University of Nottingham, Computer Science School

G54DMA Coursework

Explore The Plant Dataset By R And Weka

Pre-process, Clustering and Classification

JIN ZHAO
Student ID: 14338557

Part 1 Describe/Visualise/Pre-Process Data

A : Descriptive Statistics & Visualisation

A-i Descriptive Table with centrality, dispersion, missing values

Table 1: Summary of each attributes of the Dataset

	CentroidX	CentroidY	Mass	Width	Depth	Orientation0	Orientation1	Orientation2	Orientation3
mean	159.032395424586	239.10667340953	0.349743618454039	218.282438368802	527.521834066667	0.241884884899433	0.183577126256303	0.125053041997211	0.0850361615509066
median	158.4238314	242.56175625	0.3456485335	220.07799555	478.4548071	0.2492458815	0.1839960985	0.124691054	0.082130655
min	110.3001042	154.0731417	0.1316	138.0912337	328.6874215	0.150085676	0.149263406	0.105141912	0.072157606
max	218.1901549	301.0646561	0.5484	276.0645592	720.3505396	0.317740381	0.216980693	0.160313255	0.123009023
Range	107.8900507	146.9915144	0.4168	137.9733255	391.6631181	0.167654705	0.067717287	0.055171343	0.050851417
IQR	28.6968	52.28742635	0.08278136825	41.10206845	270.01703425	0.071629032	0.01460774	0.020008372	0.013761764
var	461.767914345689	1120.2050189335	0.00392905199014141	842.034603100545	15517.4496642612	0.00179683006415595	0.000201224595357335	0.000152104442765938	0.0001130212430825
sd	21.4887857810926	33.4694639773853	0.0626821504907211	29.017832501766	124.569055805449	0.0423890323569192	0.0141853655348509	0.0123330629920526	0.01063114495633
n_NA	0	0	6	6	4	18	10	7	7

	Orientation4	Orientation5	Orientation6	Orientation7	Orientation8	Orientation9	Leaf.weight	LeafArea	Leaf.Hue
mean	0.0670584988788732	0.0504381805287517	0.0520727881180556	0.0562529034725738	0.0638333346256983	0.0743341619777778	0.0282108626198083	9836.6201800211	61.7606097618513
median	0.0654620135	0.050377285	0.051741042	0.055046583	0.0617064615	0.0699334695	0.03	10039.50726	61.83791842
min	0.057501859	0.043237501	0.043753744	0.043725419	0.045384442	0.049406592	0.01	6629.747971	49.49600122
max	0.088841486	0.060698312	0.064907948	0.07353907	0.09152908	0.121500687	0.05	14394.75091	71.37876598
Range	0.031339627	0.017460811	0.021154204	0.029813651	0.046144638	0.072094095	0.04	7765.002939	21.88276476
IQR	0.00990252	0.00468576	0.005444988	0.00622112850000001	0.01264691775	0.017534429	0.02	1306.7849415	5.86613094999999
var	4.58242215097255e-05	1.95134332007338e-05	1.98867036026468e-05	3.48134880363692e-05	8.77594450190031e-05	0.000225930679917986	0.000201275907266323	2263594.80958908	18.0063818350014
sd	0.00676935901764159	0.00441740118177349	0.00445945104274583	0.005900295588898	0.00936800112185108	0.0150309906499201	0.0141871740408836	1504.52477865573	4.24339272693459
n_NA	14	11	4	13	8	4	411	13	11

A-ii visualise each attribute in histograms

1) Bins/Widths for Histograms: I tried 3 ways to calculate the optimal width according to Scott (1979), Izenman (1991) and Scott (1992), which is related to the standard deviation, the range, and the IQR of the data respectively. The calculator formula showed in function 'Width_decision()'. However, I found that they are kind of different and hard to decide efficiently to use which one to implement. Alternatively, a simpler but more effective way was conducted: For each histogram, I changed 'breaks' in basic R for several times by the number '8,10,15,20' as the number is easy and fast to conduct and in a rational range; then I chose one of them based on the histograms that they printed, which was judged by which performed better with clear distribution as well as adequate information displaying. I also returned back to Table 2, implementing some of the suggested bin widths and comparing output images with mine to figure out if my histograms presented appropriate visualisation.

2) Assistant Lines in Histograms: I print the histograms for each attribute for checking the distribution of them and comparing them to the standard normal distribution. Specifically, I add some other lines in these histograms for understanding them more intuitively: the blue line for MEAN of the attribute and the red line for Median; the black curve for the density of the attribute itself. The

green curves are the corresponding standard normal distribution curve for each attribute, which means, with the same mean and standard deviation.

3) Assistant Descriptive Statistics: Skewness and kurtosis for attributes were calculated, shown in Table 3, in order to assist the characterisation the shape of distribution.

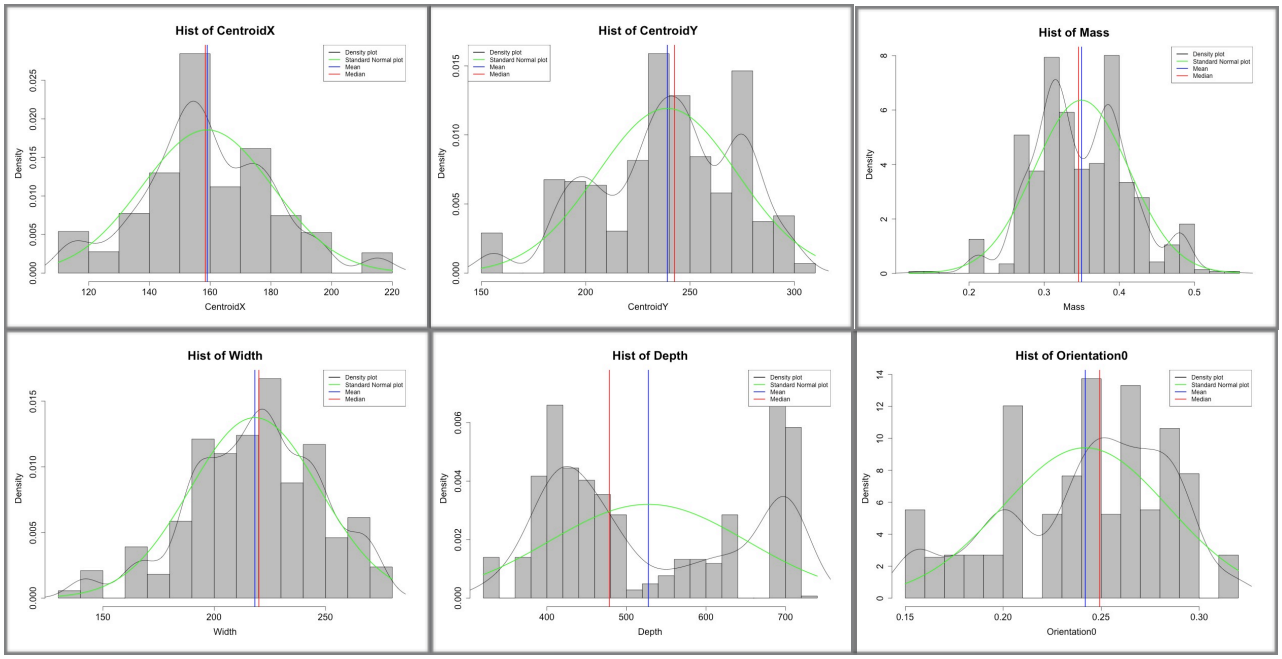
4) Distribution Information:

Table 3: Skewness & Kurtosis of each Attribute

	CX	CY	Mass	Width	Depth	Ori0	Ori1	Ori2	Ori3	Ori4
skewness	0.1522	-0.3612	0.1769	-0.3351	0.3102	-0.512	-0.2312	0.7987	1.4007	0.999
Kurtosis	0.1423	-0.4568	0.0137	-0.060	-1.4839	-0.5989	0.1721	0.4932	2.4250	0.9301

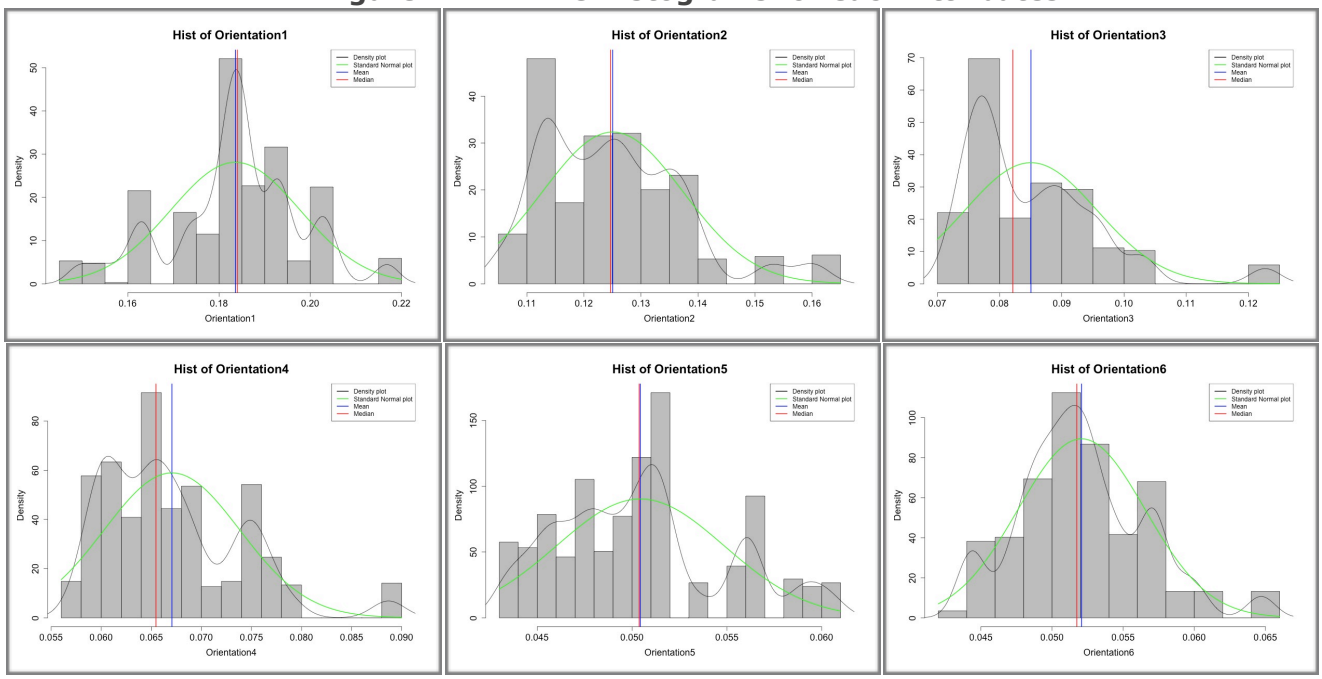
	Ori5	Ori6	Ori7	Ori8	Ori 9	Leaf weight	Leaf Area	Leaf Hue
skewness	0.4983	0.4836	0.5236	0.7664	1.0727	0.1687	0.2094	-0.2673
Kurtosis	-0.4924	0.2883	0.5978	0.5289	1.0419	-1.2957	0.9512	0.3246

Figure 1 - 6: The Histograms for each Attributes



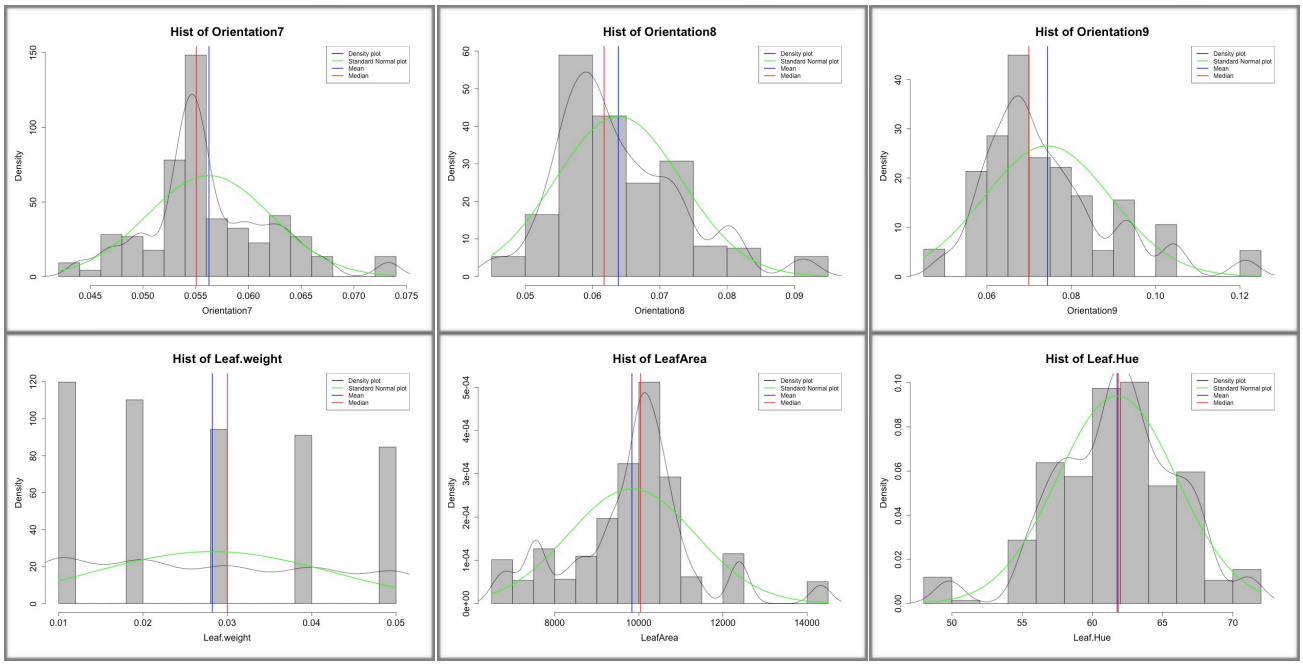
In Figure 1-6, it's clearly that distribution of *CentroidX, CentroidY, Width are highly similar to Standard Normal Distribution (SND), with Symmetrical skew and Gaussian kurtosis; Mass is also similar to SND, while it is slightly bimodal skew and relatively leptokurtic; Depth is bimodal and platykurtic; Orientation0 is negatively skew and platykurtic.*

Figure 7 - 12: The Histograms for each Attributes



In Figure7-12, Orientation6 has the shape *close to SND.* the distribution of Orientation1 is symmetric and leptokurtic; Orientation2, Orientation3 and Orientation4 show slightly positive skewness with gaussian kurtosis; Orientation5 is relatively symmetric with platykurtic shape.

Figure 13 - 18: The Histograms for each Attributes

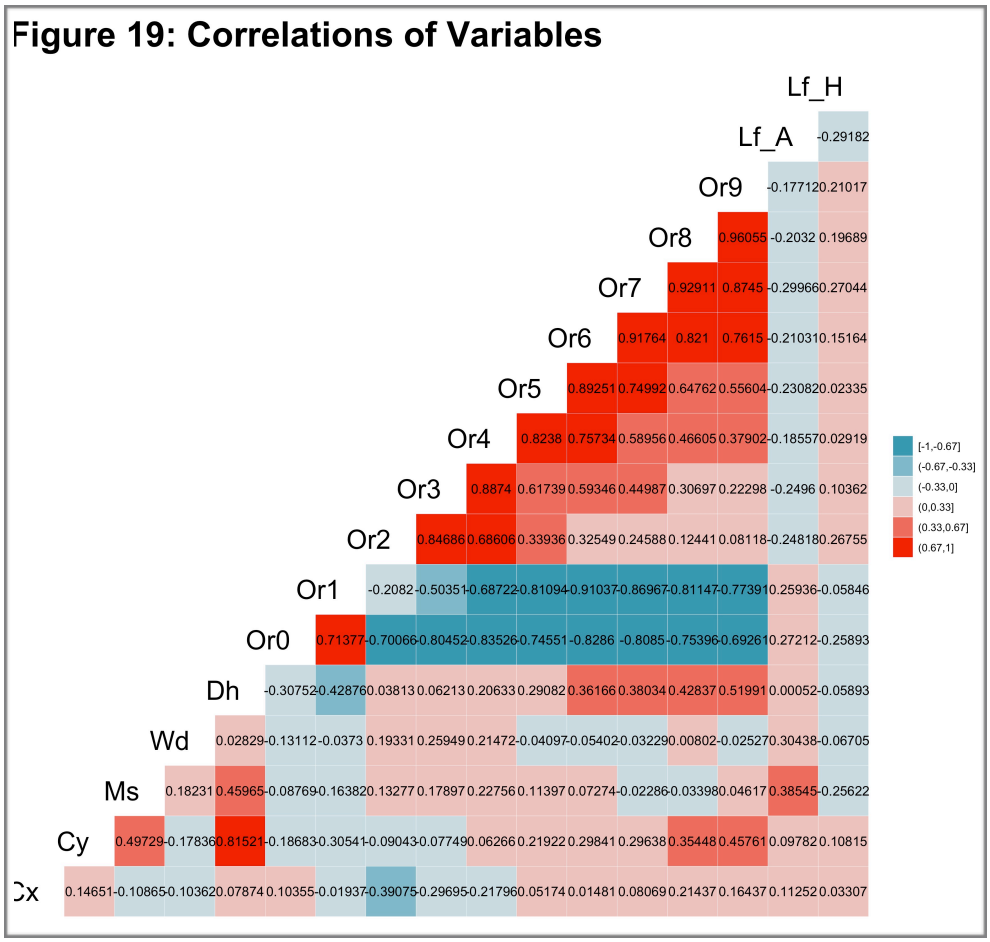


In Figure13-18, Leaf.Hue is relatively symmetric with gaussian kurtosis*; the leptokurtic shape shows in Orientation7, Orientation 8, Orientation9 and LeafArea; Orientation7 and Orientation9 are platykurtic while LeafArea has leptokurtic shape and Orientation 8 has gaussian shape. As for Leaf.weight, 5 specific values presented in these instances with slightly more numbers in 0.01 and less in 0.05.*

1-B : Correlations

1) Correlations between each attributes: Package 'GGally' has been used to get the Figure 19, with six different colours showing the levels of correlation and the numbers presenting the exact numbers in the grids.

1-B-i Calculate correlations, got the answer that: i) Correlation for Orientation1



and Orientation7 is -0.869672, which means they are High Negative Correlation; ii) for Mass and orientation0 is -0.087690, Nearly No Correlation; iii) for orientation7 and Orientation 8 is 0.929113, High Positive Correlation.

1-B-ii & **1-B-iii**: For distinguishing the two most and two least correlated variables, I firstly had a look for Figure 19 and then produced Figure 20, with red circle highlight the ones which 'abs(coefficient) > 0.92' and grey circle for 'abs(coefficient <0.01)'. Clearly, I got the answer that:

B-ii) the two most correlated variables are Orientation8 and Orientation 9 (cor=0.96055); then are Orientation8 and Orientation7 (cor=0.92911). It indicates that these attributes are high related to each other, and they embedded with more similar / redundant information than with other attributes.

B-iii) the two least correlated variables is Depth and Leaf.Area (cor=0.00052); then are Width and Orientation8 (cor=0.00802). It indicates that these

attributes are nearly no correlation with each other; they can be considered as independent attributes with more significant information.

2) Scatterplots between class and other three attributes: Three 2d scatterplots were produced separately for the class and each attributes (Figure21-23), and then a 3D scatterplot were created with an assistant line for

Figure 20: The Most & Least Correlated Variables

Lf_H																	
Lf_A	-0.29182																
Or9	-0.17712	0.21017															
Or8	0.96055	-0.2032	0.19689														
Or7	0.92911	0.8745	-0.29966	0.27044													
Or6	0.91764	0.821	0.7615	-0.21031	0.15164												
Or5	0.89251	0.74992	0.64762	0.55604	-0.23082	0.02335											
Or4	0.8238	0.75734	0.58956	0.46605	0.37902	-0.18557	0.02919										
Or3	0.8874	0.61739	0.59346	0.44987	0.30697	0.22298	-0.2496	0.10362									
Or2	0.84686	0.68606	0.33936	0.32549	0.24588	0.12441	0.08118	-0.24818	0.26755								
Or1	-0.2082	-0.50351	-0.68722	-0.81094	-0.91037	-0.86967	-0.81147	-0.77391	0.25936	-0.05846							
Or0	0.71377	-0.70066	-0.80452	-0.83526	0.74551	-0.8286	-0.8085	-0.75396	0.69261	0.27212	-0.25893						
Dh	-0.30752	-0.42876	0.03813	0.06213	0.20633	0.29082	0.36166	0.38034	0.42837	0.51991	0.00052	-0.05893					
Wd	0.02829	-0.13112	-0.0373	0.19331	0.25949	0.21472	-0.04097	-0.05402	-0.03229	0.00802	-0.02527	0.30438	-0.06705				
Ms	0.18231	0.45965	-0.08769	-0.16382	0.13277	0.17897	0.22756	0.11397	0.07274	-0.02286	-0.03398	0.04617	0.38545	-0.25622			
Cy	0.49729	-0.17836	0.81521	-0.18683	-0.30541	-0.09043	-0.07749	0.06266	0.21922	0.29841	0.29638	0.35448	0.45761	0.09782	0.10815		
Cx	0.14651	-0.10865	-0.10362	0.07874	0.10355	-0.01937	-0.39075	-0.29695	-0.21796	0.05174	0.01481	0.08069	0.21437	0.16437	0.11252	0.03307	

each points (Figure24). With both of them, we can understand the relationships between class and three attributes more clearly.

In Figure 24, **Class A** shows relatively high values in Depth (around 500-800) and low values in Orientation2 and LeafArea (around 0.11-0.14 and 6500-11000 respectively). **Class B** shows similarly high values in Depth (around 400-800) and lower range of values in Orientation2, while slightly higher in Depth. The highest values in both Depth and LeafArea are appeared in **Class C** (over 700 and 1400 respectively), while in Orientation2, all values of class C are lower than 0.14. As for **Class D**, most of its instances are in the moderate level with the lowest values in Depth from this class. **Class E** shows the most dispersed range with the highest values out of five classes in Orientation2 (over 0.160), while in Depth almost all of its instances are below the average (less than 500). Additionally, In Orientation2 and LeafArea, some classes (i.e. E and C respectively) show some distinctly values, which far away from others' ranges.

Figure 21 - 23: Between Class and other 3 attributes

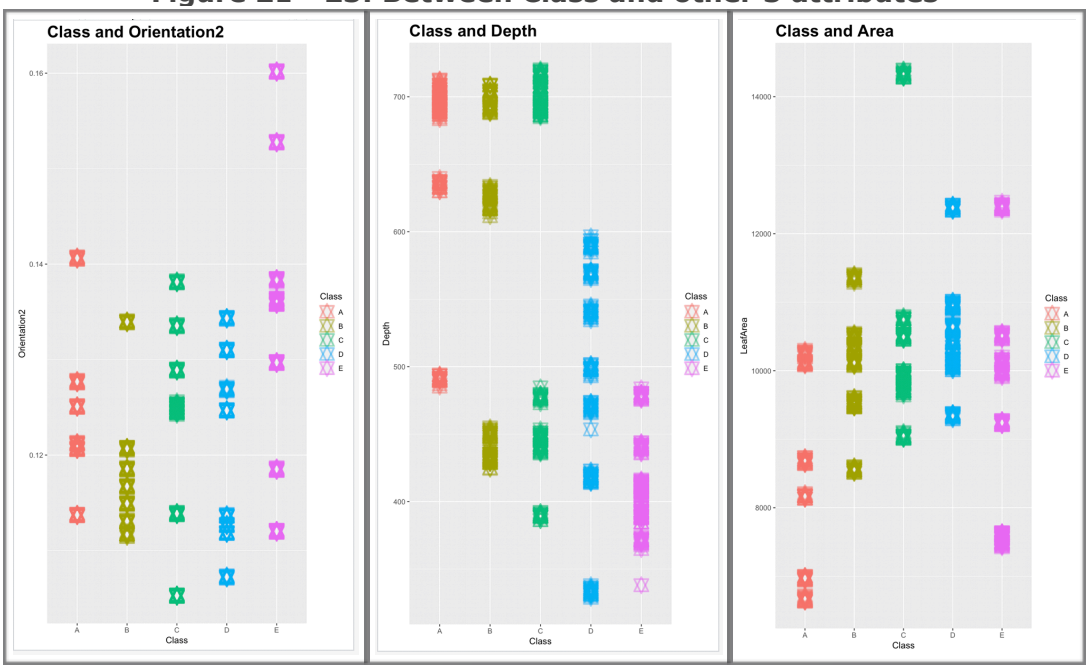


Figure 24 Class in other 3 attributes

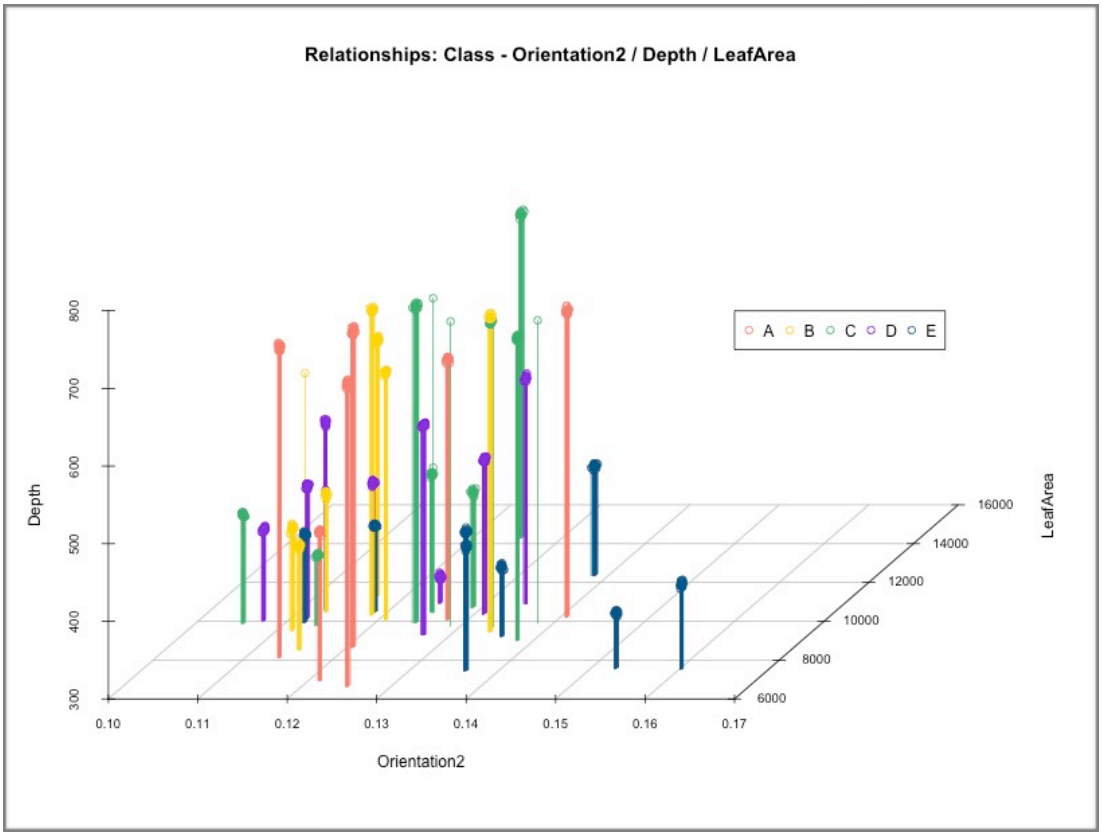
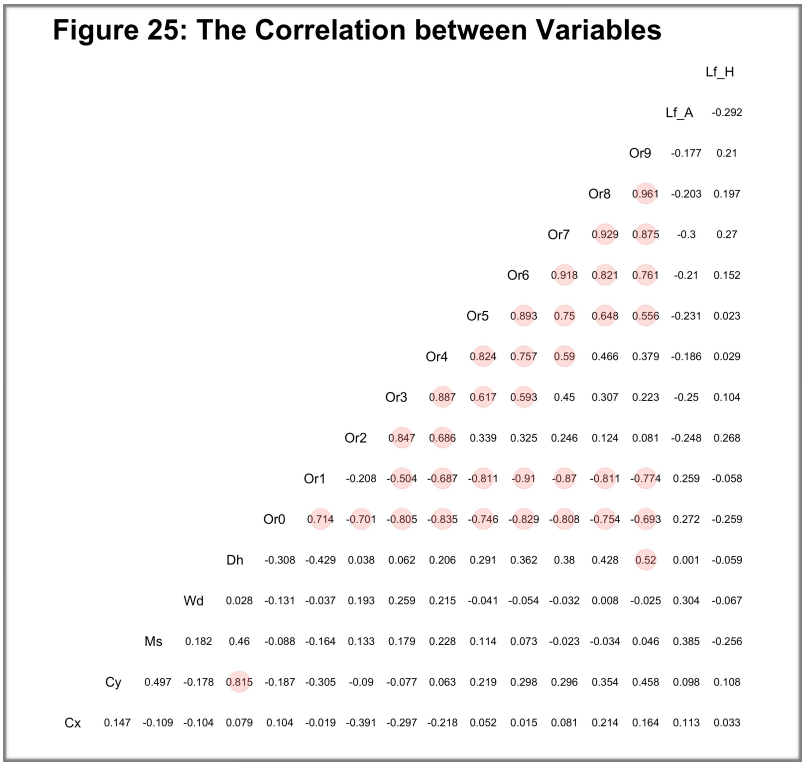


Figure 25: The Correlation between Variables



1-C : General Conclusion 5’

Considering the number of missing values, the attributes ‘Leaf.weight’ missed over 50% values. Although sometimes the missing values can tell some the significant information, in our case it might not. So ‘Leaf.weight’ is regarded as insignificant. Figure 25 is about the correlations between all other attributes with red circle for ‘ abs(coefficient) > 0.5 ’. It indicates that the attributes Orientation0, Orientation1, Orientation3, Orientation4, Orientation5, Orientation6, Orientation7, Orientation8, Orientation9 shows strongly high correlated with other over six attributes. In other words, these attributes hold extremely similar information with other attributes. For this reason, they are regarded as insignificant. Conversely, there is no missing value in CentroidX and CentroidY and they show no or less correlated to other attributes, for which they are considered as the two holding more significant information. For similar reason, the attributes Mass, Width, Depth, LeafArea and Leaf.Hue are considered as significant.

1-D : Dealing with missing values in 3 ways

1-D-i & ii Three ways were to replaced NAs: by 0, mean, median respectively. All of them are easy to implement and work well with small numerical datasets. From TableA, it seemed that 0 changed the value of min in all the attributes it replaced, thus distorted the distribution of attributes to a larger extent. From Table B and Table C we can see these two might have a less influence to the distribution of attributes than method 0. Since mean and median in our dataset are relatively close to each other, there is no significant difference between Table B and C.

Table A - replace_na_by_0

	CentroidX	CentroidY	Mass	Width	Depth	Orientation0	Orientation1	Orientation2	Orientation3
mean	159.032395	239.106673	0.34684519	216.473468	524.607349	0.23587117	0.18104153	0.12384397	0.08421399
median	158.423831	242.561756	0.34435305	219.989553	478.417018	0.24915987	0.18396053	0.12467354	0.08211157
min	110.300104	154.073142	0	0	0	0	0	0	0
max	218.190155	301.064656	0.5484	276.064559	720.35054	0.31774038	0.21698069	0.16031326	0.12300902
Range	107.890051	146.991514	0.5484	276.064559	720.35054	0.31774038	0.21698069	0.16031326	0.12300902
IQR	28.6968	52.2874264	0.08462261	41.9435508	270.618239	0.07243709	0.01750228	0.02000598	0.01357125
var	461.767914	1120.20502	0.00490314	1227.1825	16962.6745	0.00317252	0.00065812	0.00030058	0.00018126
sd	21.4887858	33.469464	0.07002244	35.0311647	130.240833	0.05632512	0.02565394	0.01733712	0.01346333
n_NA	0	0	0	0	0	0	0	0	0

	Orientation4	Orientation5	Orientation6	Orientation7	Orientation8	Orientation9	Leaf.weight	LeafArea	Leaf.Hue
mean	0.06576179	0.04967185	0.05178509	0.05524284	0.06312799	0.07392348	0.01219613	9659.99578	60.822258
median	0.06541253	0.05025573	0.05173259	0.05501911	0.06166733	0.06991752	0	10010.3492	61.6498021
min	0	0	0	0	0	0	0	0	0
max	0.08884149	0.06069831	0.06490795	0.07353907	0.09152908	0.12150069	0.05	14394.7509	71.378766
Range	0.08884149	0.06069831	0.06490795	0.07353907	0.09152908	0.12150069	0.05	14394.7509	71.378766
IQR	0.01000593	0.00482934	0.00545328	0.0062492	0.01352841	0.01779007	0.02	1470.46868	5.97752674
var	0.00013033	5.73E-05	3.47E-05	9.01E-05	0.00013138	0.00025508	0.00028245	3931444.73	74.8840401
sd	0.01141617	0.00757192	0.0058903	0.00949018	0.01146197	0.01597129	0.01680612	1982.78711	8.6535565
n_NA	0	0	0	0	0	0	0	0	0

Table B - replace_na_by_MEAN

	CentroidX	CentroidY	Mass	Width	Depth	Orientation0	Orientation1	Orientation2	Orientation3
mean	159.032395	239.106673	0.34974362	218.282438	527.521834	0.24188488	0.18357713	0.12505304	0.08503616
median	158.423831	242.561756	0.34638106	219.989553	478.858316	0.24915987	0.18396053	0.12472882	0.08221327
min	110.300104	154.073142	0.1316	138.091234	328.687422	0.15008568	0.14926341	0.10514191	0.07215761
max	218.190155	301.064656	0.5484	276.064559	720.35054	0.31774038	0.21698069	0.16031326	0.12300902
Range	107.890051	146.991514	0.4168	137.973326	391.663118	0.16765471	0.06771729	0.05517134	0.05085142
IQR	28.6968	52.2874264	0.08152088	40.9155134	269.784285	0.06972749	0.01451507	0.01993549	0.01344746
var	461.767914	1120.20502	0.00389645	835.046764	15431.5993	0.0017521	0.00019844	0.00015063	0.00011193
sd	21.4887858	33.469464	0.06242152	28.8971757	124.223989	0.04185804	0.01408692	0.01227321	0.01057956
n_NA	0	0	0	0	0	0	0	0	0

	Orientation4	Orientation5	Orientation6	Orientation7	Orientation8	Orientation9	Leaf.weight	LeafArea	Leaf.Hue
mean	0.0670585	0.05043818	0.05207279	0.0562529	0.06383333	0.07433416	0.02821086	9836.62018	61.7606098
median	0.06551608	0.05042775	0.05174898	0.05510547	0.06175281	0.06999618	0.02821086	10010.3492	61.7606098
min	0.05750186	0.0432375	0.04375374	0.04372542	0.04538444	0.04940659	0.01	6629.74797	49.4960012
max	0.08884149	0.06069831	0.06490795	0.07353907	0.09152908	0.12150069	0.05	14394.7509	71.378766
Range	0.03133963	0.01746081	0.0211542	0.02981365	0.04614464	0.0720941	0.04	7765.00294	21.8827648
IQR	0.00983519	0.00463809	0.00543592	0.0061374	0.0125829	0.01748728	0	1294.28604	5.71814494
var	4.49E-05	1.92E-05	1.98E-05	3.42E-05	8.68E-05	0.00022468	8.69E-05	2222893.93	17.7324258
sd	0.0067035	0.00438367	0.0044471	0.00584701	0.00931603	0.01498935	0.00931975	1490.93727	4.2109887
n_NA	0	0	0	0	0	0	0	0	0

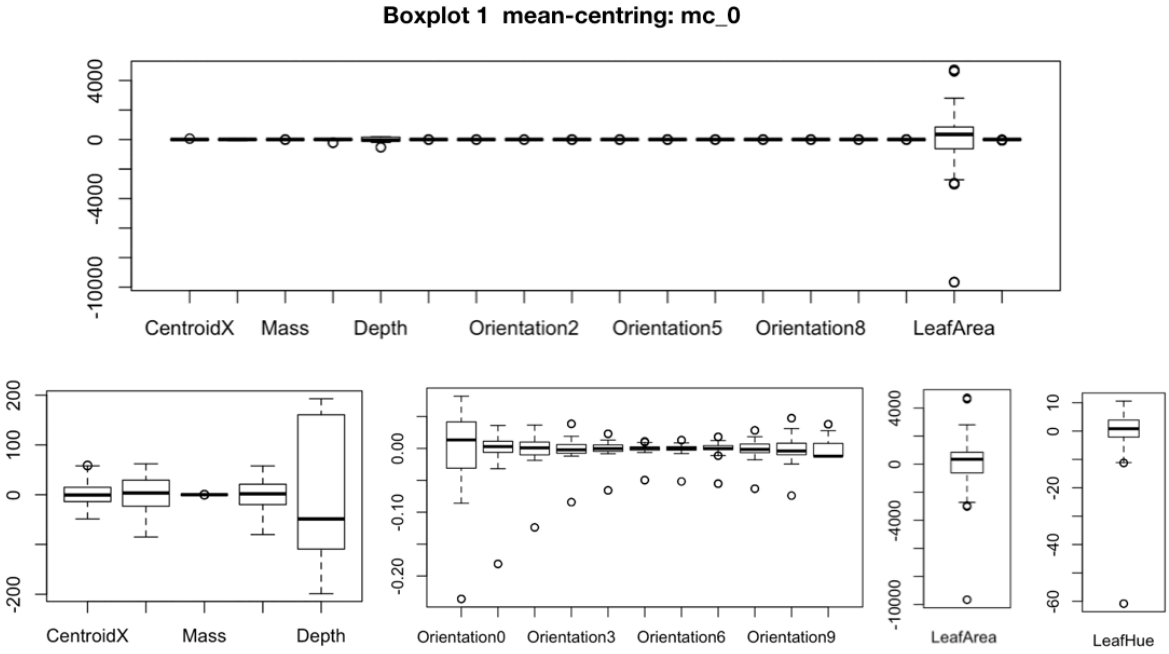
Table C - replace_na_by_Median

	CentroidX	CentroidY	Mass	Width	Depth	Orientation0	Orientation1	Orientation2	Orientation3	Orientation4
mean	159.0324	239.10667	0.3497097	218.29732	527.25075	0.24206789	0.18358291	0.12504954	0.08500807	0.06702763
median	158.42383	242.56176	0.3456485	220.078	478.45481	0.24924588	0.1839961	0.12469105	0.08213066	0.06546201
min	110.3001	154.07314	0.1316	138.09123	328.68742	0.15008568	0.14926341	0.10514191	0.07215761	0.05750186
max	218.19015	301.06466	0.5484	276.06456	720.35054	0.31774038	0.21698069	0.16031326	0.12300902	0.08884149
Range	107.89005	146.99151	0.4168	137.97333	391.66312	0.16765471	0.06771729	0.05517134	0.05085142	0.03133963
IQR	28.6968	52.287426	0.0815209	40.915513	269.78428	0.06972749	0.01451507	0.01993549	0.01344746	0.00983519
var	461.76791	1120.205	0.0038966	835.0733	15444.846	0.00175341	0.00019844	0.00015063	0.00011201	4.50E-05
sd	21.488786	33.469464	0.0624226	28.897635	124.27729	0.04187375	0.01408701	0.01227327	0.01058338	0.00670711
n_NA	0	0	0	0	0	0	0	0	0	0

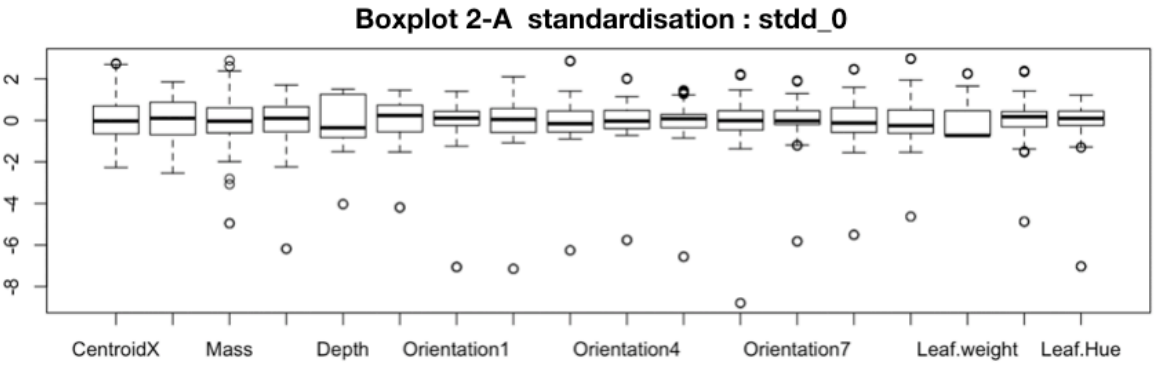
	Orientation5	Orientation6	Orientation7	Orientation8	Orientation9	Leaf.weight	LeafArea	Leaf.Hue
mean	0.05043726	0.05207096	0.05623124	0.06380983	0.07430985	0.0292265	9840.2632	61.761784
median	0.05037729	0.05174104	0.05504658	0.06170646	0.06993347	0.03	10039.507	61.837918
min	0.0432375	0.04375374	0.04372542	0.04538444	0.04940659	0.01	6629.748	49.496001
max	0.06069831	0.06490795	0.07353907	0.09152908	0.12150069	0.05	14394.751	71.378766
Range	0.01746081	0.0211542	0.02981365	0.04614464	0.0720941	0.04	7765.0029	21.882765
IQR	0.00463809	0.00543592	0.0061374	0.0125829	0.01748728	0	1294.286	5.7181449
var	1.92E-05	1.98E-05	3.42E-05	8.68E-05	0.00022479	8.76E-05	2223620.8	17.732515
sd	0.00438367	0.00444717	0.00584921	0.00931868	0.01499291	0.0093619	1491.181	4.2109993
n_NA	0	0	0	0	0	0	0	0

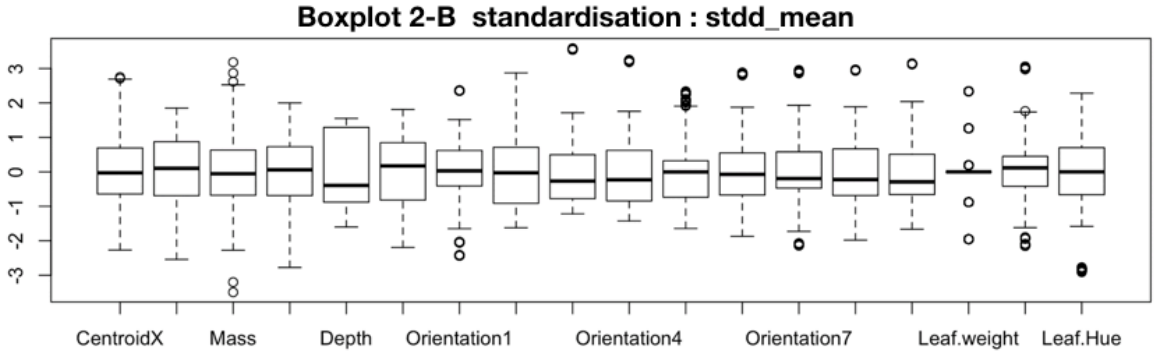
1-E : Attributes Transformation in 3 ways

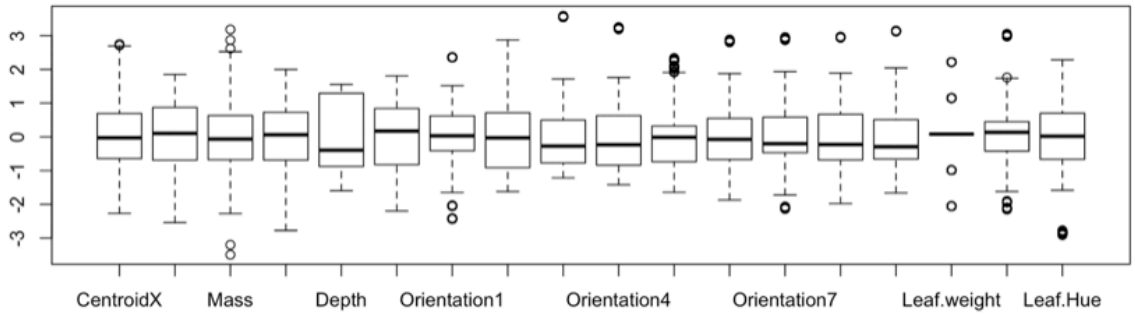
1-E-i: see functions mean_centre(), standard_z(), normal_min_max().

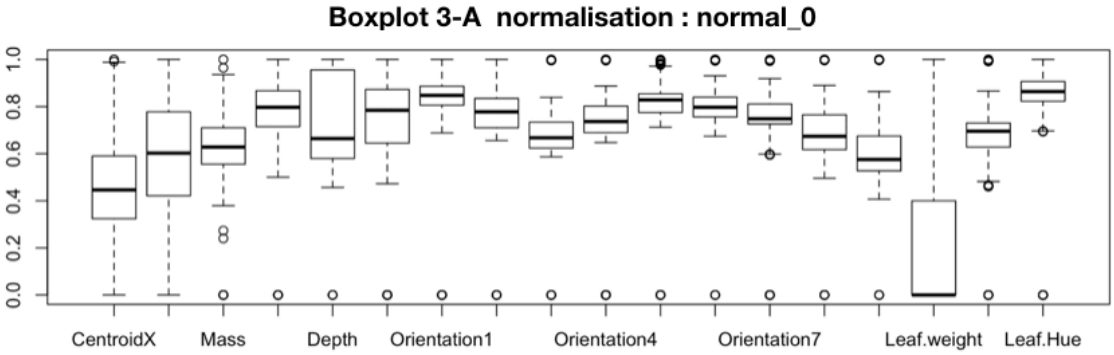


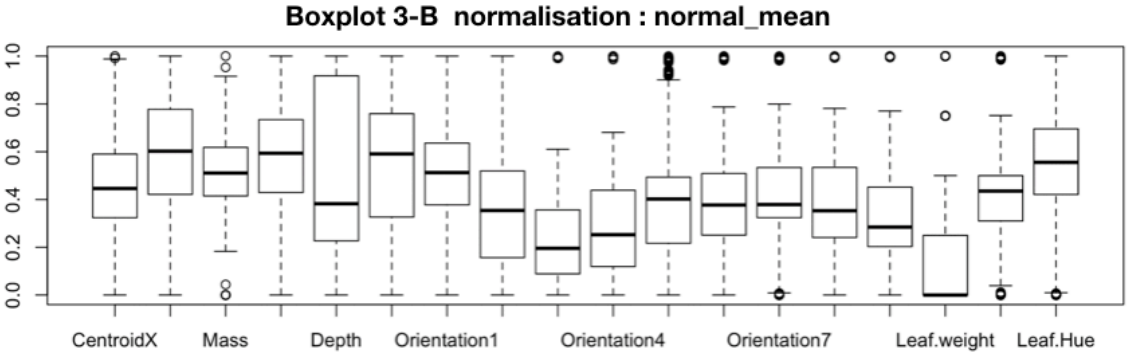
1-E-ii: Mean Centring removed mean from all values and ensured all attributes to distribute around the centre 0, see in Boxplot 1. It reflected the range of each attributes, although the distribution might be not easy to intuitively compare when the arranges of them vary. Boxplot2 illustrated the consequences of using Standardisation in three datasets. All attributes were scaled and easy to compare. It cannot reflect the arranges of attributes, while it represents the distribution of the attributes and all instances influenced the consequences. As for Normalisation, shown in Boxplot3, all the values of instances were plotted in the range of 0 to 1. It was influenced by the min and max, rather than the whole values of instances. Comparing three datasets in Boxplot2-3, it indicates that the way of replacing NA by 0 will have a larger impact on the attributes' distribution than other two replacing ways.

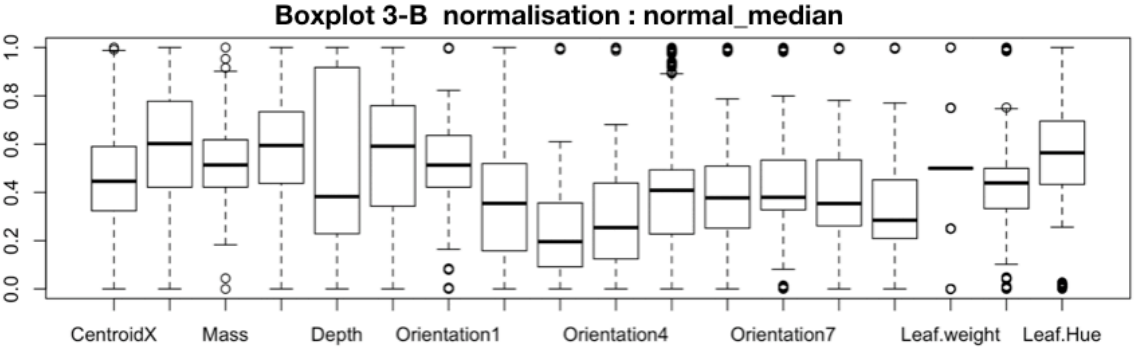



Boxplot 2-B standardisation : stdd_mean
-3 -2 -1 0 1 2 3
CentroidX
Mass
Depth
Orientation1
Orientation4
Orientation7
Leaf.weight
Leaf.Hue

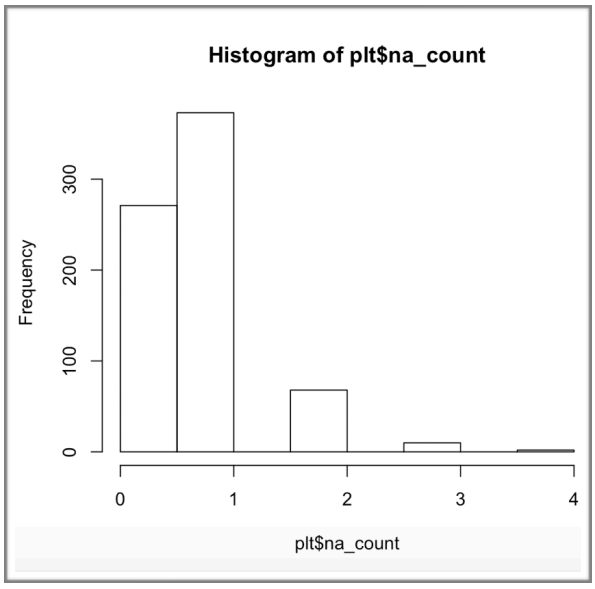

Boxplot 2-C standardisation : stdd_median
-3 -2 -1 0 1 2 3
CentroidX
Mass
Depth
Orientation1
Orientation4
Orientation7
Leaf.weight
Leaf.Hue


Boxplot 3-A normalisation : normal_0
0.0 0.2 0.4 0.6 0.8 1.0
CentroidX
Mass
Depth
Orientation1
Orientation4
Orientation7
Leaf.weight
Leaf.Hue


Boxplot 3-B normalisation : normal_mean
0.0 0.2 0.4 0.6 0.8 1.0
CentroidX
Mass
Depth
Orientation1
Orientation4
Orientation7
Leaf.weight
Leaf.Hue


Boxplot 3-B normalisation : normal_median
0.0 0.2 0.4 0.6 0.8 1.0
CentroidX
Mass
Depth
Orientation1
Orientation4
Orientation7
Leaf.weight
Leaf.Hue

1-F : Attributes/Instances Selection, PCA



1-F-i: For Instances Deletion, the number of NAs in each row was counted, showing that the range of NAs is from 0 to 4 and most of instances have only 0 or 1 missing value. Consequently, I decided to delete instances with more than one NA (>=2). Here, I left only 644 rows/ instances.

For Attributes Deletion, the number of NAs in each attributes was counted, which indicates that Leaf.weight missed over 50% values while all other attributes missed less than 2% values. Consequently, the attribute Leaf.weight are deleted.

For next steps to deal with the remained missing values, I conducted Predictive Mean Matching (PMM) methods to produce imputed values that were more like the real values, given that PMM take the specific distribution of the attribute into consideration, rather than merely based on normal distribution. I took the 'Class' into consideration to replace each NAs according to its sub-datasets for each class and then combined them together. Here I got dataset 'imp_plt', used in Part 2.

1-F-ii: Uncorrelated Attributes and No missing Values.

I deleted the Leaf.weight which missed over 50% values. For correlation consideration in Figure25, I considered to form sub-dataset in which the correlations <0.5.

1) I removed 4 attributes Orientation0, Orientation1, Orientation5, Orientation6 which is highly correlated with other 9 or 8 attributes. Got Figure26. **2)** I decided to remove attributes Orientation3 and Orientation8, as they had the highest values of correlation; For Depth and CentroidY, I remained CentroidY given their distribution, missing value and what mentioned in 1-F-i. Got Figure27. **3)** Afterwards, Orientation7 and Orientation4 were deleted. Results showed in Figure 28. **4)** Finally, the rows with missing values are reduced. I deleted the rows directly to get dataset 'plt_uncor_omit'; replaced them by median to get dataset 'plt_uncor_median'.

1-F-ii: Principle Components Analysis (PCA). Before Applying PCA, I pre-processed the dataset by: 1) deleted the duplicated rows by attributes CentroidX and CentroidY as their values are unique and the following values in other columns are likely to be the same when these two are the same. 2) deleted the attribute 'Leaf.weight' which has too many NAs; and replaced other NAs by mean; 3) standardized the variables with scale() function. Then PCA was applied.The results showed PC1 to PC5 had a higher standard deviation (more than 1); the proportion of variance is decreasing with the increasing of the No. components; PC1 to PC7 has included 92% proportion of the whole dataset.

Table 3 Importance of Components

PCs	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
sd	2.7239	1.6203	1.4471	1.10582	1.02311	0.86846	0.76940	0.63492	0.57191
Proportion of Variance	0.4365	0.1544	0.1232	0.07193	0.06157	0.04437	0.03482	0.02371	0.01924
Cumulative Proportion	0.4365	0.5909	0.7141	0.78602	0.84759	0.89195	0.92678	0.95049	0.96973

PCs	PC10	PC11	PC12	PC13	PC14	PC15	PC16	PC17
sd	0.3665	0.33172	0.29089	0.25670	0.21853	0.16901	0.15645	0.13886
Proportion of Variance	0.0079	0.00647	0.00498	0.00388	0.00281	0.00166	0.00144	0.00113
Cumulative Proportion	0.9776	0.98410	0.98908	0.99296	0.99577	0.99743	0.99887	1.00000

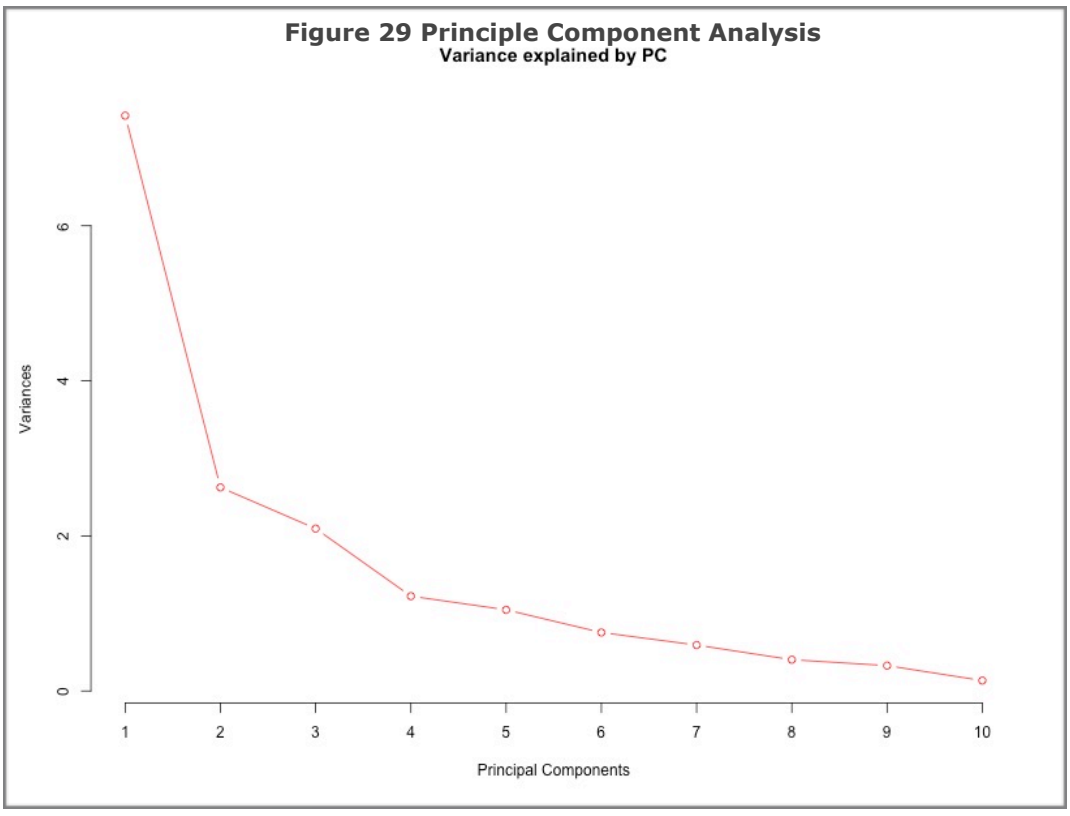


Figure 29 Principle Component Analysis
Variance explained by PC

Part 2 Clustering

2-A : Use Hierarchical, K-Means, PAM

1) Dataset & Pre-Process: The dataset 'imp_plt' produced in step'1-f-i' has been used here. Before clustering, I conducted three additional processes to this dataset in order to get better results, which are: ① Deleted the high correlated attributes, same as what did in f-ii; ②Removed the outliers by each attribute; ③ Standardised the whole datasets. After these, I got dataset 'stdd_plt', and conducted these three algorithms.

Figure 31 Results of HCA (dataset: stdd_plt)

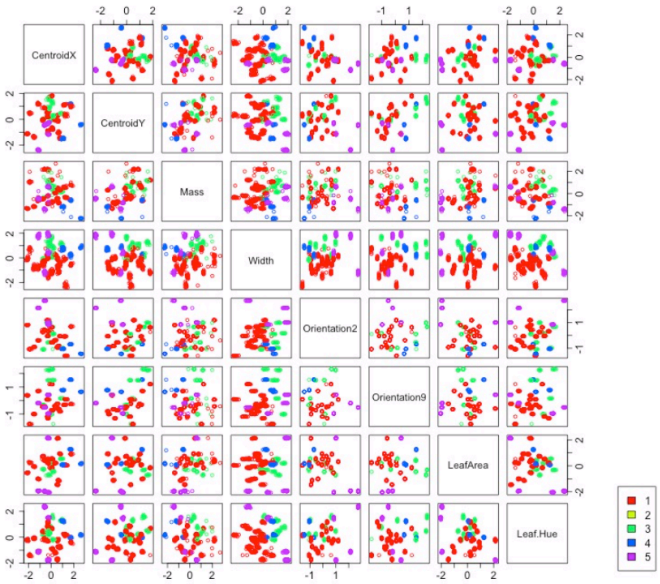


Figure 32 Results of K-Means (dataset: stdd_plt)

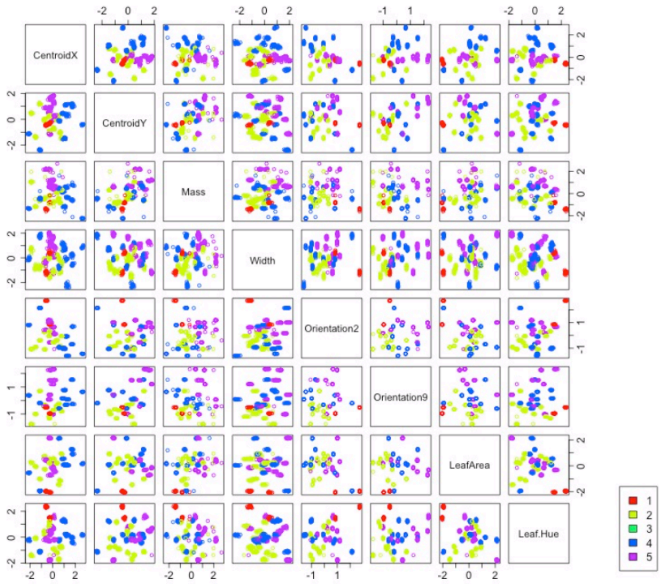


Figure 33 Results of PAM (dataset: stdd_plt)

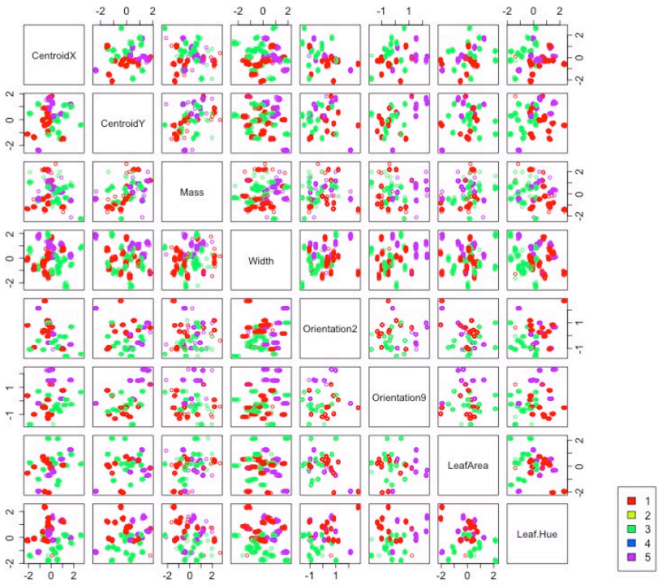
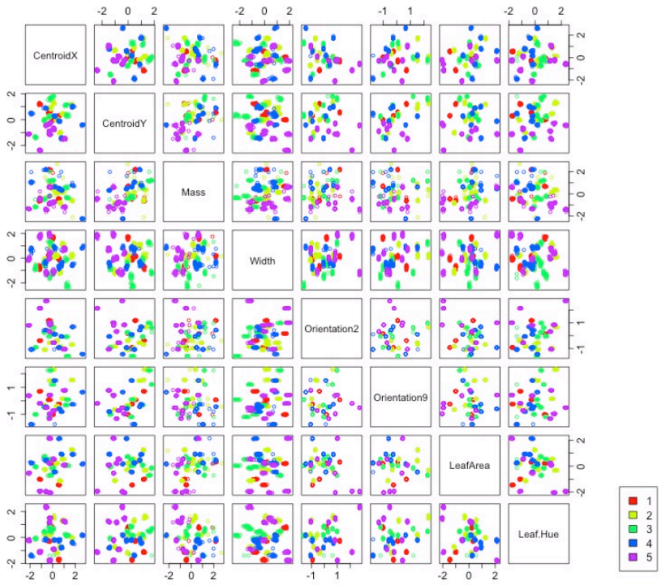


Figure 34 The Real Class(dataset: stdd_plt)



2) Compared Results: Clustering results were evaluated by the known classes of the dataset. By Confusion Matrix, firstly I adjusted the row to maximise the diagonal (Table4-6), aiming to match the clusters with the known classes. Afterwards, True Positive Rate (TPR) and Purity were calculated (Table7 with TPR&Purity). It seemed that PAM showed the better performance, with the highest rate in both TPR (44.8%) and Purity (48.2%).

Table4： Confusion Matrix 1_HCA

	1	2	3	4	5
A	102	17	0	0	0
B	34	67	18	0	0
C	84	16	10	0	0
D	55	0	20	38	35
E	54	19	0	0	0

Table5： Confusion Matrix 2_KM

	1	2	3	4	5
A	35	17	34	0	33
B	35	50	34	0	0
C	30	19	43	18	0
D	18	36	0	19	0
E	0	40	20	38	50

Table6： Confusion Matrix 3_PAM

	1	2	3	4	5
A	18	18	37	0	0
B	0	89	40	0	19
C	15	0	80	15	0
D	1	0	50	68	0
E	17	49	35	18	0

Table7： Performance of Clustering

	True Positive Rate	Purity
HCA	0.3814	0.4605
K-Means	0.3462	0.3796
PAM	0.4482	0.4815

2-B : Adjusting Parameters

HCA:

I changed the single parameter 'methods' and got Table 8, which showed that the methods 'complete' had an apparently higher value in both TPR and Purity, indicating a better performance.

K-Means:

I used all the three options in the parameter 'methods' respectively with the parameter 'iter.max' being assigned the number from 20 to 2000 by 20. The results in Figure 30-31 illustrated that the algorithm's performance / accuracy wouldn't improve because of 'iter.max'; rather, its performance is quite unstable and can be considered as random. Meanwhile, it seemed that three methods presented no significance difference in TPR and Purity, supported by the Table 9.

Table8： Results of HCA_7methods

	TruePositive	Purity
complete	0.3813708	0.4604569
ward	0.3391916	0.3989455
single	0.1642857	0.3482143
median	0.1642857	0.3482143
average	0.2321429	0.4107143
maquitty	0.2424749	0.3244147
centroid	0.1821429	0.3321429

Table9： Results of K-Means 3methods

	TruePositive	Purity
sd(HartiganWong)	0.05280315	0.041488
sd(Forgy)	0.06441809	0.0517964
sd(MacQueen)	0.0562332	0.0458742
mean(HartiganWong)	0.3847961	0.4370986
mean(Forgy)	0.3765759	0.4319273
mean(MacQueen)	0.3802772	0.4347385

Figure 30-31： True Positive Rate & Purity of K-Means （parameters: methods and iter.max)

Table10-a： Results of PAM: medoids & do.swap

Original Parameters	Corresponding Conditions	TruePositive	Purity
medoids = c(1,2,3,4,5), do.swap = FLASE	no_build & no_swap	0.319859402	0.374340949
medoids = NULL, do.swap = FLASE	yes_build & no_swap	0.448154657	0.481546573
medoids = c(1,2,3,4,5), do.swap = TRUE	no_build & yes_swap	0.448154657	0.481546573
medoids = NULL, do.swap = TRUE	yes_build & yes_swap	0.448154657	0.481546573
Default (NULL, TRUE)	Default	0.448154657	0.481546573

Table10-b： Results of PAM: metric & stand _ nostdd_plt

Dataset	parameters		TruePositive	Purity
	metric	stand		
nostdd_plt	euclidean	stand=TRUE	0.439367311	0.474516696
		stand=FALSE	0.383128295	0.423550088
	manhattan	stand=TRUE	0.41827768	0.476274165
		stand=FALSE	0.377855888	0.414762742

Table10-c： Results of PAM: metric & stand _ stdd_plt

Dataset	parameters		TruePositive	Purity
	metric	stand		
stdd_plt	euclidean	stand=TRUE	0.439367311	0.474516696
		stand=FALSE	0.448154657	0.481546573
	manhattan	stand=TRUE	0.41827768	0.476274165
		stand=FALSE	0.416520211	0.476274165

PAM:

① Firstly, the parameters 'medoids' and 'do.swap' were selected to adjust, as they represent the significant 'build' and 'swap' phase regarding to the set of medoids. According to Table10-a, when implementing in our dataset 'stdd_plt' (n=569), both build and swap phase contributed to a better PAM performance to the same extent; when both of them weren't conducted, TPR and Purity decreased considerably. The default of this two parameters is beneficial.

② Afterwards, I selected the parameters 'metric' and 'stand' to tune with others in default, due to the reason that 'metric' is for methods of calculating dissimilarities, and 'stand' is for the standardisation before that. Since it is related to the standardisation process, I used the two paired datasets with and without standardisation to implement ('stdd_plt' and 'nostdd_plt' respectively). There are two findings from Table10-b and Table10-c:

- For datasets without being standardized(e.g. nostdd_plt), 'stand=True' apparently benefits PAM's performance; while for datasets which have been standardized(e.g. stdd_plt), 'stand=True' might not contribute to PAM performance or even get worse answer. Its default is 'True' for the former datasets and 'False' for the later ones, which is rational.
- When 'metric = euclidean', the PAM shows higher value in TPR than 'metric = manhattan' in both datasets, which means that setting 'metric = euclidean' is likely to improve the PAM performance. Therefore 'euclidean' as the default is reasonable.

2-C: Choose PAM to cluster datasets:

Table 11 (a-f)： PAM for different datasets

	A: stdd_plt				
	1	2	3	4	5
A	18	18	37	0	0
B	0	89	40	0	19
C	15	0	80	15	0
D	1	0	50	68	0
E	17	49	35	18	0

a)

	C-i: pca_10				
	1	2	3	4	5
A	99	19	0	0	0
B	76	38	38	0	0
C	47	34	52	0	0
D	22	0	61	65	21
E	95	19	38	0	0

b)

	C-ii: imp_plt				
	1	2	3	4	5
A	64	0	21	35	15
B	35	54	19	21	19
C	31	0	51	37	0
D	38	0	17	61	18
E	37	53	18	0	0

c)

	C-iii: NA_by_0				
	1	2	3	4	5
A	76	0	55	2	19
B	18	76	21	3	0
C	37	0	93	4	18
D	56	19	57	1	0
E	21	62	61	3	22

d)

	C-iii: NA_by_mean				
	1	2	3	4	5
A	76	0	21	37	18
B	43	62	21	21	22
C	36	0	57	40	0
D	42	0	19	72	19
E	40	58	20	0	0

e)

	C-iii: NA_by_median				
	1	2	3	4	5
A	75	0	22	37	18
B	43	62	21	21	22
C	28	0	57	48	0
D	41	0	19	73	19
E	40	58	20	0	0

f)

Table 12: Performance of PAM in datasets

	TruePositive	Purity
stdd_plt	0.4481547	0.4815466
pca_10	0.35082873	0.39226519
imp_plt	0.357142857	0.386645963
Na_by_0	0.370165746	0.374309392
Na_by_mean	0.36878453	0.399171271
Na_by_median	0.36878453	0.399171271

I conducted PAM to these datasets, with results shown in Table11-13.

I putted the result of dataset stdd_plt from Part2-A for comparison with others, especially the 'imp_plt' where it came. This result indicated the pre-process in Part2-A makes sense: considerably improvements were observed in TPR and Purity between them, from 0.3571 and 0.3866 to 0.4481 and 0.4815 respectively.

Regarding to dataset 'pca_10', which was conducted Principle Components Analysis(PCA) in 'Na_by_mean', presented a slightly lower TPR and purity than 'Na_by_mean'. In other words, the PCA process didn't contribute to or even impaired the PAM performance.

As for the datasets with three NA replacing techniques, it seemed that the PAM performance of 'NA_by_mean' and 'NA_by_median' were extremely similar to each other, which is probably due to the high similarity of the means and medians in our datasets; while 'NA_by_0' led to a relatively lower PAM performance compared to them.

Part 3 Classification

3-A : Five ways to classification by Weka

I conducted five algorithms by using 'sttd_plt' dataset in Weka, with 2/3 training sub-dates and 1/3 for testing. Results shown in Table13 -18, the summary and confusion matrix, which showed that only ZeroR didn't perform well (TPR=25.91%); all the others, KNN, NavieBayers, J48, OneR, perforemed very well, with high TPRs 99.48%.

Table 13-18： Classification Results in Part3-A

Table 13 - ZeroR

	A	B	C	D	E
1	0	0	0	0	18
2	0	0	0	0	30
3	0	0	0	0	32
4	0	0	0	0	43
5	0	0	0	0	37

Table 14 - OneR

A	B	C	D	E
18	0	0	0	0
0	29	0	0	0
0	0	30	0	0
0	0	0	42	0
0	0	0	0	37

Table 15 - J48

	A	B	C	D	E
1	24	0	0	0	0
2	0	46	0	0	0
3	0	0	41	0	0
4	0	0	0	31	0
5	0	0	0	0	50

Table 16 - IBK

	A	B	C	D	E
1	24	0	0	0	0
2	0	47	0	0	0
3	0	0	41	0	0
4	0	0	0	30	0
5	0	0	0	0	50

Table 18 - Summary of five methods

	IBK / KNN	NaiveBayes	J48	OneR	ZeroR
True Positive Rate	99.48%	99.48%	99.48%	99.48%	25.91%
Correctly Classified Instances	192	192	192	192	50
Incorrectly Classified Instances	1	1	1	1	143
Kappa statistic	0.9934	0.9934	0.9934	0.9934	0
Mean absolute error	0.0062	0.01	0.0021	0.0021	0.3168
Root mean squared error	0.0455	0.0595	0.0455	0.0455	0.3981
Relative absolute error	1.97%	3.15%	0.65%	0.65%	100%
Root relative squared error	11.44%	14.96%	11.44%	11.44%	100%
Total Number of Instances	193	193	193	193	193

Table 17 - NaiveBayes

	A	B	C	D	E
1	24	0	0	0	0
2	0	47	0	0	0
3	0	0	41	0	0
4	0	0	1	30	0
5	0	0	0	0	50

3-B : Explore the parameters

There are two important parameters in J48, denoted by C (default = 0.25) and M (default = 2). I used datasets 'NA_by_mean' to explore J48's two parameters with the test options 'use training set' and '10-folds cross-validation'. TPR and Root Relative Squared Error (RRSE) were selected to compare the results, shown in Table 18-19. The results indicates that TPR and RRSE were less or not influenced by C while they would be improved by the decreasing of M. There will be a threshold of M, over which the TPR and RRSE will have a fall (e.g.: in Table18, the threshold is between M=50 and M=100; in Table19 between M=2 and M=10). Thus, the default seems reasonable.

Table 18： Explore Parameters of J48 (Use Training Set)

	M=2	M=10	M=50	M=100
c =0.50	98.78% (15.06%)	98.37% (19.98%)	98.37% (19.98%)	82.92%(48.80%)
c =0.25	98.78% (15.06%)	98.37% (19.98%)	98.37% (19.98%)	82.92%(48.80%)
c=0.05	98.78% (15.06%)	98.37% (19.98%)	98.37% (19.98%)	82.92%(48.80%)

Table 19： Explore Parameters of J48 (10-folds Cross-validation)

	M=2	M=10	M=50	M=100
c =0.50	99.31%(12.75%)	98.90% (16.59%)	98.90% (16.59%)	98.90% (16.59%)
c =0.25	99.31%(12.75%)	98.90% (16.59%)	98.90% (16.59%)	98.90% (16.59%)
c=0.05	99.31%(12.75%)	98.90% (16.59%)	98.90% (16.59%)	98.90% (16.59%)

3-C : Implemented in several datasets

I conducted J48 Classification to these datasets, with results shown in Table11-13. It seemed that all five datasets showed a high TPR with four of them are extremely similar. The dataset 'imp_plt' showed the 100% TPR with 0% RRSE, thus it can be considered as good impact on this algorithm's performance.

Table 20 (a-f) The results of J48 for five datasets: Confusion Matrix, TPR, and RRSE

a) Pca_10

```
=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 39  0  0  0  1 |  a = A
  0 39  0  0  0 |  b = B
  0  0 51  0  0 |  c = C
  0  0  0 49  2 |  d = D
  0  0  0  0 65 |  e = E
```

b) imp_plt

```
=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 39  0  0  0  0 |  a = A
  0 37  0  0  0 |  b = B
  0  0 48  0  0 |  c = C
  0  0  0 45  0 |  d = D
  0  0  0  0 50 |  e = E
```

d) NA_by_mean

```
  a  b  c  d  e   <-- classified as
 40  0  0  0  0 |  a = A
  0 39  0  0  0 |  b = B
  0  0 51  0  0 |  c = C
  2  0  0 49  0 |  d = D
  1  0  0  0 64 |  e = E
```

d) NA_by_mean

```
  a  b  c  d  e   <-- classified as
 40  0  0  0  0 |  a = A
  0 39  0  0  0 |  b = B
  0  0 51  0  0 |  c = C
  2  0  0 49  0 |  d = D
  1  0  0  0 64 |  e = E
```

f) The results of J48 Classification in five datasets by True Positive Rate (TPR) and Root Relative Squared Error (RRSE)

	pca_10	imp_plt	NA_by_0	NA_by_Mean	NA_by_Medain
TPR	98.78%	100%	98.78%	98.78%	98.78%
RRSE	17.50%	0%	15.06%	15.06%	15.06%

e) NA_by_median

```
  a  b  c  d  e   <-- classified as
 40  0  0  0  0 |  a = A
  0 39  0  0  0 |  b = B
  0  0 51  0  0 |  c = C
  2  0  0 49  0 |  d = D
  1  0  0  0 64 |  e = E
```

Reference

Wand, M. P. (1997). Data-based choice of histogram bin width. *The American Statistician*, *51*(1), 59-64.

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika*, *57*(3), 519-530.

Ligges, U. and Ma¨chler, M. (2003). Scatterplot3d – an R Package for Visualizing *Multivariate Data. Journal of Statistical Software,* 8(11), 1–20.

Schmitt P, Mandel J, Guedj M (2015) A Comparison of Six Methods for Missing Data Imputation *J Biomet Biostat* 6: 224.

Appendix

1. All Codes of Part 1 in R

```
library(ggplot2)
library(dplyr)
library(scatterplot3d)
library(RColorBrewer)
library(cluster)


plt <- read.table(file = "g54dma-plant-dataset.csv",header = TRUE,sep = ',')

##################################################
########## Part A - ##############################

###############Functions in
A#######################################

##################################################
##################################################
descriptive = function(plt){

  p_means = colMeans(plt[,1:(ncol(plt))],(ncol(plt)))
  p_medians = c(median(plt[,1],na.rm = TRUE))
  p_mins = c(min(plt[,1],na.rm = TRUE))
  p_maxs = c(max(plt[,1],na.rm = TRUE))
  p_ranges = p_maxs - p_mins
  p_IQRs = c(IQR(plt[,1],na.rm = TRUE)) # IQR(x) = quantile(x, 3/4) -
quantile(x, 1/4).
  p_vars = c(var(plt[,1],na.rm = TRUE))
  p_sds = c(sd(plt[,1],na.rm = TRUE))
  p_miss = sum(is.na(plt[,1]))

  for (i in 2:(ncol(plt))){
    p_medians = cbind(p_medians,median(plt[,i],na.rm = TRUE))
```

```
    p_mins = cbind(p_mins,min(plt[,i],na.rm = TRUE))
    p_maxs = cbind(p_maxs,max(plt[,i],na.rm = TRUE))
    p_ranges = cbind(p_ranges,(max(plt[,i],na.rm = TRUE)-min(plt[,i],na.rm =
TRUE)))
    p_IQRs = cbind(p_IQRs,IQR(plt[,i],na.rm = TRUE))
    p_vars = cbind(p_vars,var(plt[,i],na.rm = TRUE))
    p_sds = cbind(p_sds,sd(plt[,i],na.rm = TRUE))
    p_miss= cbind(p_miss,sum(is.na(plt[,i])))
  }

  it =
as.table(rbind(p_means,p_medians,p_mins,p_maxs,p_ranges,p_IQRs,p_vars,p_
sds,p_miss))
  row.names(it) = c('mean','median','min','max','Range','IQR','var','sd','n_NA')
  write.csv(it,file = 'Descriptive.csv')
  return(it)
}



##############################################
##############################################
distribution = function(plt){
  library(GGally)
  library(e1071)
  p_means = colMeans(plt[,1:(ncol(plt))],(ncol(plt)))
  p_medians = c(median(plt[,1],na.rm = TRUE))
  p_mins = c(min(plt[,1],na.rm = TRUE))
  p_maxs = c(max(plt[,1],na.rm = TRUE))
  p_ranges = p_maxs - p_mins
  p_IQRs = c(IQR(plt[,1],na.rm = TRUE)) # IQR(x) = quantile(x, 3/4) -
quantile(x, 1/4).
  p_vars = c(var(plt[,1],na.rm = TRUE))
  p_sds = c(sd(plt[,1],na.rm = TRUE))
  p_miss = sum(is.na(plt[,1]))
  skew = c(skewness(plt[,1],na.rm = TRUE))
  kurtosis = c(kurtosis(plt[,1],na.rm = TRUE))
```

```
  for (i in 2:(ncol(plt))){
    p_medians = cbind(p_medians,median(plt[,i],na.rm = TRUE))
    p_mins = cbind(p_mins,min(plt[,i],na.rm = TRUE))
    p_maxs = cbind(p_maxs,max(plt[,i],na.rm = TRUE))
    p_ranges = cbind(p_ranges,(max(plt[,i],na.rm = TRUE)-min(plt[,i],na.rm =
TRUE)))
    p_IQRs = cbind(p_IQRs,IQR(plt[,i],na.rm = TRUE))
    p_vars = cbind(p_vars,var(plt[,i],na.rm = TRUE))
    p_sds = cbind(p_sds,sd(plt[,i],na.rm = TRUE))
    p_miss= cbind(p_miss,sum(is.na(plt[,i])))
    skew = c(skew, skewness(plt[,i],na.rm = TRUE))
    kurtosis = c(kurtosis, kurtosis(plt[,i],na.rm = TRUE))

  }

  it =
as.table(rbind(p_means,p_medians,p_mins,p_maxs,p_ranges,p_IQRs,p_vars,p_
sds,p_miss,skew,kurtosis))
  row.names(it) =
c('mean','median','min','max','Range','IQR','var','sd','n_NA','skewness','kurtosis')
  write.csv(it,file = 'Describution.csv')
  return(it)
}

##########################################
########Replace NA by O  ####################
NA_by_0 = function(plt){
  plt_NA_0 = plt
  plt_NA_0[is.na(plt_NA_0)] = 0
  return(plt_NA_0)
}

##########################################
########Replace NA by mean  #################
NA_by_mean = function(plt){
```

```
  plt_NA_mean = plt
  for (i in 1:ncol(plt)){
    plt_NA_mean[,i][is.na(plt_NA_mean[,i])] = mean(plt_NA_mean[,i],na.rm =
TRUE)
  }
  return(plt_NA_mean)
}

##############################################
##############################################
######## Replace NA by median ###############
NA_by_median = function(plt){
  plt_NA_median = plt
  for (i in 1:ncol(plt)){
    plt_NA_median[,i][is.na(plt_NA_median[,i])] =
median(plt_NA_median[,i],na.rm = TRUE)
  }
  return(plt_NA_median)
}


##############################################
##############################################
# 1. Mean centring : subtract mean to move the mean value to zero
# y = (y - mean)
mean_centre = function(plt){
  mean_centre = plt
  # step 2: substract the mean
  for(i in 1:ncol(plt)){
    for (m in 1:nrow(plt)){
      mean_centre[m,i] = plt[m,i] - mean(plt[,i])
    }
  }
  return(mean_centre)
}
```

```
##############################################
##############################################
# 2. Standardisation (Z-score normalization)
# y = (y - mean) / sd
standard_z = function(plt){
  standard_z = plt
  # step 2: substract the mean
  for(i in 1:ncol(plt)){
    for (m in 1:nrow(plt)){
      # this 'if' part is for: when the value is equal to mean, the return will be 0
rather than NaN
      x = plt[m,i] - mean(plt[,i])
      if ( x == 0 ){
        standard_z[m,i] = 0
      }else{
        standard_z[m,i] = (plt[m,i] - mean(plt[,i]))/sd(plt[,i])
      }
    }
  }
  return(standard_z)
}

##############################################
##############################################
# 3. Normalizaton (Min-Max normailization)
# y = (y - min) / (max - min)
normal_min_max = function(plt){
  normal_min_max = plt
  # step 2: substract the mean
  for(i in 1:ncol(plt)){
    for (m in 1:nrow(plt)){
      # this 'if' part is for: when the value is equal to mean, the return will be 0
rather than NaN
      x = plt[m,i] - mean(plt[,i])
      if ( x == 0 ){
        normal_min_max[m,i] = 0
```

```
    }else{
      normal_min_max[m,i] = (plt[m,i] - min(plt[,i])) / (max(plt[,i])-
min(plt[,i]))
    }
   }
  }
  return(normal_min_max)
}


##################################################################
##################################################################
################################ 1-a-i
##################################################################
#############

# call the function written in '1-A-i-summary.R'
sum_plt = descriptive(plt[,2:19])
sum_plt

################################  1-a-ii
##################################################################
############

# get the shape values
shape_plt = distribution(plt[,2:19])
shape_plt

##############################################
#get the histogram regarding to reference.

Width_decision = function(sum_plt){
  # Methods 1 : calculate the optimal width according to Scott,1979
  # sum_plt[9,] is for N_NA
  # sum_plt[8,] is for sd
  sum_plt[10,2] = c( 3.49*sum_plt[8,2]*(724 - sum_plt[9,2] )^(-1/3) )
```

```
  # Methods 2 : calculate the optimal width according to Frredman and Diaconis
(Izenman,1991)
  # sum_plt[6,] is IQR of data in row 5
  sum_plt[11,2] = c( 2*sum_plt[6,2]*(724 - sum_plt[9,2] )^(-1/3) )

  # Methods 3 : calculate the optimal width according to Scott,1992
  # sum_plt[5,] is Range of data
  sum_plt[12,2] = c( sum_plt[5,2] / (1 + log2(724 - sum_plt[9,2])))

  for (i in 3:19){
    # sd in row8, n_na in row9
    sum_plt[10,i] = c( 3.49*sum_plt[8,i]*(724 - sum_plt[9,i] )^(-1/3))
    sum_plt[11,i] = c( 2*sum_plt[6,i]*(724 - sum_plt[9,i] )^(-1/3))
    sum_plt[12,i] = c( sum_plt[5,i] / (1 + log2(724 - sum_plt[9,i])))
  }

  return(sum_plt[10:12,2:19])
}



Width_decision(sum_plt)  # Got the suggested width for each attributes



# https://www.fmrib.ox.ac.uk/datasets/techrep/tr00mj2/tr00mj2/node24.html


########## CentroidX
##############################################################
##########


# 1. Open jpeg file
jpeg("Hist_CentroidX.jpg", width = 800, height = 600)
# 2. Create the plot



hist(plt$CentroidX, prob = T, breaks = 15, col = 8, xlab = "CentroidX", main =
"Hist of CentroidX",
```

```
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$CentroidX,na.rm = T))
abline(v = mean(plt$CentroidX,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$CentroidX,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$CentroidX,na.rm = T), sd=sd(plt$CentroidX,na.rm =
T)),
     col='green', lwd=2, add=TRUE, yaxt="n")

legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))




# 3. Close the file
dev.off()



########## CentroidY
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_CentroidY.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$CentroidY, prob = T, breaks = 15, col = 8, xlab = "CentroidY", main =
"Hist of CentroidY",
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$CentroidY,na.rm = T))
abline(v = mean(plt$CentroidY,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$CentroidY,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$CentroidY,na.rm = T), sd=sd(plt$CentroidY,na.rm =
T)),
```

```
      col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topleft", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()


########## Mass
##############################################################
##########



# 1. Open jpeg file
jpeg("Hist_Mass.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Mass, prob = T, breaks = 15, col = 8, xlab = "Mass", main = "Hist of
Mass",
     cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Mass,na.rm = T))
abline(v = mean(plt$Mass,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Mass,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Mass,na.rm = T), sd=sd(plt$Mass,na.rm = T)),
      col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))
```

```
# 3. Close the file
dev.off()



########## Width
##########################################################
##########

# 1. Open jpeg file
jpeg("Hist_Width.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Width, prob = T, breaks = 15, col = 8, xlab = "Width", main = "Hist of
Width",
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Width,na.rm = T))
abline(v = mean(plt$Width,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Width,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Width,na.rm = T), sd=sd(plt$Width,na.rm = T)),
     col='green', lwd=2, add=TRUE, yaxt="n")



legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()



########## Depth
##########################################################
##########

# 1. Open jpeg file
jpeg("Hist_Depth.jpg", width = 800, height = 600)
```

```
# 2. Create the plot
hist(plt$Depth, prob = T, breaks = 15, col = 8, xlab = "Depth", main = "Hist of
Depth",
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Depth,na.rm = T))
abline(v = mean(plt$Depth,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Depth,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Depth,na.rm = T), sd=sd(plt$Depth,na.rm = T)),
     col='green', lwd=2, add=TRUE, yaxt="n")



legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()



########## Orientation0
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation0.jpg", width = 800, height = 600)
# 2. Create the plot
ggplot(plt,aes(Orientation0)) + geom_histogram(binwidth=0.01)

hist(plt$Orientation0, prob = T, breaks = 15, col = 8,
    xlab = "Orientation0",main = "Hist of Orientation0",
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation0, na.rm = T))
abline(v = mean(plt$Orientation0, na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation0, na.rm = T), col = "red", lwd = 2)
```

```
curve(dnorm(x, mean(plt$Orientation0, na.rm = T), sd=sd(plt$Orientation0,
na.rm = T)),
      col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))


# 3. Close the file
dev.off()



########## Orientation1
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation1.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation1, prob = T, breaks = 15, col = 8, xlab = "Orientation1",
main = "Hist of Orientation1",
     cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation1,na.rm = T))
abline(v = mean(plt$Orientation1,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation1,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation1,na.rm = T),
sd=sd(plt$Orientation1,na.rm = T)),
      col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
```

```
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()



########## Orientation2
##########################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation2.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation2, prob = T,  breaks = 15, col = 8, xlab = "Orientation2",
main = "Hist of Orientation2",
     cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation2,na.rm = T))
abline(v = mean(plt$Orientation2,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation2,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation2,na.rm = T),
sd=sd(plt$Orientation2,na.rm = T)),
      col='green', lwd=2, add=TRUE, yaxt="n")



legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()
```

```
########## Orientation3
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation3.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation3, prob = T, breaks = 15, col = 8, xlab = "Orientation3",
main = "Hist of Orientation3",
     cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation3,na.rm = T))
abline(v = mean(plt$Orientation3,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation3,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation3,na.rm = T),
sd=sd(plt$Orientation3,na.rm = T)),
      col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))


# 3. Close the file
dev.off()


########## Orientation4
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation4.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation4, prob = T, breaks = 15, col = 8, xlab = "Orientation4",
main = "Hist of Orientation4",
```

```
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation4,na.rm = T))
abline(v = mean(plt$Orientation4,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation4,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation4,na.rm = T),
sd=sd(plt$Orientation4,na.rm = T)),
     col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))


# 3. Close the file
dev.off()

########## Orientation5
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation5_15.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation5, prob = T, breaks = 15, col = 8, xlab = "Orientation5",
main = "Hist of Orientation5",
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation5,na.rm = T))
abline(v = mean(plt$Orientation5,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation5,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation5,na.rm = T),
sd=sd(plt$Orientation5,na.rm = T)),
     col='green', lwd=2, add=TRUE, yaxt="n")
```

```
legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))



# 3. Close the file
dev.off()

########## Orientation6
##########################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation6.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation6, prob = T, breaks = 15, col = 8, xlab = "Orientation6",
main = "Hist of Orientation6",
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation6,na.rm = T))
abline(v = mean(plt$Orientation6,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation6,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation6,na.rm = T),
sd=sd(plt$Orientation6,na.rm = T)),
     col='green', lwd=2, add=TRUE, yaxt="n")



legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()
```

```
########## Orientation7
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation7.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation7, prob = T, breaks = 15, col = 8, xlab = "Orientation7",
main = "Hist of Orientation7",
     cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation7,na.rm = T))
abline(v = mean(plt$Orientation7,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation7,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation7,na.rm = T),
sd=sd(plt$Orientation7,na.rm = T)),
      col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))



# 3. Close the file
dev.off()

########## Orientation8
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation8.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation8, prob = T, breaks = 15, col = 8, xlab = "Orientation8",
main = "Hist of Orientation8",
```

```
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation8,na.rm = T))
abline(v = mean(plt$Orientation8,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation8,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation8,na.rm = T),
sd=sd(plt$Orientation8,na.rm = T)),
     col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()

########## Orientation9
##############################################################
##########

# 1. Open jpeg file
jpeg("Hist_Orientation9.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Orientation9, prob = T, breaks = 15, col = 8, xlab = "Orientation9",
main = "Hist of Orientation9",
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Orientation9,na.rm = T))
abline(v = mean(plt$Orientation9,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Orientation9,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Orientation9,na.rm = T),
sd=sd(plt$Orientation9,na.rm = T)),
     col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
```

```
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()



########## Leaf.weight
##########################################################
##########

# 1. Open jpeg file
jpeg("Hist_Leaf.weight.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Leaf.weight, prob = T, breaks = 15, col = 8, xlab = "Leaf.weight", main
= "Hist of Leaf.weight",
     cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Leaf.weight,na.rm = T))
abline(v = mean(plt$Leaf.weight,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Leaf.weight,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Leaf.weight,na.rm = T), sd=sd(plt$Leaf.weight,na.rm
= T)),
      col='green', lwd=2, add=TRUE, yaxt="n")



legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()
```

```
########## LeafArea
##########################################################
##########

# 1. Open jpeg file
jpeg("Hist_LeafArea.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$LeafArea, prob = T, breaks = 15, col = 8, xlab = "LeafArea", main =
"Hist of LeafArea",
     cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$LeafArea,na.rm = T))
abline(v = mean(plt$LeafArea,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$LeafArea,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$LeafArea,na.rm = T), sd=sd(plt$LeafArea,na.rm =
T)),
      col='green', lwd=2, add=TRUE, yaxt="n")



legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()



########## Leaf.Hue
##########################################################
##########

# 1. Open jpeg file
jpeg("Hist_Leaf.Hue.jpg", width = 800, height = 600)
# 2. Create the plot
hist(plt$Leaf.Hue, prob = T, breaks = 15, col = 8, xlab = "Leaf.Hue", main =
"Hist of Leaf.Hue",
```

```
    cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5)
lines(density(plt$Leaf.Hue,na.rm = T))
abline(v = mean(plt$Leaf.Hue,na.rm = T), col = "blue", lwd = 2)
abline(v = median(plt$Leaf.Hue,na.rm = T), col = "red", lwd = 2)
curve(dnorm(x, mean(plt$Leaf.Hue,na.rm = T), sd=sd(plt$Leaf.Hue,na.rm =
T)),
     col='green', lwd=2, add=TRUE, yaxt="n")


legend(x = "topright", # location of legend within plot area
       c("Density plot", "Standard Normal plot","Mean", "Median"),
       col = c("black", "green", "blue",'red'),
       lwd = c(2, 2, 2))

# 3. Close the file
dev.off()




################################  1-b-i
##############################################################
############


# use is "complete.obs" then missing values are handled by casewise deletion.
cor_o1_o7 = cor(plt$Orientation1, plt$Orientation7,use = "complete.obs")
cor_mass_o0 = cor(plt$Mass, plt$Orientation0,use = "complete.obs")
cor_o7_o8 = cor(plt$Orientation7, plt$Orientation8,use = "complete.obs")
sprintf('correlation_o1_o7 is %f, correlation_mass_o0 is %f, and
correlation_o7_o8 is %f.',cor_o1_o7,cor_mass_o0,cor_o7_o8)
# Output: "correlation_o1_o7 is -0.869672, correlation_mass_o0 is -0.087690,
and correlation_o7_o8 is 0.929113."
```

```
################################  1-b-ii & iii
#############################################################
#######

library(GGally)



########## Step 1: Get the sub dataset and rename the column:
#################################

subplt = cbind(plt[2:16],plt[18:19])
names(subplt) =
c('Cx','Cy','Ms','Wd','Dh','Or0','Or1','Or2','Or3','Or4','Or5','Or6','Or7','Or8','Or9','L
f_A','Lf_H')




########## Step 2-1: Visualise the correlation between variables
#################################
gg_cor1 = ggcorr(subplt, nbreaks = 6,
                 label = T, label_round = 5,label_alpha = F,
                 hjust = 1, size = 8)  +
  ggtitle('Figure 19: Correlations of Variables') +
  theme(plot.title = element_text(size = 30, face = "bold"))

print(gg_cor1)
ggsave('1-b-ggcor1.jpg')




########## Step 2-2: Find the least & most correlated ones
#################################

gg_cor2 = ggcorr(subplt, geom = "blank",
                 label = TRUE, label_round = 5, label_alpha = F,
                 hjust = 0.75, size = 8 )
```

```
gg_cor3 = gg_cor2 +
  geom_point(size = 20, aes(alpha = abs(coefficient) > 0.92, color ='tomato'))
+
  geom_point(size = 20, aes(alpha = abs(coefficient) < 0.01)) +
  scale_alpha_manual(values = c("TRUE" = 0.25, "FALSE" = 0)) +
  guides(color = FALSE, alpha = FALSE)

gg_cor4 = gg_cor3 + ggtitle('Figure 20: The Most & Least Correlated Variables')
+
  theme(plot.title = element_text(size = 30, face = "bold"))



print(gg_cor4)
ggsave('1-b-ggcor4.jpg')




########### Find the high correlated ones
#####################################

gg_cor5 = ggcorr(subplt, geom = "blank",
                 label = TRUE, label_round = 3, label_alpha = F,
                 hjust = 0.75, size = 5 )

gg_cor6 = gg_cor5 +
  geom_point(size = 10, aes(alpha = abs(coefficient) > 0.5, color ='tomato')) +
  scale_alpha_manual(values = c("TRUE" = 0.25, "FALSE" = 0)) +
  guides(color = FALSE, alpha = FALSE)

gg_cor7 = gg_cor6 + ggtitle('Figure 25: The Correlation between Variables') +
  theme(plot.title = element_text(size = 30, face = "bold"))



print(gg_cor7)
ggsave('1-b-ggcor7.jpg')
```

```
########## Reference
####################################

# 1. GGally package: Extension tp ggplot2 for [ correlation matrix and survival plots ]
# http://www.sthda.com/english/wiki/ggally-r-package-extension-to-ggplot2-for-correlation-matrix-and-survival-plots-r-software-and-data-visualization

# 2. ggcor()  # Good one !!!!
# https://briatte.github.io/ggcorr/



###############  1-b-iv
#################################################################################################################################
#####################


############ Step 0: load the dataset
#################################################################################
#####

plt <- read.table(file = "g54dma-plant-dataset.csv",header = TRUE,sep = ',')


############ Step 1: Use different colors
#############################################


plotvar = plt$Class  # pick a variable to plot for color
nclr = 5  # number of colors
# plotclr = brewer.pal(nclr,"PuBu") # get the colors
plotclr = c("#FA8072","#FFD700","#3CB371","#8A2BE2","#045A8D")
# the code of color: red, yellow, seagreen, blueviolet , darkblue
```

```
colornum = cut(rank(plotvar), nclr, labels=FALSE)
colcode = plotclr[colornum] # assign color



# Alternative for Step 1 :

# colors <- c("#F1EEF6","#BDC9E1","#74A9CF","#2B8CBE","#045A8D")
# colors <- colors[as.numeric(plt$Class)]
# scatterplot3d( plt$Orientation2, plt$LeafArea, plt$Depth, pch = 16,
color=colors)



############ Step 2: Several ways
##############################################################
#####

##############################################
ggplot2
##############################################

library(ggplot2)

gg_class_Ori2 =
  ggplot(plt, aes(x = Class, y = Orientation2, color = Class)) +
  geom_point(size = 5, stroke = 1.5, alpha = 1/2, shape = 11 ) +
  ggtitle("Class and Orientation2")+
  theme(plot.title = element_text(size = 20, face = "bold"))

gg_class_Depth = ggplot(plt, aes(x = Class, y = Depth,color = Class)) +
  geom_point(size = 5, stroke = 1.5, alpha = 1/2, shape = 11 ) +
  ggtitle("Class and Depth")+
  theme(plot.title = element_text(size = 20, face = "bold"))

gg_class_Area = ggplot(plt, aes(x = Class, y = LeafArea, color = Class)) +
```

```
  geom_point(size = 5, stroke = 1.5, alpha = 1/2, shape = 11 ) +
  ggtitle("Class and Area") +
  theme(plot.title = element_text(size = 20, face = "bold"))

gg_class_Mass = ggplot(plt, aes(x = Class, y = Mass, color = Class)) +
  geom_point(size = 5, stroke = 1.5, alpha = 1/2, shape = 11 ) +
  ggtitle("Class and Mass") +
  theme(plot.title = element_text(size = 20, face = "bold"))
gg_class_Mass

gg_class_Cy = ggplot(plt, aes(x = Class, y = CentroidY, color = Class)) +
  geom_point(size = 5, stroke = 1.5, alpha = 1/2, shape = 11 ) +
  ggtitle("Class and CentroidY") +
  theme(plot.title = element_text(size = 20, face = "bold"))
gg_class_Cx

gg_class_Cx = ggplot(plt, aes(x = Class, y = CentroidX, color = Class)) +
  geom_point(size = 5, stroke = 1.5, alpha = 1/2, shape = 11 ) +
  ggtitle("Class and CentroidX") +
  theme(plot.title = element_text(size = 20, face = "bold"))

gg_class_Hue = ggplot(plt, aes(x = Class, y = Leaf.Hue, color = Class)) +
  geom_point(size = 5, stroke = 1.5, alpha = 1/2, shape = 11 ) +
  ggtitle("Class and Leaf.Hue") +
  theme(plot.title = element_text(size = 20, face = "bold"))

print(gg_class_Ori2)
ggsave("1-b-iv-gg_class_Ori2.jpg")

print(gg_class_Depth)
ggsave('1-b-iv-gg_class_Depth.jpg')

print(gg_class_Area)
ggsave('1-b-iv-gg_class_Area.jpg')
```

```
print(gg_class_Mass)
ggsave('1-b-iv-gg_class_Mass.jpg')

print(gg_class_Cy)
ggsave('1-b-iv-gg_class_Cy.jpg')

print(gg_class_Cx)
ggsave('1-b-iv-gg_class_Cx.jpg')

print(gg_class_Hue)
ggsave('1-b-iv-gg_class_Hue.jpg')



##############################################
scatterplot matrixs
##############################################

#  Basic scatterplot matrix of the four measurements
# E.g - Basic R : pairs(~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,
data=iris)
# Similar plot using ggplot2
# plotmatrix(with(iris, data.frame(Sepal.Length, Sepal.Width, Petal.Length)))
# plotmatrix didn't exist any more!
# dev.off()

Orientation2 = plt$Orientation2
Depth = plt$Depth
LeafArea = plt$LeafArea
sub_plt = cbind(Orientation2, Depth, LeafArea)

pairs(sub_plt, col=colcode)



dev.off()

??pairs
```

```
####### 3D scatterplots by scatterplot3d
#############################################################

library(scatterplot3d)

x = plt$Orientation2
y = plt$LeafArea
z = plt$Depth



# 1. Open jpeg file
jpeg("1-b-iv-3Dscatterplot-classes.jpg", width = 800, height = 600)
jpeg("1-b-iv-3Dscatterplot-classes2.jpg", width = 800, height = 600)

# scatter plot
s3d = scatterplot3d(x, y, z, angle= 45, type='h',color=colcode, pch = 1,
                    grid=TRUE, box=FALSE,
                    xlab = 'Orientation2', ylab = 'LeafArea', zlab = 'Depth',
                    main = "Relationships: Class - Orientation2 / Depth / LeafArea")

# when [ type = 'h' ] , there will have a line to the botton for each points.

# add legend for each colors / classes. the xyz.convert() indicates the location.
legend(s3d$xyz.convert(0.17,6000,800), legend = levels(plt$Class),
       col =  plotclr, pch = 1, xpd = TRUE, horiz = TRUE)
# }

# Don't Run this part
# Add legend in another way
legend("right", legend = levels(plt$Class),
       col = plotclr, pch = 1, inset = 0.1,xpd = TRUE, horiz = TRUE)



# 3. Close the file
```

```
dev.off()



################################  1-d
#########################################################
#################

# call the function

head(plt)
# Replace NA by O
plt_NA_0 = NA_by_0(plt[2:19])
plt_NA_0$Class = plt$Class
write.csv(plt_NA_0,file = 'plt_NA_0.csv')
sum_plt_0 = descriptive(plt_NA_0[,1:18])
sum_plt_0[9,] # 0
write.csv(sum_plt_0,file = 'sum_plt_0.csv')

# Replace NA by mean
plt_NA_mean = NA_by_mean(plt[2:19])
plt_NA_mean$Class = plt$Class
write.csv(plt_NA_mean,file = 'plt_NA_mean.csv')
sum_plt_mean = descriptive(plt_NA_mean[,1:18])
sum_plt_mean[9,]#0
write.csv(sum_plt_mean,file = 'sum_plt_mean.csv')

# Replace NA by mean
plt_NA_median = NA_by_median(plt[2:19])
plt_NA_median$Class = plt$Class
write.csv(plt_NA_median,file = 'plt_NA_median.csv')
sum_plt_median = descriptive(plt_NA_median[,1:18])
sum_plt_median[9,]#0
write.csv(sum_plt_median,file = 'sum_plt_median.csv')
```

```
###############  1-e Transforming Data from Lec 5
#################################################################
####
###############  1-e-1 Mean Centring
#################################################################
##############

# Call the function
mc_0 = mean_centre(plt_NA_0[,1:18])
boxplot(mc_0)
mc_mean = mean_centre(plt_NA_mean[,1:18])
boxplot(mc_mean)



boxplot(mc_0[,16])
boxplot(mc_mean[,16:18])
boxplot(mc_0[,16],xlab = 'Leaf.Weight' )
boxplot(plt[,18] )
boxplot(mc_0[,18],xlab = 'LeafArea' )




boxplot(mc_mean[,6:16])
boxplot(mc_mean[,17],xlab = 'LeafArea' )
boxplot(mc_mean[,18],xlab = 'LeafHue' )

###############  1-e-2 Standardisation (Z-Score)
#################################################################
##############

# Call the function

stdd_median = standard_z(plt_NA_median[,1:18])
boxplot(stdd_median)
stdd_mean = standard_z(plt_NA_mean[,1:18])
boxplot(stdd_mean)
stdd_0 = standard_z(plt_NA_0[,1:18])
```

```
boxplot(stdd_0)

# plt_0_scale = scale(plt_NA_0[2:19])
# plt_mean_scale = scale(plt_NA_median[2:19])
# plt_median_scale = scale(plt_NA_mean[2:19])



################  1-e-3 Normalizaton (Min-Max)
##################################################################
###############



# Call the function
normal_median =normal_min_max(plt_NA_median[,1:18])
boxplot(normal_median)
normal_mean =normal_min_max(plt_NA_mean[,1:18])
boxplot(normal_mean)
normal_0 =normal_min_max(plt_NA_0[,1:18])
boxplot(normal_0)




################  1-f-i
##################################################################
##################################################
# Reference
# https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-
imputing-missing-values/
# https://statisticalhorizons.com/predictive-mean-matching

install.packages('mice')
library(mice)
library(dplyr)

# got the number of Na in each row
plt$na_count <- apply(is.na(plt), 1, sum)
```

```
plt$na_count
hist(plt$na_count)
# Delete some instances with NAs >=2
sub_plt = plt %>% filter(plt$na_count < 2)
nrow(sub_plt)

# got the number of Na in each col



# Delete the attribute with 50% NAs
sub_plt = subset(sub_plt, select = -Leaf.weight)
analysis(sub_plt[,2:18])[9,]

write.csv(sub_plt,file = 'sub_plt_fi1.csv')

analysis(sub_plt[,2:18])[9,]

# using pmm methods to replace NAs

subA = filter(sub_plt, Class=='A')
nrow(subA) #108
impA <- mice(subA, method = 'pmm')
impA <- complete(impA,2)

subB = filter(sub_plt, Class=='B')
impB <- mice(subB, method = 'pmm')
impB <- complete(impB,2)
nrow(subB) #119

subC = filter(sub_plt, Class=='C')
impC <- mice(subC, method = 'pmm')
impC <- complete(impC,2)
nrow(subC) #135

subD = filter(sub_plt, Class=='D')
```

```
impD <- mice(subD, method = 'pmm')
impD <- complete(impD,2)
nrow(subD) # 134

subE = filter(sub_plt, Class=='E')
impE <- mice(subE, method = 'pmm')
impE <- complete(impE,2)
nrow(subE) # 148

imp_plt = rbind(impA,impB,impC,impD,impE)
ncol(imp_plt)
analysis(imp_plt[,2:18])[9,]

imp_plt = subset(imp_plt, select = -na_count)
head(imp_plt)

write.csv(imp_plt,file = 'imp_plt.csv')

################  1-f-ii
##########################################################
##########################################################

uncor_f = subset(plt, select = -Leaf.weight)
uncor_f = subset(uncor_f, select = -Orientation0)
uncor_f = subset(uncor_f, select = -Orientation1)
uncor_f = subset(uncor_f, select = -Orientation5)
uncor_f = subset(uncor_f, select = -Orientation6)

uncor_f = subset(uncor_f, select = -Orientation3)
uncor_f = subset(uncor_f, select = -Orientation8)
uncor_f = subset(uncor_f, select = -Depth)

uncor_f = subset(uncor_f, select = -Orientation4)
uncor_f = subset(uncor_f, select = -Orientation7)

ncol(uncor_f)
```

```
head(uncor_f)

gg_cor = ggcorr(uncor_f[2:9], geom = "blank",
              label = TRUE, label_round = 3, label_alpha = F,
              hjust = 0.75, size = 5 )

gg_cor = gg_cor +
  geom_point(size = 10, aes(alpha = abs(coefficient) > 0.5, color ='tomato')) +
  scale_alpha_manual(values = c("TRUE" = 0.25, "FALSE" = 0)) +
  guides(color = FALSE, alpha = FALSE)

gg_cor = gg_cor + ggtitle('Figure 28: Uncorrelated Attributes') +
  theme(plot.title = element_text(size = 40, face = "bold"))



print(gg_cor)
ggsave('1-f-ggcor-3.jpg')



sum_uncor_f = analysis(uncor_f[2:9])
sum_uncor_f[9,]

# delete the NA by na.omit()
plt_uncor_noNA = na.omit(uncor_f)
plt_uncor_mean = NA_by_mean(uncor_f)
plt_uncor_median = NA_by_median(uncor_f)

# check if NAs have been deleted seccussfully, and output the dataset
nrow(plt_uncor_noNA)  # return 679 (instances)
analysis(plt_uncor_noNA[2:9])[9,] # returns all 0

write.csv(plt_uncor_noNA,file = 'plt_uncor_omit.csv')
write.csv(plt_uncor_median,file = 'plt_uncor_median.csv')
write.csv(plt_uncor_mean,file = 'plt_uncor_mean.csv')
```

```
###############  1-f-iii
#########################################################
#########################################
```

```
plt_f_iii = plt_NA_mean[2:20]
head(plt_f_iii)

# 4. Before applying PCA: we must standardize our variables with scale()
function:
plt.stand = as.data.frame(scale(plt_f_iii[1:18])) # 'standardize' means, (it -
average value) / sd
head(plt.stand)
sapply(plt.stand,sd) #now, standard deviations are 1
sapply(plt.stand,mean) #now, mean should be 0 (or very very close to 0)

plt_pca = prcomp(plt.stand,scale=T)
summary(plt_pca)

# 1. Open jpeg file
jpeg("plt_pca.jpg", width = 800, height = 600)
# 2. Create the plot
screeplot(plt_pca, type="lines",col=2, main="Variance explained by PC")
title(xlab="Principal Components")

# 3. Close the file
dev.off()

plt_pca_7= plt_pca$x[,1:7]
plt_pca_10= plt_pca$x[,1:10]
pca_10 = data.frame(plt_pca_10)
pca_10$Class = plt_f_iii$Class
```

2. All Codes of Part 2-3 in R

```
library(ggplot2)
library(dplyr)
library(GGally)
source('Normal_Standard_Mean.R')
source('summary.R')

plt = read.table(file = "g54dma-plant-dataset.csv",header = TRUE,sep = ',')

imp_plt =  read.table(file = "imp_plt.csv",header = TRUE,sep = ',')

################################### Part 2 Clustering data
#########################################

###### Before clustering, I did further pre-process to the dataset
'imp_plt' (from 1-a-fi) ###################################

# 1: delete the high correlated columns
#########################################
imp_plt1 = subset(imp_plt, select = -Orientation0)
imp_plt1 = subset(imp_plt1, select = -Orientation1)
imp_plt1 = subset(imp_plt1, select = -Orientation5)
imp_plt1 = subset(imp_plt1, select = -Orientation6)
imp_plt1 = subset(imp_plt1, select = -Orientation3)
imp_plt1 = subset(imp_plt1, select = -Orientation8)
imp_plt1 = subset(imp_plt1, select = -Orientation4)
imp_plt1 = subset(imp_plt1, select = -Orientation7)
imp_plt1 = subset(imp_plt1, select = -Depth)
imp_plt2 = imp_plt1[4:12]
ncol(imp_plt2)
head(imp_plt2)

# 2: delete the outliers
#########################################
```

```
round(descriptive(imp_plt2[,1:8]),3)

# choose attributes with highest sd, 'LeafArea'
outliers <- boxplot(imp_plt2[,7])$out # get the outliers
imp_plt3 <- imp_plt2[-which(imp_plt2[,7] %in% outliers),] #remove the rows containing the outliers
round(descriptive(imp_plt3[,1:8]),3)
nrow(imp_plt3) #595

# check the outliers of each attribute
boxplot(imp_plt3[,1])$out # the outliers are near the Q3+1.5*IQR, so I kepted it.
outliers <- boxplot(imp_plt3[,1])$out # get the outliers
imp_plt4 <- imp_plt3[-which(imp_plt3[,1] %in% outliers),] #remove the rows containing the outliers
round(descriptive(imp_plt4[,1:8]),1)
nrow(imp_plt4) #589

boxplot(imp_plt4[,2])$out # without outliers
boxplot(imp_plt4[,3])$out
outliers <- boxplot(imp_plt4[,3])$out # get the outliers
imp_plt5 <- imp_plt4[-which(imp_plt4[,3] %in% outliers),] #remove the rows containing the outliers
round(descriptive(imp_plt5[,1:8]),3)
nrow(imp_plt5) #586

boxplot(imp_plt5[,4])$out  # without outliers
boxplot(imp_plt5[,5])$out  # without outliers
boxplot(imp_plt5[,6])$out  # without outliers
boxplot(imp_plt5[,8])$out  # without outliers
outliers <- boxplot(imp_plt5[,8])$out # get the outliers
imp_plt6 <- imp_plt5[-which(imp_plt5[,8] %in% outliers),] #remove the rows containing the outliers
boxplot(imp_plt6[,8])$out
outliers <- boxplot(imp_plt6[,8])$out # only 1 outliers and deleted it
```

```
imp_plt6 <- imp_plt6[-which(imp_plt6[,8] %in% outliers),]
round(descriptive(imp_plt6[,1:8]),3)
nrow(imp_plt6) #569



# 3: Finally, Standardized the dataset by the function written
before.########################################
stdd_plt = standard_z(imp_plt6[,1:8])
stdd_plt$Class = imp_plt6$Class
boxplot(stdd_plt[,1:8])
head(stdd_plt) # 8 attributes
ncol(stdd_plt) # 9
nrow(stdd_plt) # 569 instances

write.csv(stdd_plt,file='stdd.csv')



###################################### Class itself
##############################################
########################

plotclr = rainbow(5)
# the code of color: red, yellow, seagreen, blueviolet , darkblue
colornum = cut(rank(stdd_plt$Class), 5)
colcode = plotclr[colornum] # assign color

jpeg("Figure 34 Classes in Attributes.jpg", width = 800, height = 700)
pairs(stdd_plt[,1:8],col=colcode,oma=c(5,5,5,15),
      main = 'Figure 34 Classes in Attributes (dataset: stdd_plt)',
      cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5) #shows pairs for all
clusters/attributes
par(xpd = TRUE)
legend("bottomright",
       fill = rainbow(5),
       legend = c("A", "B","C", "D",'E'))
```

```
dev.off()



################## A. CLUSTERING METHODS: 层次聚类算法
##########################################################################
#########################

# IMPORTANT: Clustering is an Unsupervised method. Therefore, label/class
attribute cannot be used!

####################################
# A.1. HCA: Hierarchical clustering:
hc = hclust(dist(stdd_plt[,1:8])) #applies hierarchical clustering
plot(hc) #shows the whole hierarchy
stdd_plt$HC5 = cutree(hc,5) #stops hierarchy at level 5 and saves it in
stdd_plt$HC5
stdd_plt$HC5

##########################################################################
##########
plotclr = rainbow(5)
# the code of color: red, yellow, seagreen, blueviolet , darkblue
colornum = cut(rank(stdd_plt$HC5), 5)
colcode = plotclr[colornum] # assign color
##########################################################################
##########

jpeg("Figure 31 Results of HCA.jpg", width = 800, height = 700)
pairs(stdd_plt[,1:8],col=colcode,oma=c(5,5,5,15),
      main = 'Figure 31 Results of HCA (dataset: stdd_plt)',
      cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5) #shows pairs for all
clusters/attributes
par(xpd = TRUE)
legend("bottomright",
       fill = rainbow(5),
       legend = c("1", "2","3", "4",'5'))
dev.off()
```

```
################## A.2. K-means clustering 均值聚类算法
##########################################################
##########################################################
####################

# Note: we are not giving the kmeans the initial means! As a consequence,
these will be chosen randomly each time. So, the results of K-means might be
different each time!
km5 = kmeans(stdd_plt[,1:8],5,iter.max=100) #applies k-means with 3 clusters
and 100 iterations
km5



stdd_plt$KM5= km5$cluster #saves clusters in stdd_plt$KM3
# IMPORTANT: Check the value of stdd_plt$KM3. If you notice, the clustering
result is just a list of the cluster each point belongs to, nothing more!
stdd_plt$KM5
stdd_plt$Class

##########################################################
###########
plotclr = rainbow(5)
# the code of color: red, yellow, seagreen, blueviolet , darkblue
colornum = cut(rank(stdd_plt$KM5), 5)
colcode = plotclr[colornum] # assign color
##########################################################
###########

jpeg("Figure 32 Results of K-Means.jpg", width = 800, height = 700)
pairs(stdd_plt[,1:8],col=colcode,oma=c(5,5,5,15),
      main = 'Figure 32 Results of K-Means (dataset: stdd_plt)',
      cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5) #shows pairs for all
clusters/attributes
par(xpd = TRUE)
legend("bottomright",
       fill = rainbow(5),
```

```
      legend = c("1", "2","3", "4",'5'))
dev.off()



################## A.3. PAM clustering - Partitioning Around Medoids 中心点算法
##################################################################################################################################

# A.3. PAM clustering:
library(cluster)

PAM5 = pam(stdd_plt[,1:8], 5)

#Similarly to k-means, PAM also returns several values:
PAM5

stdd_plt$PAM5 = PAM5$clustering #Saves clustering result only



##################################################################################################################################
plotclr = rainbow(5)
# the code of color: red, yellow, seagreen, blueviolet , darkblue
colornum = cut(rank(stdd_plt$PAM5), 5)
colcode = plotclr[colornum] # assign color
##################################################################################################################################

jpeg("Figure 33 Results of PAM.jpg", width = 800, height = 700)
pairs(stdd_plt[,1:8],col=colcode,oma=c(5,5,5,15),
      main = 'Figure 33 Results of PAM (dataset: stdd_plt)',
      cex.lab=1.5, cex.axis=1.5, cex.main=2, cex.sub=1.5) #shows pairs for all clusters/attributes
par(xpd = TRUE)
legend("bottomright",
```

```
      fill = rainbow(5),
      legend = c("1", "2","3", "4",'5'))
dev.off()



############ Compare the results ######################

t1= table(stdd_plt$Class,stdd_plt$HC5)
t1
t2 = table(stdd_plt$Class,stdd_plt$KM5)
t2
t3 = table(stdd_plt$Class, stdd_plt$PAM5)
t3

#################################################################
#################################################################
max_diag = function(x){
 final = x
 d = x
 for(i in 1:nrow(x)){
  a = which(d == max(d),arr.ind = TRUE)
  a = a[-2,]
  final[a[2],] = x[a[1],]
  d[a[1],] = -100
  d[,a[2]] = -100
 }

 return(final)
}
#################################################################
#################################################################
performance = function(x){
 r = max_diag(x)
 TruePositive = sum(diag(r))/sum(r)
```

```
  col_max = apply(r, 2, max)
  Purity = sum(col_max)/sum(r)
  performance = c(TruePositive,Purity)

  return(performance)
}

##################################################################
##################################################################
r1 = max_diag(t1)
r1
r2 = max_diag(t2)
r2
r3 = max_diag(t3)
write.csv(t1,'t1.csv')
write.csv(r1,'r1.csv')
write.csv(r2,'r2.csv')
write.csv(r3,'r3.csv')

result = rbind(c('True Positive Rate','Purity'), round(performance(t1),
4),round(performance(t2),4),round(performance(t3),4))
rownames(result) = c(' ','HCA','K-Means','PAM')

##################################################################
##################################################################
#### 2-B
##################################################################
##################################################################

##################################################################
##################################################################
##### Change Parameters of HCA##########################

#1 hc-default = complete
result_hca = performance(t1)

#2 hc-ward
```

```
hc= hclust(dist(stdd_plt[,1:8]),method = "ward")
stdd_plt$HC5= cutree(hc,5)
result_hca =
rbind( result_hca,performance(table(stdd_plt$Class,stdd_plt$HC5)))

#3 hc-single
hc= hclust(dist(stdd_plt[,1:8]),method = "single")
stdd_plt$HC5= cutree(hc,5)
result_hca =
rbind( result_hca,performance(table(stdd_plt$Class,stdd_plt$HC5)))

#4 hc-median
hc= hclust(dist(stdd_plt[,1:8]),method = "median")
stdd_plt$HC5= cutree(hc,5)
result_hca =
rbind( result_hca,performance(table(stdd_plt$Class,stdd_plt$HC5)))

#5 hc-average
hc= hclust(dist(stdd_plt[,1:8]),method = "average")
stdd_plt$HC5= cutree(hc,5)
result_hca =
rbind( result_hca,performance(table(stdd_plt$Class,stdd_plt$HC5)))

#6 hc-mcquitty
hc= hclust(dist(stdd_plt[,1:8]),method = "mcquitty")
stdd_plt$HC5= cutree(hc,5)
result_hca =
rbind( result_hca,performance(table(stdd_plt$Class,stdd_plt$HC5)))

#7 hc-centroid
hc= hclust(dist(stdd_plt[,1:8]),method = "centroid")
stdd_plt$HC5= cutree(hc,5)
result_hca =
rbind(result_hca,performance(table(stdd_plt$Class,stdd_plt$HC5)))

###################################
# name the column & row
```

```
colnames(result_hca) = c('TruePositive','Purity')
rownames(result_hca)=
c('complete','ward','single','median','average','maquitty','centroid')
write.csv(result_hca,'results_7methods_HCA.csv')



##################################################################
#####################################
##### Change Parameters of K-Means
##################################################################
####

#km-Hartigan-Wong
km5_1= kmeans(stdd_plt[,1:8],5,iter.max=20,algorithm = c("Hartigan-Wong"))
stdd_plt$KM5_1= km5_1$cluster
result_km_al1 = performance(table(stdd_plt$Class,stdd_plt$KM5_1))

km5_2= kmeans(stdd_plt[,1:8],5,iter.max=20,algorithm = c("Forgy"))
stdd_plt$KM5_2= km5_2$cluster
result_km_al2 = performance(table(stdd_plt$Class,stdd_plt$KM5_2))

km5_3= kmeans(stdd_plt[,1:8],5,iter.max=20,algorithm = c("MacQueen"))
stdd_plt$KM5_3= km5_3$cluster
result_km_al3 = performance(table(stdd_plt$Class,stdd_plt$KM5_3))

iter_step = seq(40, 2000, by = 20)
iter_step

for (i in iter_step){
  #####
  km5_1= kmeans(stdd_plt[,1:8],5,iter.max=i,algorithm = c("Hartigan-Wong"))
  stdd_plt$KM5_1= km5_1$cluster
  result_km_al1 = rbind(result_km_al1,
performance(table(stdd_plt$Class,stdd_plt$KM5_1)))
  #####
  km5_2= kmeans(stdd_plt[,1:8],5,iter.max=i,algorithm = c("Forgy"))
```

```
  stdd_plt$KM5_2= km5_2$cluster
  result_km_al2 = rbind(result_km_al2,
performance(table(stdd_plt$Class,stdd_plt$KM5_2)))
  #####
  km5_3= kmeans(stdd_plt[,1:8],5,iter.max=i,algorithm = c("MacQueen"))
  stdd_plt$KM5_3= km5_3$cluster
  result_km_al3 = rbind(result_km_al3,
performance(table(stdd_plt$Class,stdd_plt$KM5_3)))
}

  ###################################
  km_al1 = data.frame(result_km_al1)
  km_al2 = data.frame(result_km_al2)
  km_al3 = data.frame(result_km_al3)
  #####
  km_TP = cbind(km_al1[,1], km_al2[,1], km_al3[,1])
  colnames(km_TP) = c('HartiganWong','Forgy','MacQueen')
  rownames(km_TP) = seq(20, 2000, by = 20)
  km_TP = data.frame(km_TP)

  #####
  km_Purity = cbind(km_al1[,2], km_al2[,2], km_al3[,2])
  colnames(km_Purity) = c('HartiganWong','Forgy','MacQueen')
  rownames(km_Purity) = seq(20, 2000, by = 20)
  km_Purity = data.frame(km_Purity)

  ggplot(km_TP, aes(x=seq(20, 2000, by = 20))) +
    geom_line(aes(y = HartiganWong, colour = "Hartigan-Wong"))+
    geom_line(aes(y = Forgy, colour = "Forgy"))+
    geom_line(aes(y = MacQueen, colour = "MacQueen"))+
    labs(y='Ture Positive Rate', x = 'iter.max')+
    ggtitle('Ture Positive Rate of K-Means')

  ggplot(km_Purity, aes(x=seq(20, 2000, by = 20))) +
    geom_line(aes(y = HartiganWong, colour = "Hartigan-Wong"))+
    geom_line(aes(y = Forgy, colour = "Forgy"))+
```

```
    geom_line(aes(y = MacQueen, colour = "MacQueen"))+
    labs(y='Purity', x = 'iter.max')+
    ggtitle('Purity of K-Means')

sd(km_TP$HartiganWong)
mean(km_TP$HartiganWong)
sd(km_TP$Forgy)
mean(km_TP$Forgy)
sd(km_TP$MacQueen)
mean(km_TP$MacQueen)

sd(km_Purity$HartiganWong)
mean(km_Purity$HartiganWong)
sd(km_Purity$Forgy)
mean(km_Purity$Forgy)
sd(km_Purity$MacQueen)
mean(km_Purity$MacQueen)



##############################################################
##########################################
##############################################################
##########################################

## no_build_ *and* no_swap_ phase: just cluster
pam5_nn = pam(stdd_plt[,1:8], 5, medoids = c(1,2,3,4,5), do.swap = FALSE)
stdd_plt$PAM5_nn = pam5_nn$cluster
result1_pam = performance(table(stdd_plt$Class,stdd_plt$PAM5_nn))

## yes_build_ *and* no_swap_ phase
pam5_yn = pam(stdd_plt[,1:8], 5, medoids = NULL, do.swap = FALSE)
stdd_plt$PAM5_yn = pam5_yn$cluster
result1_pam = rbind(result1_pam,
performance(table(stdd_plt$Class,stdd_plt$PAM5_yn)))

## no_build_ *and* yes_swap_ phase
```

```
pam5_ny = pam(stdd_plt[,1:8], 5, medoids = c(1,2,3,4,5), do.swap = TRUE)
stdd_plt$PAM5_ny = pam5_ny$cluster
result1_pam = rbind(result1_pam,
performance(table(stdd_plt$Class,stdd_plt$PAM5_ny)))

## yes_build_ *and* yes_swap_ phase
pam5_ny = pam(stdd_plt[,1:8], 5, medoids = NULL, do.swap = TRUE)
stdd_plt$PAM5_ny = pam5_ny$cluster
result1_pam = rbind(result1_pam,
performance(table(stdd_plt$Class,stdd_plt$PAM5_ny)))



###################################
# name the column & row
colnames(result1_pam ) = c('TruePositive','Purity')
rownames(result1_pam )=
c('N_N_build&swap','Y_N_build&swap','N_Y_build&swap','Y_Y_build&swap')
result1_pam
write.csv(result1_pam ,'results_medioid&do.swap_PMA.csv')

#################################################################
#####################################

nostdd_plt = imp_plt6

## yes_build_ *and* yes_swap
## 'euclidean' & stand = True ##############################
pam5_ny_et = pam(nostdd_plt[,1:8], 5,
           metric = 'euclidean', stand = TRUE)
nostdd_plt$PAM5_ny_et = pam5_ny_et$cluster
result2_pam = performance(table(nostdd_plt$Class,nostdd_plt$PAM5_ny_et))

## 'euclidean' & stand = False ##############################
pam5_ny_ef = pam(nostdd_plt[,1:8], 5,
             metric = 'euclidean', stand = FALSE)
nostdd_plt$PAM5_ny_ef = pam5_ny_ef$cluster
```

```
result2_pam = rbind(result2_pam,
performance(table(nostdd_plt$Class,nostdd_plt$PAM5_ny_ef)))

##  'manhattan' & stand = True
#############################
pam5_ny_mt = pam(nostdd_plt[,1:8], 5,
             metric = 'manhattan', stand = TRUE)
nostdd_plt$PAM5_ny_mt = pam5_ny_mt$cluster
result2_pam = rbind(result2_pam,
performance(table(nostdd_plt$Class,nostdd_plt$PAM5_ny_mt)))

##  'manhattan' & stand = False
#############################
pam5_ny_mf = pam(nostdd_plt[,1:8], 5,
             metric = 'manhattan', stand = FALSE)
nostdd_plt$PAM5_ny_mf = pam5_ny_mf$cluster
result2_pam = rbind(result2_pam,
performance(table(nostdd_plt$Class,nostdd_plt$PAM5_ny_mf)))

pam5_ny_default= pam(nostdd_plt[,1:8], 5)
stdd_plt$PAM5_ny_default = pam5_ny_default$cluster
result2_pam = rbind(result2_pam,
performance(table(nostdd_plt$Class,stdd_plt$PAM5_ny_default)))

result2_pam

#################################
# name the column & row
colnames(result2_pam ) = c('TruePositive','Purity')
rownames(result2_pam )=
c('euclidean&stand_T','euclidean&stand_F','manhattan&stand_T','manhattan&sta
nd_F','Default')
result2_pam
write.csv(result2_pam ,'results_metric&stand_PMA.csv')

#################################################
#####################################
```

```
## yes_build_ *and* yes_swap
## 'euclidean' & stand = True ##################################
pam5_ny_et = pam(stdd_plt[,1:8], 5,
                 metric = 'euclidean', stand = TRUE)
stdd_plt$PAM5_ny_et = pam5_ny_et$cluster
result3_pam = performance(table(stdd_plt$Class,stdd_plt$PAM5_ny_et))

## 'euclidean' & stand = False #################################
pam5_ny_ef = pam(stdd_plt[,1:8], 5,
                 metric = 'euclidean', stand = FALSE)
stdd_plt$PAM5_ny_ef = pam5_ny_ef$cluster
result3_pam = rbind(result3_pam,
performance(table(stdd_plt$Class,stdd_plt$PAM5_ny_ef)))

##  'manhattan' & stand = True
#############################
pam5_ny_mt = pam(stdd_plt[,1:8], 5,
                 metric = 'manhattan', stand = TRUE)
stdd_plt$PAM5_ny_mt = pam5_ny_mt$cluster
result3_pam = rbind(result3_pam,
performance(table(stdd_plt$Class,stdd_plt$PAM5_ny_mt)))

##  'manhattan' & stand = False
##############################
pam5_ny_mf = pam(stdd_plt[,1:8], 5,
                 metric = 'manhattan', stand = FALSE)
stdd_plt$PAM5_ny_mf = pam5_ny_mf$cluster
result3_pam = rbind(result3_pam,
performance(table(stdd_plt$Class,stdd_plt$PAM5_ny_mf)))

pam5_ny_default = pam(stdd_plt[,1:8], 5)
stdd_plt$PAM5_ny_default = pam5_ny_default$cluster
result3_pam = rbind(result3_pam,
performance(table(stdd_plt$Class,stdd_plt$PAM5_ny_default)))
```

```
result3_pam

#################################
# name the column & row
colnames(result3_pam ) = c('TruePositive','Purity')
rownames(result3_pam )= c('euclidean&stand_T','euclidean&stand_F','manhattan&stand_T','manhattan&stand_F','Default')
result3_pam
write.csv(result3_pam ,'results_metric&stand_PMA_stdd.csv')

#################################################################################
#### 2-C#########################################################

##### get the dataset ###################################
p1 = pca_10  # based on plt_NA_mean
p2 = imp_plt[2:19]
p3_0 = plt_NA_0[2:20]
p3_mean = plt_NA_mean[2:20]
p3_median = plt_NA_median[2:20]

ncol(p3_0)
head(p3_0)
##### apply the pam & get performance
#################################

## p1: pca_10,stdd
pam5 = pam(p1[,1:10], 5)
p1$PAM5 = pam5$cluster
sum_pam = performance(table(p1$Class,p1$PAM5))

## p2: nostdd
pam5 = pam(p2[,1:17], 5)
p2$PAM5 = pam5$cluster
sum_pam = rbind(sum_pam, performance(table(p2$Class,p2$PAM5)))
```

```
## p3_0: nostdd,noNA
pam5 = pam(p3_0[,1:19], 5)
p3_0$PAM5 = pam5$cluster
sum_pam = rbind(sum_pam, performance(table(p3_0$Class,p3_0$PAM5)))

## p3_mean: nostdd,noNA
pam5 = pam(p3_mean[,1:19], 5)
p3_mean$PAM5 = pam5$cluster
sum_pam = rbind(sum_pam,
performance(table(p3_mean$Class,p3_mean$PAM5)))

## p3_median: nostdd,noNA
pam5 = pam(p3_median[,1:19], 5)
p3_median$PAM5 = pam5$cluster
sum_pam = rbind(sum_pam,
performance(table(p3_median$Class,p3_median$PAM5)))

####################################
# name the column & row, get the table for performance
colnames(sum_pam) = c('TruePositive','Purity')
rownames(sum_pam)=
c('pca_10','imp_plt','Na_by_0','Na_by_mean','Na_by_median')
sum_pam
write.csv(sum_pam,'results_PAM_2-C.csv')

####################################
#get the confusion matrix
stdd_plt_matrix = max_diag(table(stdd_plt$Class,stdd_plt$PAM5))
performance(table(stdd_plt$Class,stdd_plt$PAM5))
write.csv(stdd_plt_matrix,'matrix_stdd_plt.csv')

p1_matrix = max_diag(table(p1$Class,p1$PAM5))
performance(table(p3_mean$Class,p3_mean$PAM5))
write.csv(p1_matrix,'matrix_p1.csv')
```

```
p2_matrix = max_diag(table(p2$Class,p2$PAM5))
write.csv(p2_matrix,'matrix_p2.csv')
p3_0_matrix = max_diag(table(p3_0$Class,p3_0$PAM5))
write.csv(p3_0_matrix,'matrix_p3_0.csv')
p3_mean_matrix =max_diag(table(p3_mean$Class,p3_mean$PAM5))
write.csv(p3_mean_matrix,'matrix_p3_mean.csv')
p3_median_matrix =max_diag(table(p3_median$Class,p3_median$PAM5))
write.csv(p3_median_matrix,'matrix_p3_median.csv')
```

```
##############################################
#### PART 3 -A ###############################
# read the confusion matrix from results of WEKA
stdd = read.csv(file = 'stdd.csv',header = T, sep=',')
oneR = read.csv(file = 'OneR.csv',header = T, sep=',')
zeroR = read.csv(file = 'ZeroR.csv',header = T, sep=',')
IBK = read.csv(file = 'IBK.csv',header = T, sep=',')
J48 = read.csv(file = 'J48.csv',header = T, sep=',')
Bayes = read.csv(file = 'Bayes.csv',header = T, sep=',')

tests = c()
tests = rbind(tests, test(zeroR))
tests = rbind(tests, test(Bayes))
tests = rbind(tests, test(IBK))
tests = rbind(tests, test(J48))

#################################
# name the column & row, get the table for performance
colnames(tests) = c('TruePositive','Purity')
rownames(tests)= c('OneR','ZeroR','Bayes','IBK','J48')
tests
write.csv(sum_pam,'results_PAM_2-C.csv')
```

```
#### PART 3 -C ###################################
# same as PART 2 - C. Write them into files for WEKA use.
p1 = pca_10  # based on plt_NA_mean
write.csv(p1,'p1.csv')
p2 = imp_plt[2:19]
write.csv(p2,'p2.csv')
p3_0 = plt_NA_0[2:20]
write.csv(p3_0,'p3_0.csv')
p3_mean = plt_NA_mean[2:20]
write.csv(p3_mean,'p3_mean.csv')
p3_median = plt_NA_median[2:20]
write.csv(p3_median,'p3_median.csv')
```