EE460J - Lab 1

Can Gokalp, (EID: CG39283), Priyadarshan Patil (EID:PP22352)

September 10, 2020

1 Programming questions

The following libraries have been used to solve various questions: Numpy, Pandas, Matplotlib, Scipy, Zipfile, Seaborn, os. The code snippet to import them can be found with the code for problem 1.

1.1 Problem 1

Correlations

- When given a data matrix, an easy way to tell if any two columns are correlated is to look at a scatter plot of each column against each other column. For a warm up, do this: Look at the data in DF1 in Lab2 Data.zip. Which columns are (pairwise) correlated? Figure out how to do this with Pandas, and also how to do this with Seaborn.
- Compute the covariance matrix of the data. Write the explicit expression for what this is, and then use any command you like (e.g., np.cov) to compute the 4 * 4 matrix. Explain why the numbers that you get fit with the plots you got.
- The above problem in reverse. Generate a zero-mean multivariate Gaussian random variable in 3 dimensions, $Z = (X_1, X_2, X_3)$ so that (X_1, X_2) and (X_1, X_3) are uncorrelated, but (X_2, X_3) are correlated. Specifically: choose a covariance matrix that has the above correlations structure, and write this down. Then find a way to generate samples from this Gaussian. Choose one of the non-zero covariance terms (C_{ij} , if C denotes your covariance matrix) and plot it vs the estimated covariance term, as the number of samples you use scales. The goal is to get a visual representation of how the empirical covariance converges to the true (or family) covariance.

1.1.1 Solution to problem 1

The pairwise scatter plots using Pandas and Seaborn can be found in Figures 1 and 2 respectively. From the scatterplots, it appears that column 0 and column 2 are correlated, as well as column 1 and column 3. Note that the original column zero has been dropped as it is just the observation number.

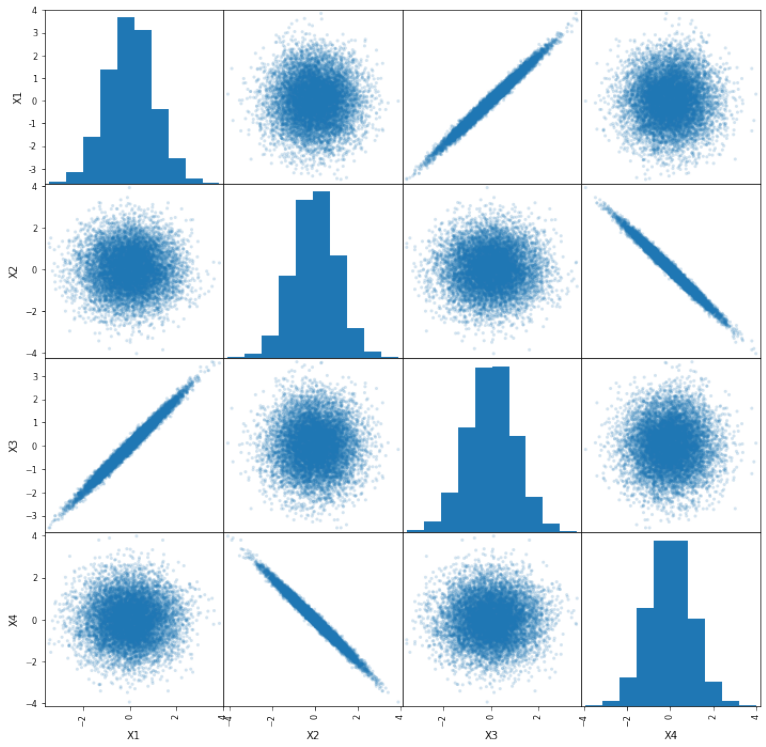

Figure 1: Pairwise scatterplot using pandas

The formula for the covariance matrix of a dataset is given by: $E[XX^T] - E[X](E[X]^T)$. The following covariance matrix is calculated using *np.cov* command:

$$Q = \begin{pmatrix} 1.00155793 & -0.00401176 & 0.99162409 & 0.00412485 \\ -0.00401176 & 1.00537841 & -0.00409877 & -0.99545662 \\ 0.99162409 & -0.00409877 & 1.00158867 & 0.00408108 \\ 0.00412485 & -0.99545662 & 0.00408108 & 1.00516828 \end{pmatrix}$$

The estimated matrix corroborates the observations from the scatterplots. The columns that appeared correlated have covariance values close to 1 or -1. The values on the main diagonal are effectively 1, while the rest are near zero.

Let $Z = (X_1, X_2, X_3)$, where K is the correlation matrix. For the required relations, we have the correlation matrix as $K = ([1, 0, 0], [0, 1, -1], [0, -1, 1])$. Choosing $C_{23} = -1$ for estimation, data is generated for $n = 5$ to $n = 1000$. The estimated value for C_{23} and the true value of C_{23} can be seen in Figure 3.

1.2 Problem 2

Outliers

Consider the two-dimensional data in DF2 in Lab2 Data.zip. Look at a scatter plot of the data. It contains two points that look like potential outliers. Which one is "more" outlying? Propose a transformation of the data that makes it clear that the point at $(-1, 1)$ is more outlying than the point at (5.5, 5), even though the latter point is "farther away" from the nearest points. Plot the data

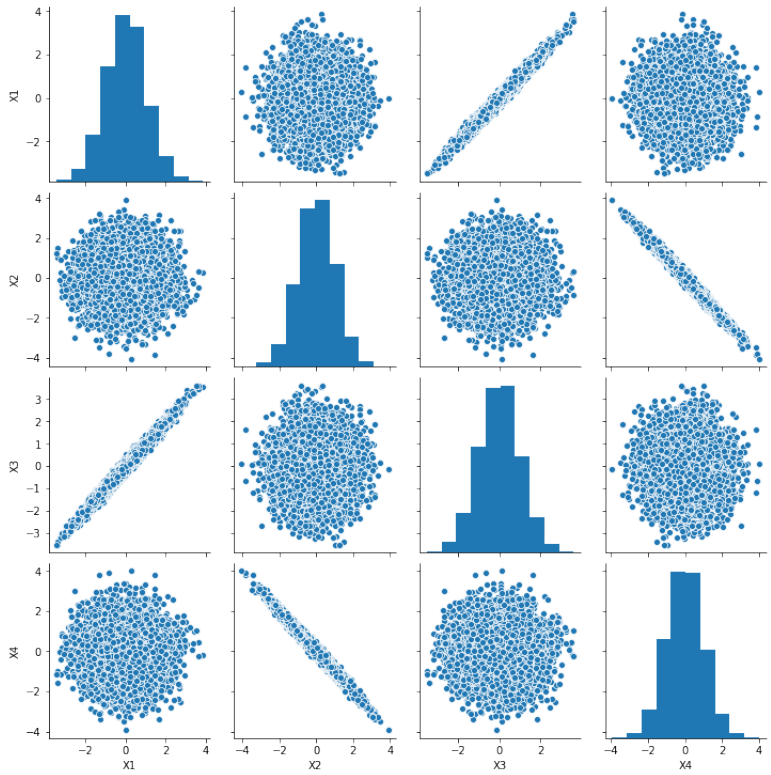

Figure 2: Pairwise scatterplot using seaborn

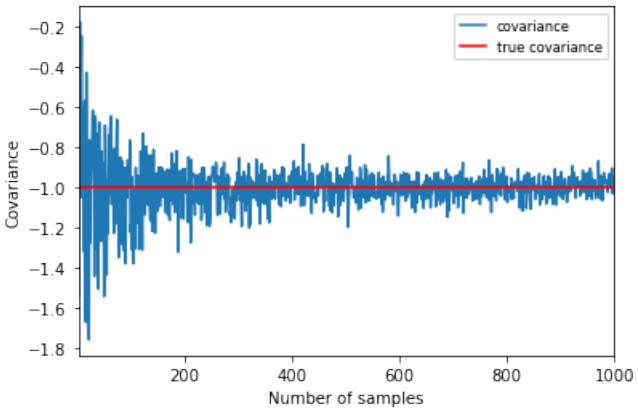

Figure 3: Empirical estimation of covariance

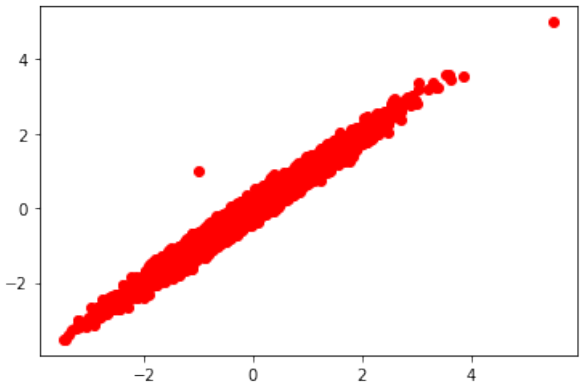

Figure 4: Scatterplot without transformation

again after performing this transformation. Provide discussion as appropriate to justify your choice of transformation.

Hint: if y comes from a standard Gaussian in two dimensions (i.e., with covariance equal to the two by two identity matrix), and

$$Q = \begin{pmatrix} 2 & \frac{1}{2} \\ \frac{1}{2} & 2 \end{pmatrix},$$

what is the covariance matrix of the random variable z = Qy? If you are given z, how would you create a random Gaussian vector with covariance equal to the identity, using z?

1.2.1 Solution to problem 2

Consider the scatterplot of data in Figure 4. As mentioned, two points appear to be outliers. We would like to find the directions of the larger spread of data, and values representing the magnitude —which are the eigenvectors and eigenvalues. If we apply some linear transformation to matrix M (m by m); $Mv_1 = b_1$ we will obtain a $mX1$ vector b_1. If we select the vector v_1 such that after the transformation the length is changed but not the direction $Mv_1 = \lambda v_1$, this vector is called the eigenvector and the λ represents the magnitude/eigenvalue (indicating the magnitude of the spread in the direction of the corresponding eigenvector). Lets consider the covariance matrix and find the directions of the spread of the data.

$$Covariance\ of\ features = \begin{pmatrix} 1.00464777 & 0.9942424 \\ 0.9942424 & 1.00415964 \end{pmatrix}$$

The eigenvalues are $[1.99864, 0.01016]$, and the eigenvectors are: $\begin{pmatrix} 0.70719 & -0.70701 \\ 0.70701 & 0.707193 \end{pmatrix}$. Looking at the eigenvalues we see that much of the variation is along the first eigenvector: [0.707, 0.707] and tiny variation along the second eigenvector. These two vectors are orthogonal to each other - they define a new space. We can project the data points onto this new space. After the projection, the points can be visualized on a scatter plot as shown in Figure 5. In this scatter plot we can see that the first data point is more outlying then the second. The spread along that direction is small and so that point looks farther out.

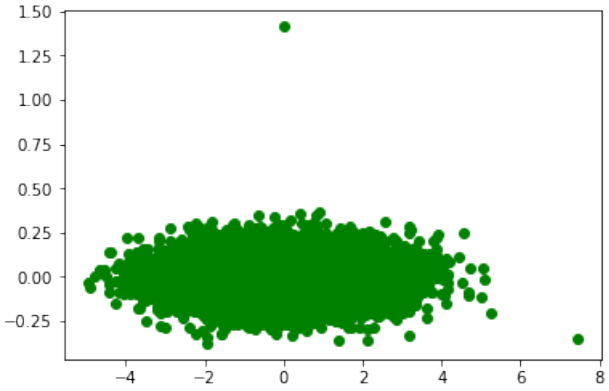

Figure 5: Scatterplot after transformation

1.3 Problem 3

Even more standard error

In one of the written exercises below, you derive an expression for what is called the Standard Error: where β denotes the "truth,", $\hat{\beta}$ denotes the value we compute using least squares linear regression, and Z and e are as in the exercise below, you find:

$$\hat{\beta} - \beta = Ze$$

If we know the distribution of the noise (the distribution generating the noise vectors, e_i), then we know the distribution for the error, $(\hat{\beta} - \beta)$. This allows us to answer the question: if we solve a regression and obtain value $\hat{\beta}$, how can we tell if it is statistically significant? The answer is: we compare the size of $\hat{\beta}$ to the spread introduced by the noise (i.e., the standard error), and we ask: what is the likelihood that the true $\beta = 0$, and what we observed was purely due to the noise.

If the noise is Gaussian (normal), i.e., $e_i \sim N(0, \sigma^2)$, and if the values of the x_i are normalized, then we expect error of the size $\sigma/\sqrt{(n)}$, as this is roughly the standard deviation of the expression for the error that you derive above. This means: if you have twice the data points, you should expect the error to be reduced by about 1.4 (the formula says that the standard deviation of the error would decrease by a factor of $1/\sqrt{(2)}$).

Compute this empirically, as follows: We will generate data for a regression problem, solve it, and see what the error is: Generate data as follows: $xi \sim N(0, 1), e_i \sim N(0, 1)$. Generate y by $y_i = \beta_0 + x_i\beta + e_i$, where $\beta_0 = -3$ and $\beta = 0$. Note that since $\beta = 0$, this means that y and x are unrelated! The question we are exploring here is as follows: when we solve a regression problem, we are not going to find $\hat{\beta} = 0$ - we will find that $\hat{beta}$ takes some other values, hopefully close to zero. How do we know if the value of $\hat{\beta}$ we get is statistically meaningful?

- By creating fresh data and each time computing $\hat{\beta}$ and recording $\hat{\beta} - \beta$, compute the empirical standard deviation of the error for n = 150. By running a linear regression of y vs. noise, we find $\hat{\beta} = -0.15$. Given your empirical computation of the standard deviation of the error, how significant is the value -0.15

- Now repeat the above experiment for different values of n. Plot these values, and on the same plot, plot $1/sqrt(n)$. How is the fit?

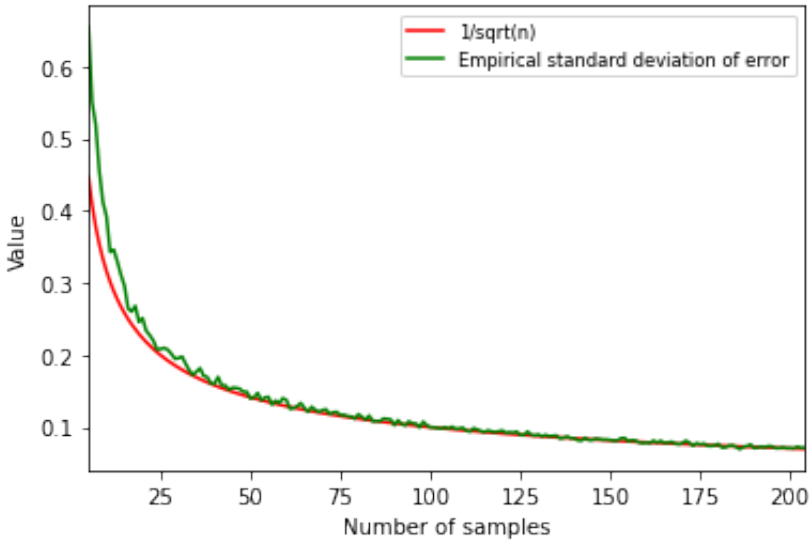

Figure 6: Linear regression estimates vs. $1/sqrt(n)$

1.3.1 Solution to problem 3

We generate the data, fit k linear regression models using scikit, and obtain the standard deviation of the calculated error. Here, we set $k = 1000$. The calculated error is 0.084053. Given $\hat{\beta} = -0.15$, it is over one standard deviation away from zero. Therefore, we can conclude that the $\hat{\beta}$ estimate is not statistically significant.

The experiment is repeated for $n = 5$ to $n = 204$, for $k = 1000$. The results can be seen in Figure 6. As n increases, the estimates asymptotically approach the true value.

1.4 Problem 4

Names and frequencies

The goal of this exercise is for you to get more experience with Pandas, and to get a chance to explore a cool data set. Download the file Names.zip from Canvas. This contains the frequency of all names that appeared more than 5 times on a social security application from 1880 through 2015.

- Write a program that on input k and $XXXX$, returns the top k names from year $XXXX$.
- Write a program that on input **Name** returns the frequency for men and women of the name **Name**.
- It could be that names are more diverse now than they were in 1880, so that a name may be relatively the most popular, though its frequency may have been decreasing over the years. Modify the above to return the relative frequency. Note that in the next coming lectures we will learn how to quantify diversity using entropy.
- Find all the names that used to be more popular for one gender, but then became more popular for another gender.

1.4.1 Solution to problem 4

The code with the required functions can be found below. As proof of concept, the answers to parts a, b, and c are provided for sample cases below in Appendix A. Additionally, the list of names which change popularity with respect to gender is also found in Appendix A.

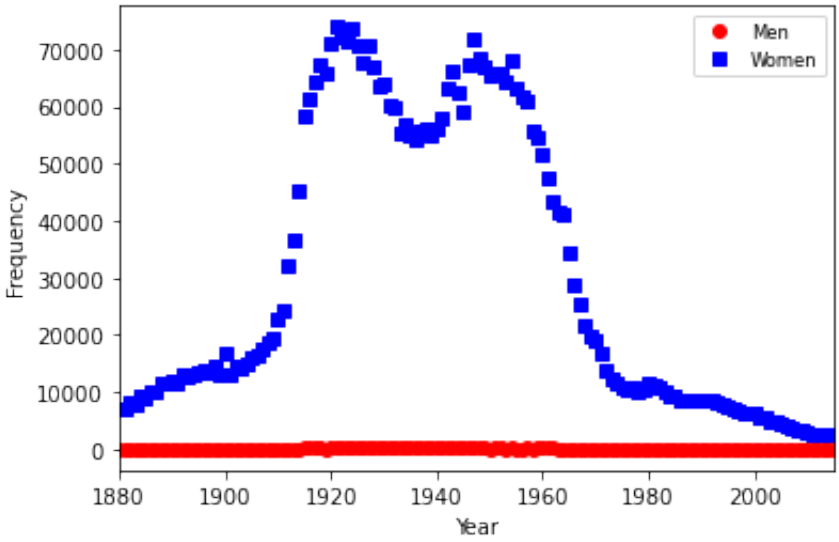

Figure 7: Frequency of name ”Mary”

Appendix A: Output for problem 4

The top 10 names in 2015 are : ’Emma’ ’Olivia’ ’Noah’ ’Liam’ ’Sophia’ ’Mason’ ’Ava’ ’Jacob’ ’William’ ’Isabella’.

The list of number of males with name Mary over the years is: [27, 29, 30, 32, 36, 38, 32, 47, 50, 41, 35, 39, 50, 55, 48, 47, 57, 63, 50, 50, 75, 58, 56, 63, 58, 68, 76, 70, 79, 80, 99, 98, 118, 126, 127, 159, 164, 159, 169, 155, 195, 187, 186, 203, 224, 260, 272, 283, 294, 327, 340, 325, 330, 316, 303, 276, 293, 288, 284, 272, 303, 315, 276, 254, 252, 198, 186, 184, 164, 170, 119, 161, 158, 146, 168, 149, 143, 184, 141, 170, 169, 167, 158, 136, 134, 136, 108, 127, 101, 108, 99, 86, 74, 54, 61, 66, 62, 52, 61, 65, 52, 67, 73, 57, 56, 67, 47, 69, 48, 74, 45, 32, 25, 18, 10, 13, 15, 13, 14, 11, 10, 13, 12, 9, 31, 10, 10, 10, 0, 0, 0, 0, 6, 0, 5, 0]
The list of number of females with name Mary over the years is: [7065, 6919, 8148, 8012, 9217, 9128, 9889, 9888, 11754, 11648, 12078, 11703, 13172, 12784, 13151, 13446, 13811, 13413, 14406, 13172, 16707, 13136, 14486, 14275, 14962, 16067, 16370, 17580, 18666, 19259, 22848, 24390, 32303, 36641, 45345, 58187, 61437, 64281, 67368, 65843, 70975, 73983, 72170, 71633, 73532, 70602, 67827, 70638, 66862, 63514, 64137, 60294, 59869, 55493, 56921, 55072, 54362, 55642, 56215, 54901, 56196, 58035, 63241, 66172, 62463, 59287, 67464, 71685, 68592, 66864, 65485, 65693, 65660, 64368, 67987, 63167, 61753, 61095, 55850, 54474, 51477, 47669, 43493, 41551, 40985, 34274, 28886, 25317, 21720, 19851, 19204, 16698, 13761, 12320, 11745, 10966, 10323, 10661, 10046, 10555, 11475, 11041, 10852, 9895, 9290, 9240, 8505, 8394, 8508, 8642, 8665, 8757, 8454, 8115, 7750, 7438, 6946, 6631, 6434, 6360, 6184, 5725, 5455, 5010, 4805, 4451, 4086, 3674, 3493, 3155, 2863, 2704, 2566, 2639, 2624, 2602]

This frequency distribution is visualized in Figure 7.

The list of fraction of males with name Mary over the years is: [0.013400700807019982, 0.015049611823805372, 0.013541939386279307, 0.014750282329622716, 0.014786641091254113, 0.015777192822207645, 0.01253334064444871, 0.01899788193826901, 0.016695940215177278, 0.014189404321884907, 0.011612436587801632, 0.013604255690241563, 0.014953181588446574, 0.016911525050580835, 0.014172168223636815, 0.01338939787592871, 0.015944680351118644, 0.018157817372707937, 0.013107499849263751, 0.014739045204651642, 0.016655119117411927, 0.016771924954310964, 0.014480237062166761, 0.016526540837607063, 0.01437436802347483,

0.016039967825711595, 0.017737903799879103, 0.01504098668872678, 0.01616672601287608,
0.015648503318460735, 0.016759237471623563, 0.015210990406211052, 0.011942606666201108,
0.011080878098028836, 0.008966868480045187, 0.00867677064784917, 0.008477980255611105,
0.007923069486814318, 0.007783810870897897, 0.007345028197800188, 0.00861790083845546,
0.008010397238597606, 0.008124977885644868, 0.008816454022192274, 0.009405240314386953,
0.011142935143403147, 0.011847381336715644, 0.012202657247192635, 0.01300492506243691,
0.014918781605461643, 0.015294364791292649, 0.015448273239550432, 0.015633120043886433,
0.015808275632293506, 0.014592634331700691, 0.013209211414864478, 0.014105173565845647,
0.013519489188633 49, 0.012837945350132357, 0.01234586953502369, 0.013160752291187074,
0.012931283977974944, 0.01010477109947598, 0.00900063535981103, 0.00936895270351912,
0.007464160719213353, 0.005821711038433622, 0.005108354018928118, 0.004750529148117154,
0.004878628336336081, 0.003395223405700095, 0.004372617636314997, 0.004159856437562135,
0.003790195542937723 5, 0.004222010395695358, 0.003713157811449833 5, 0.003469800843137340 6,
0.004380800095996663, 0.003412569485723358, 0.00408986442580589, 0.004067509098343946,
0.00403377745142559, 0.0039151890751357, 0.003435223927345014, 0.003446476350870839 7,
0.003750234389649353, 0.003107093753964781 2, 0.003740414084455605, 0.002989031790684448,
0.003106385952922721, 0.002743980773287042, 0.002505198286444372 3, 0.002353712026514248,
0.001789518788456013, 0.002006152639932435 4, 0.002185395728809608 3, 0.002042733322153212 2,
0.001636830190199668, 0.001921776755070497 7, 0.001953574650896209 8, 0.001509815250876128 3,
0.001937137292726309 6, 0.002081533970776974, 0.001646384842168017 3, 0.001605916195263350 2,
0.001878121150026840 3, 0.001322032858986068 4, 0.001914653621123346 7, 0.001299848161486636 3,
0.001925450210589612, 0.001139105255603828 3, 0.000821774560530373 3, 0.000651062860119144 5,
0.000477581020293478 85, 0.000269070579096391 06, 0.000355096079438817 07, 0.000411419694167056 14,
0.000358691240511582, 0.000380797057200341 95, 0.000297959895681531 8, 0.000264739784618300 83,
0.000347537828155910 8, 0.000321257874499005 03, 0.000236896903257358 75, 0.000812073383142247 3,
0.000260365475017281 8, 0.000253073066243393 5, 0.000250483433025739 7, 0.0, 0.0, 0.0, 0.0,
0.000164557451531659 63, 0.0, 0.000135570499642365 03, 0.0]
The list of fraction of females with name Mary over the years is: [3.5065167111702285, 3.5906298003072195,
3.67799073731346, 3.6931019382792876, 3.7857908593913656, 3.789847791608194, 3.8731939260297903,
3.9968309916085953, 3.9248816257838746, 4.0311751595442775, 4.007285974499089, 4.08232318827941,
3.9392661576603656, 3.93085338630228, 3.882878839771828, 3.8305073157390948, 3.8633680759526245,
3.8658857844465326, 3.7765328565698724, 3.8828540687134288, 3.710094334594681, 3.798551831031531
3.7457270371883515, 3.7447042929657273, 3.7080912821936276, 3.7899141625839445, 3.82065112110553
3.7774363712545256, 3.819849465270189, 3.767181567627941, 3.867828866178335, 3.78567404089273,
3.269339179138088, 3.222336939602179, 3.201595678957866, 3.175316060920753, 3.1759858107559724,
3.20316245082963, 3.1028388801813582, 3.1201205911468244, 3.136694933381417, 3.169161598412656,
3.1525787849838176, 3.1110790688261045, 3.087438083917417, 3.0258211807482653, 2.954310051196367
3.045834991615524, 2.957603059607676, 2.897710993545232, 2.885101984173931, 2.865963651401396,
2.836179587598293, 2.776103290072353, 2.7413443524578716, 2.635716271881944, 2.61701517196758,
2.6119840883123078, 2.541144710766516, 2.4919139093468217, 2.440863484341745, 2.3824351290850028
2.315347206891161, 2.3448426890921867, 2.322273383809186, 2.234988366464657, 2.1115909327789564,
1.990175857863381, 1.9868798495588524, 1.9188506181222102, 1.868371468254376, 1.7841637911952863
1.7287099600653781, 1.6710089500535301, 1.7085822665008354, 1.574154627354709, 1.498395884379441
1.4545922927441093, 1.3517163530329757, 1.310536910184412, 1.2389536441150966, 1.151413996000038
1.077742521803019, 1.049536686802299, 1.0541330838838907, 0.9451142166973671, 0.8310325016391358
0.7456382943004924, 0.6427621490036299, 0.5709709958469346, 0.5322768360626703, 0.48641629054707
0.4376950161738184 5, 0.4082753976625570 5, 0.3862666025574828 5, 0.3631068115473661, 0.3401150981 3
0.3355816664945896 5, 0.3164945783842331, 0.3172304683109153 4, 0.3331755769962226, 0.319222878343

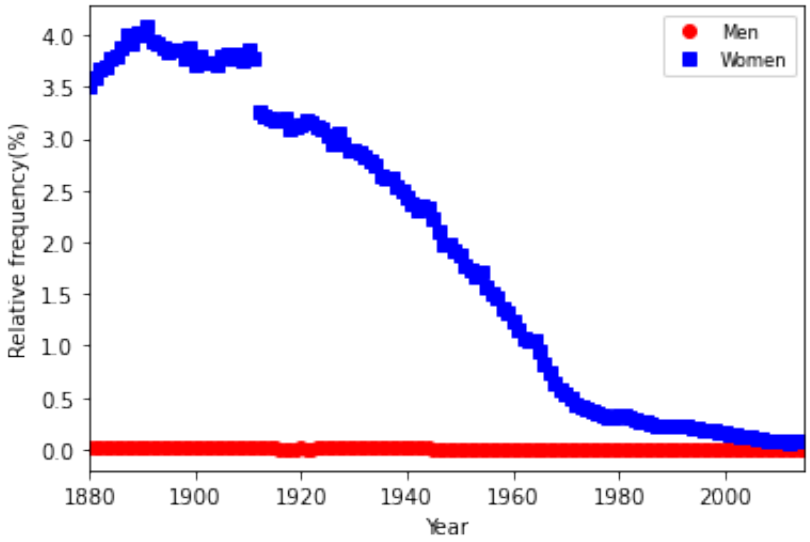

Figure 8: Relative frequency of name ”Mary”

0.30943570754618793, 0.28580663181144794, 0.26641002596422364, 0.2590125287499702, 0.23923169075
0.23292177530013583, 0.2303980866235063, 0.22486136107993818, 0.21934104532904827, 0.22488374458
0.2201634167778899, 0.2153094433156434, 0.20852969879970307, 0.2031695876050709, 0.1905147463789
0.18296012429479233, 0.17500344757335715, 0.17227499423041293, 0.16371508280795724,
0.15305031278404535, 0.14603847544933937, 0.1318726094799297, 0.12587137438704832, 0.11588867293
0.1034056548670506, 0.09202761329365676, 0.08900700997092556, 0.08273552220144147, 0.07762869946
0.07410887097437269, 0.07037573677170643, 0.07266991583892998, 0.07114739821231317,
0.07093430180555332]

This relative frequency distribution is visualized in Figure 8.

The list of names that change popularity with gender can be found in the accompanying csv file named *Names_flip.csv*.

Appendix B: Code for all problems

Code for problem 1

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy
from zipfile import ZipFile
import seaborn as sb
import os
from sklearn.linear_model import LinearRegression
import time

file_name = "Lab2_data.zip"

with ZipFile(file_name, 'r') as zip:
    # extracting all the files
    # print('Extracting all the files now...')
    zip.extractall()
    # print('Done')

header_list = ["X0", "X1", "X2", "X3", "X4"]
df1 = pd.read_csv("DF1", header = None, names = header_list)
df1 = df1.iloc[1:]
df1 = df1.drop(columns = ["X0"])
print(df1.head())

#df1.drop(0, axis=1)

from pandas.plotting import scatter_matrix
scatter_matrix(df1, alpha=0.2, figsize=(12, 12) );
#diagonal='kde'

sb.pairplot(df1)

print("From the scatterplots, it appears that column 0 and column 2
are correlated, as well as column 1 and column 3. Note that the original
column zero has been dropped as it is just the observation number.\n")

print("The formula for the covariance matrix of a dataset is given by:
E[XX^T] - E[X](E[X]^T)")

print("The covariance matrix of the dataset is: ")
print(np.cov(np.transpose(df1)))


print("Let Z = (X1, X2, X3), where K is the correlation matrix.")
```

```
print("For the required relations, we have the correlation matrix as
K = ([1, 0, 0], [0, 1, −1], [0, −1, 1])")

K = [[1, 0, 0], [0, 1, −1], [0, −1, 1]]
cov_obs = []
for i in range(998):
    Obs= np.random.multivariate_normal(mean = [0,0,0], cov=K, size = i+2)
    cov_mat = np.cov(np.transpose(Obs))
    cov_obs.append(cov_mat[2,1])
#print(cov_obs)


fig, ax = plt.subplots()
ax.set_xlabel('Number of samples')
ax.set_ylabel('Covariance')
# set the xlim
ax.set_xlim(3, 1000)
dim = np.arange(3,1001,1);
actual_cov = [−1]*998
ax.plot(dim, cov_obs, label="covariance")
ax.plot(dim, actual_cov, "r", label="true covariance")
legend = ax.legend(loc='best', shadow=False, fontsize='small')
plt.show()
```

Code for problem 2

```
file_name = "Lab2_data.zip"

with ZipFile(file_name, 'r') as zip:
    # extracting all the files
    # print('Extracting all the files now...')
    zip.extractall()
    # print('Done')

header_list = ["Col0", "Col1", "Col2"]
df2 = pd.read_csv("DF2", header=None, names = header_list)
df2 = df2.iloc[1:]
df2 = df2.drop(columns = ["Col0"])
print(df2.head())

plt.scatter(df2['Col1'].to_numpy(), df2['Col2'].to_numpy())

Z = df2.to_numpy()
Z_cov = np.cov(Z, rowvar=False)
print('covariance of features: \n {}'.format(Z_cov))

e_val, e_vec = np.linalg.eig(Z_cov)
print('eigen_values: {}, \n eigen_vectors: \n {}'.format(e_val, e_vec))
```

```
projected = Z@e_vec

plt.scatter(projected[:,0], projected[:,1],  c = "g")
```

Code for problem 3

```
n_obs = 150
mu_x, sigma_x = 0, 1
mu_e, sigma_e = 0, 1
beta_0, beta_1 = -3, 0

k=1000
err_arr=[]

for i in range(k):
    Obs_x = np.random.normal(mu_x, sigma_x, n_obs)
    Obs_e = np.random.normal(mu_e, sigma_e, n_obs)

    Obs_y = ([beta_0]*n_obs) + (Obs_x * beta_1) + Obs_e
    Obs_x_reshape = Obs_x.reshape((-1, 1))

    model = LinearRegression(fit_intercept=True).fit(Obs_x_reshape,
    Obs_y)
    beta_est =  model.coef_[0]
    #print(model.intercept_)
    #print(model.coef_)
    beta_hat = -0.15
    err_arr.append(beta_est-beta_hat)
#e_stdev = np.std(Obs_e)
#print("The error is: ",e_stdev/np.sqrt(n_obs))
print(np.std(err_arr))

beta = []
e_stdev_array = []
sqrt_array = []
for i in range(200):
    n_obs = i+5
    err_arr=[]
    for i in range(k):
        Obs_x = np.random.normal(mu_x, sigma_x, n_obs)
        Obs_e = np.random.normal(mu_e, sigma_e, n_obs)

        Obs_y = ([beta_0]*n_obs) + (Obs_x * beta_1) + Obs_e
        Obs_x_reshape = Obs_x.reshape((-1, 1))

        model = LinearRegression(fit_intercept=True).fit(Obs_x_reshape,
        Obs_y)
        beta_est =  model.coef_[0]
        #print(model.intercept_)
```

```
            #print(model.coef_)
            beta_hat = -0.15
            e_stdev = beta_est-beta_hat
            err_arr.append(e_stdev)
        e_stdev_array.append(np.std(err_arr))
        sqrt_array.append(1/np.sqrt(n_obs))
        #print("The error is: ",e_stdev/np.sqrt(n_obs))

fig, ax = plt.subplots()
ax.set_xlabel('Number of samples')
ax.set_ylabel('Value')
# set the xlim
ax.set_xlim(5, 204)
dim = np.arange(5, 205,1)
actual_stdev = [1]*200
#ax.plot(dim, beta, "b", label="Estimated beta")
ax.plot(dim, sqrt_array, "r", label="1/sqrt(n)")
ax.plot(dim, e_stdev_array, "g", label="Empirical standard
deviation of error")
#ax.plot(dim, actual_stdev, "c", label="Actual standard
deviation of error")

legend = ax.legend(loc='best', shadow=False, fontsize='small')
plt.show()
```

Code for problem 4

```
file_name = "Names.zip"

with ZipFile(file_name, 'r') as zip:
    # extracting all the files
    # print('Extracting all the files now...')
    zip.extractall()
    # print('Done')

def top_k_year(k,year):
    file_name_parta = "Names/" + "yob" + str(year) + ".txt"
    header_list = ["Name", "Sex", "Freq"]
    df = pd.read_csv(file_name_parta, header=None, names = header_list)
    #print(df.head())
    print(df.sort_values(by='Freq', ascending=False)[:k]['Name'].values)
    return df.sort_values(by='Freq', ascending=False)[:k]['Name'].values


def name_freq(name):
    list_men = []
    list_women = []
    directory="Names/"
    for filename in os.listdir(directory):
```

```
        #print(filename)
        file_address = "Names/" + filename
        #print(file_address)
        header_list = ["Name", "Sex", "Freq", "Temp_label"]
        df = pd.read_csv(file_address, header=None, names= header_list)
        df['Temp_label'] = df['Name']+df['Sex']
        #print(df.head())
        df_count_sorted = df['Temp_label'].value_counts()

        name_m = name+"M"
        x=0
        try:
            x = df_count_sorted.at[name_m]
        except KeyError:
            x = 0
        if(x>0):
            x=df[df.Temp_label == name_m].iloc[0].Freq
        else:
            x=0
        list_men.append(x)


        name_f = name+"F"
        x=0
        try:
            x = df_count_sorted.at[name_f]
        except KeyError:
            x = 0
        if(x>0):
            x=df[df.Temp_label == name_f].iloc[0].Freq
        else:
            x=0
        list_women.append(x)
        #print(df[df.Name == "Emma"].iloc[0].Freq)

print("The list of number of males with name ",name,"
over the years is:",list_men)
print("The list of number of females with name ",name,"
over the years is:",list_women)
#a = plt.plot(list_men,'ro')

fig, ax = plt.subplots()
ax.set_xlabel('Year')
ax.set_ylabel('Frequency')
# set the xlim
ax.set_xlim(1880, 2015)
dim = np.arange(1880,2016,1);
ax.plot(dim, list_men, 'ro', label="Men")
```

```
    ax.plot(dim, list_women,"bs", label="Women")
    legend = ax.legend(loc='best', shadow=False, fontsize='small')
    plt.show()



def name_relative_freq(name):
    list_men = []
    list_women = []
    directory="Names/"
    for filename in os.listdir(directory):
        #print(filename)
        file_address = "Names/" + filename
        #print(file_address)
        header_list = ["Name", "Sex", "Freq", "Temp_label"]
        df = pd.read_csv(file_address, header=None, names= header_list)
        df['Temp_label'] = df['Name']+df['Sex']
        #print(df.head())
        df_count_sorted = df['Temp_label'].value_counts()

        name_m = name+"M"
        x=0
        try:
            x = df_count_sorted.at[name_m]
        except KeyError:
            x = 0
        if(x>0):
            x=df[df.Temp_label == name_m].iloc[0].Freq
        else:
            x=0
        y=df.sum(axis=0, skipna=True)
        #print(y.Freq)
        list_men.append((100*x)/(y.Freq))


        name_f = name+"F"
        x=0
        try:
            x = df_count_sorted.at[name_f]
        except KeyError:
            x = 0
        if(x>0):
            x=df[df.Temp_label == name_f].iloc[0].Freq
        else:
            x=0
        list_women.append((100*x)/(y.Freq))
        #print(df[df.Name == "Emma"].iloc[0].Freq)
```

```
    print("The list of percentage of males with name ",name,"
    over the years is:",list_men)
    print("The list of percentage of females with name ",name,"
    over the years is:",list_women)
    #a = plt.plot(list_men,'ro')

    fig, ax = plt.subplots()
    ax.set_xlabel('Year')
    ax.set_ylabel('Relative frequency(%)')
    # set the xlim
    ax.set_xlim(1880, 2015)
    dim = np.arange(1880,2016,1);
    ax.plot(dim, list_men, 'ro', label="Men")
    ax.plot(dim, list_women,"bs", label="Women")
    legend = ax.legend(loc='best', shadow=False, fontsize='small')
    plt.show()

top_k_year(10, 2013)
name_freq("Mary")
name_relative_freq("Mary")

########### Part d code #############

folder_path = 'Names'

def get_file(year=1880):
    year = str(year)
    text_file = os.path.join(folder_path, 'yob' + year + '.txt')
    df = pd.read_csv(text_file, header=None)
    df.rename(columns = {0:'Name', 1:'Sex', 2:'Freq'}, inplace=True)
    return df

df = get_file(1880)


df.head(5)

df.sort_values(by='Freq', ascending=False)

def topk(k=5, year=1880):
    df = get_file(year)
    return df.sort_values(by='Freq', ascending=False)
    [:k-1]['Name'].values

#topk()

def name_freq(name='John', years=list(np.arange(1880,2016,1))):
    frames = []
```

```
    if not isinstance(years,list):
        years = [years]
    for year in years:
        df = get_file(year)
        cur_df = df[df['Name']==name]
        cur_df['Year'] = year
        cur_df['Rel_Freq'] = df['Freq']/df['Freq'].sum()
        cur_df['Pop'] = df['Freq']/df['Freq'].sum()
        frames.append(cur_df)
        yearly_freq = pd.concat(frames)
        yearly_freq.reset_index(drop=True,inplace=True)
    return yearly_freq

#name_freq_df = name_freq()

frames = []
years = np.arange(1880, 2016, 1)
for year in years:
    df = get_file(year)
    df.groupby(['Sex','Name']).sum()
    df['MF_diff'] = df.sort_values(['Sex','Name','Freq'])
    .groupby(['Name'])['Freq'].diff()
    df['Year'] = year
    df.dropna(inplace=True)
    frames.append(df)


yearly_freq = pd.concat(frames)
yearly_freq.reset_index(drop=True,inplace=True)



unique_names = yearly_freq['Name'].unique()
for name in unique_names:
    mf_diff = yearly_freq[yearly_freq['Name']==name]['MF_diff'].to_numpy()
    num_flips = ((np.diff(np.sign(mf_diff)) != 0)*1).sum()
    if num_flips > 0:
        print('name: {}'.format(name))
```