MP2 Report

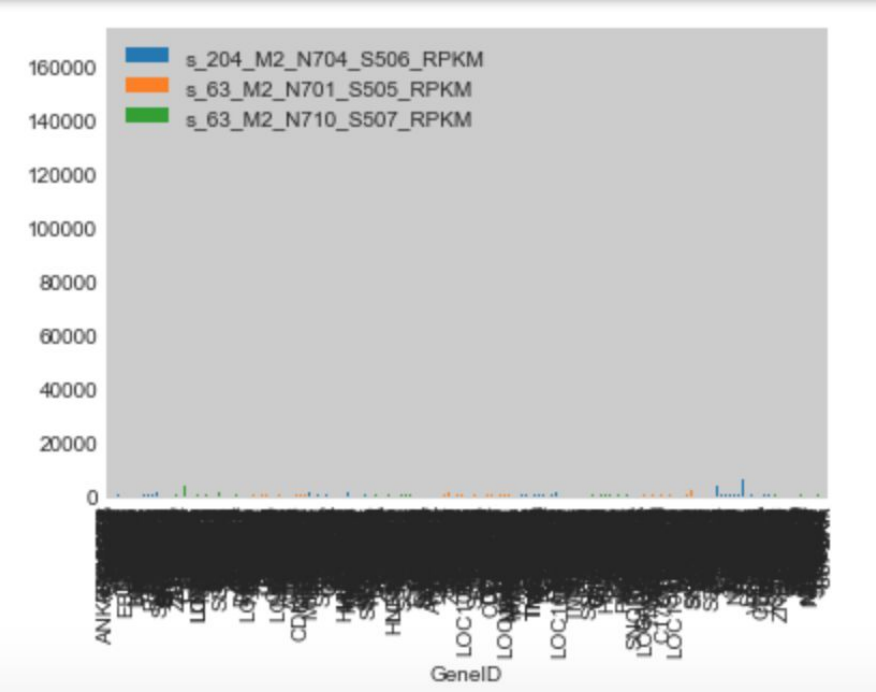
Akhil Chainani

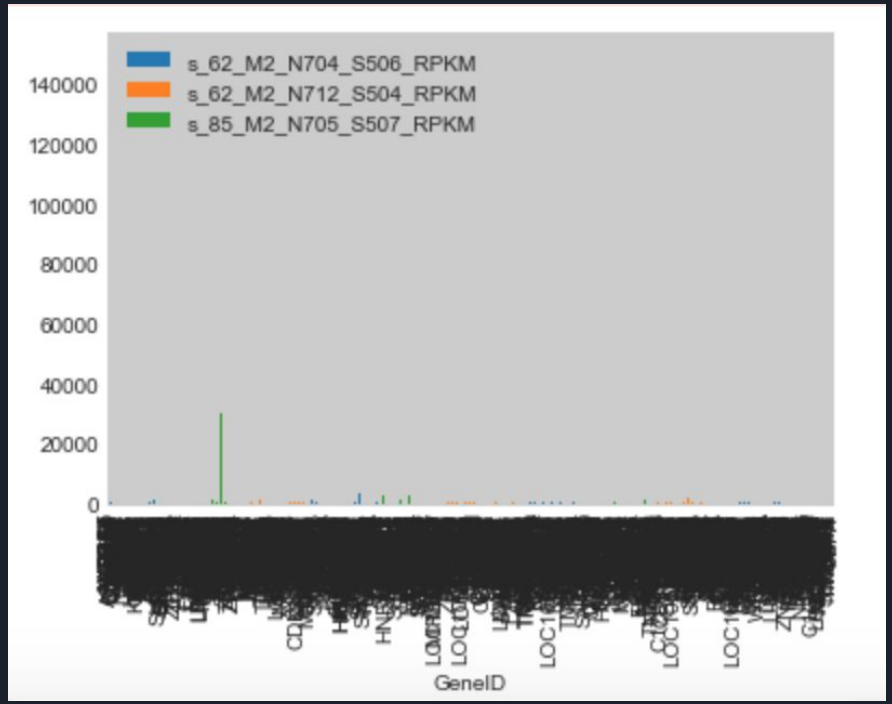
Task 1

General Information about Datasets

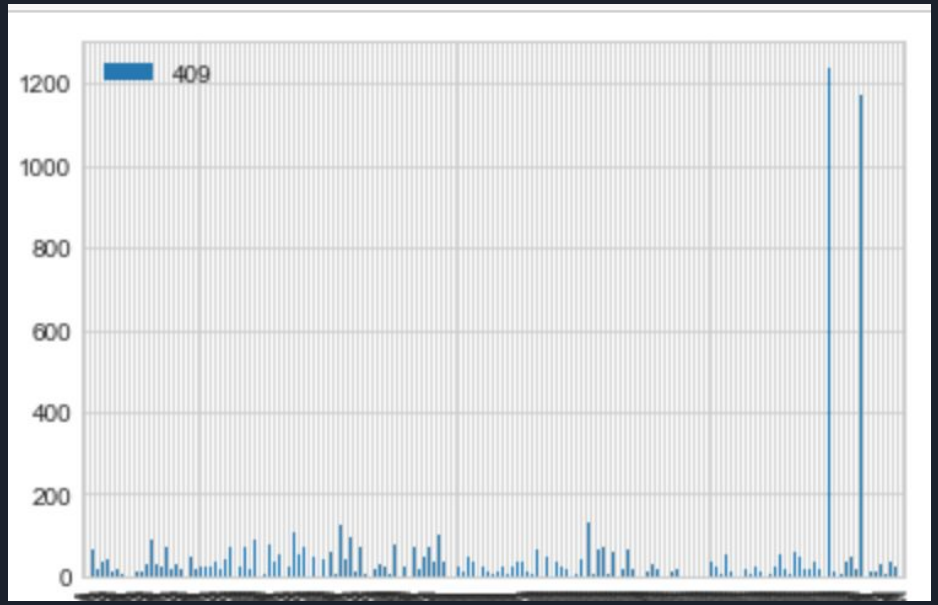
- There are 169 Cells in the baseline dataset
- There are 177 Cells in the metformin dataset
- There are 1170 Genes in Both datasets
- There are 833 overlapping genes between both datasets

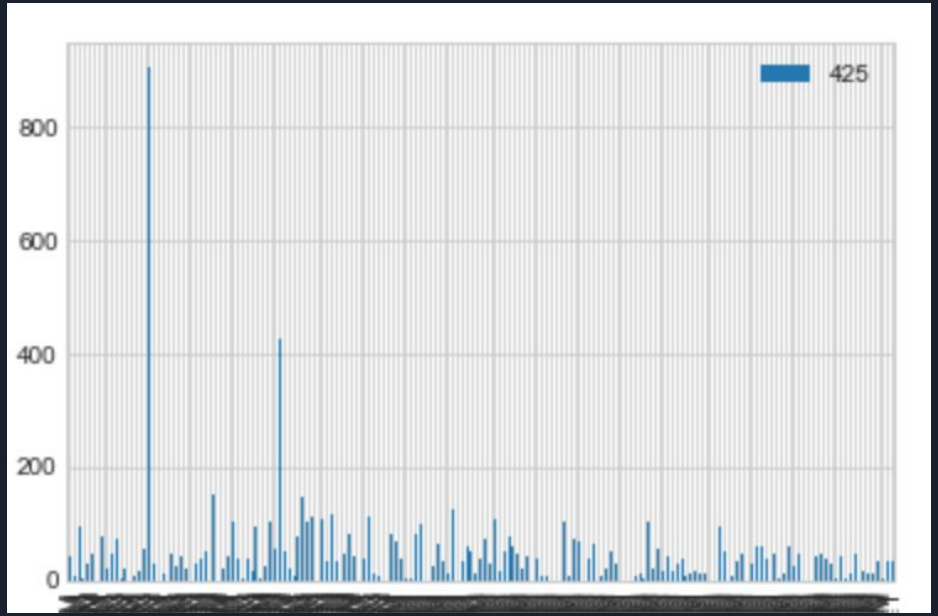
Variation Across Different Genes for Assigned Cells





Variation Across Different Cells for Assigned Genes





KS test on Baseline and Metformin Variance distributions

	Alpha	Different Genes Count
0	0.100	309
1	0.050	222
2	0.025	173
3	0.010	138
4	0.005	109

Task 2: GMM Clustering

GMM Clustering

- Using the GaussianMixture function in sklearn library, I created a Gaussian Mixture Model for each both datasets using N = 2, and N= 3 clusters for baseline and metformin cells, respectively
- The means_ attribute for both Gaussian Mixture Model represents the average expression for each cluster for each gene.
    - I.e for each of the 1170 genes, there are 2 means in the baseline dataset, one for each cluster
    - I.e. for each of the 1170 genes, there are 3 means in the metformin dataset, one for each cluster
- The baseline dataset is split up into 2 clusters: one with 152 Cells and one 17 cells
- The metformin dataset is split up into 3 clusters: one with 161 cells, one with 11 cells, and one with 5 cells

Modifying the original datasets

	Chr	GeneID	Start	Stop	CodingLength	1	2	3
0	chr1	ANKRD36BP1	168214819	168216668	1850	6.961783	0.000000	209.918657
1	chr1	ANP32E	150190717	150208504	3553	589.414935	335.422835	369.797641
2	chr1	APOA1BP	156561558	156564091	1121	34.144335	78.297295	0.000000
3	chr1	ARF1	228270361	228286913	2198	129.281934	128.121970	191.948024
4	chr1	ARPC5	183595332	183604985	1982	132.848917	191.115200	0.000000

	Chr	GeneID	StartCoding	Stop	CodingLength	1	2
0	chr1	AKR1A1	46016455	46035723	1580	23.878180	59.520515
1	chr1	ANP32E	150190717	150208504	3553	551.845311	314.138091
2	chr1	ARF1	228270361	228286913	2198	113.391566	76.383464
3	chr1	ARPC5	183595332	183604985	1982	106.269814	51.795593
4	chr1	ATP5F1	111991743	112004525	2101	214.599276	71.062689

- I dropped each of the original cells columns in both datasets
- I appended the mean of each cluster for each gene to the original dataset
- Now, we have a smaller dataset that is much easier to do various analysis on

Re-Defining Clusters

- Bu should be cluster 2 because it has 17 cells in it
- Bv should be cluster 1 because it has 152 cells in it
- Mx should be cluster 3 because it has 5 cells in it
- My should be cluster 2 because it has 11 cells in it
- Mz should be cluster 1 because it has 161 cells in it
- Based on this definition, Bu < Bv and Mx < My < Mz

Filtering Datasets

```
metformin_upMy_downMx_upMz_DF = metformin_means
metformin_upMy_downMx_upMz_DF = metformin_upMy_downMx_upMz_DF[metformin_upMy_downMx_upMz_DF['3'] < rpkmThreshold]
metformin_upMy_downMx_upMz_DF = metformin_upMy_downMx_upMz_DF[metformin_upMy_downMx_upMz_DF['2'] > rpkmThreshold]
metformin_upMy_downMx_upMz_DF = metformin_upMy_downMx_upMz_DF[metformin_upMy_downMx_upMz_DF['1'] > rpkmThreshold]
```

```
baseline_upBv_downBu_DF = baseline_means
baseline_upBv_downBu_DF = baseline_upBv_downBu_DF[baseline_upBv_downBu_DF['2'] < rpkmThreshold]
baseline_upBv_downBu_DF = baseline_upBv_downBu_DF[baseline_upBv_downBu_DF['1'] > rpkmThreshold]
```

- Using these 2 snippets of code, I filitered both datasets to only include genes that met the following conditions:
  - For Metformin, Mx < rpkmThreshold < My and rpkmThreshold < Mz
    - This left only 338 remaining genes
  - For Baseline, Bu < rpkmThreshold < Bv
    - This left only 82 remaining genes

Further Filtration of Datasets

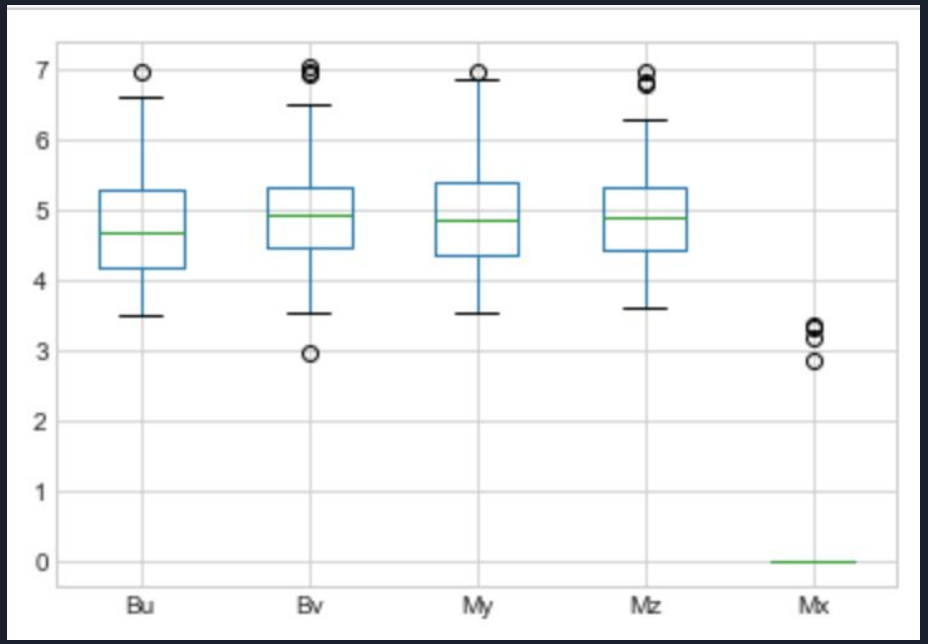
- After removing genes that were common to both baseline_upBv_downBu_DF and metformin_upMy_downMx_upMz_DF from metformin_upMy_downMx_upMz_DF, there were only 303 remaining genes
- After this, I filtered the original baseline and metformin datasets, to only include genes that overlapped between the mean baseline dataset and the remaining 303 Genes that were filtered.
    - This left both datasets with only 221 Genes that represent the downregulated genes.

Combined Dataframe

	GeneID	Bu	Bv	My	Mz	Mx	Mean	STD
0	ARPC5	3.966428	4.675347	5.258095	4.896712	0.0	4.699145	0.421639
1	ATP5F1	4.277536	5.373421	3.963450	5.336792	0.0	4.737800	0.561118
2	ATP6V0B	5.294395	5.100574	3.604205	4.868313	0.0	4.716872	0.590207
3	C1orf31	3.910229	4.499485	4.939882	4.543935	0.0	4.473383	0.328762
4	CDC20	5.081032	4.551463	6.026550	4.457334	0.0	5.029095	0.557253

- This dataframe was created by appending the Bu, Bv, My, and Mz clusters for each gene
- I then applied the log + 1 transformation
- Then, I found the mean and STD
- Lastly, I appended Mx to the dataframe and applied the log + 1 transformation to that column

Visualization Boxplot



- This boxplot is a visualization of the relationship between the various clusters
- The reason Mx was not included from the beginning was because due to its very low values, it shift the mean and standard deviation drastically
- Since Mx is essentially an outlier cluster, it should be neglected from the mean and STD

Task 2: K-Means Clustering

K-Means Clustering

- Using the KMeans function in sklearn library, I created a Gaussian Mixture Model for each both datasets using N = 2, and N= 2 clusters for baseline and metformin cells, respectively
  - The reason this is different from the GMM clustering, was because in order to find the optimum cluster number, I performed a silhouette analysis
    - The baseline dataset had the highest score of .919 for N = 2
    - The metformin dataset had the highest score of .913 for N = 2

K-Means Clustering Analysis

- The means_ attribute for both K-means cluster analysis represents the average expression for each cluster for each gene.
  - I.e for each of the 1170 genes, there are 2 means in the baseline dataset, one for each cluster
  - I.e. for each of the 1170 genes, there are 2 means in the metformin dataset, one for each cluster
- The baseline dataset is split up into 2 clusters: one with 152 Cells and one with 17 cells
- The metformin dataset is split up into 2 clusters: one with 161 cells and one with 16 cells.

Modifying the original datasets

	Chr	GeneID	StartCoding	Stop	CodingLength	1	2
0	chr1	AKR1A1	46016455	46035723	1580	23.878180	59.520515
1	chr1	ANP32E	150190717	150208504	3553	551.845311	314.138091
2	chr1	ARF1	228270361	228286913	2198	113.391566	76.383464
3	chr1	ARPC5	183595332	183604985	1982	106.269814	51.795593
4	chr1	ATP5F1	111991743	112004525	2101	214.599276	71.062689

	Chr	GeneID	Start	Stop	CodingLength	1	2
0	chr1	ANKRD36BP1	168214819	168216668	1850	6.961783	65.599580
1	chr1	ANP32E	150190717	150208504	3553	589.414935	346.164962
2	chr1	APOA1BP	156561558	156564091	1121	34.144335	53.829391
3	chr1	ARF1	228270361	228286913	2198	129.281934	148.067612
4	chr1	ARPC5	183595332	183604985	1982	132.848917	131.391700

- I dropped each of the original cells columns in both datasets
- I appended the mean of each cluster for each gene to the original dataset
- Now, we have a smaller dataset that is much easier to do various analysis on

Re-Defining Clusters

- Bu should be cluster 2 because it has 17 cells in it
- Bv should be cluster 1 because it has 152 cells in it
- Mx should be cluster 2 because it has 16 cells in it
- My should be cluster 1 because it has 161 cells in it
- Based on this definition, Bu < Bv and Mx < My

Filtering Datasets

```
Kmetformin_upMy_downMxDF = metformin_Kmeans
Kmetformin_upMy_downMxDF = Kmetformin_upMy_downMxDF[Kmetformin_upMy_downMxDF['2'] < rpkmThreshold]
Kmetformin_upMy_downMxDF = Kmetformin_upMy_downMxDF[Kmetformin_upMy_downMxDF['1'] > rpkmThreshold]

Kbaseline_upBv_downBu_DF = baseline_Kmeans
Kbaseline_upBv_downBu_DF = Kbaseline_upBv_downBu_DF[Kbaseline_upBv_downBu_DF['2'] < rpkmThreshold]
Kbaseline_upBv_downBu_DF = Kbaseline_upBv_downBu_DF[Kbaseline_upBv_downBu_DF['1'] > rpkmThreshold]
```

- Using these 2 snippets of code, I filitered both datasets to only include genes that met the following conditions:
  - For Metformin, Mx < rpkmThreshold < My
    - This left only 77 remaining genes
  - For Baseline, Bu < rpkmThreshold < Bv
    - This left only 82 remaining genes

Further Filtration of Datasets

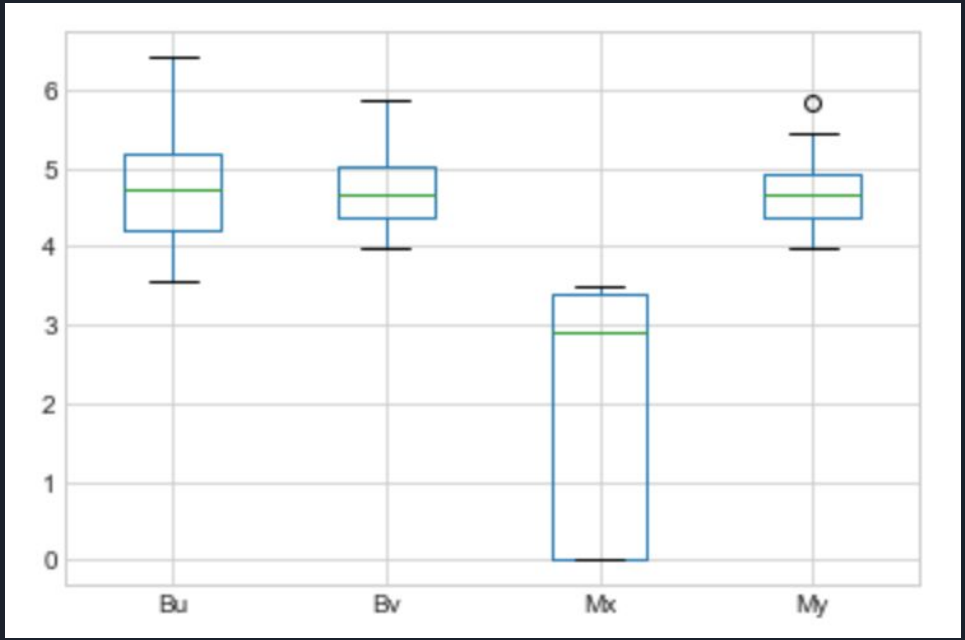
- After removing genes that were common to both baseline_upBv_downBu_DF and metformin_upMy_downMx_upMz_DF from metformin_upMy_downMx_upMz_DF, there were only 61 remaining genes
- After this, I filtered the original baseline and metformin datasets, to only include genes that overlapped between the mean baseline dataset and the remaining 61 Genes that were filtered.
    - This left both datasets with only 35 Genes that represent the downregulated genes.

Combined Dataframe

	Bu	Bv	Mx	My	Mean	STD
0	5.294395	5.100574	3.241804e+00	4.868313	4.626271	0.727556
1	5.366940	4.113803	1.421085e-14	4.209736	3.422620	1.821647
2	5.098730	4.709254	2.842171e-14	4.681207	3.622298	1.876361
3	5.165655	4.901507	3.286394e+00	4.906031	4.564897	0.667107
4	4.830385	4.283786	0.000000e+00	4.376657	3.372707	1.751454

- This dataframe was created by appending the Bu, Bv, Mx, and Mz clusters for each gene
- I then applied the log + 1 transformation
- Then, I found the mean and STD

Visualization Boxplot



- This boxplot is a visualization of the relationship between the various clusters
- For K-means, I wound up including Mx in the mean and std calculation, but based on the boxplot, the interquartile range is quite large
- As a result, if i were to replicate this, I would rethink whether or not to include the Mx in the mean/std calculation

GMM Clustering vs. K-Means Clustering

- After using both types of clustering, K-means clustering was unsuccessful in replicating the results of GMM
  - K-Means clustering resulted in only 35 total downregulated genes as opposed to the 221 downregulated genes
  - K-means was not comprehensive in including all of the pertinent downregulated genes between Metformin and Baseline cells

Task 3: Principal Component Analysis

Intro to Principal Component Analysis

- After transposing the original datasets, I found the covariance of gene expression across both datasets as well as the mean vectors
- Using the np.linalg.eig function, I found the eigenvalues and eigenvectors of the data

Ideal Number of Principal Components

```
total = sum(baseline_eig_vals)
baseline_var_exp = [(i / total)*100 for i in sorted(baseline_eig_vals, reverse=True)]
baseline_cum_var_exp = np.cumsum(baseline_var_exp)
print(baseline_cum_var_exp)

total = sum(metformin_eig_vals)
metformin_var_exp = [(i / total)*100 for i in sorted(metformin_eig_vals, reverse=True)]
metformin_cum_var_exp = np.cumsum(metformin_var_exp)
print(metformin_cum_var_exp)
```

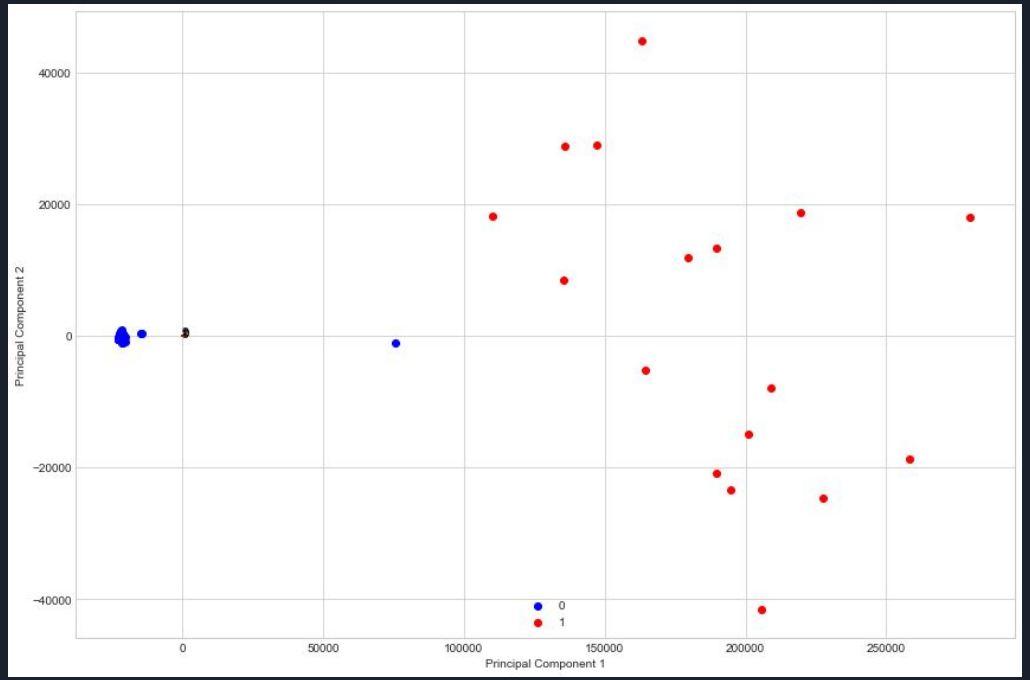
```
[ 96.30770128+0.00000000e+00j  97.53051717+0.00000000e+00j
  97.9197728 +0.00000000e+00j ... 100.        +2.27327242e-17j
 100.        +0.00000000e+00j 100.        +0.00000000e+00j]
[ 96.5753673 +0.0000000e+00j  97.34256751+0.0000000e+00j
  98.02445192+0.0000000e+00j ... 100.        +1.1967404e-17j
 100.        +0.0000000e+00j 100.        +0.0000000e+00j]
```

- For each dataset, I calculated the cumulative covariance
- For the baseline and metformin datasets, the first two components account for 97.53% and 97.34% of the total variance, respectively
- Hence, a two principal component analysis should suffice for this problem

Finding and Using the Projection Matrix

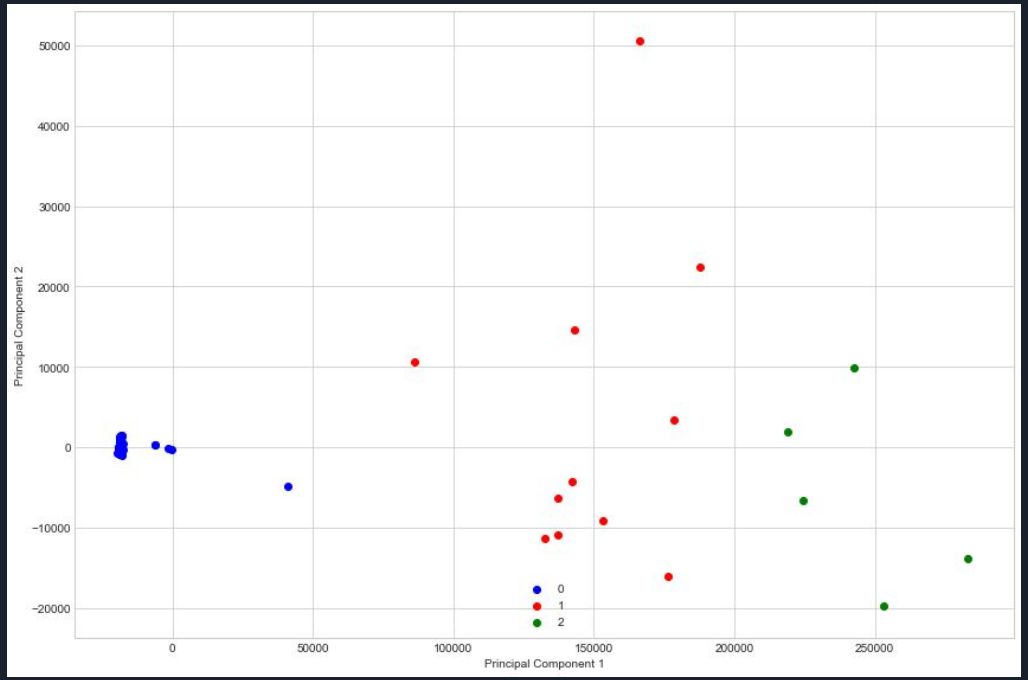
- Using the np.hstack function, I reshaped each dataset to 2 dimensions for each of the 1170 genes in order to find the projection matrix for each dataset
- Then, I projected the original dataset onto this 2-dimensional space by taking the product of the original matrix and the projection matrix
- This entire process can be simplified using the sklearnPCA function in the sklearn library
    - I did it both ways so I can understand the underlying linear algebra concepts for my own benefit; I wind up using the projected matrix from the library when creating my scatter plot

PCA Visualization: Baseline Clusters



- Using the values from the projected matrix baselineY_sklearn, I plotted all of the values for each cell in the baseline dataset
- I also used the original clusters from my GMM analysis to divide them into two clusters in order to see how the points vary within clusters

PCA Visualization: Metformin Clusters



- Using the values from the projected matrix metforminY_sklearn, I plotted all of the values for each cell in the baseline dataset
- I also used the original clusters from my GMM analysis to divide them into three clusters in order to see how the points vary within clusters

PCA Conclusion

- In both datasets, the metformin and the baseline, cluster 0, which is the largest clusters in both datasets by far have minimal variance between principal component values
    - Cluster 1 in the baseline cells and clusters 1 and 2 in the metformin cells are widely scattered across the plot
- This shows that the cells that are in Cluster 0 for both datasets are minimally impacted by the use of metformin and most of the differences can be accounted for in the cells that are parts of the other clusters
- This analysis conclusively shows which cells metformin has the most impact on