COMP4030-Cwk-20128825

Name: Wei Chuen Sea

Table of Contents

1 Analysis and Preprocessing

1.1.i

	min	max	mean	median	q2	q3	na
objid	1.24E+18	1.24E+18	1.24E+18	1.24E+18	1.24E+18	1.24E+18	0
dia	27.54043	848171.8	2001.357	348.9469	229.2518	693.5872	6646
rerun	301	301	301	301	301	301	30
ra	21.09996	249.9498	170.2456	170.6178	152.1987	206.1278	48
dec	-5.38263	64.97873	1.485673	0.0548	-0.66043	0.542058	49
u	14.72825	19.59975	18.59933	18.84282	18.14064	19.25774	50
g	13.20555	19.67224	17.34179	17.48185	16.75438	17.99925	51
r	12.44427	24.80204	16.80718	16.84079	16.10533	17.5118	53
i	12.06544	24.36181	16.54853	16.5376	15.79239	17.26209	50
z	11.75742	22.82691	16.38544	16.35763	15.56186	17.14328	53
run	308	1334	815.3432	756	752	756	50
m_unt	2.30E-05	0.000351	0.000222	0.000228	0.000163	0.00027	46
native	0	1	0.493388	0	0	1	50
flux	22.9907	250.6136	171.4883	170.3775	152.9922	208.8994	50
camcol	1	6	3.441375	4	2	5	50
field	11	625	322.9806	305	227	415.5	50
specobjid	3.00E+17	8.39E+18	1.65E+18	3.90E+17	3.24E+17	3.21E+18	50
redshift	-0.00333	4.106183	0.138358	0.035119	9.66E-05	0.088071	50
plate	266	7456	1461.94	346	287.5	2852	50
mjd	51608	57093	52932.91	51985	51882	54468	50
fiberid	1	1000	336.4652	319	152	500	32

Table 1: summary of raw dr14 dataset

Based on the summary, there are missing values from all columns except the “class” and “objid” column. The number of missing values in most of the columns ranges from 30 to 53. However, when inspecting the “dia” attribute which represents the diameter of the identified object, there is a large number of missing values. The “dia” attribute contains over 6646 missing values, which exceeds half of the total number of observations in the dr14 dataset.

1.1.ii

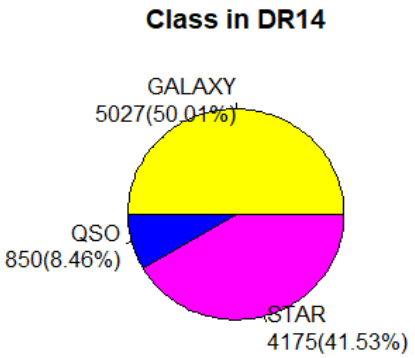

Figure 1: Pie chart of "class" attribute

Based on figure 1, the mode class is "GALAXY" followed by "QSO" and "STAR"

1.1.iii

Attributes in figure 2 have no unique values (uniform distribution)

Figure 2: Histogram of "objid" and "rerun" attributes

Attributes in figure 3 have a high positive (right) skew.

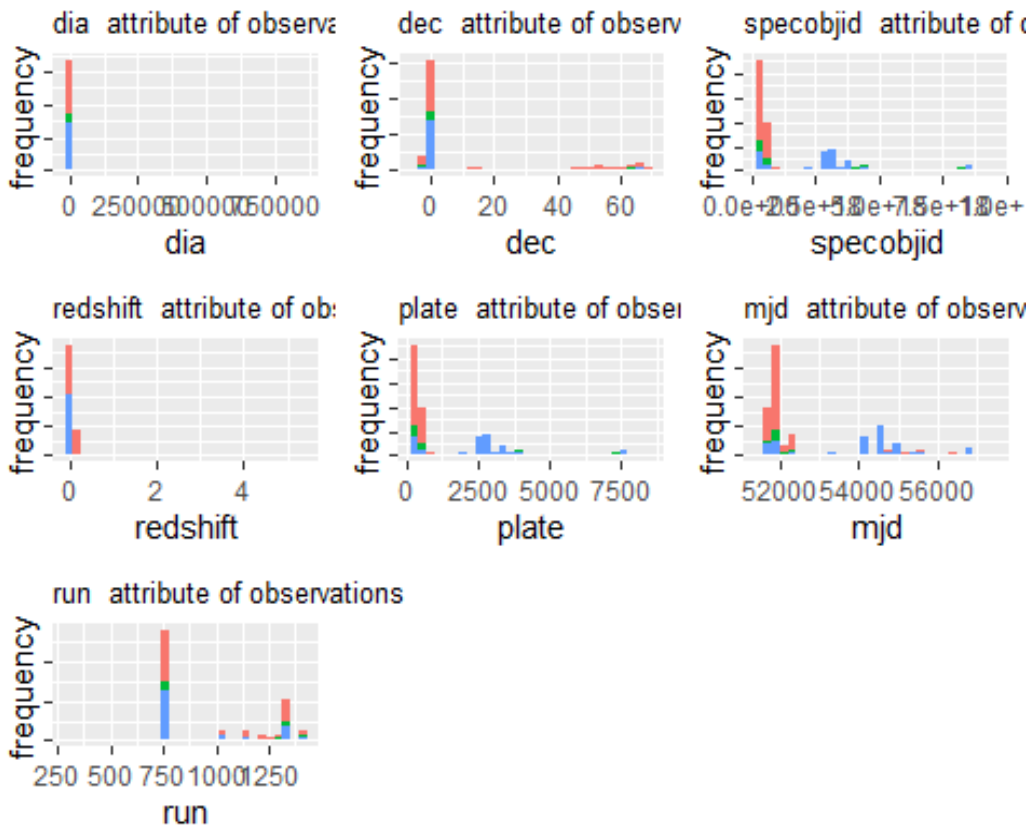

Figure 3: Histogram of "dia" , "dec", "specobjid", "redshift", "plate", "mjd", and "run" attributes

Attributes in figure 4 Uniform distribution

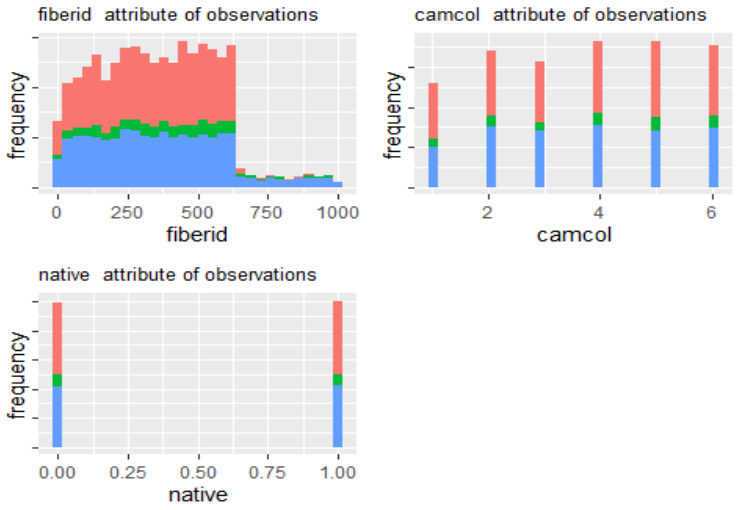

Figure 4: Histogram of "fiberid" , "camcol", and "native" attributes

The "u", "g", "ra" attributes have a negative (left) skew. Whereas the "r" and "i" attributes have positive (right) skew as shown in figure 5.

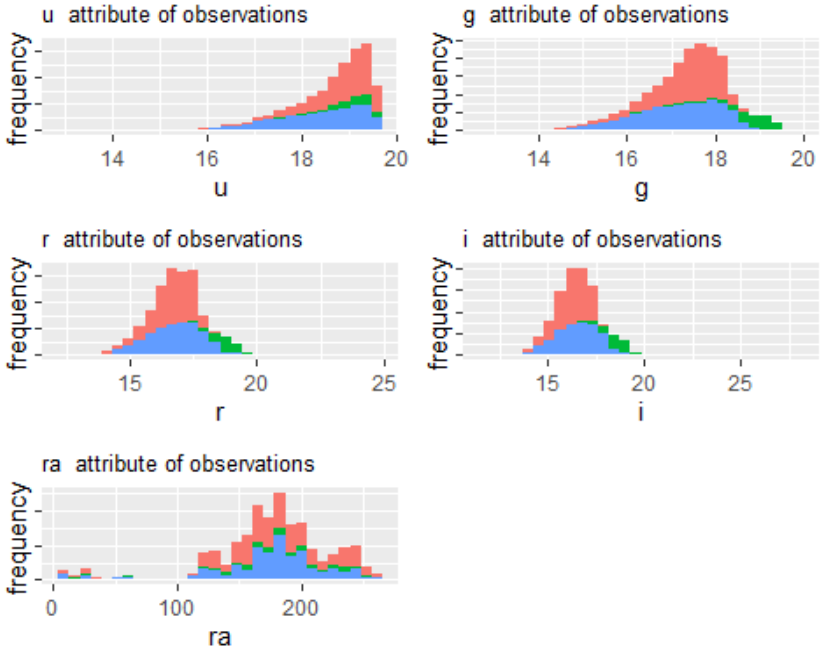

Figure 5: Histogram of "u" , "g", "r", "i", and "ra" attributes

Attributes in figure 6 have a normal distribution and leptokurtic kurtosis.

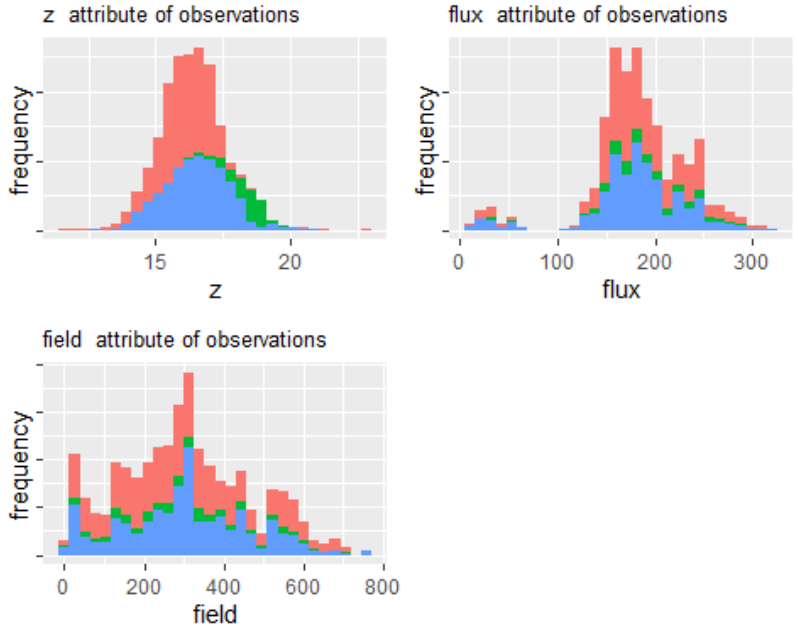

Figure 6: Histogram of "z" , "flux", and "field" attributes

The “m_unt” attribute in figure 7 has a bimodal shape which has two peaks.

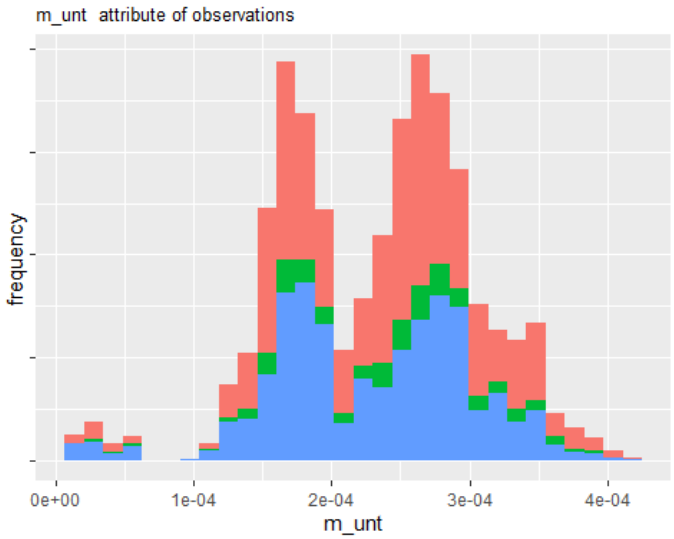

Figure 7: Histogram of “m_unt” attribute

1.2.

1.2.i

Using the “cor” function, “r” and “g” has a pearson coefficient of +0.96. Based on figure 8, “r” increases linearly as “g” increases. “r” and “g” have a perfect positive correlation.

Figure 8: Scatterplot of “r” vs “g” attribute

1.2.ii

"mjd" and "r" has a pearson coefficient of -0.034. Based on the figure 8, there is no pattern between "mjd" and "r". Therefore, "mjd" and "r" have no correlation.

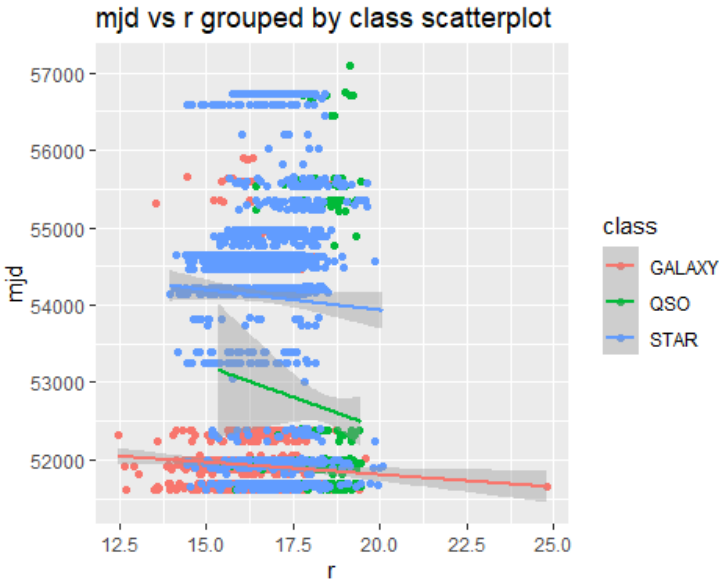

Figure 9: Scatterplot of "mjd" vs "r" attribute

1.2.iii

The 3 classes in the dr14 dataset are not separable when using the "u" attribute as shown in figure 10.

Figure 10: Scatterplot of "u" vs "class" attribute

The 3 classes in the dr14 dataset are not separable when using the “z” attribute. This is because the three classes overlap as shown in figure 11.

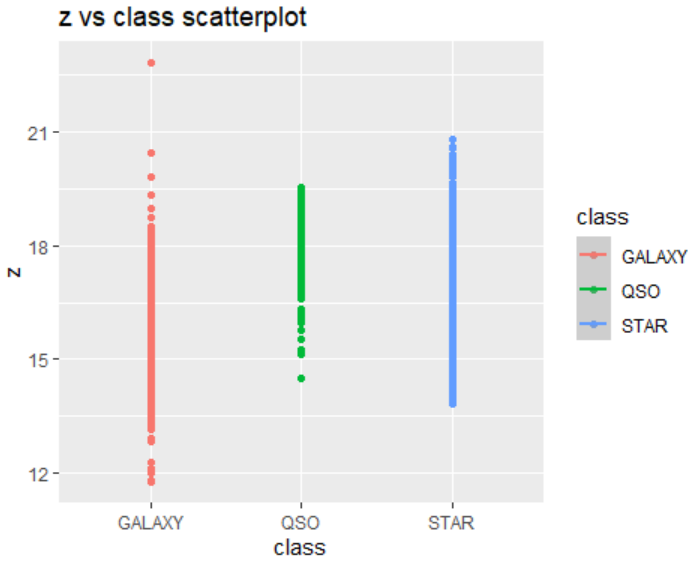

Figure 11: Scatterplot of “z” vs “class” attribute

Based on figure 12, the 3 classes in the dr14 dataset are separable when using the “redshift” attribute. This is because the three classes do not overlap at a “redshift” value. This is difficult to see in this graph because the “redshift”" values of the “STAR” class are very small and negative.

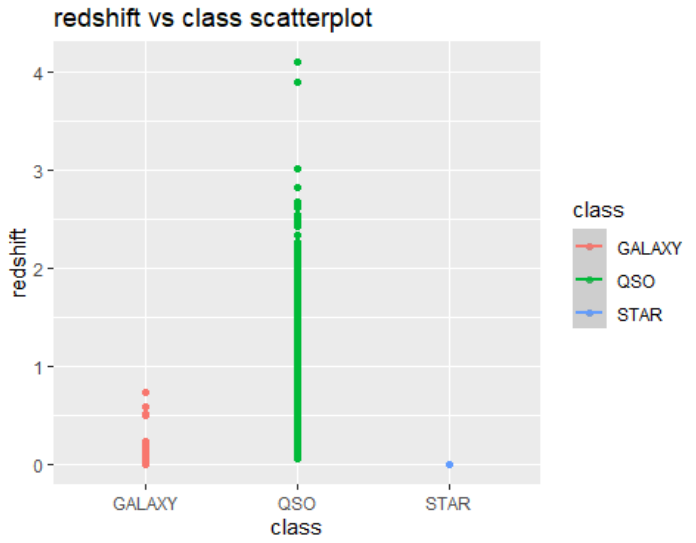

Figure 12: Scatterplot of “redshift” vs “class” attribute

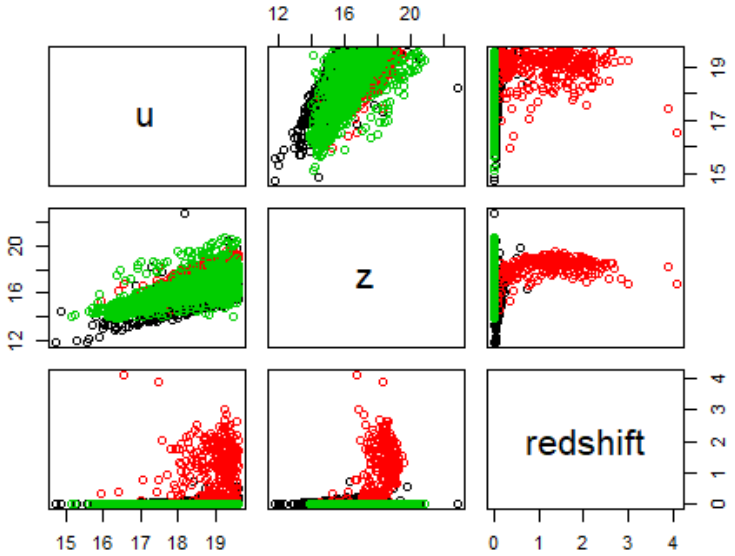

Figure 13: Scatterplot of “u” vs “z” vs “redshift” attribute. With the classes coloured.

Based on the scatter plot of the “u”, “z” and “redshift” values using the “pairs” function, it can be inferred that “u” and “z” are highly correlated. They also do not separate the three classes well, whereas the “redshift” values are not correlated with “u” or “z” and “redshift” can separate the classes well as seen in the non-overlapping colours.

1.2.iv

I have determined the appropriate attributes to be values which are continuous and do not represents object identification numbers.

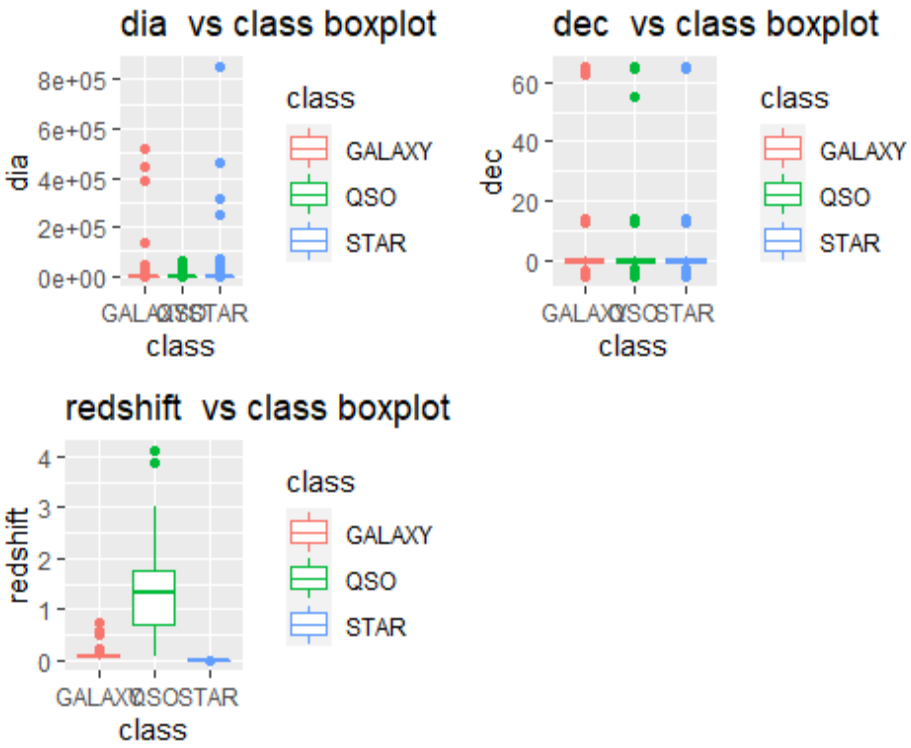

Figure 14: Boxplots of "dia", "dec", and "redshift" attribute vs "class"

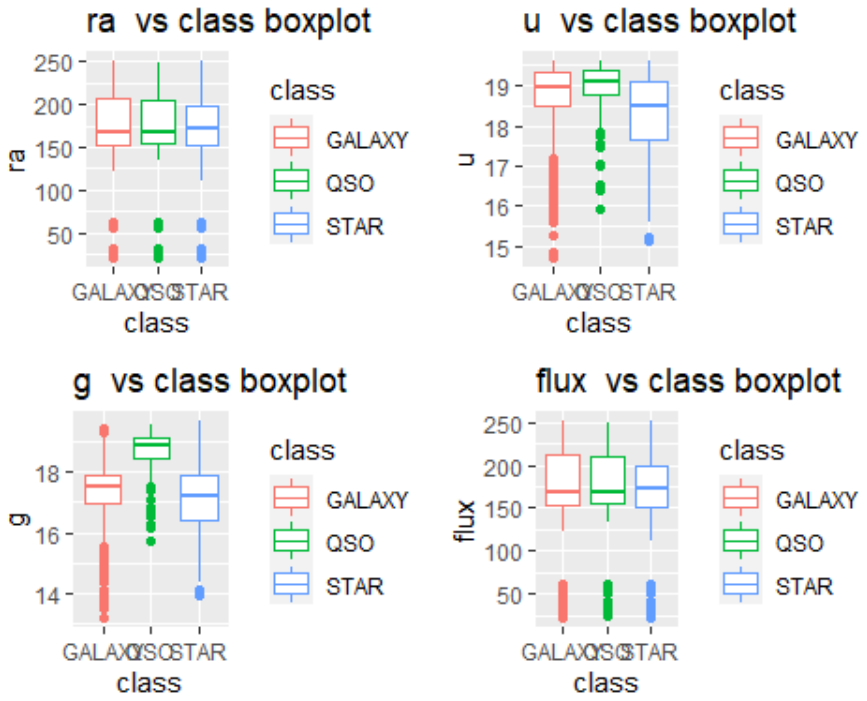

Figure 15: Boxplots of "ra", "u", "g", and "flux" attribute vs "class"

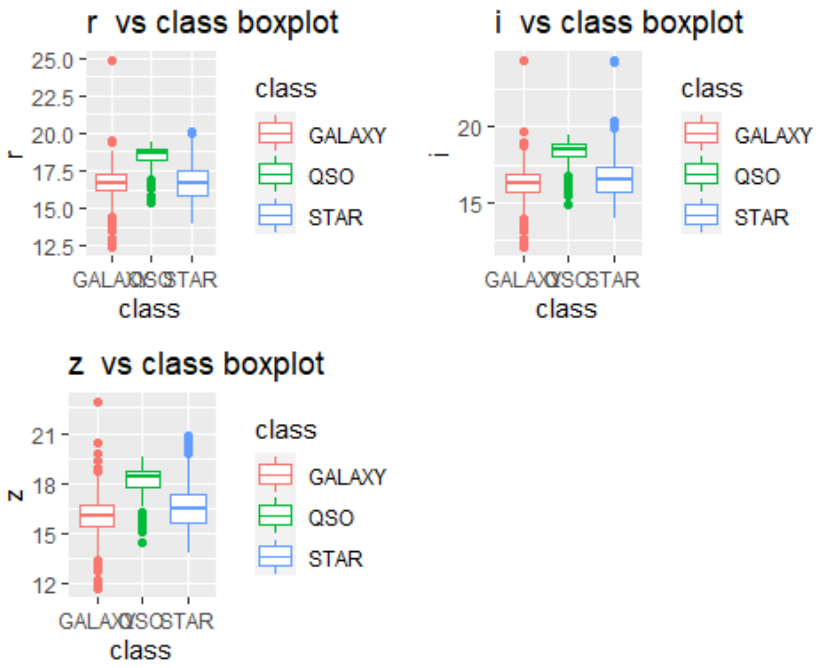

Figure 16: Boxplots of "r", "i", and "z" attribute vs "class"

1.3.

After observing the dataset and the visualisation produced, some deductions were made about the information held by particular attributes.

There are 2 attributes which only has 1 unique value which can be seen in the histograms plotted, these attributes are the "objid" and "rerun" attributes. Attributes with only 1 unique value does not hold significant information to classify or cluster the dataset into right classes.

A uniform histogram is where the histogram produces a rectangular shape. The attributes with an even distribution are the "fiberid", "camcol", and "native" attributes. The frequency of these attributes is very consistent throughout all values of "fiberid", "camcol", and "native". These attributes do not seem to hold significant information as all object classes have similar values.

Based on the scatterplots produced, there are several pairs of attributes which are highly correlated. These pairs of attributes are "u" and "z" as well as "r" and "g". Due to the high correlation between these pairs of attributes, "z" and "g" are considered redundant as they carry the same information as "u" and "r".

Based on the scatterplots, there are also some attributes which contain significant information of the dataset such as "redshift", "g", "r", "i" and "z". These attributes are able to separate at least the "QSO" class from "GALAXY" and "STAR". However, "redshift" is able to completely separate the classes.

1.4

	min	max	mean	median	q2	q3
dia	27.54043	848171.8	1989.388	1912.346	666.3243341	2245.324028
ra	8.2351	260.8844	175.5456	180.1969	157.5729357	201.4645886
dec	-5.38263	68.54227	14.82519	0.409613	-0.534460893	34.62493882
u	12.98897	19.5999	18.619	18.848	18.18093	19.2575
g	12.79955	19.91897	17.37144	17.49073	16.819	18.0066775
r	12.4316	24.80204	16.8403	16.84871	16.1783175	17.5094025
i	11.94721	28.17963	16.58295	16.55411	15.8576375	17.25441
z	11.61041	22.83306	16.42228	16.38787	15.6242825	17.1377175
run	308	1412	980.9167	756	752	1331
m_unt	1.05E-05	0.000415	0.000234	0.000244	0.000179493	0.000283961
flux	9.508854	319.0946	183.5168	183.3547	161.7081006	211.9883025
field	11	768	302.3916	300	185	413
specobjid	3.00E+17	9.47E+18	1.65E+18	4.99E+17	3.39E+17	2.88E+18
redshift	-0.00414	5.353854	0.143209	0.042689	7.96E-05	0.092182813
plate	266	8410	1461.356	443	301	2559
mjd	51578	57481	52944.03	51997	51900	54468
fiberid	1	1000	352.7195	349.5	186	510

Table 2: summary of dataframe when using mean values of class to replace na values

	min	max	mean	median	q2	q3
dia	0	848171.8	677.6854	0	0	234.426329
ra	0	260.8844	174.7066	180.1969	156.7541933	201.4645886
dec	-5.38263	68.54227	14.75447	0.393429	-0.534460893	34.62493882
u	0	19.5999	18.52641	18.848	18.162675	19.2575
g	0	19.91897	17.28371	17.4906	16.7994825	18.006235
r	0	24.80204	16.75197	16.84834	16.16289	17.50858
i	0	28.17963	16.50137	16.54832	15.843705	17.25441
z	0	22.83306	16.33647	16.38162	15.60259	17.137315
run	0	1412	976.0506	756	752	1331
m_unt	0	0.000415	0.000233	0.000244	0.000178193	0.000283961
flux	0	319.0946	182.6048	183.1608	161.5219886	211.9883025
field	0	768	300.8894	298.5	183	413
specobjid	0	9.47E+18	1.64E+18	4.97E+17	3.38E+17	2.88E+18
redshift	-0.00414	5.353854	0.14297	0.041923	7.60E-05	0.092182813
plate	0	8410	1454.595	441	300	2559
mjd	0	57481	52681.11	51997	51900	54468
fiberid	0	1000	351.6323	349	184	510

Table 3: summary of dataframe when using zero to replace na values

	min	max	mean	median	q2	q3
dia	27.54043	848171.8	908.6757	348.6454	347.3194662	348.6453718
ra	8.2351	260.8844	175.565	180.3708	157.5729357	201.4645886
dec	-5.38263	68.54227	14.75644	0.405793	-0.534460893	34.62493882
u	12.98897	19.5999	18.61989	18.85334	18.18093	19.2575
g	12.79955	19.91897	17.3721	17.49581	16.819	18.0066775
r	12.4316	24.80204	16.84072	16.84871	16.1783175	17.5094025
i	11.94721	28.17963	16.58323	16.55411	15.8576375	17.25441
z	11.61041	22.83306	16.42251	16.38787	15.6242825	17.1377175
run	308	1412	979.8111	756	752	1331
m_unt	1.05E-05	0.000415	0.000234	0.000244	0.000179493	0.000283961
flux	9.508854	319.0946	183.5124	183.2885	161.7081006	211.9883025
specobjid	3.00E+17	9.47E+18	1.65E+18	4.97E+17	3.39E+17	2.88E+18
redshift	-0.00414	5.353854	0.1432	0.042689	7.96E-05	0.092182813
plate	266	8410	1461.261	441	301	2559
mjd	51578	57481	52944.42	51997	51900	54468
fiberid	1	1000	352.7316	349.5	186	510

Table 4: summary of dataframe when using median to replace na values by class

The effects of replacing "na" values with mean, median and zero has minimal effects on the overall dispersion of the dataset. Only the "dia" attribute is affected, this is due to the large amount of "na" values in the attribute. Mean replacement has caused "dia" to have a larger interquartile range. Besides that, the minimum values of when replacing na values with zeros have become zero for most attributes.

1.5.

Mean centering is subtracting the mean from all values in the dataset. Normalisation is the rescaling of values to a range of 0 to 1. Finally, standardisation rescales the data to have a mean of 0 and a standard deviation of 1.

Comparing and contrasting the approaches, mean centering only makes the mean become 0 but does not change the scale of the data, whereas normalisation and standardisation does. Normalisation is used to make attributes with different ranges to be the same (0 to 1) and standardization is used for most machine learning algorithms.

1.6.

1.6.i.

Based on number of missing values per attribute, diameter contains too many missing values (over 50% of dataset is empty) so it was removed.

Based on number of missing values per instance, 50 instances have more than 3 missing values and will be removed. Out of the values removed, 30 belonged to the galaxy class, 20 belonged to the star class.

The effect of the data is that 50 observations have been deleted and 1 attribute has been deleted. Therefore, the size of the dataset has changed from 10052 x 22 matrix to a 10002 x 21 matrix.

1.6.ii

Based on the pearson correlation table produced by the "cor" function, there are several attributes which have a high positive correlation. The scatterplots of these variables have been plotted below.

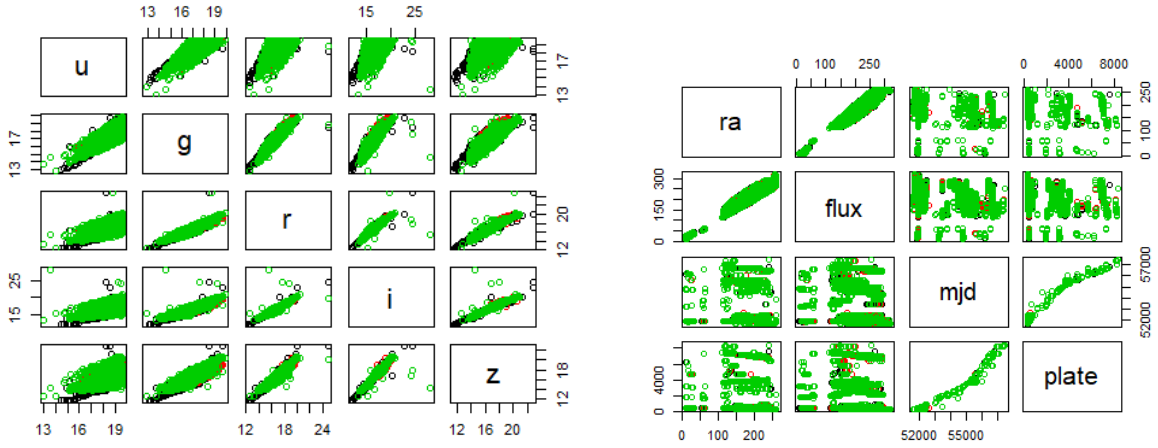

Figure 17: Scatterplot of "u", "g", "r", "i", "z", ra", "flux", "mjd", and "plate" attributes coloured by class

As shown in figure 17, the "u", "g", "r", "i", and "z" attribute have a high positive correlation with each other. Therefore, I have decided to remove all attributes except the "z" attribute.

Subsequently, "ra" and "flux" attribute have a high positive correlation. Whereas the "mjd" and "plate" attribute have a positive correlation. Therefore, I have decided to remove the "flux" and "plate" attributes as they are redundant.

The effect of attribute selection is the deletion of 6 correlated attributes.

1.7.

1.7.i

PCA as a transformation technique produces more features than PCA as a dimensionality reduction. The cumulative proportion of the full PCA features is higher than when PCA is used for dimensionality reduction.

Based on table 5, when using more than 10 principle components, the cumulative proportion increase is very small.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Standard deviation	2.15	1.8181	1.6762	1.4182	1.16529	0.99941	0.93513	0.73278	0.67859
Proportion of Variance	0.2568	0.1836	0.1561	0.1117	0.07544	0.05549	0.04858	0.02983	0.02558
Cumulative Proportion	0.2568	0.4405	0.5965	0.7083	0.78371	0.8392	0.88778	0.91761	0.9432
	PC10	PC11	PC12	PC13	PC14	PC15	PC16	PC17	PC18
Standard deviation	0.65805	0.60322	0.35932	0.20481	0.13934	0.12538	0.11623	0.07658	0.000528
Proportion of Variance	0.02406	0.02021	0.00717	0.00233	0.00108	0.00087	0.00075	0.00033	0
Cumulative Proportion	0.96725	0.98747	0.99464	0.99697	0.99805	0.99892	0.99967	1	1

Table 5: internal matrix of clustering methods

1.7.ii

8 PCs need to reach cumulative variance of 90%

2. Clustering

2.1.

I have used the dataset reduced dataset featuring 12 Principal Components.

	hca	kmeans	pam
cluster.number	3	3	3
max.diameter	15.51883338	15.68556	19.13324
average.between	6.16695938	6.198541	6.097584
average.within	5.245533726	4.887209	5.002827
dunn	0.02862233	0.013515	0.02102
min.separation	0.444185164	0.211987	0.402187
avg.silwidth	0.124183637	0.161614	0.138315

Table 6: internal matrix of clustering methods

The internal matrix above were calculated using the “cluster.stats” function from the “cluster” package. This is a short description of the internal matrix used: max.diameter = maximum cluster diameter, average.between = average distance between clusters, average.within = average distance within clusters (reweighted so that every observation, rather than every distance, has the same weight), dunn = minimum separation / maximum diameter (higher value is better), min.separation = minimum cluster separation, avg.silwidth = average silhouette width.

The hca clustering method has the lowest maximum cluster diameter (“max.dia”), the highest dunn index (“dunn”) and the largest minimum cluster separation (min.separation).

Meanwhile, kmeans clustering method has the highest average silhouette width (“avg.silwidth”) and the highest average distance between clusters (“average.between”).

Lastly, pam clustering method has the lowest average weighted distance between clusters (“average.within”).

HCA	1	3	2	accuracy	recall	precision	f1_score
GALAXY	3958	0	1039	0.70096	0.792075	0.673358	0.727908
QSO	756	62	32	0.70096	0.072941	1	0.135965
STAR	1164	0	2991	0.70096	0.719856	0.736337	0.728003

Table 7: external matrix produced by hca clustering method

kmeans	2	3	1	accuracy	recall	precision	f1_score
GALAXY	3561	1331	105	0.542592	0.712628	0.710211	0.711417
QSO	678	15	157	0.542592	0.017647	0.005217	0.008054
STAR	775	1529	1851	0.542592	0.445487	0.876006	0.590619

Table 8: external matrix produced by kmeans clustering method

pam	1	3	2	accuracy	recall	precision	f1_score
GALAXY	3267	1625	105	0.621976	0.653792	0.732183	0.690771
QSO	428	296	126	0.621976	0.348235	0.111656	0.169095
STAR	767	730	2658	0.621976	0.639711	0.920042	0.754685

Table 9: external matrix produced by pam clustering method

The confusion matrix of the hca clustering method shows that the majority of the "GALAXY" and "STAR" observations have been correctly clustered, but the majority of "QSO" observations have been wrongly clustered as "GALAXY". The hca clustering method produced the best accuracy, recall, precision and f1 score compared to kmeans and pam. The f1 score matrix was obtained by aligning the confusion matrix and calculating the accuracy, recall and precision.

2.2

For hca method, the "distance" and "method" parameters were tuned. The "distance" parameter is the distance measure used to calculate the distance matrix. Whereas the "method" parameter determines the method used to compare clusters. I have found that distance = euclidean and method = complete-linkage performs best.

The parameters tuned in the kmeans method are "centers" and "iter.max". The "centers" parameter determines the number of clusters and the "iter.max" parameter is the maximum iterations allowed to define the best cluster.

Lastly, the parameters tuned in the pam method are the "k" and "metric". These parameters are the same as the "distance" and "method" parameters in the hca method.

The best value for “distance”, “centers” and “k” parameters is 3. The parameter tuning technique used was the elbow method. This method was used to prevent over-fitting.

2.3.

The hca clustering method was chosen to perform clustering on all the following datasets.

2.3.i

i	hca
cluster.number	3
max.diameter	17.35028
average.between	6.505497
average.within	5.502381
dunn	0.030466
min.separation	0.528592
avg.silwidth	0.131445

Table 10: internal matrix of transformed dataset featuring all Principal Components

I HCA	1	3	2	accuracy	recall	precision	f1_score
GALAXY	4559	0	438	0.491602	0.912347408	0.49549	0.64220313
QSO	841	4	5	0.491602	0.004705882	1	0.00936768
STAR	3801	0	354	0.491602	0.085198556	0.444166	0.14297254

Table 11: external matrix of transformed dataset featuring all Principal Components

2.3.ii

ii	hca	kmeans	pam
cluster.number	3	3	3
max.diameter	15.51883338	15.68556	19.13324
average.between	6.16695938	6.198541	6.097584
average.within	5.245533726	4.887209	5.002827
dunn	0.02862233	0.013515	0.02102
min.separation	0.444185164	0.211987	0.402187
avg.silwidth	0.124183637	0.161614	0.138315

Table 12: internal matrix of reduced dataset featuring 12 Principal Components

Ii HCA	1	3	2	accuracy	recall	precision	f1_score
GALAXY	3958	0	1039	0.70096	0.792075	0.673358	0.727908
QSO	756	62	32	0.70096	0.072941	1	0.135965
STAR	1164	0	2991	0.70096	0.719856	0.736337	0.728003

Table 13: external matrix of reduced dataset featuring 12 Principal Components

2.3.iii

iii	hca	kmeans	pam
cluster.number	3	3	3
max.diameter	27.449579	65.82945	65.82945
average.between	40.405309	5.111639	5.142152
average.within	4.6973922	4.062792	4.124075
dunn	0.2964047	0.00359	0.003215
min.separation	8.1361838	0.236326	0.211637
avg.silwidth	0.8633214	0.16391	0.159507

Table 14: internal matrix of the dataset after deletion of instances and attributes.

iil HCA	1	2	3	accuracy	recall	precision	f1_score
GALAXY	5024	3	0	0.499901	0.999403	0.50005	0.666578
QSO	850	0	0	0.499901	0	0	NaN
STAR	4173	1	1	0.499901	0.00024	1	0.000479

Table 15: external matrix of the dataset after deletion of instances and attributes.

2.3.iv

Iv mean	hca	kmeans	pam
cluster.number	3	3	3
max.diameter	3.99E+18	3.69E+18	3.69E+18
average.between	3.89E+18	3.68E+18	3.68E+18
average.within	6.14E+17	3.02E+17	3.02E+17
dunn	0.047865	0.078591	0.102981
min.separation	1.91E+17	2.90E+17	3.80E+17
avg.silwidth	0.718397	0.893986	0.894036

Table 16: internal matrix of the mean version of the mean-centred data.

Ivhcamean	1	3	2	accuracy	recall	precision	f1_score
GALAXY	4903	29	95	0.667728	0.975333	0.640831	0.773466
QSO	693	65	92	0.667728	0.076471	0.138298	0.098485
STAR	2055	376	1744	0.667728	0.417725	0.903159	0.571241

Table 17: external matrix of the mean version of the mean-centred data.

Iv med	hca	kmeans	pam
cluster.number	3	3	3
max.diameter	3.99E+18	3.69E+18	3.69E+18
average.between	3.89E+18	3.68E+18	3.68E+18
average.within	6.06E+17	3.02E+17	3.02E+17
dunn	0.080201	0.078591	0.102981

min.separation	3.20E+17	2.90E+17	3.80E+17
avg.silwidth	0.722338	0.894244	0.894294

Table 18: internal matrix of the median version of the mean-centred data.

Ivhcamed	1	3	2	accuracy	recall	precision	f1_score
GALAXY	4903	29	95	0.669718	0.975333	0.642511	0.774688
QSO	693	65	92	0.669718	0.076471	0.138298	0.098485
STAR	2035	376	1764	0.669718	0.422515	0.904152	0.575906

Table 19: external matrix of the median version of the mean-centred data.

Iv zero	hca	kmeans	pam
cluster.number	3	3	3
max.diameter	4.37E+18	3.69E+18	3.69E+18
average.between	3.68E+18	3.69E+18	3.69E+18
average.within	3.09E+17	3.05E+17	3.04E+17
dunn	0.121281	0.078591	0.102981
min.separation	5.30E+17	2.90E+17	3.80E+17
avg.silwidth	0.891452	0.893415	0.893466

Table 20: internal matrix of the zero version of the mean-centred data.

ivhcazero	1	2	3	accuracy	recall	precision	f1_score
GALAXY	4892	50	85	0.76154	0.973145	0.735639	0.837886
QSO	692	77	81	0.76154	0.090588	0.14	0.11
STAR	1066	423	2686	0.76154	0.643353	0.941795	0.76448

Table 21: internal matrix of the zero version of the mean-centred data.

2.3.v

The best dataset for hca clustering is the mean-centred data that had missing values replaced with zeros. This dataset achieves the highest accuracy of 76%. Based on the internal matrix, it has a very high average silhouette index which indicates that a sample is very similar to its cluster and very different from other clusters.

3. Classification

3.1

The reduced dataset with 12 principal components was used. This dataset was used because the features are uncorrelated and the number of attributes is reduced, so the number of dimensions of the features are lower.

To prevent overfitting, the dataset has been split into training set (80%) and testing set (20%). I have also used evaluation metrics (accuracy, recall, precision, f1_score).

Algorithm	accuracy	TP-Rate	FP-Rate	Precision	Recall	F-Measure	ROC-Area
ZeroR	50.6468	0.506	0.506	?	0.506	?	?
OneR	75.7214	0.757	0.229	0.747	0.757	0.724	0.764
NaiveBayes	86.9652	0.87	0.118	0.873	0.87	0.868	0.954
IBK	90.0498	0.9	0.092	0.905	0.9	0.9	0.956
J48	92.8358	0.928	0.057	0.928	0.928	0.928	0.94

Table 22: Evaluation results of datasets using weka.

The best performing algorithm is the J48 classifier, it outperforms every other classifier in all the matrix except "ROC-Area".

3.2

split	k	algorithm	accuracy	precision	recall	GALAXY	QSO	STAR
						978	2	38
						27	132	10
80,20	5	linear	90.0498	0.905	0.9	121	2	700
						1643	4	76
						59	218	15
66,33	5	linear	89.3798	0.898	0.894	207	2	1194
						2423	3	126
						89	298	29
50,50	5	linear	88.3207	0.888	0.883	332	8	1718
						980	2	36
						29	127	13
80,20	7	linear	88.9055	0.895	0.889	140	3	680
						972	2	44
						24	133	12
80,20	3	linear	90.199	0.905	0.902	114	1	708
						972	2	44
						24	133	12
80,20	3	ball tree	90.199	0.905	0.902	114	1	708
						972	2	44
						24	133	12
80,20	3	kd tree	90.199	0.905	0.902	114	1	708

Table 23: Evaluation results of datasets using weka.

The "split" parameter determines the percentage of the dataset that is used for training and testing. The "k" parameter determines the number of neighbours used to determine which cluster an observation belongs to. Finally, the "algorithm" parameter is the algorithm used for spatial division of data points and their allocation into certain regions. The best parameters for clustering of the dataset used is "split" = 80:20, "k" = 3, and "algorithm" = (linear, ball tree or kd tree)

3.3

Dataset	accuracy	TP-Rate	FP-Rate	Precision	Recall	F1_score	ROC-Area
3.3.i	94.6578	0.947	0.042	0.947	0.947	0.947	0.952
3.3.ii	92.9865	0.93	0.055	0.93	0.93	0.93	0.947
3.3.iii	98.7702	0.988	0.009	0.988	0.988	0.988	0.991
3.3.iv mean	99.5324	0.995	0.003	0.995	0.995	0.995	0.997
3.3.iv median	99.1146	0.991	0.006	0.991	0.991	0.991	0.994
3.3.iv zero	98.5973	0.986	0.009	0.986	0.986	0.986	0.99

Table 24: Evaluation results of datasets using weka.

Generally, all datasets performed well when using the when using the J48 classifier. The dataset with the worst f1_score is the reduced dataset featuring 12 principle components. This may be due to the removal of principle components which may help to separate the classes better.

3.3.v

The dataset where J48 classifier performed best is the normalised dataset where na values were replaced with the class mean. This may be due to the separability of the "dia" attribute when the na values have been replaced by the class mean.

word count excluding tables = 2112

Appendix

```{r}

dr14 = read.csv(file="cw_data.csv", header=TRUE,stringsAsFactors = TRUE)

# names(dr14)

dr14omit = na.omit(dr14)

```

1.1

Looking at the raw SDSS dr14 dataset, there are 21 attributes with the last column ("class"") being the label and 10052 instances in total. The attributes can be separated into several data types: nominal, interval, ordinal and ratio. The columns can be grouped into the appropriate nominal, interval, ordinal or ratio data type based on the attribute description provided.

The columns categorized as the nominal data type are the "objid", "native", "camcol", "run", "rerun", "fiberid", "specobjid", and "class". These columns were categorized as nominal because they represent identification numbers, type of objects used to observe the instance as well as boolean values. The data is unordered and distinct therefore, they are nominal data. The column "mjd" have been categorized as ordinal as it is a date and can be ordered, but the difference would be meaningless. Subsequently, the column of an interval data type is the "redshift" column, because the data is ordered and the differences are meaningful but it doees not have a true 0. Finally, the columns with the ratio data type are the "diameter", "ra", "dec", "u", "g", "r", "i", "z", "m_unt", and "flux" columns. They are of the ratio data type because they have a true 0 and the ratios between them are meaningful.

1.1.i

```{r}

summdisp = c(min = min,max = max,mean = mean,median = median,

      q2={function(x) quantile(x, 0.25)},q3={function(x) quantile(x, 0.75)})

sum_tab = sapply(summdisp, {function(x) summarise_all(dr14omit[,1:21],x)})

na_sum = c(na = {function(x) sum(is.na(x))})
```

```
na_table = (sapply(na_sum, {function(x) summarise_all(dr14[,1:21],x)}))


sum_tab = as.data.frame(sum_tab)
sum_tab$na = na_table
```


```{r}
#sum_tab
write_xlsx(as.data.frame(sum_tab),"first_sum.xlsx")
```

```{r}


df_char = apply(sum_tab,2,as.character)
rownames(df_char) = rownames(sum_tab)
write.csv(df_char, "first_sum.csv", row.names = TRUE)
```


Based on the summary, there are missing values from all columns except the "class" and "objid" column. The number of missing values in most of the columns ranges from 30 to 53. However, when inspecting the "dia" attribute which represents the diameter of the identified object, there is a large number of missing values. The "dia" attribute contains over 6646 missing values, which exceeds half of the total number of observations in the dr14 dataset.

1.1.ii

```{r}
class_table = table(dr14$class)
pie(class_table,
```

```
  labels = paste(names(class_table), "\n",
                 class_table,
                 "(",
                 round(class_table*100/nrow(dr14), digits = 2),
                 "%)",
                 sep=""),
  col=c("yellow","blue","magenta"),
  main = "Class in DR14")


barplot(class_table, main="Class frequency",
  xlab="Class")
```

```

Based on the visualisations above, there are 5027 observations are classified as "GALAXY", 50.01% of all observations are classified as "GALAXY". This is the mode class of the dr14 dataset, followed by the "STAR" class with 4175 observations, and "QSO" with the smallest number of observations of 850. These visualisations show that the dr14 dataset provided is an imbalanced dataset. Therefore, special care has to be taken when evaluating the clustering and classification model.

```{r}
plothist = function(df, col, binNum){
  plot = ggplot(df, aes(x = df[,col], fill=class)) +
    geom_histogram(bins = 30,  na.rm = TRUE) +
    theme(legend.position = "none",
          # axis.text.x = element_blank(),
          axis.text.x = element_text(angle=0),
```
```

```
        axis.text.y = element_blank(),
        plot.title = element_text(size=10)) +
   scale_x_continuous(name = col) +
   scale_y_continuous(name = "frequency") +
   ggtitle(paste(col, " attribute of observations"))

 return(plot)
}

plotAllHist = function (df, colnames, binNum){
 plot = ggplot(df) +
        geom_histogram(binwidth = 1000000,
              aes(x = objid, fill = class),
              na.rm = TRUE) +
        theme(legend.position = "none",
            plot.title = element_text(size=10),
            axis.text.x = element_blank(),
            axis.text.y = element_blank()) +
        ggtitle("objid attribute of observations")
 list_plots = list(plot)
 for (i in 2:length(colnames)){
  if(colnames[i]=="objid"){
    plot = plothist(df, colnames[i], 1000000)
  }
  else{
    plot = plothist(df, colnames[i], binNum)
  }
```

```
   list_plots = append(list_plots,list(plot))
  }
  names(list_plots) = colnames
  return(list_plots)
}

attName = colnames(dr14[,1:ncol(dr14)-1])
list_plots = plotAllHist(dr14, attName, 30)
```

```{r}
#plot shape

#no unique values
grid.arrange(list_plots[["objid"]], list_plots[["rerun"]], nrow = 1, ncol=2)

#extremely skewed histograms (might be equivalent to no unique values)
grid.arrange(list_plots[["dia"]], list_plots[["dec"]],  list_plots[["specobjid"]],
          list_plots[["redshift"]], list_plots[["plate"]], list_plots[["mjd"]],
          list_plots[["run"]],
          nrow = 3, ncol=3)

#uniform distribution
grid.arrange(list_plots[["fiberid"]], list_plots[["camcol"]], list_plots[["native"]],
          nrow=2, ncol=2)

#skew
grid.arrange(list_plots[["u"]], list_plots[["g"]], list_plots[["r"]],
```

```
        list_plots[["i"]], list_plots[["ra"]],
        nrow = 3, ncol=2)

# bellshaped
grid.arrange(list_plots[["z"]],list_plots[["flux"]],
        list_plots[["field"]],
        nrow = 2, ncol=2)

#bimodal
grid.arrange(list_plots[["m_unt"]],nrow = 1, ncol=1)
```

1.1.iii

The histograms have been plot using the "ggplot"" function from the "ggplot2"" library. Orange was used to represent the "GALAXY" class, green represents the "QSO" class and blue represents the "STAR" class.

no unique values

The "objid" and the "rerun" attribute have the simplets looking graph. This is because they do not have unique values. It is a peculiar characteristic present in the "objid" as an identification attribute. This may be due to the "objid" being an absurdly large number (more than 10^18). Conversely, the "rerun" attribute represents how the image was processed. Therefore, it can be assumed that all the images in the dataset was processed in the same way, given that the "rerun" value is the same for every observation.

extremely skewed

The "dia", "dec", "specobjid", "redshift", "plate", "mjd", and "run" attributes are extremely skewed. Causing the histogram to look as if there is almost no values on the other side. This behaviour is due to the presence of outliers in the far reaches of the other side. The presence of outliers in a dataset containing galaxy, star, and quasar objects may be explained by the huge difference in size between the three class objects. For example, the

largest recorded "dia" attribute in the dr14 dataset is 848171 and the smallest recorded "dia" attribute is 27.

uniform distribution

The "fiberid", "camcol", and "native" attributes have a uniform distribution. Upon further inspection, the classes are also evenly distributed.

skew

The "u", "g", "ra" attributes have a negative (left) skew. Whereas the "r" and "i" attributes have positive (right) skew.

bellshaped

The "z", "field", and "flux" attributes have a bellshaped distribution.

bimodal

The "m_unt" attribute has a bimodal shape which has two peaks.

1.2.i

1.2.

1.2.i

"r" and "g" has a pearson coefficient of +0.96.

```{r}
#calculate correlation
cor(dr14omit$r, dr14omit$g, method = c("pearson", "kendall", "spearman"))
```

Based on the scatterplot produced, "r" increases linearly as "g" increases. "r" and "g" have a perfect positive correlation.

```{r}
ggplot(dr14omit,
      aes(x=g, y = r, color = class)) +
  geom_point() + geom_smooth(method = "lm", se=TRUE) +
  ggtitle("r vs g grouped by class")
```

1.2.ii

"mjd" and "r" has a pearson coefficient of -0.034.

```{r}
#calculate correlation
cor(dr14omit$mjd, dr14omit$r, method = c("pearson", "kendall", "spearman"))
```

Based on the scatterplot produced, there is no pattern between "mjd" and "r". Therefore, "mjd" and "r" have no correlation.

```{r}
ggplot(dr14omit,
      aes(x=r, y = mjd, color = class)) +
  geom_point() + geom_smooth(method = "lm", se=TRUE) +
  ggtitle("mjd vs r grouped by class scatterplot")
```

1.2.iii

Scatterplot between class and u, z and redshift

```{r}
ggplot(dr14omit,
```

```
    aes(x=class, y = u, color = class)) +
  geom_point() + geom_smooth(method = "lm", se=TRUE) +
  ggtitle("u vs class scatterplot")
```

The 3 classes in the dr14 dataset is not separable when using the "u" attribute. This is because the three classes overlap in at a "u" value of 17.5 to 19.5. However, the "GALAXY" and "STAR" class has a larger range than "QSO" with the "GALAXY" class having a slightly smaller minimum "u" value compared to the minimum "z" value of "STAR". The "GALAXY" class has the smallest minimum "u" value as well due to the presence of outliers. Generally, There are outliers present in the "u" values of all three classes.

```{r}
ggplot(dr14omit,
    aes(x=class, y = z, color = class)) +
  geom_point() + geom_smooth(method = "lm", se=TRUE) +
  ggtitle("z vs class scatterplot")
```

The 3 classes in the dr14 dataset is not separable when using the "z" attribute. This is because the three classes overlap at a "z" value of 15 to 19.5. However, the "GALAXY" and "STAR" class have a larger range than "QSO" with the "GALAXY" class having a smaller minimum "z" value and "STAR" having a higher minimum "z" value. The "z" value of the "GALAXY" class also has the highest maximum value due to the presense of outliers which lie very far away from the mean "z" value.

```{r}
ggplot(dr14omit,
    aes(x=class, y = redshift, color = class)) +
  geom_point() + geom_smooth(method = "lm", se=TRUE) +
  ggtitle("redshift vs class scatterplot")
```

The 3 classes in the dr14 dataset is highly separable when using the "redshift" attribute. This is because the three classes do not overlap at a "redshift" value. This is difficult to see in this graph because the "redshift"" values of the "STAR" class are very small and negative.

However, the "GALAXY" and "QSO" class has a larger range than "STAR" with the "QSO" class having a larger maximum "redshift" value. The presence of outliers in the "redshift" values in the three classes are minimal as well.

```{r}
a = c("u", "z", "redshift")
pairs(dr14omit[,a], col=dr14omit$class)
```

Based on the scatter plot of the "u", "z" and "redshift" values using the "pairs" function , it can be inferred that "u" and "z" are highly correlated. They also do not separate the three classes well, whereas the "redshift" values are not correlated with "u" or "z" and "redshift" can can separate the classes well as seen in the non overlapping colours.

1.2.iv

I have determined the approprate attributes to be values which are continous and do not represents object identification numbers.

```{r}
plotbox = function(df, col){
  plot = ggplot(df, aes(x=class, y = df[,col], color = class)) +
    geom_boxplot() +
    ggtitle(paste(col, " vs class boxplot"))+
    # theme(legend.position = "none",
    #       # axis.text.x = element_blank(),
    #       axis.text.y = element_blank(),
    #       plot.title = element_text(size=10))
    scale_y_continuous(name = col)
  return(plot)
}
```

```
# plot = ggplot(df, aes(x = df[,col], fill=class)) +
#     geom_histogram(bins = 30,  na.rm = TRUE) +
#     theme(legend.position = "none",
#           # axis.text.x = element_blank(),
#           axis.text.x = element_text(angle=0),
#           axis.text.y = element_blank(),
#           plot.title = element_text(size=10)) +
#     scale_x_continuous(name = col) +
#     scale_y_continuous(name = "frequency") +
#     ggtitle(paste(col, " attribute of observations"))

plotAllBox= function (df, colnames){
  list_plots = list()
  for (i in 1:length(colnames)){
    plot = plotbox(df, colnames[i])
    list_plots = append(list_plots,list(plot))
  }
  names(list_plots) = colnames
  return(list_plots)
}

attName = colnames(dr14)
list_plot_box = plotAllBox(dr14omit, attName[1:length(attName)-1])
```

```{r}
# fig.cap= "caption"
# grid.arrange(list_plot_box[["dia"]], list_plot_box[["ra"]], list_plot_box[["dec"]],
```

```
#           list_plot_box[["u"]], list_plot_box[["g"]], list_plot_box[["r"]],
#           list_plot_box[["i"]], list_plot_box[["z"]], list_plot_box[["flux"]],
#           list_plot_box[["redshift"]],
#           nrow =4, ncol=3)



#+ve skew
grid.arrange(list_plot_box[["dia"]], list_plot_box[["dec"]], list_plot_box[["redshift"]],
        nrow =2, ncol=2)


#-ve skew
grid.arrange(list_plot_box[["ra"]], list_plot_box[["u"]],  list_plot_box[["g"]],
        list_plot_box[["flux"]],
        nrow =2, ncol=2)


#normal distribution
grid.arrange(list_plot_box[["r"]], list_plot_box[["i"]], list_plot_box[["z"]],
        nrow =2, ncol=2)
```

These boxplots show that the data is positively skewed. Based on the boxplots, the "dia" and "dec"" attribute have outliers a lot higher than the whiskers of the boxplots.This is shown by the presence of values far away from the median. Excluding the outliers, the values of the "dia" and "dec" attribute are very close to the median. This is shown by the small interquartile ranges of the two values in all classes.

The "redshift" values of the "GALAXY" and "STAR" class have outliers which are not as extreme as the outliers in the "dia" and "dec" attribute. The values also lie close to the median as shown by the low interquartile range. Whereas the "redshift" value of the "QSO"

class have outliers and has a larger interquartile range compared to the "GALAXY" and "STAR" class.

These boxplots show that the data is negatively skewed. Based on the boxplots, the "ra", "u", "g" and "flux" attribute have outliers a lot lower than the whiskers of the boxplots.This is shown by the presence of values lower than the median. Excluding the outliers, the interquartile range of "ra", "u", "g" and "flux" attributes shown are larger than in the previously shown "."dia", "dec", and "redshift". This is shown by the bigger boxplots of the values in all classes.

Based on the boxplots of "r", "z" and "i" values vs "class", these values have a normal distribution. The "r", "i", and "z" values of the "GALAXY" and "STAR" class have larger interquartile range compared to the "QSO" class. Furthermore, the "r", "i", and "z" values for the "GALAXY" class have many outliers as shown by the abundence of dotted points located above and below the whiskers.

1.3.

After observing the dataset and the visualisation produced, some deuctions were made about the information held by particular attributes.

There are 2 atrributes which only has 1 unique value which can be seen in the histograms plotted, these attributes are the "objid" and "rerun" attributes. Attributes with only 1 unique value does not hold significant information to classify or cluster the dataset into right classes.

A uniform histogram is where the histogram produces a rectangular shape. The attributes with an even distribution are the "fiberid", "camcol", and "native" attributes. The frequency of these attributes are very consistent throughout all values of "fiberid", "camcol", and "native". These attributes do not seem to hold significant information as all object classes have similar values.

Based on the scatterplots produced, there are several pairs of attributes which are highly correlated. These pairs of attributes are "u" and "z" as well as "r" and "g". Due to the high correlation between these pairs of attributes, "z" and "g" are considered redundant as they carry the same information as "u" and "r".

Based on the scatterplots, there are also some attributes which contain significant information of the dataset such as "redshift", "g", "r", "i" and "z". These attributes are able to separate at least the "QSO" class from "GALAXY" and "STAR". However, "redshift" is able to completely separate the classes.

1.4

```{r}
dr14drop = select(dr14, -c(objid,rerun,run,field,specobjid,camcol))
```

```{r}
##functions for replacing na values with mean and median according to class
medColsByClass = function (df){
  dfreplaced = df
  for (i in 1:(ncol(df) -1) ){
    #median of all classes

    tempcol = df[df$class=="GALAXY",i]
    med = median(tempcol[!is.na(tempcol)])
    tempcol[is.na(tempcol)] = med

    dfreplaced[dfreplaced$class=="GALAXY",i] = tempcol

    tempcol = df[df$class=="STAR",i]
    med = median(tempcol[!is.na(tempcol)])
    tempcol[is.na(tempcol)] = med
```

```
  dfreplaced[dfreplaced$class=="STAR",i] = tempcol


  tempcol = df[df$class=="QSO",i]
  med = median(tempcol[!is.na(tempcol)])
  tempcol[is.na(tempcol)] = med

  dfreplaced[dfreplaced$class=="QSO",i] = tempcol
 }
 return (dfreplaced)
}

meanColsByClass = function (df){
 dfreplaced = df
 for (i in 1:(ncol(df)-1)){
  #median of all classes

  tempcol = df[df$class=="GALAXY",i]
  mean = mean(tempcol[!is.na(tempcol)])
  tempcol[is.na(tempcol)] = mean

  dfreplaced[dfreplaced$class=="GALAXY",i] = tempcol

  tempcol = df[df$class=="STAR",i]
  mean = mean(tempcol[!is.na(tempcol)])
  tempcol[is.na(tempcol)] = mean

  dfreplaced[dfreplaced$class=="STAR",i] = tempcol
```

```
   tempcol = df[df$class=="QSO",i]
   mean = mean(tempcol[!is.na(tempcol)])
   tempcol[is.na(tempcol)] = mean

   dfreplaced[dfreplaced$class=="QSO",i] = tempcol
 }
 return (dfreplaced)
}
```

```{r}
dr14dropmed = medColsByClass(dr14)
dr14dropmean = meanColsByClass(dr14)
dr14dropzero = dr14
dr14dropzero[is.na(dr14dropzero)] = 0
```

```{r}
# summary(dr14)
# summary(dr14dropmed)
# summary(dr14dropmean)
# summary(dr14dropzero)
summdisp = c(min = min,max = max,mean = mean,median = median,
```

```
      q2={function(x) quantile(x, 0.25)},q3={function(x) quantile(x, 0.75)})
sum_med = sapply(summdisp, {function(x) summarise_all(dr14dropmed[,1:21],x)})
sum_med = as.data.frame(sum_med)
df_med_char = apply(sum_med,2,as.character)
rownames(df_med_char) = rownames(sum_med)

summdisp = c(min = min,max = max,mean = mean,median = median,
      q2={function(x) quantile(x, 0.25)},q3={function(x) quantile(x, 0.75)})
sum_mean = sapply(summdisp, {function(x) summarise_all(dr14dropmean[,1:21],x)})
sum_mean = as.data.frame(sum_mean)
df_mean_char = apply(sum_mean,2,as.character)
rownames(df_mean_char) = rownames(sum_mean)

summdisp = c(min = min,max = max,mean = mean,median = median,
      q2={function(x) quantile(x, 0.25)},q3={function(x) quantile(x, 0.75)})
sum_zero = sapply(summdisp, {function(x) summarise_all(dr14dropzero[,1:21],x)})
sum_zero = as.data.frame(sum_zero)
df_zero_char = apply(sum_zero,2,as.character)
rownames(df_zero_char) = rownames(sum_zero)

######## this determine if wanna put summary in or not

write.csv(df_med_char, "sum_med.csv", row.names = TRUE)
write.csv(df_mean_char, "sum_mean.csv", row.names = TRUE)
write.csv(df_zero_char, "sum_zero.csv", row.names = TRUE)
```

1.5.

```{r}
#mean centering
meanCentering = function (df){
  dfcenter = df
  for (i in 1:(ncol(df) -1) ){
    temp = df[,i]
    dfcenter[,i] = scale(temp,center = TRUE, scale = FALSE)
  }
}

standardize = function (df){
  dfstand = df
  for (i in 1:(ncol(df) -1) ){
    temp = df[,i]
    dfstand[,i] = scale(temp,center = TRUE, scale = TRUE)
  }
  return(dfstand)
}

normalizedf = function(df){
  dfnorm = df
  for (i in 1:(ncol(df) - 1) ){
    temp = df[,i]
    dfnorm[,i] = (temp - min(temp)) / (max(temp) - min(temp))
  }
  return (dfnorm)
}
```

```{r}
#dr14dropmedMeanC = meanCentering(dr14dropmed)
dr14dropmedStand = standardize(dr14dropmed)
dr14dropmedNorm = normalizedf(dr14dropmed)

#dr14dropmeanMeanC = meanCentering(dr14dropmean)
dr14dropmeanStand = standardize(dr14dropmean)
dr14dropmeanNorm = normalizedf(dr14dropmean)

# dr14dropzeroMeanC = meanCentering(dr14dropzero)
dr14dropzeroStand = standardize(dr14dropzero)
dr14dropzeroNorm = normalizedf(dr14dropzero)
```

```{r}
dr14dropmedMeanC = scale(dr14dropmed[,1:ncol(dr14dropmed)-1],center = TRUE, scale = FALSE)
dr14dropmeanMeanC = scale(dr14dropmean[,1:ncol(dr14dropmean)-1],center = TRUE, scale = FALSE)
dr14dropzeroMeanC = scale(dr14dropzero[,1:ncol(dr14dropzero)-1],center = TRUE, scale = FALSE)

dr14dropmedMeanC = as.data.frame(dr14dropmedMeanC)
dr14dropmeanMeanC = as.data.frame(dr14dropmeanMeanC)
dr14dropzeroMeanC = as.data.frame(dr14dropzeroMeanC)

dr14dropmedMeanC$class = dr14dropmed$class
dr14dropmeanMeanC$class = dr14dropmed$class
```

```
dr14dropzeroMeanC$class = dr14dropmed$class
```

```{r}
export_csv(dr14dropmedMeanC, "dr14dropmedMeanC.csv")
export_csv(dr14dropmedStand, "dr14dropmedStand.csv")
export_csv(dr14dropmedNorm, "dr14dropmedNorm.csv")

# export_csv(dr14dropmeanMeanC, "dr14dropmeanMeanC.csv")
# export_csv(dr14dropmeanStand, "dr14dropmeanStand.csv")
# export_csv(dr14dropmeanNorm, "dr14dropmeanNorm.csv")
#
# export_csv(dr14dropzeroMeanC, "dr14dropzeroMeanC.csv")
# export_csv(dr14dropzeroStand, "dr14dropzeroStand.csv")
# export_csv(dr14dropzeroNorm, "dr14dropzeroNorm.csv")
```

```{r}
#check method success
sapply(dr14dropmeanStand[,1:ncol(dr14dropmeanStand)-1], sd)
sapply(dr14dropmeanStand[,1:ncol(dr14dropmeanStand)-1], mean)

sapply(dr14dropmeanNorm[,1:ncol(dr14dropmeanNorm)-1], max)
sapply(dr14dropmeanNorm[,1:ncol(dr14dropmeanNorm)-1], min)

```

1.6.

1.6.i.

```{r}
#missing value per attribute
naCountCol = sapply(dr14, function(y) sum(length(which(is.na(y)))))

#missing values per instance
naCountRow = apply(dr14, MARGIN = 1, function(x) sum(is.na(x)))

sum(naCountRow > 3)
summary(dr14[naCountRow > 3,"class"])

colname = colnames(dr14[,1:21])
colrem = colname[naCountCol>5000]
```

```{r warning = FALSE}
#remove rowws
dr14narem = dr14[naCountRow <= 3,]

#remove column
dr14narem = select(dr14narem, -c(colrem))

summary(dr14narem)
```

Based on number of missing values per attribute, diameter contains too many missing values (over 50% of dataset is empty) so it was removed.

Based on number of missing values per instance, 50 instances have more than 3 missing values and will be removed.Out of the values removed, 30 belonged to the galaxy class, 20 belonged to the star class.

The effect of the data is that 50 overvations have been deleted and 1 attribute has been deleted. Therefore, the size of the dataset has changed from 10052 x 22 matrix to a 10002 x 21 matrix. (talk about changes in mean and median if any)

```{r}
originaldf = dr14
```

1.6.ii

```{r}
cor_pearson = cor(as.matrix(originaldf[,1:ncol(originaldf)-1]), method = "pearson", use="complete.obs")

cor_spearman = cor(as.matrix(originaldf[,1:ncol(originaldf)-1]), method = "spearman", use="complete.obs")
```

(insert correlation table here)

Based on the correlation table, there are several attributes which have a high positive correlation. The scatterplots of these variables have been plotted below.

```{r}
a = c("u","g","r","i","z")

pairs(originaldf[,a], col=originaldf$class)
```

```

As shown in the figure above, the "u", "g", "r", "i", and "z" attribute have a high positive correlation with each other. Therefore, I have decided to remove all attributes except the "z" atrribute.

```{r}
a = c("ra", "flux","mjd", "plate")
pairs(originaldf[,a], col=originaldf$class)
```

As shown in the figure above, the "ra" ,"flux" attribute have a high positive correlation. Whereas the "mjd" and "plate" attribute have a positive correlation. Therefore, I have decided to remove the "flux" and "plate" attributes as they are redundant.

```{r}
dr14correm = dr14dropmeanStand
dr14correm = select(dr14correm, -c("objid","rerun","flux","plate","u","g","r","i"))
```

The effects of attribute selction using the correlations the deletion of 6 correlated attributes.

```{r}
# objid and rerun has only 1 unique values, which holds no meaning to determine the class
# ra and flux high positive correlation
# ugriz data high positive correlation
# mjd and plate high positive correlation
# spaceobjid and plate high positive correlation
# spaceobjid and mjd high + cor
# ra and m_unt 0.7
```
```

```
# flux and m_unt 0.7
# field and flux 0.7

export_csv(as.data.frame(cor_pearson), "cor_pearson.csv")
```

1.7.
1.7.i
conducting pca
```{r}
#pca dataset
#dr14dropmedStand = standardize(dr14dropmed)
pca_df = dr14dropmedStand

pca_df = select(pca_df, -c("objid","rerun","native"))
# pca_df = meanColsByClass(pca_df)
# pca_df = standardize(pca_df)
# pca_df = select(pca_df, -c("objid","rerun","plate","u","g","r","i"))

```

```{r}
#pca with the class attribute removed
dr14_pca = prcomp(pca_df[,1:ncol(pca_df)-1], scale = T)
dr14_pca_sum = summary(dr14_pca)

# prob_export_csv(dr14_pca_sum, "dr14_pca_sum.csv")
```

(1.7.i insert pca summary)

```{r}
#original correlation (export to table for comparison in report)
original_df_cor = cor(as.matrix(pca_df[,1:ncol(pca_df)-1]), method = "pearson")

#pca for data transformation
dr14_pca_cor = cor(dr14_pca$x)
```

```{r}
export_csv(original_df_cor, "ori_cor.csv")
export_csv(dr14_pca_cor, "pca_cor.csv")
################################ until here
```

```{r}
#pca for dimensionality reduction
dr14_pca_dreduc = dr14_pca$x[,1:12]

cor(dr14_pca_dreduc)
```

compare the effects:

```{r}
source("bips.R")
bip(dr14_pca, col=pca_df$class, main = "PC1/PC2 according to class")
```

1.7.ii

8 PCs need to reach cumulative variance of 90%

```{r}
screeplot(dr14_pca, type="lines",col=3, main="Variance explained by PC")
title(xlab="Principal Components")
```

```{r}
#remove outliers
rmv_outliers = function(df, field){
m = mean(df[,field])
s = sd(df[,field])
thrs = 3*s
out = df[df[,field]<=(m+thrs) & df[,field]>=(m-thrs),]
return(out)
}
```

dataset for clustering

```{r}
clustering_df = dr14dropzeroMeanC[,1:ncol(dr14dropzeroMeanC)-1]
df_with_class = dr14dropzeroMeanC
```

2.

2.1.

I have used the dataset which has been transformed using principle components and reduced to 12 dimensions using dimensionality reduction.

(insert 2.1 internal matrix table here)

The internal matrix above were calculated using the "cluster.stats" function from the "cluster" package. This is a short description of the internal matrix used:

max.diameter = maximum cluster diameter.

average.between = average distance between clusters. ()

average.within = average distance within clusters (reweighted so that every observation, rather

than every distance, has the same weight).

dunn = minimum separation / maximum diameter. (higher value is better)

min.separation = minimum cluster separation.

avg.silwidth = average silhouette width.

The hca clustering method has the lowest maximum cluster diameter ("max.dia"), the highest dunn index ("dunn") and the largest minimum cluster separation (min.separation).

Meanwhile, kmeans clustering method has the highest average silhouette width ("avg.silwidth") and the highest average distance between clusters ("average.between").

Lastly, pam clustering method has the lowest average weighted distance betwen clusters ("average.within").

(insert 2.1 external matrix here)

The confusion matrix of the hca clustering method shows that the majority of the "GALAXY" and "STAR" observations have been correctly clustered, but the majority of "QSO"

observations have been wrongly clustered as "GALAXY". The hca clustering method produced the best accuracy, recall, precision adn f1 score compared to kmeans and pam.

```{r}
#HCA
hc= hclust(dist(clustering_df)) #method = complete linkage, dist = euclidean

#plot(hc, xlab="", ylab="Cluster",main="HCA applied to dr14") #shows the whole hierarchy
```

```{r}
k = 3 #number of groups in data

#HCA (linkage-method, dist metric)
res = data.frame(class = df_with_class[,"class"], hca = 0, kmeans = 0, pam = 0)
res$hca= cutree(hc,k) #stops hierarchy at level 3 and saves
hca_tab = table(res$class,res$hca) #shows clusters class label according to clusters
#visualization
#pairs(clustering_df,col=res$hca) #shows pairs for all clusters/attributes

#k-means ()
km3= kmeans(clustering_df,k,iter.max=100) #applies k-means with 3 clusters and 100 iterations
res$kmeans= km3$cluster #saves clusters in dr14$KM3
km3_tab = table(res$class, res$kmeans) #shows clusters class label according to clusters
#visualization
#pairs(clustering_df,col=res$kmeans) #shows pairs for all clusters/attributes

#pam
```

```
pam3 = pam(clustering_df, k) #k=3
res$pam = pam3$clustering #Saves clustering result only
pam_tab = table(res$class,res$pam)
#visualization
#pairs(clustering_df,col=res$pam)
```

```{r}
#exporting
# hca_tab
# km3_tab
# pam_tab
# write.csv(hca_tab, "hca_tab.csv", row.names = TRUE)
# write.csv(km3_tab, "km3_tab.csv", row.names = TRUE)
# write.csv(pam_tab, "pam_table.csv", row.names = TRUE)
```

```{r}
#visualization

#scatterplot
#pairs(clustering_df,col=res$pam)

#Boxplot by class (to compare before and after clustering)
# par(mfrow=c(2,2))
# boxplot(clustering_df,las=3,main="whole data")
# boxplot(clustering_df[df_with_class$class=="GALAXY",],las=3,main="GALAXY")
```

```
# boxplot(clustering_df[df_with_class$class=="QSO",],las=3,main="QSO")

# boxplot(clustering_df[df_with_class$class=="STAR",],las=3,main="STAR")

#

# #Boxplot by clusters (clusters that are mixed up will be different, clusters well separated same)

# par(mfrow=c(2,2))

# boxplot(clustering_df,las=3,main="whole data")

# boxplot(clustering_df[res$kmeans==1,],las=3,main="Cluster 1")

# boxplot(clustering_df[res$kmeans==2,],las=3,main="Cluster 2")

# boxplot(clustering_df[res$kmeans==3,],las=3,main="Cluster 3")
```

```{r}
#internal matrix

distance = dist(clustering_df)

all_res = res[,c(2:4)]

summ=sapply(all_res,
        FUN = function(x){
          cluster.stats(distance,
                  clustering = x,
                  silhouette = TRUE)
          })#takes awhile to load
```

```{r}
#choose internal matrix here for comparison
```

```
total =
as.data.frame(summ[c("cluster.number","max.diameter","average.between","average.withi
n",
                     "dunn", "min.separation","avg.silwidth"),])


total = apply(total,2,as.character)


rownames(total) =
c("cluster.number","max.diameter","average.between","average.within",
                     "dunn", "min.separation","avg.silwidth")


write.csv(total, "internal_matrix.csv", row.names = TRUE)

total
```

```{r}
#configure distance

#internal matrix calculated using hca (Davies-Bouldin Index)

intra.inter.hca = cls.scatt.data(clustering_df,
                     res$hca,
                     dist="manhattan")


intra.inter.kmeans = cls.scatt.data(clustering_df,
                     res$kmeans,
                     dist="manhattan")
intra.inter.pam = cls.scatt.data(clustering_df,
                     res$pam,
                     dist="manhattan")
```

```
intraclust = c("complete","average","centroid")
interclust = c("single", "complete", "average","centroid")

hca_db = clv.Davies.Bouldin(intra.inter.hca, intraclust, interclust)
kmeans_db = clv.Davies.Bouldin(intra.inter.kmeans, intraclust, interclust)
pam_db = clv.Davies.Bouldin(intra.inter.pam, intraclust, interclust)
```

```{r}
# hca_db
# kmeans_db
# pam_db
```

```{r}
#exporting

# write.csv(hca_db, "hca_db.csv", row.names = TRUE)
# write.csv(kmeans_db, "kmeans_db.csv", row.names = TRUE)
# write.csv(pam_db, "pam_db.csv", row.names = TRUE)
```

```{r}
t.hca = table(res$class,res$hca)
t.kmeans = table(res$class,res$kmeans)
t.pam = table(res$class,res$pam)
```

```
# t.hca
# t.kmeans
# t.pam
```

```{r}
#External matrix
maximise_diag = function(table, nentity){
 new_table = table
 maxdiagcols = table[1,]
 maxdiag = 0

 combs = perm(c(1:ncol(table)))

 for (i in 1:nrow(combs)){
  colcomb = combs[i,]
  new_table = table[,colcomb]
  sumdiag = 0
  for (j in 1:nrow(new_table)){
   sumdiag = sumdiag + new_table[j,j]
  }

  if (sumdiag > maxdiag){
   maxdiag = sumdiag
   maxdiagcols = colcomb
  }
 }
```

```
 maxdiagtab = table[,maxdiagcols]
 accuracy = maxdiag/nentity
 recall = recall(maxdiagtab)
 precision = precision (maxdiagtab)
 f1 = 2*(precision*recall)/(precision+recall)

 listout = list(maxdiagtab,accuracy,recall,precision,f1)
 names(listout) = c("aligned_mat", "accuracy", "recall","precision","f1_score")
 return (listout)
}

perm = function(v) {
 n = length(v)
 if (n == 1) v
 else {
  X = NULL
  for (i in 1:n) X = rbind(X, cbind(v[i], perm(v[-i])))
  X
 }
}

recall = function(table){
 new_table = table[,1]
 for (i in 1:nrow(table)){
  new_table[i] = table[i,i]/sum(table[i,])
 }
 return(new_table)
```

```
}

precision = function(table){
  new_table = table[,1]
  for (i in 1:nrow(table)){
    new_table[i] = table[i,i]/sum(table[,i])
  }
  return(new_table)
}
```

```{r}
#external matrices
numEntity =  nrow(clustering_df)
a.hca = maximise_diag(t.hca,numEntity)
a.kmeans = maximise_diag(t.kmeans,numEntity)
a.pam = maximise_diag(t.pam,numEntity)

#need to export to csv by hand
# a.hca
# a.kmeans
# a.pam
#export confusion matrix
write.csv(a.hca[[1]], "ahca.csv", row.names = TRUE)
write.csv(a.kmeans[[1]], "akmeans.csv", row.names = TRUE)
write.csv(a.pam[[1]], "apam.csv", row.names = TRUE)
```

```{r}

#need to export to csv by hand

# a.hca[1]

# a.kmeans[1]

# a.pam[1]


a.hca

# a.kmeans

# a.pam
```


2.2

For hca method, the "distance" and "method" parameters were tuned. The "distance" parameter is the distance measure used to calculate the distance matrix. Whereas the "method" parameter determines the method used to compare clusters. I have found that distance = euclidean and method = complete-linkage performs best.

The parameters tuned in the kmeans method are "centers" and "iter.max". The "centers" parameter determines the number of clusters and the "iter.max" parameter is the maximum iterations allowed to define the best cluster.

Lastly, the parameters tuned in the pam method are the "k" and "metric". These parameters are the same as the "distance" and "method" parameters in the hca method.

The best value for "distance", "centers" and "k" parameters is 3. The parameter tuning technique used was the elbow method. This method was used to prevent over-fitting.

2.3.

2.3.i

(insert transformed pca cluster results)

2.3.ii

(insert transformed reduced pca cluster results)

2.3.iii

(insert redueced dataset results)

2.3.iv

(insert all 3 mean centering results)

2.3.v

3

3.1

3.2

3.3

3.3.i

3.3.ii

3.3.iii

3.3.iv

3.3.v