Course Project

Topic 1

Colon Cancer Data Analysis Using Logistic Regression model.

Topic 2

Protein Expression Analysis Using ANOVA.

Topic 1

Colon Cancer Data Analysis Using Logistic Regression model.

Background

Colon cancer is a prevalent gastrointestinal malignancy worldwide [1]. Advances in diagnostic, prognostic, predictive, and treatment approaches have been explored. Understanding colon cancer features, especially gene expression patterns, is crucial for diagnosis and treatment. This analysis will use the gene expression feature to explore the logistic regression model, aiding in the identification of tumor tissue for further diagnosis or targeted therapy based on gene features.

Specific Aims

This analysis aims to identify the most relevant genes as inputs and employ them to classify tumor tissue from normal tissues.

- **Aim 1:** Identify significant genes for regression inputs by comparing expression levels between carcinoma and normal tissues.
- **Aim 2:** Implement logistic regression using selected genes to mitigate issues stemming from high feature dimensionality.

Approach

1. Gene Selection:
    - Use paired T-test to identify significantly expressed genes. By comparing the gene expression level in carcinoma tissue and normal tissue.
    - Apply Principal component analysis (PCA) to reduce the feature dimension.
2. Logistic Regression:
    - Employ logistic regression on cleaned gene data.
    - Utilize cross-validation for model assessment due to limited cases.

Data description

The original data comprises expression levels of 2000 genes across 62 samples. However, after cleaning and normalization, only 18 carcinoma and paired

normal tissue samples, along with 4 adenoma samples and their controls, are retained. This analysis focuses on the 18 carcinoma and 18 paired normal tissue samples for logistic regression, utilizing Python version 3.9.12.

Additionally, gene descriptions and patient information are present in the data. Duplicates in Accession numbers exist in the original data, with different accession numbers sharing the same description, indicating the analysis of the same gene. Further details on data cleaning will be elaborated.

Data link: http://genomics-pubs.princeton.edu/oncology/

Data Processing and Analysis

1. Import data and data check

In Python, import the dataset for analysis. The initial inspection reveals the first 5 rows of the original dataset named "df," comprising 7,457 entries and 38 columns. The "Accession numbers" correspond to the gene chip version used for detection, and the "Description" signifies the gene description. The columns with label "Tumor" and "Normal" in the dataset denote the origin of the data, distinguishing between samples from tumor and normal tissues, respectively.

	Accession Number	Description	\
0	D00003	"Human liver cytochrome P-450 mRNA, complete cds"	
1	D00003	"Human liver cytochrome P-450 mRNA, complete cds"	
2	D00003	"Human liver cytochrome P-450 mRNA, complete cds"	
3	D00015	"Human prion protein mRNA, human PrP 27-30 mRN…	
4	D00102	"Human lymphotoxin (LT) mRNA, complete cds"	

	Tumor 27	Tumor 29	Tumor 34	Tumor 28	Tumor 35	Tumor 8	Tumor 3	\
0	1.990224	7.173473	0.658535	-2.776356	0.444464	10.222021	14.886598	
1	-5.473116	1.471482	6.749979	7.634980	6.222496	-6.133213	7.145567	
2	0.995112	-3.862639	-2.634138	2.776356	-3.259403	-13.970095	-7.741031	
3	26.370469	12.507594	12.841423	9.023159	1.629701	9.881287	20.245774	
4	4.668679	2.497183	0.473626	9.592017	-5.261476	2.989451	-5.448153	

	Tumor 9	…	Normal 4	Normal 32	Normal 39	Normal 10	Normal 33	\
0	-2.016190	…	625.782641	0.312526	0.000000	74.232961	4.325167	
1	-14.113332	…	418.377846	10.000824	-1.880271	79.899599	6.920267	
2	-15.457459	…	433.988959	-6.875566	-15.668925	64.599676	4.757684	
3	17.473649	…	23.639686	79.381540	21.936495	51.566408	58.822270	
4	-1.841628	…	1.226149	-11.386648	-2.871692	5.067382	2.019791	

	Normal 5	Normal 11	Normal 6	Normal 12	Normal 40
0	13.378411	23.585107	4.373810	3.879600	-0.750229
1	-1.254226	19.197180	-2.783334	3.879600	14.254345
2	-4.180753	0.000000	3.180953	-17.458199	-1.125343
3	39.717157	42.233796	71.173821	42.675598	60.393407
4	-5.922169	-0.355968	-1.785324	7.533962	-6.562813

2. Data cleaning

2.1 Check for Duplicates

Utilize the following code (left) to identify duplicate rows in the dataset. Upon inspection, 1,546 duplicate rows are identified. Subsequently, we will calculate the averages for these duplicates and aggregate the data using the code below:

Find the duplicate rows.

```
#Identify duplicates based on 'Accession Number'
duplicates = df.duplicated(subset='Accession Number', keep=False)
summary = duplicates.value_counts()

# Print the summary
print(summary)
```

```
False    5911
True     1546
dtype: int64
```

Removed the duplicate rows.

```
# Step 1: Handling non-numeric columns (like 'Description')
# First, we'll aggregate these columns by joining unique non-null values
description_agg = df[duplicates].groupby('Accession Number')['Description'].apply(
    lambda x: ' '.join(x.dropna().unique())).reset_index()

# Step 2: Calculate averages for duplicates and aggregate data for numeric columns
# Select only the columns with float64 type for averaging
float_cols = df.select_dtypes(include=['float64']).columns

# Aggregate the data for numeric columns
# Group by 'Accession Number', calculate mean for float columns
numeric_agg = df[duplicates].groupby('Accession Number')[float_cols].mean().reset_index()

# Merge the aggregated description and numeric columns
aggregated_df = pd.merge(description_agg, numeric_agg, on='Accession Number')

# Step 3: Include the non-duplicate rows in the final DataFrame
non_duplicates_df = df[~duplicates]
df_duplicate_removed = pd.concat([non_duplicates_df, aggregated_df], ignore_index=True)

# Display the info of the final DataFrame
df_duplicate_removed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6597 entries, 0 to 6596
```

2.2 Adjust Values Less Than 10 to 10.

Referring to microarray publications[3], to circumvent division by 0 or dealing with negative numbers, values equal to or less than 10 are adjusted to 10. This adjustment helps mitigate the impact of low-level expression transcripts on reported changes in expression.

Execute the following code to obtain the adjusted dataframe, denoted as "adjust_df," where intensity values less than 10 are replaced by 10:

```
# replace the value  less than 10 to 10
adjust_df = df_duplicate_removed.applymap(lambda x: 10 if isinstance(x, (int, float)) and x < 10 else x)

# Display the modified DataFrame
display(adjust_df.head(3))
display(adjust_df.info())
```

	Accession Number	Description	Tumor 27	Tumor 29	Tumor 34	Tumor 28	Tumor 35	Tumor 8	Tumor 3	Tumor 9	...	Normal 4	Normal 32	Normal 39
0	D00015	"Human prion protein mRNA, human PrP 27-30 mRN...	26.370469	12.507594	12.841423	10.000000	10.0	10.000000	20.245774	17.473649	...	23.639686	79.38154	21.936495
1	D00102	"Human lymphotoxin (LT) mRNA, complete cds"	10.000000	10.000000	10.000000	10.000000	10.0	10.000000	10.000000	10.000000	...	10.000000	10.00000	10.000000
2	D00137	"Human class I alcohol dehydrogenase beta-1 su...	17.912017	12.691530	10.865819	28.457654	10.0	63.035796	26.200413	135.756815	...	47.279373	137.51133	118.457074

3 rows × 38 columns

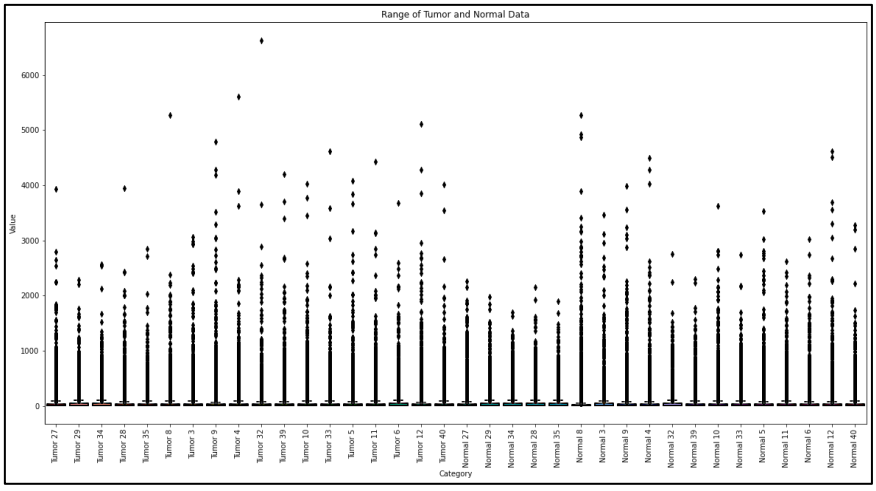
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6597 entries, 0 to 6596
```

2.3 Check for Missing Values

Prior to conducting any analysis, a preliminary check for missing values is imperative, alongside an exploration of the value ranges.

Upon inspection, only one missing value is identified in the "Description" column, allowing for further analysis. The figure depicting the "Range of Tumor and Normal Data" highlights the presence of outliers and a broad dimension and value range. Recognizing the substantial impact of unscaled features on feature importance, the values will be scaled using a scaler. Subsequently, PCA will be employed for dimension reduction.

	Missing Values	% of Total Values
Accession Number	0	0.000000
Description	1	0.015158
Tumor 27	0	0.000000
Tumor 29	0	0.000000
Tumor 34	0	0.000000
Tumor 28	0	0.000000
Tumor 35	0	0.000000
Tumor 8	0	0.000000
Tumor 3	0	0.000000
Tumor 9	0	0.000000
Tumor 4	0	0.000000
Tumor 32	0	0.000000
Tumor 39	0	0.000000
Tumor 10	0	0.000000
Tumor 33	0	0.000000
Tumor 5	0	0.000000
Tumor 11	0	0.000000
Tumor 6	0	0.000000
Tumor 12	0	0.000000
Tumor 40	0	0.000000
Normal 27	0	0.000000
Normal 29	0	0.000000
Normal 34	0	0.000000
Normal 28	0	0.000000
Normal 35	0	0.000000
Normal 8	0	0.000000
Normal 3	0	0.000000
Normal 9	0	0.000000
Normal 4	0	0.000000
Normal 32	0	0.000000
Normal 39	0	0.000000
Normal 10	0	0.000000
Normal 33	0	0.000000
Normal 5	0	0.000000
Normal 11	0	0.000000
Normal 6	0	0.000000
Normal 12	0	0.000000
Normal 40	0	0.000000



3. Paired T-Test for Selection of Significantly Expressed Genes

The expression data originate from paired tumor and normal tissues, necessitating the execution of a paired t-test to identify genes with significant expression differences.

Let μ_1 and μ_2 denote the average expression levels of each gene in tumor and normal tissues, respectively.

(1) H_0: $\mu_1 = \mu_2$ or $\mu_D = \mu_1 - \mu_2 = 0$.
(2) H_1: $\mu_1 \neq \mu_2$ or $\mu_D = \mu_1 - \mu_2 \neq 0$.
(3) α=0.05.
(4) The p-value is computed for each gene. Given the multitude of t-tests conducted (one for each gene), it becomes imperative to perform an adjustment for multiple comparisons to control the rate of false positives, also known as Type I errors. To accomplish this, the Benjamini-Hochberg procedure is employed to adjust the p-values.

```
#Loop through each gene to perform a paired t-test between its tumor and normal samples.
p_values = []
for index, row in adjust_df.iterrows():
    tumor_values = row[2:20]  # Tumor columns
    normal_values = row[20:]  # Normal columns
    t_stat, p_val = stats.ttest_rel(tumor_values, normal_values)
    p_values.append(p_val)

adjust_df['p_value'] = p_values
```

```
# Use Benjamini-Hochberg procedure to adjust these p-values.
from statsmodels.stats.multitest import multipletests

# Apply multiple testing correction
adjusted_p_values = multipletests(p_values_nonan, method='fdr_bh')[1]

# Add the adjusted p-values to the DataFrame
adjust_df['adjusted_p_value'] = adjusted_p_values

display(adjust_df.head(3))
```

	Accession Number	Description	Tumor 27	Tumor 29	Tumor 34	Tumor 28	Tumor 35	Tumor 8
0	D00015	"Human prion protein mRNA, human PrP 27-30 mRN…	26.370469	12.507594	12.841423	10.000000	10.000000	10.000000
1	D00102	"Human lymphotoxin (LT) mRNA, complete cds"	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000
2	D00137	"Human class I alcohol dehydrogenase beta-1 su…	17.912017	12.691530	10.865819	28.457654	10.000000	63.035796
3	D00173	Human liver cytochrome P-450 mRNA	10.000000	10.000000	10.340111	10.000000	10.041955	10.000000
4	D00265	"Human cytochrome c mRNA, carboxyl-terminal re…	153.220213	144.986886	131.925550	72.900102	183.579488	77.928955

	Tumor 3	Tumor 9	…	Normal 39	Normal 10	Normal 33	Normal 5
0	20.245774	17.473649	…	21.936495	51.566408	58.822270	39.717157
1	10.000000	10.000000	…	10.000000	10.000000	10.000000	10.000000
2	26.200413	135.756815	…	118.457074	92.366203	95.586188	50.169040
3	10.000000	10.000000	…	10.000000	10.000000	10.000000	10.000000
4	51.374323	72.006462	…	139.556942	79.908722	86.652650	91.273064

	Normal 11	Normal 6	Normal 12	Normal 40	p_value	adjusted_p_value
0	42.233796	71.173821	42.675598	60.393407	0.000053	0.001850
1	10.000000	10.000000	10.000000	10.000000	NaN	1.000000
2	85.016083	78.728584	81.471597	206.312882	0.000041	0.001586
3	10.000000	71.621286	27.035137	10.000000	0.112747	0.347258

NaN p-values in the "p_value" column resulting from the paired t-tests are attributed to sample pairs with identical values, as values less than 10 were replaced with 10. To address this, a approach is taken by assigning a default value, such as 1, to NaN p-values. This value signifies no statistical significance, enabling subsequent p-value adjustments.

Genes with p-values less than the significance level of 0.05 are chosen as significant for subsequent logistic regression analysis. Following this filtration process, a total of 823 genes are retained for further analysis.

```
significant_genes = adjust_df[adjust_df['adjusted_p_value'] < 0.05]

display(significant_genes.shape)
```

```
(823, 40)
```

4. Dimension reduction using Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is employed as a dimensionality reduction method to condense large datasets by transforming a multitude of variables into a smaller set that still preserves the majority of information from the original set. In this context, PCA facilitates the use of a reduced set of features that captures approximately 95% of the pertinent information crucial for training the logistic regression model.

4.1 Reconstructing Data for Enhanced Analysis and Visualization

To facilitate easier analysis handling and visualization, the data is reconstructed. The process involves transposing the data and categorizing all tumor tissue samples as "Tumor" and normal samples as "Normal." This entails separating the features from the labels. This reconstructed data frame, denoted as "new_df", serves as the basis for subsequent analysis, with features and labels separated for ease of processing.

Reconstruct the dataframe

```
#1. Use the gene accession number as the column name in the new DataFrame.
access_number = significant_genes.iloc[:, 0]

#2. Use the gene expression values as the input values: X
significant_genes_expression = significant_genes.iloc[:, 2:38]

#3. Transpose the expression data
significant_genes_expression_transposed = significant_genes_expression.transpose()

#4. Create a new DataFrame with gene accession number as columns
new_df = pd.DataFrame(data=significant_genes_expression_transposed.values, columns= access_number)

#5. Create the sample labels
num_samples = new_df.shape[0]  # Total number of samples
num_tumor = num_samples // 2  # Number of tumor samples
num_normal = num_samples - num_tumor  # Number of normal samples, in case the total is odd

# Create a list of sample labels with 'Tumor' for the first half and 'Normal' for the second half
sample_labels = ['Tumor'] * num_tumor + ['Normal'] * num_normal

# Add the labels as a new column of the new DataFrame
new_df['Sample Type'] = sample_labels
```

Show the new dataframe

```
display(access_number.shape)
display(significant_genes_expression.shape)
display(significant_genes_expression_transposed.info())
```

```
(823,)

(823, 36)

<class 'pandas.core.frame.DataFrame'>
Index: 36 entries, Tumor 27 to Normal 40
Columns: 823 entries, 0 to 6594
dtypes: float64(823)
memory usage: 231.8+ KB

None
```

```
display(new_df.shape)
display(new_df.head(2))
display(new_df.tail(2))
```

```
(36, 824)
```

Accession Number	D00015	D00137	D00596	D00749	D00761	…	X55187	X56597	X61587	X68277	X84707	Z24725	Z36531	Z46629	Z49269	Sample Type
0	26.370469	17.912017	70.030183	65.050259	202.505298	…	113.635491	116.915691	29.983293	120.664491	103.491651	10.0	10.000000	22.961398	39.306925	Tumor
1	12.507594	12.691530	138.010991	64.094369	260.268323	…	231.949080	83.812388	89.121694	200.604610	69.527510	10.0	28.482584	35.251172	26.364047	Tumor

[2 rows x 824 columns]

Accession Number	D00015	D00137	D00596	D00749	D00761	…	X55187	X56597	X61587	X68277	X84707	Z24725	Z36531	Z46629	Z49269	Sample Type
34	42.675598	81.471597	22.601887	105.475471	149.364594	…	75.855433	20.612889	56.923270	118.730243	112.508395	10.0	22.261921	10.000000	71.125997	Normal
35	60.393407	206.312882	40.783197	103.598695	183.806023	…	288.740051	32.914760	135.162702	212.741743	108.408042	10.0	13.112384	14.807182	197.810291	Normal

[2 rows x 824 columns]

Separate features and labels

```
# Separate the features and the labels
X = new_df.iloc[:, :-1]  # All columns except the last one
y = new_df.iloc[:, -1]   # The last column contains the tissue type

# Convert 'Tumor'/'Normal' labels to a binary variable if they're not already
y = y.map({'Tumor': 1, 'Normal': 0})

display(X.head(2))
display(y.head(2))
```

Accession Number	D00015	D00137	D00596	D00749	D00761	D10523	D11086	D13627	D13634
0	26.370469	17.912017	70.030183	65.050259	202.505298	92.047863	47.765378	95.594393	152.763588
1	12.507594	12.691530	138.010991	64.094369	260.268323	84.426262	96.382050	156.994912	113.385214

2 rows × 823 columns

```
0    1
1    1
Name: Sample Type, dtype: int64
```

4.2 Use PCA for dimension reduction.

To mitigate the impact of scale on variable importance, scaling is initially applied. Subsequently, Principal Component Analysis (PCA) is employed to identify components that capture 95% of the information in the features. The percentage of information retained is showed in the cumulative explained variation results.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Standardize the features by using the standardScaler library
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=0.95)  # Retain 95% of the variance
X_pca = pca.fit_transform(X_scaled)

print("The shape of data before PCA:", X_scaled.shape)

print("-----------------------------------")
print("The shape of data after PCA:", X_pca.shape)

# View the explained variance to justify the results
pcaSummary_df = pd.DataFrame({'Standard deviation': np.sqrt(pca.explained_variance_),
                              'Proportion of variance': pca.explained_variance_ratio_,
                              'Cumulative proportion': np.cumsum(pca.explained_variance_ratio_)})
pcaSummary_df = pcaSummary_df.transpose()
pcaSummary_df.columns = ['PC{}'.format(i) for i in range(1, len(pcaSummary_df.columns) + 1)]
display(pcaSummary_df.round(4))

# Cumulative explained variation
cumulative_var = np.cumsum(pca.explained_variance_ratio_)
print('The cumulative explained variation ratio for data is:', cumulative_var[-1])
```

```
The shape of data before PCA: (36, 823)
-----------------------------------
The shape of data after PCA: (36, 28)
                            PC1     PC2     PC3     PC4     PC5     PC6  \
Standard deviation      16.8578  9.6405  6.3473  5.9333  5.6708  5.1270
Proportion of variance   0.3357  0.1098  0.0476  0.0416  0.0380  0.0311
Cumulative proportion    0.3357  0.4455  0.4931  0.5347  0.5727  0.6037

                            PC7     PC8     PC9    PC10  …    PC19    PC20  \
```



```
Standard deviation      4.8826  4.7386  4.5343  4.3512  …  3.3279  3.2974
Proportion of variance  0.0282  0.0265  0.0243  0.0224  …  0.0131  0.0128
Cumulative proportion   0.6319  0.6584  0.6827  0.7051  …  0.8596  0.8725

                           PC21    PC22    PC23    PC24    PC25    PC26  \
Standard deviation      3.1729  3.1610  3.0265  2.9430  2.9103  2.8208
Proportion of variance  0.0119  0.0118  0.0108  0.0102  0.0100  0.0094
Cumulative proportion   0.8844  0.8962  0.9070  0.9172  0.9272  0.9366

                           PC27    PC28
Standard deviation      2.7296  2.7244
Proportion of variance  0.0088  0.0088
Cumulative proportion   0.9454  0.9542

[3 rows x 28 columns]
The cumulative explained variation ratio for data is: 0.9542050047190851
```

From the results, it is observed that the first 28 principal components can explain about 95% of the variances. These 28 components will be utilized for the subsequent logistic regression analysis.

5. Logistic regression analysis using 28 principal components

The dataset is split into 75% training data and 25% test data for model training and testing purposes.

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.25, random_state=42)

display(X_train.shape)
display(y_train.shape)
display(X_test.shape)
display(y_test.shape)
```

```
(27, 28)

(27,)

(9, 28)

(9,)
```

```
from sklearn.linear_model import LogisticRegression
# Initialize and train logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
from sklearn.metrics import classification_report, confusion_matrix

# Predict on test set
y_pred = model.predict(X_test)
```

6. Model evaluation with K-fold cross validation

Due to the limited sample size, K-fold cross-validation is employed for model evaluation. This approach involves splitting the dataset randomly into K groups, using some for training and others for testing, and repeating this process to obtain a less biased estimate of the model's skill.

The cross-validation scores, indicating the accuracy achieved in each fold, are [1.0, 1.0, 1.0, 0.85714286, 1.0] (results are as below), suggesting consistently high accuracy. The mean accuracy, representing the average across folds, is 0.97, indicating strong overall performance. The standard deviation of accuracy measures the variability in scores, and a lower value (0.06 in this case) signifies more consistent performance across folds.

This comprehensive evaluation underscores the robustness and reliability of the logistic regression model trained on the 28 principal components.

```
from sklearn.model_selection import cross_val_score, KFold

# Define the number of folds (e.g., 5)
num_folds = 5
kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)
```

```
scores = cross_val_score(model, X_pca, y, cv=kf, scoring='accuracy')
# Print the cross-validation scores
print("Cross-Validation Scores:", scores)

# Calculate the mean and standard deviation of the scores
mean_accuracy = scores.mean()
std_accuracy = scores.std()

print(f"Mean Accuracy: {mean_accuracy:.2f}")
print(f"Standard Deviation of Accuracy: {std_accuracy:.2f}")
```

```
Cross-Validation Scores: [1.         1.         1.         0.85714286 1.        ]
Mean Accuracy: 0.97
Standard Deviation of Accuracy: 0.06
```

7. Discussion

1. Significant Expressed Genes:

Utilizing paired t-tests, we identified 823 genes with significantly different expression levels, emphasizing their potential role in distinguishing between tumor and normal tissues.

#2. Principal Component Analysis (PCA):

Through PCA, 28 principal components were discovered, collectively explaining approximately 95% of the variances in the dataset. This reduction in dimensionality facilitates effective feature representation.

#3. Logistic Regression Model Evaluation:
Employing logistic regression and K-fold cross-validation (with a mean accuracy of 0.97), the model consistently demonstrated high performance across various data subsets.
The low standard deviation (0.06) underscores the model's consistent accuracy.
Exceptional accuracy scores (1.0 in most folds) highlight the robust and reliable predictive capability of the logistic regression model, distinguishing between Tumor and Normal tissue types based on the 28 selected gene features.

Reference

[1] Mokhtari K, et al. Colon cancer transcriptome. Prog Biophys Mol Biol. 2023;180-181:49-82.
[2] Ben-Dor A, et al. Tissue classification with gene expression profiles. J Comput Biol. 2000;7(3-4):559-83.
[3] Notterman, Daniel A., et al. "Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma, and normal tissue examined by oligonucleotide arrays." Cancer research 61.7 (2001): 3124-3130.

Topic 2

Protein Expression Analysis Using ANOVA.

Background

Protein expression signals are crucial in biological research. In this topic, the data if the expression of 77 proteins in the brains of mice from different experimental groups will be used. The original article utilized 72 mice, divided into 38 controls and 34 trisomic mice (a model for Down syndrome), across eight classes differentiated by genotype, cerebral stimulation, and pharmacological treatments[1]. The experimental design aimed to investigate the effects of three distinct factors on protein expression levels:

(1) Genetic background: comparing control mice with genetically modified trisomic mice. (2) Environmental stimulation: assessing the effects of cerebral stimulation (c/s or s/c).(3) Drug treatment: evaluating the impact of Memantine versus saline as a control.
This comprehensive analysis is geared toward understanding how these factors influence disease progression, response to environmental stimuli, and pharmacological efficacy. Furthermore, identifying variations in specific proteins among the groups may pinpoint potential therapeutic targets for future drug development.

Aims

This project employs a three-factor ANOVA to examine variances in protein expression across different sample groups. The aim is to assess the impact of genetic background, drug treatment, and environmental factors on protein levels in a Down syndrome mouse model.

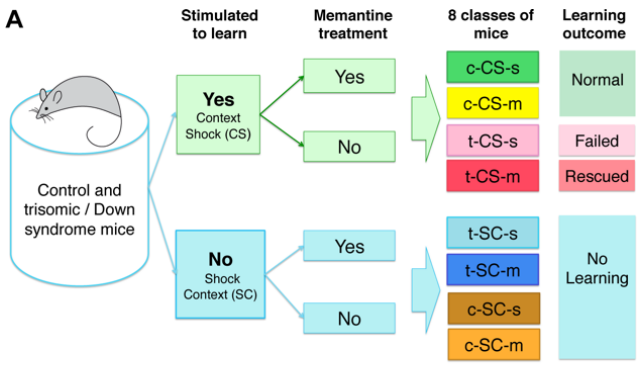
Approaches

Implement three-factor ANOVA to dissect the influence of each factor, using expression data from 77 proteins.

Data description

This dataset comprises 77 protein samples from the brains of Down syndrome model mice and their corresponding wild type controls. These subjects underwent context fear conditioning (CFC), both with and without memantine injections[1]. Each mouse was measured 15 times, with each measurement treated as a distinct observation. The figure below categorizes the mice based on class, treatment, learning outcome, and group size.

We presuppose normal distribution of the data, permitting the use of ANOVA for independent observations. This analysis will facilitate understanding the interplay among various factors influencing the study outcomes.



B

Control mice	#Mice
c-SC-s	9
c-SC-m	10
c-CS-s	9
c-CS-m	10
Trisomic mice	
t-SC-s	9
t-SC-m	9
t-CS-s	7
t-CS-m	9
Total	72

Classes of mice, adapted from the article by Clara Higuera, et al [1].

Data Processing and Analysis

1. Import data and data check

In Python, import the dataset for analysis. The column "MouseID" contains the mouse ID numbers, while columns such as "DYRK1A_N," "ITSN1_N," ..., "CaNA_N" represent observed protein names. The "Genotype" column denotes the genetic background, distinguishing between control and Down syndrome mice. The "Treatment" column indicates whether mice received drug treatment (Memantine) or were part of the control group (saline). The "Behavior" column reflects exposure to shock stimulation. Lastly, the "class" column corresponds to the classes displayed in the figure titled "Classes of Mice" above.

```
  MouseID  DYRK1A_N   ITSN1_N    BDNF_N     NR1_N    NR2A_N    pAKT_N  \
0   309_1  0.503644  0.747193  0.430175  2.816329  5.990152  0.218830
1   309_2  0.514617  0.689064  0.411770  2.789514  5.685038  0.211636
2   309_3  0.509183  0.730247  0.418309  2.687201  5.622059  0.209011

    pBRAF_N  pCAMKII_N   pCREB_N  ...   pCFOS_N     SYP_N  H3AcK18_N  \
0  0.177565   2.373744  0.232224  ...  0.108336  0.427099   0.114783
1  0.172817   2.292150  0.226972  ...  0.104315  0.441581   0.111974
2  0.175722   2.283337  0.230247  ...  0.106219  0.435777   0.111883

     EGR1_N  H3MeK4_N    CaNA_N Genotype  Treatment  Behavior   class
0  0.131790  0.128186  1.675652  Control  Memantine       C/S  c-CS-m
1  0.135103  0.131119  1.743610  Control  Memantine       C/S  c-CS-m
2  0.133362  0.127431  1.926427  Control  Memantine       C/S  c-CS-m

[3 rows x 82 columns]

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1080 entries, 0 to 1079
```

2. Missing value check

Verify the presence of missing values. The table below displays a portion of the missing value report. Upon inspection, proteins such as "BAD_N," "BCL2_N," "pCFOS_N," "H3AcK18_N," "EGR1_N," and "H3MeK4_N" exhibit a higher percentage of missing values. Consequently, these columns will be excluded from subsequent analyses. Although there are still some missing values in other columns, rows containing missing values will be removed during the ANOVA analysis.

	Missing Values	% of Total Values
MouseID	0	0.000000
DYRK1A_N	3	0.277778
ITSN1_N	3	0.277778
BDNF_N	3	0.277778
NR1_N	3	0.277778
NR2A_N	3	0.277778
pAKT_N	3	0.277778
pBRAF_N	3	0.277778
pCAMKII_N	3	0.277778
pCREB_N	3	0.277778
pELK_N	3	0.277778
SHH_N	0	0.000000
BAD_N	213	19.722222
BCL2_N	285	26.388889
pS6_N	0	0.000000
pCFOS_N	75	6.944444
SYP_N	0	0.000000
H3AcK18_N	180	16.666667
EGR1_N	210	19.444444
H3MeK4_N	270	25.000000
CaNA_N	0	0.000000
Genotype	0	0.000000
Treatment	0	0.000000

3. Three-factor ANOVA analysis

3.1 Three-factor ANOVA analysis in protein “DYRK1A_N”

A three-factor ANOVA analysis was employed to examine the interplay among genetic factors ("Genotype"), drug treatment ("Treatment"), and environmental stimulation ("Behavior"). Taking the protein "DYRK1A_N" as an example, the ANOVA table is presented below, labeled as the "ANOVA for a Three-Factor Experiment in protein “ DYRK1A_N”.

ANOVA for a Three-Factor Experiment in protein " DYRK1A_N"

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
model_1 = ols("""DYRK1A_N ~ C(Genotype) + C(Treatment) + C(Behavior) +
               C(Genotype):C(Treatment) + C(Genotype):C(Behavior) + C(Treatment):C(Behavior) +
               C(Genotype):C(Treatment):C(Behavior)""", data=df_pro).fit()

sm.stats.anova_lm(model_1, typ=2)
```

	sum_sq	df	F	PR(>F)
C(Genotype)	0.718954	1.0	16.156844	6.239697e-05
C(Treatment)	0.042170	1.0	0.947678	3.305313e-01
C(Behavior)	17.131538	1.0	384.992233	1.879031e-73
C(Genotype):C(Treatment)	0.645793	1.0	14.512718	1.471480e-04
C(Genotype):C(Behavior)	0.024089	1.0	0.541351	4.620349e-01
C(Treatment):C(Behavior)	0.012204	1.0	0.274253	6.006013e-01
C(Genotype):C(Treatment):C(Behavior)	0.775767	1.0	17.433602	3.217340e-05
Residual	47.568788	1069.0	NaN	NaN

In the ANOVA results, the "PR(>F)" is the p-value associated with the F-statistic. This is the probability of observing the data assuming the null hypothesis is true when comparing the significant level (set to 0.05 in this case).

Genotype: The Genotype factor has a significant effect on the dependent variable ($p < 0.0001$). With an F-statistic of 16.16. We can conclude that the different genotypes contribute to the variability in protein expression levels.

Treatment: The Treatment factor alone does not have a significant effect ($p = 0.3305$), indicated by the F-statistic of 0.9477. This suggests that treatment with memantine, by itself, doesn't contribute to a significant variation in the protein expression levels.

Behavior: The Behavior factor has a highly significant effect ($p < 0.0001$) with a very high F-statistic of 384.99, indicating that behavioral changes contribute substantially to the differences in protein expression levels.

Interaction: The interaction between Genotype and Treatment (with $p = 0.000147$, F-statistic of 14.51), and the three-way interaction between Genotype, Treatment, and Behavior is significant ($p < 0.0001$, F-statistic of 17.43). This means that the effect of the treatment varies depending on the genotype while the combined effect of these three factors is also significant

and should be considered when interpreting the impact on protein expression levels.

In summary, both the main effects of Genotype and Behavior and their three-way interaction with Treatment are significant contributors to DYRK1A protein expression levels. However, Treatment alone does not show a significant effect, nor do the two-way interactions of “Genotype: Treatment” and “Treatment: Behavior”.

3.2 Summary of the Three-factor ANOVA analysis in 71 proteins.

For the extended three-factor ANOVA analysis across 71 proteins(exclude the protein with high percentage of missing values), a similar code was employed for the analysis. To facilitate visualization, the results were summarized by selecting p-values less than 0.05, and the summary table is presented below. Additionally, the number of significant differences was calculated based on the comparison conditions.

From the summary ANOVA results (“ANOVA Summary in 71 proteins”), there are 349 rows, indicating that 349 factors or interactions among these 71 proteins exhibit significant differences compared to the 0.05 significance level. The detailed ANOVA summary (“Detailed ANOVA summary”) reveals that:

The "Genotype" factor demonstrates a significant effect on 46 proteins.
The "Treatment" factor exhibits a significant effect on 49 proteins.
The "Behavior" factor shows a significant effect on 63 proteins.
The interaction between "Genotype" and "Treatment" is significant in 39 proteins.
The interaction between "Treatment" and "Behavior" is significant in 40 proteins.
The interaction between "Genotype" and "Behavior" is significant in 57 proteins.
The interaction among these three factors is significant in 55 proteins.

This comprehensive analysis opens avenues for further exploration of specific proteins that may play a role in disease development.

Code for the ANOVA analysis in 71 proteins

```
import statsmodels.api as sm
import pandas as pd
from statsmodels.formula.api import ols

# 1. List of proteins. Do not include the protein with big portion of missing values.
columns_of_interest = ['DYRK1A_N', 'ITSN1_N', 'BDNF_N', 'NR1_N', 'NR2A_N', 'pAKT_N', 'pBRAF_N', 'pCAMKII_N', 'pCREB_

# 2. Initialize a summary dataframe
summary_df = pd.DataFrame(columns=['Protein', 'Comparison'])

# 3. Loop through each protein
for protein in columns_of_interest:
    # Define the formula for the model
    formula = f"{protein} ~ C(Genotype) * C(Treatment) * C(Behavior)"

    # Fit the model
    model = ols(formula, data=df_pro).fit()

    # Perform ANOVA
    anova_results = sm.stats.anova_lm(model, typ=2)

    # Check p-values and store significant results
    significant_comparisons = anova_results[anova_results['PR(>F)'] < 0.05][['F', 'PR(>F)']]
    significant_comparisons['Protein'] = protein
    significant_comparisons['Comparison'] = significant_comparisons.index
    significant_comparisons.reset_index(inplace=True, drop=True)

    # Append to the summary dataframe
    summary_df = pd.concat([summary_df, significant_comparisons[['Protein', 'Comparison', 'F', 'PR(>F)']]])

# 4.Display the summary dataframe
display(summary_df)
```

ANOVA Summary in 71 proteins

	Protein	Comparison	F	PR(>F)
0	DYRK1A_N	C(Genotype)	16.156844	6.239697e-05
1	DYRK1A_N	C(Behavior)	384.992233	1.879031e-73
2	DYRK1A_N	C(Genotype):C(Treatment)	14.512718	1.471480e-04
3	DYRK1A_N	C(Genotype):C(Treatment):C(Behavior)	17.433602	3.217340e-05
0	ITSN1_N	C(Genotype)	53.137252	6.037727e-13
...	...	...	...	...
1	CaNA_N	C(Treatment)	59.795808	2.415649e-14
2	CaNA_N	C(Behavior)	1563.044578	1.386525e-211
3	CaNA_N	C(Genotype):C(Treatment)	6.751325	9.496326e-03
4	CaNA_N	C(Treatment):C(Behavior)	40.983389	2.291671e-10
5	CaNA_N	C(Genotype):C(Treatment):C(Behavior)	19.741403	9.786117e-06

349 rows × 4 columns

Detailed ANOVA summary

```
# Count the number of occurrences for each label in the 'Comparison' column
comparison_counts = summary_df['Comparison'].value_counts()

# Display the counts
display(comparison_counts)
```

```
C(Behavior)                                63
C(Genotype):C(Behavior)                    57
C(Genotype):C(Treatment):C(Behavior)       55
C(Treatment)                               49
C(Genotype)                                46
C(Treatment):C(Behavior)                   40
C(Genotype):C(Treatment)                   39
Name: Comparison, dtype: int64
```

4. Discussion

1. Three-Factor ANOVA Analysis: Unraveling Relationships.

Conducting a three-factor ANOVA sheds light on the intricate relationship among genetic background, drug treatment, and environmental stimulation concerning protein expression levels in the mouse brain. The comprehensive ANOVA results presented here lay the foundation for in-depth analyses, especially for proteins that pique interest.

#2. Utilizing 15 Repeated Measurements.

This dataset leverages 15 repeated measurements as independent samples, providing an expanded sample size. Alternatively, averaging the 15 replicated measurements could serve as a representative record for individual mice.

#3. Exploring Factor Interactions in Protein Expression

A compelling avenue for exploration involves delving into the interactions between different factors within specific proteins. Understanding how these factors collectively influence protein expression requires consideration of established knowledge, including biological function, protein location, and protein-protein interactions. Such analyses contribute to interpreting the effects of these factors and unraveling the potential role of proteins in disease development.

References

[1] Higuera C, et al. Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome. PLoS One. 2015;10(6):e0129126.