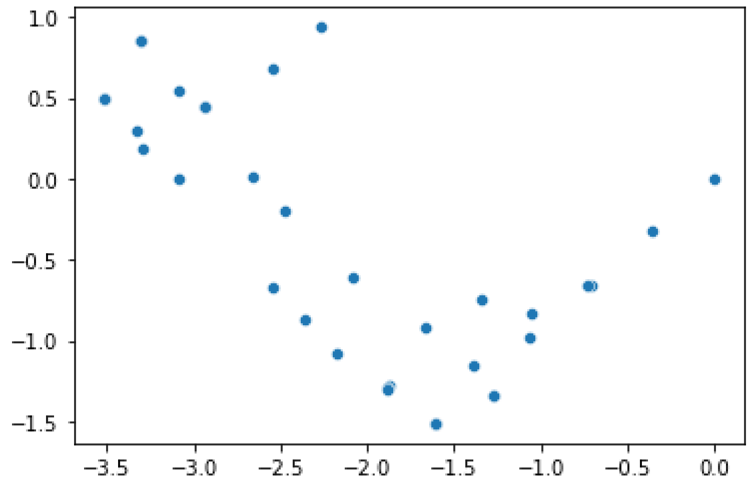
Note: Principal components beyond the first are often hard to interpret (but not always; see question 1 earlier in this homework).

To interpret pc1, it can be simply found out that the states with larger pc1 are less likely is going to vote for the Republican party. The state will tend to vote Republican with small pc1. It could be told by the negative principle component values in the graph, showing a negative correlation. By interpreting PC2, I found out that state with the larger the PC2 value tend to support a single party during most of the elections and it rawly changes. But for state with smaller PC2 values.

things go the opposite way. Furthermore, I noticed that the elections have two separate term(first half and second half), and these two terms have contrasting relations.

```
# feel free to use this cell for scratch work. If you need more scratch space, add
pcs = u*s
sns.scatterplot(x=pcs[:, 0], y = pcs[:, 1])
# Make sure to put your actual answer in the cell above where it says "Type your a
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feefb3be810>
```



Question 3i

To get a better sense of whether our 2D scatterplot captures the whole story, create a scree plot for this data. On the y-axis plot the fraction of the total variance captured by the ith principal component. You should see that the first two principal components capture much more of the variance than we were able to capture when using the Data 100 Midterm 1 data. It is partially for this reason that the 2D scatter plot was so much more useful for this dataset.

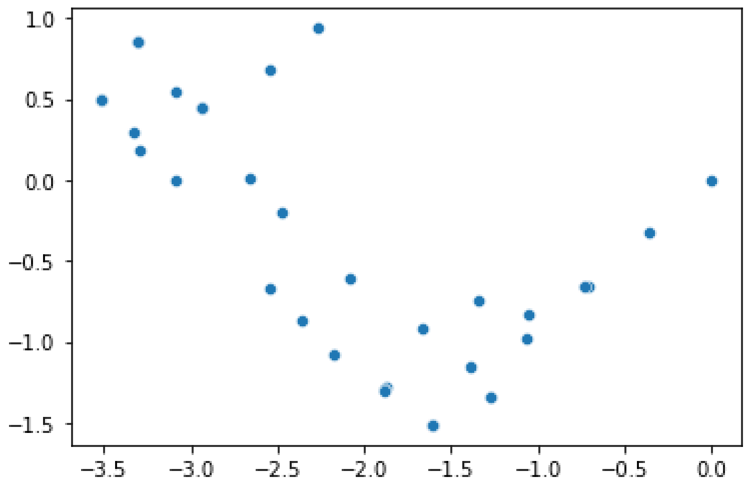
Hint: Your code will be very similar to the scree plot from problem 1d. Be sure to label your axes appropriately!

```
df_numerical.head(5)
```

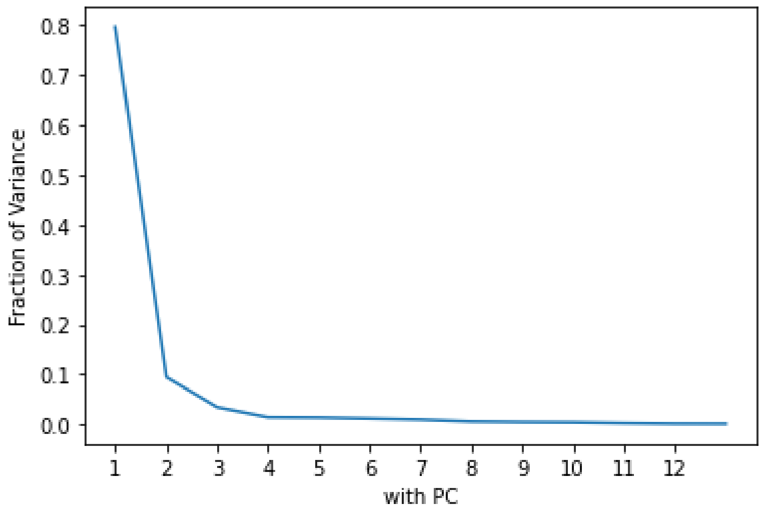
	1972	1976	1980	1984	1988	1992	1996	2000	2004	2008	2012	2016
State												
Alabama	1	0	1	1	1	1	1	1	1	1	1	1
Alaska	1	1	1	1	1	1	1	1	1	1	1	1
Arizona	1	1	1	1	1	1	0	1	1	1	1	1
Arkansas	1	0	1	1	1	0	0	1	1	1	1	1
California	1	1	1	1	1	0	0	0	0	0	0	0

```
pcs = u*s
sns.scatterplot(x=pcs[:, 0], y = pcs[:, 1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feef8c07c90>
```



```
plt.plot(np.arange(1,14), s**2 / sum(s**2));
plt.xticks(np.arange(1,13), np.arange(1,13));
plt.xlabel('with PC');
plt.ylabel('Fraction of Variance');
```



Question 4: Interpreting Diagonal Banding (Optional)

↳ *7 cells hidden*

Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

[ ] ↳ *2 cells hidden*

+ Code + Text