STAT3006 Assignment 1

Tim Ryall

August 23, 2024

Please see zip for all full code. Code snippets will reference the relevant file as no code is to be included in this paper as per the instructions.

Question 1.

We are given a multivariate normal distribution with the following mean vector $\hat{\mu}_h$ and covariance matrix $\hat{\Sigma}_h$.

Given Information 1: Multivariate Normal Distribution

$$\hat{\mu}_h = \begin{pmatrix} 5.66 \\ 38.1 \\ 505 \\ 0.891 \\ 137 \end{pmatrix}, \hat{\Sigma}_h = \begin{pmatrix} 1.29 & -0.928 & 1.01 & 0.0235 & 0.0953 \\ -0.928 & 140 & 77.8 & 0.514 & 9.19 \\ 1.01 & 77.8 & 8757 & 1.82 & 23.0 \\ 0.0235 & 0.514 & 1.82 & 0.100 & 0.354 \\ 0.0953 & 9.19 & 23.0 & 0.354 & 19.5 \end{pmatrix}$$

This distribution represents the following variables:

$$\begin{pmatrix} \text{logcp} \\ \text{ejection\_fraction} \\ \text{sqrtplat} \\ \text{recipsc} \\ \text{serum\_sodium} \end{pmatrix}$$

This can be represented in python via arrays as follows

Code Snippet 1: Multivariate Normal Distribution

```
See "question_1_a.ipynb"
```

Listing 1: Parameters for the multivariate normal distribution given in question 1.a)

Question 1. (a)

We wish to generate 100 observations from the above distribution using only random numbers generated from the uniform distribution from 0 to 1 i.e. $U[0, 1]$. To do this will use the inverse transform approach mentioned in section 1.6.2 of the lecture notes.

The first step to this approach is to generate our random observations from the uniform distribution from 0 to 1 i.e. $U[0, 1]$. For each observation of our final distribution that we want, we will need to generate k observations from the uniform distribution, where k represents the number of variables in our final multivariate distribution. In this way we will be essentially generating data from a "multivariate uniform distribution".

So in our case we will be generating vectors of length five where each variable is generated from a uniform distribution from 0 to 1 i.e. $U[0, 1]$. i.e. generated observation U is generated as follows

$$\mathbf{U} = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}, \quad s.t. \quad u_i \overset{iid}{\sim} U[0, 1], \quad i = 0, 1, 2, 3, 4$$

We can generate random uniform observation vectors of size five from the uniform distribution (as shown above) through the use of numpy as follows.

Code Snippet 2: Generating Uniform Random Observations

```
See "question_1_a.ipynb"
```

Listing 2: Generating 100 Uniform random observation vetors of the same size as the desired observations from the final distribution

The next step is to convert these multivariate uniform observations into multivariate standard normal observations. This is done through the inverse transform approach that is convered in lecture notes section 1.6.2.

In summary we first let $Z = \Phi^{-1}(U)$ where we have that Φ is the cdf of the multivariate standard normal distribution, and Φ^{-1} is therefore the inverse cdf. Then we have that $P(Z \leq z) = P(\Phi^{-1}(U) \leq z)$. This inequality is maintained through transformation provided the sign is retained. We can therefore take the monotonic increasing function $\Phi()$ of both sides of the inequality giving $P(U \leq \Phi(z)) = \Phi(z)$. Thus we have that Z has a multivariate standard normal distribution.

We can apply this approach to convert the multivariate uniform observations into multivariate standard normal observations in python as follows.

Code Snippet 3: Generating Standard Normal Observations

```
See "question_1_a.ipynb"
```

Listing 3: Converting the multivariate uniform observations into multivariate standard normal observations

Now that we have multivariate standard normal observations we must transform these observations to follow the distribution given in "Given Information 1". To do this we need a decomposition of the covariance matrix such that $AA^T = \hat{\Sigma}_h$. We will do this via the Cholesky decomposition method which can be done in python as follows.

Code Snippet 4: Decomposition of covariance matrix

```
See "question_1_a.ipynb"
```

Listing 4: Decomposition of covariance matrix using Cholesky decomposition

Now we have the decomposition we can now transform our multivariate standard normal observations by letting $X = \hat{\mu}_h + \text{A Z}$. Using Property 1.5 from the lecture notes we can see that $AZ \sim N(0, AIA^T)$ which gives $X \sim N(\hat{\mu}_h, \hat{\Sigma}_h)$. Thus we have transformed our observations to follow the desired distribution. It should be noted that this approach is an approximate as the numbers generated are pseudo-random rather than truly random however modern random number generators, such as the ones used in python are very advanced so it is highly unlikely to detect any differences with smaller data sets. We preform this in python as follows.

Code Snippet 5: Observation transformation to desired Normal Distribution

```
See "question_1_a.ipynb"
```

Listing 5: Observation transformation to desired Normal Distribution

These final observations were saved to a csv which can be found in the zip attached to this report.

Using these observations we now wish to estimate the parameters of the multivariate normal distribution, namely the mean vector μ and covariance matrix Σ based on the above observations.

If we consider X as our matrix of observations and X_i $i = 0, ..., 99$ represents each observation (which will be a vector with a value for each variable). We can compute the estimated μ, which we can call $\tilde{\mu}$ as follows (Using the maximum likelihood derived formula from the lecture notes).

$$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Which can be done in python as follows.

Code Snippet 6: Estimation of mean vector

```
See "question_1_a.ipynb"
```

Listing 6: Estimation of mean vector

Similarly we can compute the estimated Σ, which we can call $\tilde{\Sigma}$ as follows(Using the maximum likelihood derived formula from the lecture notes).

$$\tilde{\Sigma} = \frac{1}{n-1}(X - \tilde{\mu})^\top (X - \tilde{\mu})$$

Which can be done in python as follows.

Code Snippet 7: Estimation of covariance matrix

```
See "question_1_a.ipynb"
```

Listing 7: Estimation of covariance matrix

Running these computations we achieve the following results:

$$\tilde{\mu} = \begin{pmatrix} 5.5101 \\ 39.127 \\ 506.65 \\ 0.87898 \\ 136.50 \end{pmatrix}, \tilde{\Sigma} = \begin{pmatrix} 1.1266 & -1.4698 & 5.8935 & -0.029180 & -0.02360 \\ -1.4698 & 150.84 & -38.346 & 1.0062 & 16.898 \\ 5.8935 & -38.346 & 9725.3 & 0.34479 & 30.115 \\ -0.02918 & 1.0062 & 0.34479 & 0.07184 & 0.2428 \\ -0.0236 & 16.898 & 30.115 & 0.2428 & 18.044 \end{pmatrix}$$

This can be seen to be similar to the initial distribution that generated the results especially for the mean vector. The variations seen are likely due to the small sample size of observations that these estimated parameters were generated from.

Finally we can estimate the correlation matrix for this data. We can compute the correlation matrix $\tilde{R}$ via the following formula where $\tilde{R}_{ij}$ represents the value in the ith row and jth column of $\tilde{R}$ (i.e. the correlation between variables i and j) and similarly $\tilde{\Sigma}_{ij}$ represents the value in the ith row and jth column of $\tilde{\Sigma}$ (i.e. the covariance between variables i and j).

$$\tilde{R}_{ij} = \frac{\tilde{\Sigma}_{ij}}{\sqrt{\tilde{\Sigma}_{ii}\tilde{\Sigma}_{jj}}}$$

Computing each of these values to generate the whole matrix can be done in a single command via python.

Code Snippet 8: Estimation of correlation matrix

```
See "question_1_a.ipynb"
```

Listing 8: Estimation of correlation matrix

Running these computations we achieve the following results:

$$\tilde{R} = \begin{pmatrix} 1.0000 & -0.11275 & 0.056300 & -0.10256 & -0.0052300 \\ -0.11275 & 1.0000 & -0.031660 & 0.30567 & 0.32390 \\ 0.056300 & -0.031660 & 1.0000 & 0.013050 & 0.071890 \\ -0.10256 & 0.30567 & 0.013050 & 1.0000 & 0.21326 \\ -0.0052300 & 0.32390 & 0.071890 & 0.21326 & 1.0000 \end{pmatrix}$$

Question 1. (b)

Given Information 2: Given Observation

We are given an observation where several variables are known. We will denote this observation X and let X_i denote the value of the ith variable. i.e. X_0 denotes the value for "logcp" of the given observation. We are given that:

$$\text{logcp} = X_0 = 8$$
$$\text{recipsc} = X_3 = 0.4$$
$$\text{serum\_sodium} = X_4 = 142$$

We will let B denote the set of variables that have known values (i.e. $B = \{0, 3, 4\}$) and A denote the set of variables that are unknown (i.e. $A = \{1, 2\}$) (ejection_fraction and sqrtplat). We can also let the given values be $x_b = \{8, 0.4, 142\}$

We know from the result (1.36) in the lecture notes we can calculate the the conditional distribution of the variables in A via the following formula.

$$X_A | (X_b = x_b) \sim N(\mu_{A|B}(x_b), \Sigma_{A|B}) \tag{1}$$

These terms can be computed via the definitions given in the lecture notes starting with definition (1.33) which gives us that the mean vector can be represented as:

$$\begin{aligned}
\mu_{A|B}(x_b) &= \hat{\mu}_{hA} + \hat{\Sigma}_{hAB}\hat{\Sigma}_{hBB}^{-1}(x_b - \hat{\mu}_{hB}) \\
&= \begin{pmatrix} 38.1 \\ 505 \end{pmatrix} + \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} 1.29 & 0.0235 & 0.0953 \\ 0.0235 & 0.100 & 0.354 \\ 0.0953 & 0.354 & 19.5 \end{pmatrix}^{-1} \left(\begin{pmatrix} 8 \\ 0.4 \\ 142 \end{pmatrix} - \begin{pmatrix} 5.66 \\ 0.891 \\ 137 \end{pmatrix} \right) \\
&= \begin{pmatrix} 38.1 \\ 505 \end{pmatrix} + \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} 1.29 & 0.0235 & 0.0953 \\ 0.0235 & 0.100 & 0.354 \\ 0.0953 & 0.354 & 19.5 \end{pmatrix}^{-1} \begin{pmatrix} 2.34 \\ -0.491 \\ 5 \end{pmatrix} \\
&= \begin{pmatrix} 38.1 \\ 505 \end{pmatrix} + \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} 0.77853 & -0.18112 & -0.00051669 \\ -0.18112 & 10.7289 & -0.19389 \\ -0.00051669 & -0.19389 & 0.054804 \end{pmatrix} \begin{pmatrix} 2.34 \\ -0.491 \\ 5 \end{pmatrix} \\
&= \begin{pmatrix} 38.1 \\ 505 \end{pmatrix} + \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} 1.9081 \\ -6.6612 \\ 0.36801 \end{pmatrix} \\
&= \begin{pmatrix} 38.1 \\ 505 \end{pmatrix} + \begin{pmatrix} -1.8126 \\ -1.7319 \end{pmatrix} \\
&= \begin{pmatrix} 36.287 \\ 503.27 \end{pmatrix}
\end{aligned}$$

Similarly we can compute the covariance matrix using a different result from the lecture notes:

$$\begin{aligned}
\Sigma_{A|B} &= \hat{\Sigma}_{hAA} - \hat{\Sigma}_{hAB}\hat{\Sigma}_{hBB}^{-1}\hat{\Sigma}_{hAB}^{T} \\
&= \begin{pmatrix} 140 & 77.8 \\ 77.8 & 8757 \end{pmatrix} - \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} 1.29 & 0.0235 & 0.0953 \\ 0.0235 & 0.100 & 0.354 \\ 0.0953 & 0.354 & 19.5 \end{pmatrix}^{-1} \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix}^{T} \\
&= \begin{pmatrix} 140 & 77.8 \\ 77.8 & 8757 \end{pmatrix} - \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} 0.77853 & -0.18112 & -0.00051669 \\ -0.18112 & 10.7289 & -0.19389 \\ -0.00051669 & -0.19389 & 0.054804 \end{pmatrix} \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix}^{T} \\
&= \begin{pmatrix} 140 & 77.8 \\ 77.8 & 8757 \end{pmatrix} - \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} 0.77853 & -0.18112 & -0.00051669 \\ -0.18112 & 10.7289 & -0.19389 \\ -0.00051669 & -0.19389 & 0.054804 \end{pmatrix} \begin{pmatrix} -0.928 & 1.01 & \\ 0.514 & 1.829.19 & 23.0 \end{pmatrix} \\
&= \begin{pmatrix} 140 & 77.8 \\ 77.8 & 8757 \end{pmatrix} - \begin{pmatrix} -0.928 & 0.514 & 9.19 \\ 1.01 & 1.82 & 23.0 \end{pmatrix} \begin{pmatrix} -0.82032 & 0.44478 & \\ 3.9009 & 14.8840.40447 & 0.90711 \end{pmatrix} \\
&= \begin{pmatrix} 140 & 77.8 \\ 77.8 & 8757 \end{pmatrix} - \begin{pmatrix} 6.4835 & 15.574 \\ 15.574 & 48.402 \end{pmatrix} \\
&= \begin{pmatrix} 133.52 & 62.226 \\ 62.226 & 8708.6 \end{pmatrix}
\end{aligned}$$

Thus, by substituting these results into the first equation, we have:

$$X_A|(X_b = x_b) \sim N\left(\begin{pmatrix} 36.287 \\ 503.27 \end{pmatrix}, \begin{pmatrix} 133.52 & 62.226 \\ 62.226 & 8708.6 \end{pmatrix} \right)$$

These same steps can be completed in python, yielding the same results.

Code Snippet 9: Calculation of conditional distribution

```
See "question_1_b.ipynb"
```

Listing 9: Calculation of conditional distribution

We can compute the partial correlation matrix R via the following formula where R_{ij} represents the value in the ith row and jth column of $\tilde{R}$ (i.e. the correlation between variables i and j) and similarly Σ_{ij} represents the value in the ith row and jth column of Σ (i.e. the covariance between variables i and j).

$$R_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}$$

For example we have:

$$R_{12} = \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} = \frac{62.226}{\sqrt{133.52 \times 8708.6}} = 0.057706$$

The above steps can be completed for each value in the correlation matrix. As before this can be done with python as follows.

Code Snippet 10: Calculation of partial correlation matrix

```
See "question_1_b.ipynb"
```

Listing 10: Calculation of partial correlation matrix

Thus yielding the partial correlation matrix of:

$$R = \begin{pmatrix} 1 & 0.057706 \\ 0.057706 & 1 \end{pmatrix}$$

We can predict the ejection_fraction and platelet count values for this patient by looking at the expected values of the conditional distribution. Since the expected values of a normal distribution is given by the mean vector we have that the predicted ejection_fraction is 36.287% and the predicted sqrtplat is 503.27 (giving a predicted platelet count of $503.27^2 = 253280$ (thousand platelets/mL of blood)).

There are many ways to consider the unusualness of these values by as it is not specified we will look at the observations in a board general sense. We can see that the correlation between these two variables is very low so we can approximately consider them as independently normally distributed from each other. Looking at them under that lens we can see that the standard distribution of ejection_fraction is $\sqrt{140} = 11.832$ and the standard distribution of sqrtplat is $\sqrt{8757} = 93.579$. Comparing the predicted values of these variables, 36.287 and 503.27, to the original distribution means of 38.1 and 505, we can see that the data for this patient, in terms of this pair of values is well within one standard deviation of the mean for both values. This indicates that these observed values are well within the expected overall patient distribution.

Question 1. (c)

For this question we will assume that the "contours of the pdf" mentioned refer to the pdf defined by the normal distribution in "Given Information 1".

The following python code is used to produce a plot of the (joint) marginal distribution of ejection_fraction and sqrtplat. The plot includes all the sample observations (generated from part 1.a) as well as three "visually well-chosen" contours of the pdf of this distribution (normal distribution in "Given Information 1").

Code Snippet 11: Generate marginal distribution of ejection_fraction and sqrtplat

```
See "question_1_c.ipynb"
```

Listing 11: Generate marginal distribution of ejection_fraction and sqrtplat

The above code generated the figure that can be seen bellow (Figure 1). It should be noted that the contour intervals were chosen such that the pdf was best "visually" represented as per request.

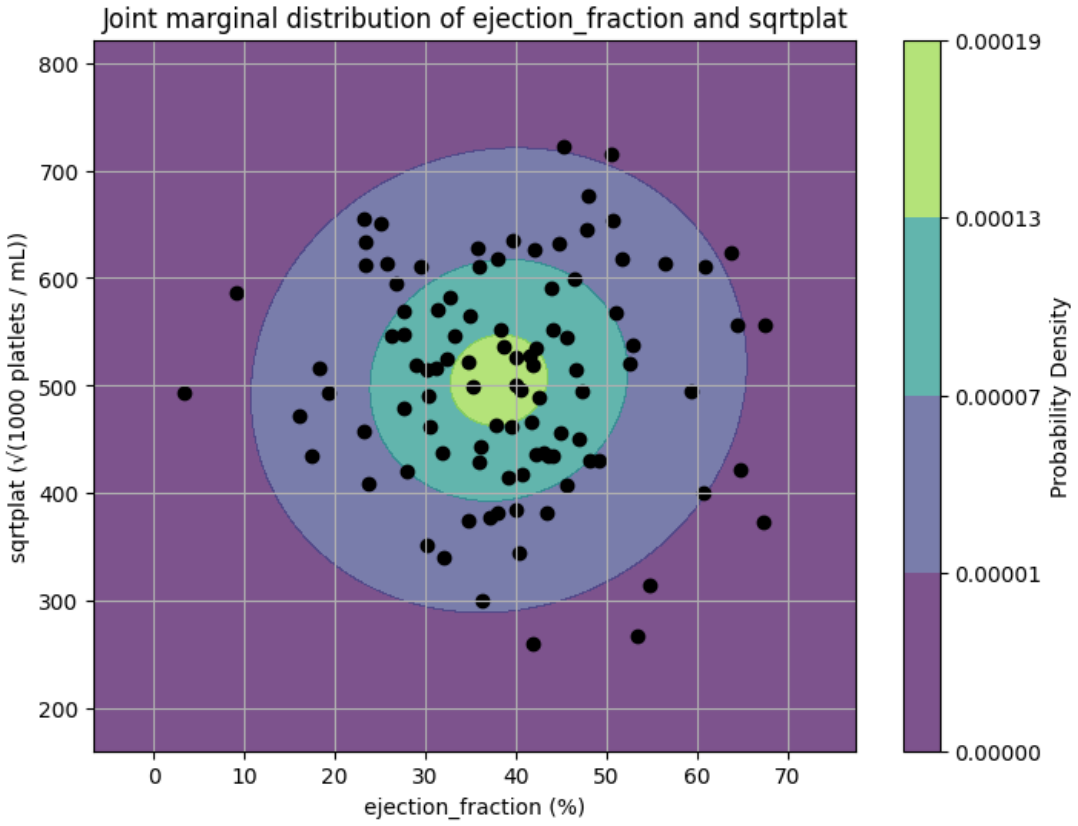

Figure 1: Joint marginal distribution of ejection_fraction and sqrtplat observations with contours based off the Multivariate Normal Distribution that generated the data

We next wish to produce a plot of the conditional distribution of ejection_fraction and sqrtplat given the other variables are fixed to the values given in question 1. b). Instead of using the normal distribution in "Given Information 1" to generate the contours, the conditional distribution from question 1. b) will be used.

Instead of plotting all the observations, as was done in the previous figure, we will just plot the 10 closest points to the hyperplane described by the given observation. To determine the "closeness" of a point we will compute the L2 distance between the hyperplane (i.e. plane produced when we fix variables 0, 3 and 4 to the given values in the observation in question 1. c)) and each observation i.e. if we consider the given observation from question 1. b) as x and an observation generated from question 1. a) as y the "closeness" can be considered as:

$$\sqrt{(x_0 - y_0)^2 + (x_3 - y_3)^2 + (x_4 - y_4)^2}$$

Similar to before we can see that the following code will produce the desired figure.

Code Snippet 12: Generate conditional distribution of ejection_fraction and sqrtplat

```
See "question_1_c.ipynb"
```

Listing 12: Generate conditional distribution of ejection_fraction and sqrtplat

The above code generated the figure that can be seen bellow (Figure 2).

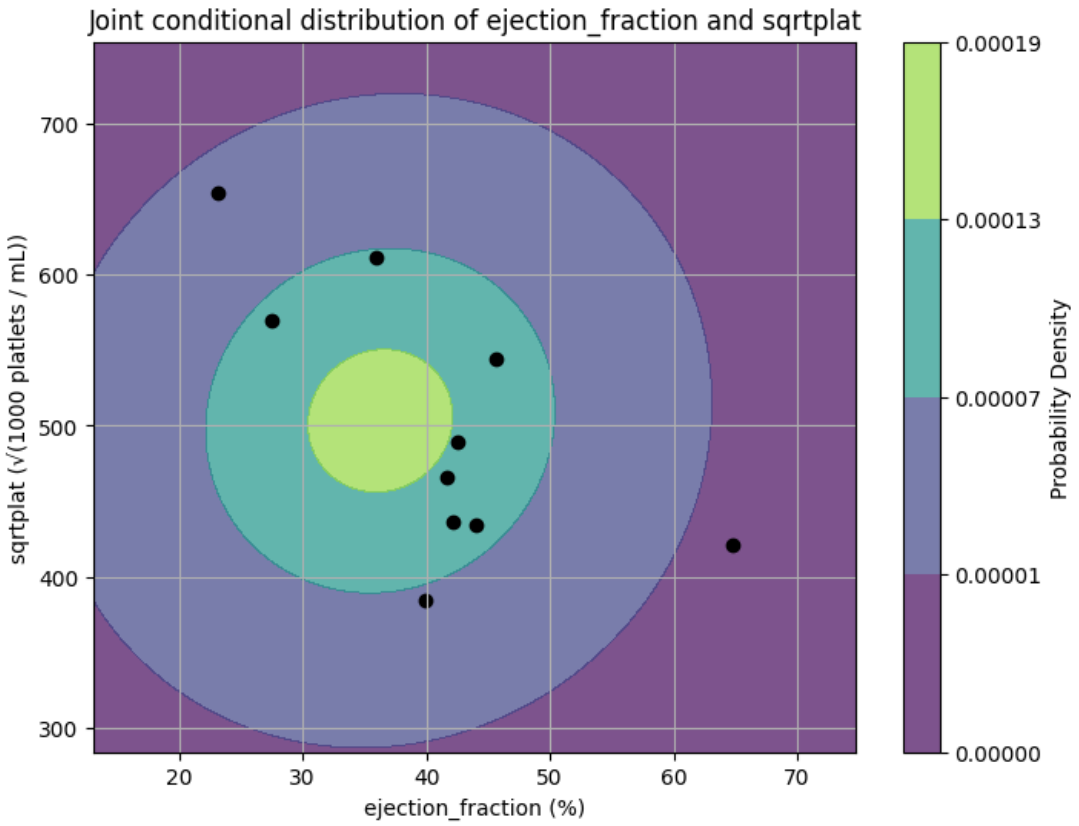

Figure 2: Conditional distribution of ejection_fraction and sqrtplat observations with contours based off the Conditional Multivariate Normal Distribution based on the given values for the observation

The reason why only observations close to the hyperplane were displayed on the above figure is because observations close to the hyperplane represent hypnotically "similar" patients to the observed patient

as they have similar recorded values for the three variables given. Thus, these observations will give a good idea of the distribution of the other two variables when the given three variables are around the given values. If we included points furthest from the hyperplane they would not be good at representing the "conditional distribution" we are trying to show, as the points do not match the condition (i.e. do not have values near the given observed values for the three given variables).

Question 1. (d)

The residual vectors for the two variables (ejection_fraction and sqrtplat) can be calculated by determining the difference between the mean vector of the conditional distribution and each observation vector. i.e. if we consider the conditional mean vector from question 1. b) as μ and an observation generated from question 1. a) as y the residual can be considered as:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \mu$$

We can calculate the residuals for each observation and plot them on a 2D plot using python as follows.

Code Snippet 13: Plot residual vectors for ejection_fraction and sqrtplat

```
See "question_1_d.ipynb"
```

Listing 13: Plot residual vectors for ejection_fraction and sqrtplat

The above code generated the figure that can be seen bellow (Figure 3).

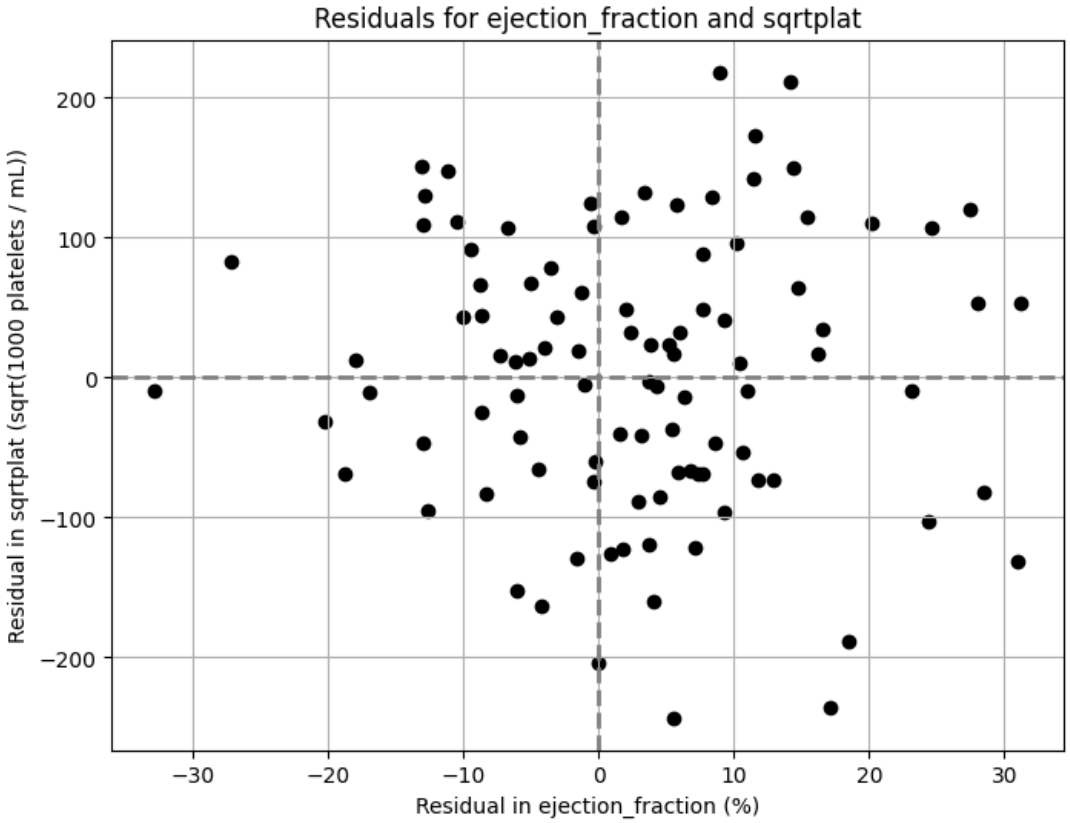

Figure 3: Plot of residual vectors for ejection_fraction and sqrtplat for each of the 100 observations when compared to the conditional distribution.

By plotting residuals for each of the dimensions (ejection_fraction and sqrtplat) individually on histograms we can see that the distribution for the residuals of both variables is approximately normal (See Figure 4).

We can fit normal distributions to the residuals using maximum likelihood estimation. We know from the lecture notes that the maximum likelihood estimates for the parameters of the normal distribution are given as follows.

The MLE for the mean μ is the sample mean:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The MLE for the variance σ^2 is the sample variance:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

Through the use of these formulas and python we can compute maximum likelihood estimates for the normal distribution parameters that best fit the residuals for ejection_fraction $(\hat{\mu}_e, \hat{\sigma}_e)$ and sqrtplat $(\hat{\mu}_s, \hat{\sigma}_s)$.

$$\hat{\mu}_e = 2.8405, \hat{\sigma}_e = 12.220$$
$$\hat{\mu}_s = 3.3787, \hat{\sigma}_s = 98.122$$

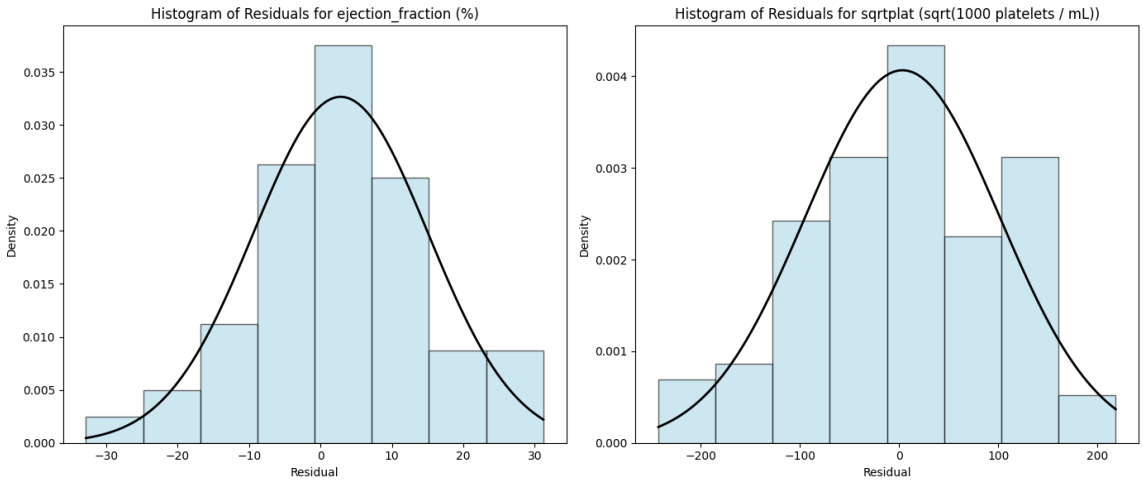

Figure 4: Histograms of residuals for ejection_fraction and sqrtplat for each of the 100 observations when compared to the conditional distribution. A normal distribution is fitted to each histogram. See above for parameters of the fitted distributions.

Given the marginal distributions of both variables are normal, and from Figure 3 we can see that the variables seem to be independent we can make the assumption that we can model the joint distribution as a multivariate normal distribution. The reasoning is provided in the Lecture notes. Thus, following a similar process as before, we can generate maximum likelihood estimates for the parameters. However the maximum likelihood estimates for the parameters of the multivariate Normal distribution are given by:

The MLE for the mean vector μ is the sample mean vector:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The MLE for the covariance matrix Σ is the sample covariance matrix:

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^\top$$

Python gives us the parameter estimates of:

$$\hat{\mu} = \begin{pmatrix} 2.8405 \\ 3.3787 \end{pmatrix}, \hat{\Sigma} = \begin{pmatrix} 150.84 & -38.346 \\ -38.346 & 9725.3 \end{pmatrix}$$

We can notice that these estimates give non-zero means. If we were calculating the residuals compared to the generating distribution we would expect the means to be 0 as the as the generating distribution is symmetric (multivariate normal). The non-zero, positive means show that the conditional distribution and the generating distribution are clearly not centered at the same point. And shows that the condition distribution, based on the given observation is skewed towards the negative negative quadrant for these two variables as both means in the mean vector for the residuals are positive.

We can also notice that the covariance matrix in the residuals shows a negative covariance between the two variables. this indicates that as the residual in one variables increases the other decreases, and vice versa. However the covariance is small in magnitude when compared to the variances of the variables, thus the correlation is small.

Question 1. (e)

We know from the lecture notes (result 1.69) that A $100(1-\alpha)\%$ confidence region consists of all vectors μ for which:

$$n(\bar{x}-\mu)^T S^{-1}(\bar{x}-\mu) \leq \frac{(n-1)p}{n-p} F_{\alpha;p,n-p}$$

This region has an ellipsoidal boundary and is centered at $\bar{x}$.

Substituting in the known parameters we get:

$$\begin{aligned} n(\bar{x}-\mu)^T S^{-1}(\bar{x}-\mu) &\leq \frac{(n-1)p}{n-p} F_{\alpha;p,n-p} \\ 100(\bar{x}-\mu)^T S^{-1}(\bar{x}-\mu) &\leq \frac{(100-1)2}{100-2} F_{\alpha;2,100-2} \\ 100(\bar{x}-\mu)^T S^{-1}(\bar{x}-\mu) &\leq \frac{198}{99} F_{\alpha;2,98} \end{aligned}$$

We can then substitute in the know parameters form the previous part to yield the region of interest. Through the use of python we can calculate that this region represents an ellipse of width of 48.278 and height of 6.0094, with an angle of 90.229^o. The region is centered around the sample mean of $(39.127, 506.65)$ which was calculated in the previous parts. The resulting 95% confidence region can be seen on Figure 5.

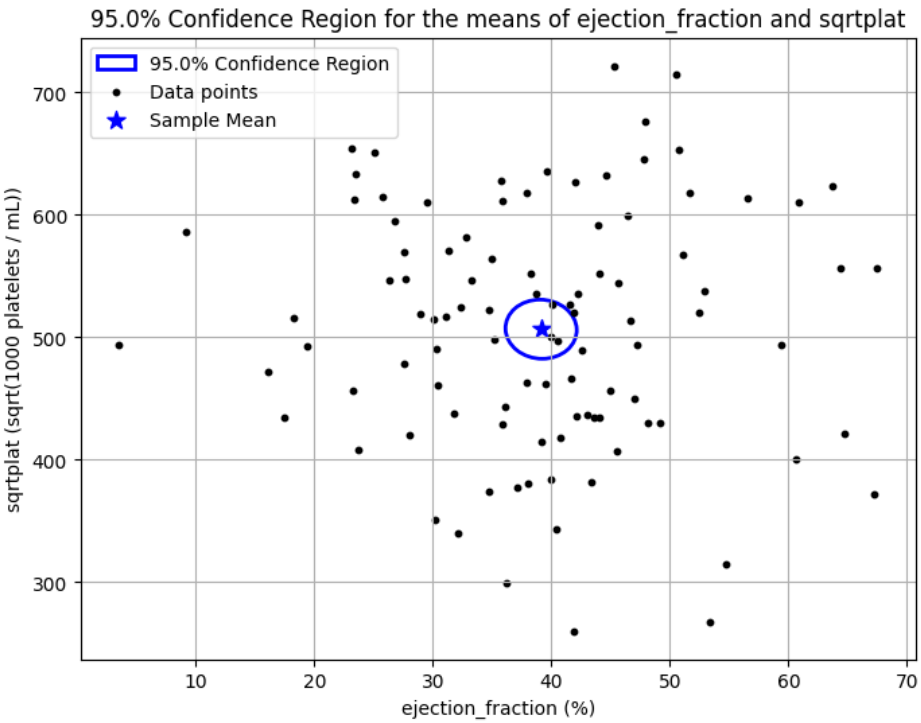

Figure 5: 95% confidence region for the means of ejection_fraction an sqrtplat based on the 100 generated observations. Observations are plotted in black on graph.

Question 2.

Question 2. (a)

The original dataset from them UCI site can be accessed and downloaded via a python package. The five required columns from the data set can then be transformed and extracted using the following code. (We are assuming that the only columns that we want to use for this question are the columns used in Question 1).

Code Snippet 14: Fetch, transform and extra desired data from UCI dataset

```
See "question_2_a.ipynb"
```

Listing 14: Fetch transform and extra desired data from UCI dataset using the ucimlrepo package for python

We can then use "scikitlearn"'s (python package) "StandardScaler" to normalise each of the variables to have a mean of 0 (this is done by subtraccting the mean of the variable from each data point), and a variance of 1 (this is done by dividing each data point by the standard deviation of the variable).

Code Snippet 15: Normalise UCI dataset

```
See "question_2_a.ipynb"
```

Listing 15: Normalise UCI dataset using use "scikitlearn"'s (python package) "StandardScaler"

We can then use "scikitlearn"'s "PCA" to calculate the principal components for both the normalised and the un-normalised datasets. (we assume we wish to have a number of PCs equal to the number of existing variables)

Code Snippet 16: Calculating PCs for both datasets

```
See "question_2_a.ipynb"
```

Listing 16: Calculating the principal components for both the normalised and the un-normalised datasets

The results of the operations above are as follows. The principle components for the "non-normalised" data are as follows:

$$\left\{ \begin{pmatrix} 0.00011 \\ 0.00903 \\ 0.99996 \\ 0.00021 \\ 0.00264 \end{pmatrix}, \begin{pmatrix} -0.00669 \\ 0.99717 \\ -0.00920 \\ 0.00373 \\ 0.07422 \end{pmatrix}, \begin{pmatrix} 0.00930 \\ -0.07423 \\ -0.00197 \\ 0.01673 \\ 0.99706 \end{pmatrix}, \begin{pmatrix} 0.99973 \\ 0.00730 \\ -0.00016 \\ 0.02009 \\ -0.00911 \end{pmatrix}, \begin{pmatrix} -0.02022 \\ -0.00263 \\ -0.00014 \\ 0.99965 \\ -0.01678 \end{pmatrix} \right\}$$

Similarly the principle components for the "normalised" data:

$$\left\{ \begin{pmatrix} 0.04381 \\ 0.48677 \\ 0.25249 \\ 0.57883 \\ 0.60195 \end{pmatrix}, \begin{pmatrix} 0.87622 \\ -0.41037 \\ -0.04939 \\ 0.24117 \\ 0.05689 \end{pmatrix}, \begin{pmatrix} 0.09024 \\ -0.04778 \\ 0.95792 \\ -0.16265 \\ -0.21332 \end{pmatrix}, \begin{pmatrix} 0.46437 \\ 0.75945 \\ -0.12267 \\ -0.36123 \\ -0.24913 \end{pmatrix}, \begin{pmatrix} 0.08090 \\ -0.12490 \\ 0.03391 \\ -0.67071 \\ 0.72584 \end{pmatrix} \right\}$$

Question 2. (b)

(We assume when the question mentions "Top Two" principle components it is meant that we preform PCA with 2 PCs) Using a similar method as above we can extract the top two principle complements and use them to plot the observations (coloured by whether or not the patient survived during the study period). This can be done for both data sets (non-normalised and normalised) using the following snippet.

Code Snippet 17: Plot both data sets using top two PCs

See "question_2_b.ipynb"

Listing 17: Ploting both data sets using top two PCs and colouring based on whether or not the patient survived during the study period

The graphs produced can be seen in Figure 6.

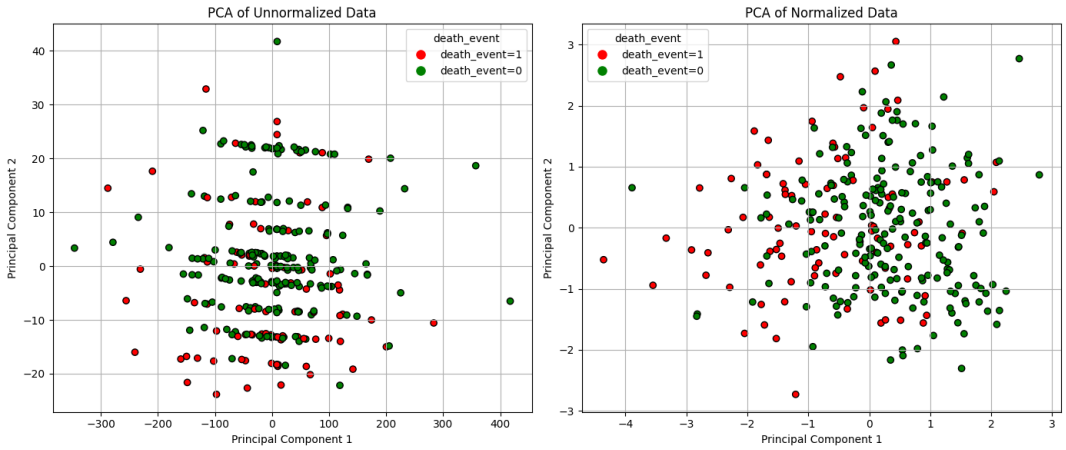

Figure 6: Plot of both data sets (normalised and non-normalised) using the top two PCs as axes and colouring based on whether or not the patient survived during the study period

We can clearly see from inspection that the normalised data is better separated by its top two principle components when compared to the normalised data. This is likely due to some of the variables having a much larger range (larger variance) than others when un-normalised and thus having a larger impact on the principle components produced (due to how principle components are formed), when in fact this variable may not be impacting the separation of data points as much.

For additional information: we can attempt to look beyond the surface level of how it looks to the human eye by fitting a simple SVM (Support vector machine) model to each data set. This model will essentially attempt to fit a decision boundary between the two groups of data aiming to split the data as best as possible.

Code Snippet 18: Fit basic SVM model to both 2 PC data sets

See "question_2_b.ipynb"

Listing 18: Fit basic SVM model to both data sets that have been fitted to 2 PCs

From this code we get the results of 58% test accuracy for the un-normalised data and 67% test accuracy for the normalised data. Higher accuracy indicates that the data can be better separated into its groups which matches our conclusion from above. While this is not conclusive it does provide supporting evidence that the normalised data can be better split into groups.

Question 2. (c)

We can produce a scree plot of the variance explained by each of the principal components for both the normalised and the un-normalised datasets using the following python code bellow.

Code Snippet 19: Produce a scree plot of the variance explained

See "question_2_c.ipynb"

Listing 19: Produce a scree plot of the variance explained by each of the principal components for both the normalised and the un-normalised datasets

The above code produced the Scree plots that can be seen in Figure

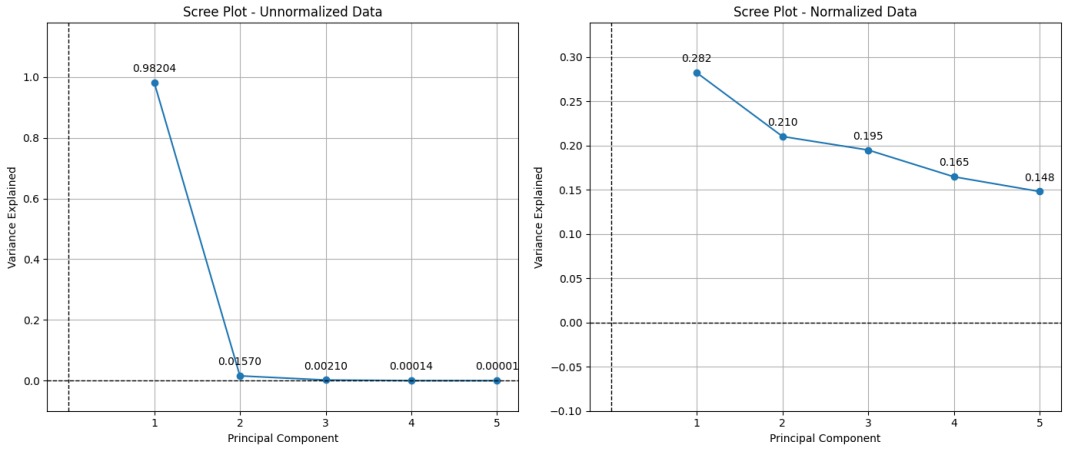

Figure 7: Scree plots of the variance explained by each of the principal components for both the normalised and the un-normalised datasets

We can see that for the Scree plot of the un-normalised data, the first principle component explains a disproportionately large amount of the variance. This is because there is one feature (namely, ejection_fraction) that has a much larger range (i.e. large variance) than the other features and therefore will contribute much more to the variance.

This can be misleading as it may not the case that this variable is actually more important than the others at explaining the data, it is just that it has a larger variance than the others. In the case of dimension reduction this could result in choosing less effective dimensions that do not explain the data as a whole as well, and are instead greatly impacted by the variable with a large range (i.e. large variance) (ejection_fraction).

However for the Scree plot of the normalised data, the variance explained by each principle component is more evenly distributed. This is because for the normalised data we have standardised each feature to have a mean of 0 and a variance of 1. This means the PCA will focus more on the structure of the data (i.e. which variables are the most impactful to the data) rather than the scale of the data (i.e. instead of which variables have the larger range (variance)).

So, in summary the data should be normalised before preforming dimensional reduction in order to produce more meaningful principle components that better represent the data structure and not just focus on the variables with the largest variance.

Question 2. (d)

The Mahalanobis distance is a measure of the distance between a point and the (sample) mean of the dataset, it is a multivariate generalization of the square of the z-score (used to determine distance from mean in a normal distribution). By its definition, it can be calculated for a given data point via the following formula:

$$D_M(x_i) = \sqrt{(x_i - \hat{\mu})^\top \hat{\Sigma}^{-1} (x_i - \hat{\mu})}$$

Where x_i is the data point we are computing, $\hat{\mu}$ is the sample mean vector and $\hat{\Sigma}$ is the sample covariance matrix. We can be assured that this will work as our covariance matrix is positive semi-definite and thus the inverse is also positive semi definite, this means that the square roots will always be defined.

We can compute this distance for an example data point as follows (note, that we know from q1.a) the definitions for the sample mean and variance):

$$\begin{aligned}
D_M \begin{pmatrix} 7.0992 \\ 35.000 \\ 513.18 \\ 0.55556 \\ 113.00 \end{pmatrix} &= \sqrt{\left(\begin{pmatrix} 7.0992 \\ 35.000 \\ 513.18 \\ 0.55556 \\ 113.00 \end{pmatrix} - \hat{\mu} \right)^\top \hat{\Sigma}^{-1} \left(\begin{pmatrix} 7.0992 \\ 35.000 \\ 513.18 \\ 0.55556 \\ 113.00 \end{pmatrix} - \hat{\mu} \right)} \\
&= \sqrt{\begin{pmatrix} 1.4393 \\ -3.0836 \\ 8.5754 \\ -0.33546 \\ -23.625 \end{pmatrix}^\top \begin{pmatrix} 0.7855 & 0.0062 & -0.0001 & -0.20328 & -0.00295 \\ 0.0062 & 0.00751 & -0.00005 & -0.02839 & -0.00299 \\ -0.0001 & -0.00005 & 0.00012 & -0.0015 & -0.00008 \\ -0.20328 & -0.02839 & -0.0015 & 10.82453 & -0.18091 \\ -0.00295 & -0.00299 & -0.00008 & -0.18091 & 0.05618 \end{pmatrix} \begin{pmatrix} 1.4393 \\ -3.0836 \\ 8.5754 \\ -0.33546 \\ -23.625 \end{pmatrix}} \\
&= \sqrt{31.305} \\
&= 5.5951
\end{aligned}$$

The following code will transform the un-normalised data using the Mahalanobis transformation and create a transformed dataset. We can then use the data set to determine the point in the original dataset had the largest Mahalanobis distance from the sample mean.

Code Snippet 20: Produce Mahalanobis distance transformed dataset

```
See "question_2_d.ipynb"
```

Listing 20: Transforms the un-normalised data using the Mahalanobis transformation and creates a transformed dataset. Using this we will determine the point with the largest Mahalanobis distance from the sample mean

The results of the code show that the point from the original dataset with the largest Mahalanobis distance is the bellow point, with a Mahalanobis distance of 5.5951

$$\begin{pmatrix} 7.0992 \\ 35.000 \\ 513.18 \\ 0.55556 \\ 113.00 \end{pmatrix}$$

For additional insight, we can plot the data (with the data point above highlighted) with two of the variables to observe how this data point is positioned relative to the rest of the data. For an example we have chosen "ejection_fraction" and "serum_sodium" as the two features to graph for Figure

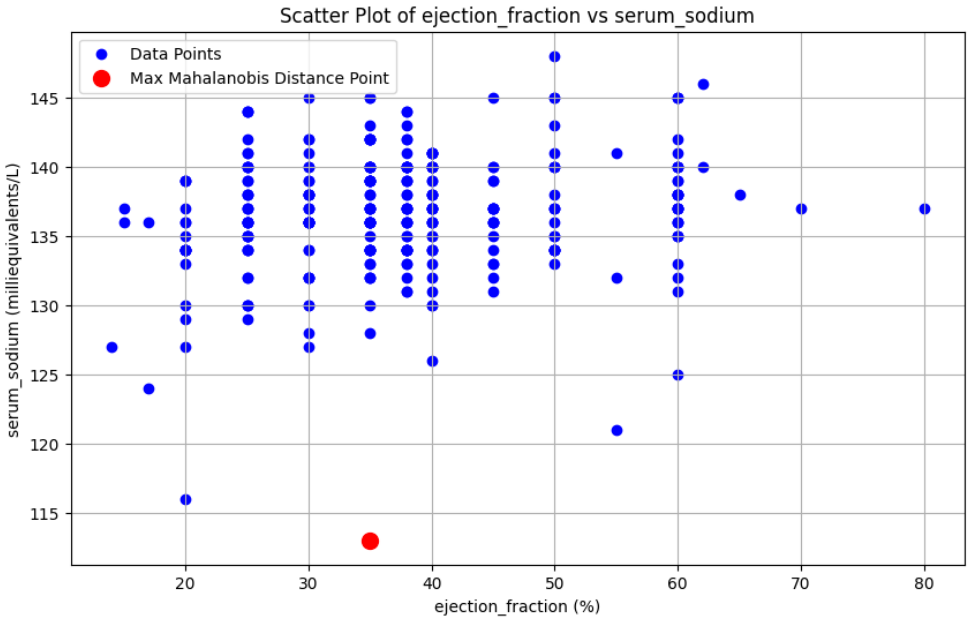

Figure 8: Plot of full un normalised data set for the features "ejection_fraction" and "serum_sodium" with the data point with the max Mahalanobis Distance highlighted

In terms of if this point should be considered an outlier, typically a "large" Mahalanobis distance indicates that the point is an outlier, however how to determine what "large" is can be done in several ways. Since our data is normally distributed it is typical to compare the Mahalanobis distance squared with a chi-squared distribution (with degrees of freedom equal to the number of dimensions). For a confidence level of 99.99% we can expect that outliers will have a squared Mahalanobis distance in excess of 25.745. Since the max distance of $5.5951^2 = 31.305$ is above this critical value, we can say that for a confidence level of 99.99% this point is considered an outlier. This is a reasonable confidence level to be set, however this can be modified based on what a person would consider an "outlier" to see if this point is considered on. See the bellow python snippet for the calculations

Code Snippet 21: Compare chi-squared distribution to Mahalanobis distance

```
See "question_2_d.ipynb"
```

Listing 21: Compare chi-squared distribution to Mahalanobis distance to determine if point is considered an outlier

Question 3.

Using python (taking advantage of the rpy2 package to interface with R) it is possible to preform Hotelling's T^2 test on all three datasets (Normalised, Un-normalised, and Normalised + PCA) to compare the two groups (death_event=1 and death_event=0). We will normalise and preform PCA on the data as seen before to generate the Normalised and Normalised + PCA data sets. The code and the results from these tests are as follows. (Note, that for this question we are assuming we preform PCA with 5 PCs as in the previous question).

Code Snippet 22: Preform Hotelling's T^2 test on all three datasets

```
See "question_3.ipynb"
```

Listing 22: Hotelling's T^2 test preformed on the Normalised Un-normalised and Normalised + PCA data

Dataset	Test stat	Numerator df	Denominator df	P-value
Normalised	68.488	5	293	7.19×10^{-12}
Un-normalised	68.488	5	293	7.19×10^{-12}
Normalised + PCA	68.488	5	293	7.19×10^{-12}

Table 1: Results from Hotelling's T^2 test preformed on the Normalised, Un-normalised, and Normalised + PCA data. These results were generated using the code snippet mentioned above

Further more, we can also preform t-tests for each variable separately for each of the three data sets. The code and the results from these tests are as follows.

Code Snippet 23: Preform t-tests for each variable on all three datasets

```
See "question_3.ipynb"
```

Listing 23: t-tests for each variable preformed on the Normalised Un-normalised and Normalised + PCA data

Variable	**t-stat**	**df**	**P-value**
logcp	-0.55058	184.19	0.58260
ejection_fraction	4.5670	164.76	9.647×10^{-6}
sqrtplat	0.99282	171.15	0.3222
recipsc	6.5259	175.61	6.987×10^{-10}
serum_sodium	3.1645	154.01	0.001872

Table 2: Results from t-tests for each variable separately preformed on the **Normalised** dataset. These results were generated using the code snippet mentioned above

Variable	**t-stat**	**df**	**P-value**
logcp	-0.55058	184.19	0.58260
ejection_fraction	4.5670	164.76	9.647×10^{-6}
sqrtplat	0.99282	171.15	0.3222
recipsc	6.5259	175.61	6.987×10^{-10}
serum_sodium	3.1645	154.01	0.001872

Table 3: Results from t-tests for each variable separately preformed on the **Un-normalised** dataset. These results were generated using the code snippet mentioned above

Variable (Principle Component)	**t-stat**	**df**	**P-value**
PC1	6.985	159.78	7.259×10^{-11}
PC2	-0.71052	178.8	0.4783
PC3	-1.0101	175.7	0.3138
PC4	0.045165	165	0.964
PC5	-2.6338	169.41	0.009226

Table 4: Results from t-tests for each variable separately preformed on the **Normalised + PCA** dataset. These results were generated using the code snippet mentioned above

We can see from the above tests that the multivariate Hotelling's T^2 shows that for all three data sets the difference between the groups is significant (At a significance level of 0.05) indicating there is an underling difference between the two groups.

For the single variable t-tests, for the normalised and un-normalised data sets, ejection_fraction, recipsc, and serum_sodium are considered to have a significant difference between groups (At a significance level of 0.05). Where as for the normalised + PCA data set, PC1 and PC5 are considered to have a significant difference between groups (At a significance level of 0.05).

These tests indicate a singificant difference between the two groups and indicate that the underlying generating distributions are different.

When looking at Hotelling's T^2 we can see that all three data sets produce identical test results. This

indicates that the test is scale invariant and is not effected by the normalisation. This matches the theory that also states that this test should be scale invariant. This is because this test is based off the Mahalanobis distances which will normalise all variables.

We can also note that PCA on the normalised data does not effect the test results. This is because we are conducting PCA with an equal number of principle components to original variables. Thus the data contains all the original information (and will produce the same statistic) just with a rotated coordinate system.

In terms of single variable t-test we can see that the results are not identically across all three data sets. This is in constrast to the multi variable Hotelling's T^2 test discussed previously that had the same results for all three data sets.

We can see that, similar to Hotelling's T^2 test, the normalised and un-normalised data have the same results for each varaible of the t-tests. This is becuase, like the previous test, the t-test is scale invariant, meaning the normalisation before the test will have no effect.

However, unlike before, the results for the Normalisation + PCA data set differs from the other two data sets. This is because the variables investigated in the t-tests are now the principle components. And thus the data has been transformed.

For these three data sets, there is no real difference between the normalised and un-normalised data when it comes to the test results. However in general the advantages to the normalised data is that future tests that may be effected by the variance of variables may benefit from having the data normalised. An advantage of the un-nromalised data is that the data - and potentially some tests become less interpretable as the data points lose their original units and it can be harder to interprete the real world meaning behind the results.

In terms of the PCA data set one advantage is that PCA could be used for dimensionality reduction so that the data can be more easily expressed via graphs and tables or could reduce the computation time of some tests or models. However one disadvantage is the major loss of interoperability as the variables do not represent the original real world measurements at all. For example in the t-tests we can see that PC1 is significantly different between the two groups however it is hard to relate this back to any real world variable that we can interprete.

So in conclusion PCA may hinder such testing due to its effect on interpretability making the results of the test hard to understand and relate back to the real world problem space. Normalisation has its advantages but also disadvantages as described above.