CSC3060 AIDA – Assignment 1

Dewei Liu

40216004

15 November 2019

Introduction

Report Structure

This report contains three sections of the assignment (Devereux, 2019).

Each section contains the following headings 1) Requirements. This part states the requirement and the expectations of the current section. 2) Assumption. If applicable, this part states any assumption we use when interpreting the requirement. The implementation is also based on the assumption. 3) Implementation. This part describes how the section is implemented in high level. 4) Result. This part states the result of the implementation and how to rerun the program.

Section 1

Requirements

The section 1 of the assignment (Devereux, 2019) requires 160 CSV files that represent images of four living things (cherry, pear, banana and flower) and four non-living things (envelope, golf club, pencil and wineglass). Each CSV file is 50 pixels by 50 pixels image which represents an instance of one thing, and there should be 20 instances for each thing.

The CSV file should be tab-delimited. Each value is either 0 or 1, representing if the corresponding pixel is black.

Implementation

Doodle Drawing

160 doodle images with 50 by 50 points were drawn on GIMP using my finger. They were then exported to PGM format with raw type.

Doodle Examples

Data Transformation

The PGM images were transformed into CSV files using Python. With the functions $\mathrm{PIL.Image.open}()$ and $numpy.array()$, the data were loaded into Numpy arrays. They were then exported to CSV files with delimitations of tabs.

Result

160 PGM files named as STUDENTNR_LABEL_INDEX. STUDENTNR is my student number which is 40216004. LABEL is the name of the thing/object. INDEX is the index of the instance. They are stored at the directory *section1_images*.

The Python script stored at the directory *section1_code* is used to generate the CSV files.

To run the Python script, make sure

- The Python package Pyyaml is installed
- The Python package Pillow is installed
- The current working directory is *section2_code*

Then execute the command

$$python\ runme.py$$

The 160 corresponding CSV file will be generated which meet the Requirements.

Section 2

Requirements

Section 2 of the assignment (Devereux, 2019) requires to calculate twenty features for each instance, and store them into a CSV file.

The CSV file should be tab-delimited and contain 160 rows representing 160 instances. The rows should be sorted first by the label and secondly by the index. Each row contains the label and index of the instance, followed by the values of twenty features.

Assumption

We assume any pixel out of the range of the image is white. For example, in a 50 by 50 image, the pixels $(-1, 0)$ and $(51, 51)$ are white.

Implementation

This section is implemented in Python.

configuration.yml
This file contains the configurations (e.g. names of objects, index range, names of features) of this section.

It makes the Python script more dynamic, and it is easier to control what the features and objects are processed. For example, if there is a need to drop features, change object labels or change the number of instances for each object, changing the configuration file is the only thing needed.

runme.py
This is the main Python script to be run to generate the CSV file containing feature values.

This is a controller of all objects. It reads a list of labels (i.e. object names) from the configuration file and calls the class Label for each label.

label.py

This is a controller of each object. The constructor of the class Label receives the label of the current object. It gets the range of the indices of instances from the configuration file, and call the class Instance for each instance.

instance.py

This is a controller of each instance. The constructor of the class Instance receives the label and index of the current instance.

It retrieves a list of feature's names from the configuration file and calls the classes for each feature in the list. The classes of each feature will return the feature value. Then it writes the feature values to the output file.

features.py

This is a controller of each feature. The class of each feature receives the image data and the feature dictionary containing feature values of the current instance.

Each class of a specific feature extends the superclass Feature, which provides facilities and variables that helps compute and return the value of the current feature.

Features

Label

Name & Type

The name is $label$. The type is string.

Implementation

It indicates the name of the thing.

Index

Name & Type

The name is $index$. The type is integer.

Implementation

It indicates the index of the instance of the thing.

Feature 1

Name & Type

The name is nr_pix. The type is integer.

Implementation

We iterates each pixel and check it is black. A counter is used to count the total number of black pixels.

Feature 2

Name & Type

The name is $height$. The type is integer.

Implementation

Two pixels on the top row and the bottom row among all black pixels are selected respectively. The value of the feature is the vertical distance between these two pixels.

Feature 3

Name & Type

The name is $width$. The type is integer.

Implementation

Two pixels on the most-left row and the most-right row among all black pixels were selected respectively. The value of the feature is the horizontal distance between these two pixels.

Feature 4

Name & Type

The name is $span$. The type is float.

Implementation

The distances between every two black pixels are calculated. Then they are compared, and the maximum value is selected as the value of this feature.

Feature 5

Name & Type

The name is $rows\_with\_5$. The type is integer.

Implementation

The image data is iterated row by row. For each row, if five black pixels are found, this row is counted. The value of this feature is the number of counted rows.

Feature 6

Name & Type

The name is $cols\_with\_5$. The type is integer.

Implementation

The image data is iterated column by column. For each column, if five black pixels are found, this column is counted. The value of this feature is the number of counted columns.

Feature 7

Name & Type

The name is $neigh1$. The type is integer.

Implementation

For each black pixel in the image, if only one of its neighbours is black, the current black pixel is counted. The value of this feature is the number of counted black pixels.

Feature 8

Name & Type

The name is $neigh5$. The type is integer.

Implementation

For each black pixel in the image, if only five of its neighbours are black, the current black pixel is counted. The value of this feature is the number of counted black pixels.

Feature 9

Name & Type

The name is $left2tile$. The type is integer.

Implementation

For each 2-tile containing at least one pixel which is in the range of the image, if the two pixels of its most two left entries are black, and the remaining pixels are white, this 2-tile is counted. The value of this feature is the number of counted 2-tiles.

Feature 10

Name & Type

The name is $right2tile$. The type is integer.

Implementation

For each 2-tile containing at least one pixel which is in the range of the image, if the two pixels of its most two right entries are black, and the remaining pixels are white, this 2-tile is counted. The value of this feature is the number of counted 2-tiles.

Feature 11

Name & Type

The name is $verticalness$. The type is float.

Implementation

The value of this feature is $(left2tile + right2tile) \div nr\_pix$.

Feature 12

Name & Type

The name is $top2tile$. The type is integer.

Implementation

For each 2-tile containing at least one pixel which is in the range of the image, if the two pixels of its two top entries are black, and the remaining pixels are white, this 2-tile is counted. The value of this feature is the number of counted 2-tiles.

Feature 13

Name & Type

The name is $bottom2tile$. The type is integer.

Implementation

For each 2-tile containing at least one pixel which is in the range of the image, if the two pixels of its two bottom entries are black, and the remaining pixels are white, this 2-tile is counted. The value of this feature is the number of counted 2-tiles.

Feature 14

Name & Type

The name is $verticalness$. The type is float.

Implementation

The value of this feature is $(top2tile + bottom2tile) \div nr\_pix$.

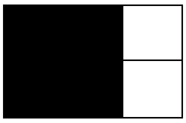
Feature 15

Name & Type

This is a custom feature. The name is $concentration$. The type is integer.

Feature Description

The value of this feature is the number of 3-tiles that the number of its black pixels is greater than the number of its white pixels. As a 3-tile has 9 pixels, it is equivalent to only counting the 3-tiles that has at least five black pixels. For example, Figure 1 is a 3-tile meeting this requirement (six black pixels > three white pixels), while Figure 2 is not.



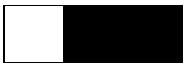


Figure 1 Counted 3-tile

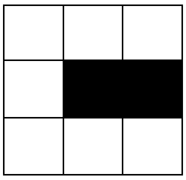


Figure 2 Not Counted 3-tile

Reason for Design

This measures how much the black pixels are concentrated together. This feature will have obvious values for the images of bananas and pencils. Both of them (Figure 3) have a considerable large area fulfilled with black pixels, while others do not. This will help us distinguish these two things from others. Also, then the size of the area in pencils is slightly larger than that in bananas on average, it also helps differentiate between these two things.

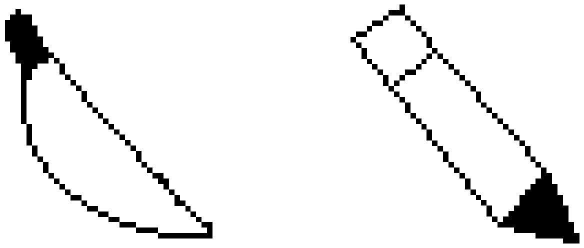


Figure 3 Images of banana and pencil

Implementation

For each 3-tile containing at least one pixel which is in the range of the image, if five black pixels are found in the tile, this 3-tile is counted. The value of this feature is the number of counted 3-tiles.

Feature 16

Name & Type

This is a custom feature. The name is $crossness$. The type is integer.

Feature Description

The value of this feature is the number of unique 3-tiles in the image that meet either of the following requirement.

- Only these three pixels $(0,0),(1,1),(2,2)$ in the tile are black. Other pixels are white.
- Only these three pixels $(0,2),(1,1),(0,2)$ in the tile are black. Other pixels are white.

For example, Figure 4 is a 3-tile meeting the requirement.

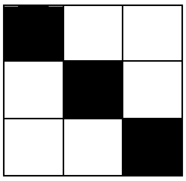


Figure 4 Counted tile in Crossness

Reason of Design

This feature measures the crossness of things/objects. Especially, the images of bananas, envelope, golf clubs and pencils (Figure 5) contain many cross lines (neither vertical nor horizontal, but straight lines) that contribute to this feature. However, the deviation of this feature for golf clubs may be large, because some handles in the images of golf clubs are double-lines which do not meet the requirement of this feature.

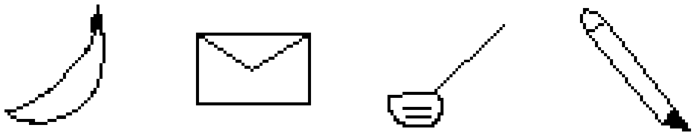


Figure 5 Things for Feature 16

Implementation

For each 3-tile containing at least one pixel which is in the range of the image, we find all black pixels in the tile. If the black pixels meet the requirement, this tile is counted. The final value is the number of counted 3-tiles.

Feature 17

Name & Type

The name is $nr\_regions$. The type is integer.

Implementation

To calculate this feature, we find all black pixels in the image stored in list A. In a LOOP, we first create a list X, and move the first pixel of list A to list X. Then we keep searching for the neighbours of the pixels of list X from list A, until all neighbours of pixels of list X are included in list X. We continue the LOOP until the list A is empty. The value of this feature is the number of list X’s.

The logic of the implementation is that each list X is a connected region of black pixels. We move pixels from list A to list X to prevent the same pixel is searched more than once.

For example, for Figure 6, three list X’s of black pixels will be generated.

1. List of black pixels for the outer shape
2. List of black pixels for the first black bar in the middle
3. List of black pixels for the second black bar in the middle

Thus, the value of this feature for this image is 3.

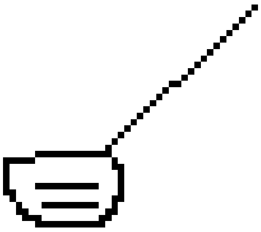


Figure 6 An Example for Feature 17

Feature 18

Name & Type

The name is nr_eyes. The type is integer.

Implementation

We find all white pixels in the image, and use a similar technique in Feature 17 to get a list of list X’s. each list X is a connected region of white pixels. A small difference for this feature is that we only consider two pixels are neighbours if they share the same edge.

After this, each list X is a potential eye. A list X is considered as an eye if it does not contain any pixel that is at the edge of the image. The value of this feature is the number of counted list X's.

For example, for Figure 7, three list X's of white pixels will be generated.

1. List of white pixels outside of the envelop shape
2. List of white pixels in the top region of the envelope
3. List of white pixels in the bottom region of the envelope

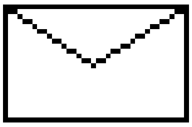


Figure 7 An example of feature 18

The list 1 is not counted, because it contains at least one pixel which is at the edge of the image (e.g. pixel $(0,0)$), while both lists 2 and 3 do not. Lists 2 and 3 are counted, and the final value is 2.

Feature 19

Name & Type
The name is $hollowness$. The type is float.

Implementation
We use the technique in Feature 18 to get a list of eyes (i.e. a list of counted list X's). We count the number of white pixels in these eyes, marked as value $WHITE$. The value of this feature is $(\mathrm{WHITE} \div \mathrm{nr\_pix})$.

Feature 20

Name & Type
This is a custom feature. The name is $straightness$. The type is integer.

Feature Description
The value of this feature is the maximum number of black pixels in the image that these black pixels are LINEAR. A group of pixels are LINEAR if all pixels are in the same vertical or horizontal line. (e.g. pixels $(14,0), (14,3), (14,4), (14,6)$ are LINEAR because they are in the line $x = 14$).

Reason for Design
The feature counts the max length of black pixels in the images. Some things have long straight lines (Figure 8), such as envelopes, flowers and wine glasses. Because their vertical or horizontal lines contribute to this feature, they have large values for this feature. Thus, it will be easy to distinguish them from other things by this feature.

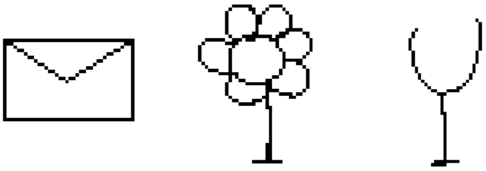


Figure 8 Examples for Feature 20

Implementation
First, all vertical and horizontal lines are found, and a list is generated where each element in the list is a line. We then count the black pixels in each line. The line with the greatest number of black pixels is the line we select. The value of this feature is the number of black pixels in the selected line.

Result

All features are correctly implemented in Python.

To run the Python script, make sure

- The Python package Pyyaml is installed
- The current working directory is *section2_code*

Then execute the command

$$python\ runme.py$$

A file called *40216004_features.csv* will be generated and stored under the folder *section2_features*.

The format of the CSV file meets the Requirements.

To examine the implementation of each feature, please see *features.py* under the directory *section2_code/python_modules.*

Section 3

Requirements

Implementation

For each question, there is a corresponding folder under the directory *section3_code* with the name of the index of the question. Under that folder, there is a runme.R file where is the start point of the R script. To rerun the R script for each question, make sure

- The current working directory is *section3_code/questionX*, where X is the index of the question. (e.g. *section3_code/question1*)
- All R libraries in the R script *section3_code/utilities/libraries.R* are installed

Then execute the command

$$r <\ runme.R - -no - save$$

The figures/graphs will be generated at the directory *section3_code/questionX/output*, where X is the index of the question.

Subtask 1

Requirement

Three features (nr_pix, height and cols_with_5) are needed to be analysed, and we need to perform them on three groups (full set, non-living and living) of things. Hence, there are nine combinations.

Implementation

For each combination, we have two figures. The first figure is the histogram of the distribution with a green line indicating the density of the distribution, and a red line showing the normal distribution of the sample mean and standard deviation. The second figure is a GG Plot which demonstrates the association between theoretical quantiles and actual quantiles.

Reasoning

We use histogram and Q-Q line for the analysation. By using histogram with density line, we can identify the shapes of the distributions. We use Q-Q line to see the normality of the distributions.

Result

nr_pix

For the shape of the distribution of feature *nr_pix* for full set, Figure 9 shows that the modality is unimodal, and the skewness is right skew. Figure 10 illustrates that the normality of the distribution is right skew.

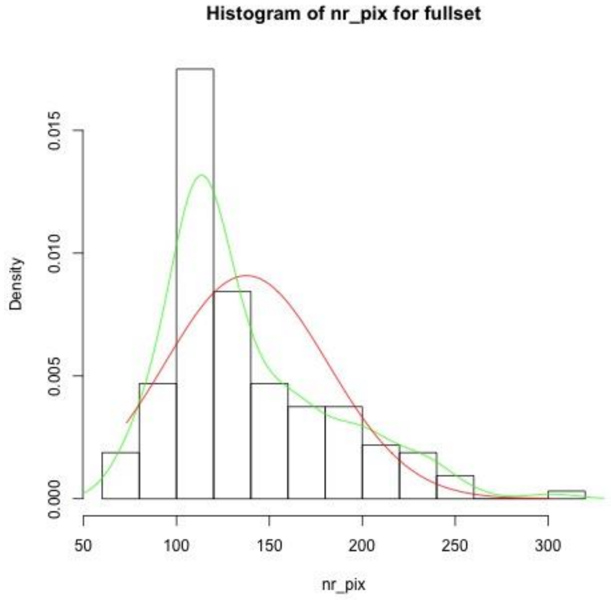


Figure 9 Histogram of the distribution of nr_pix for fullset

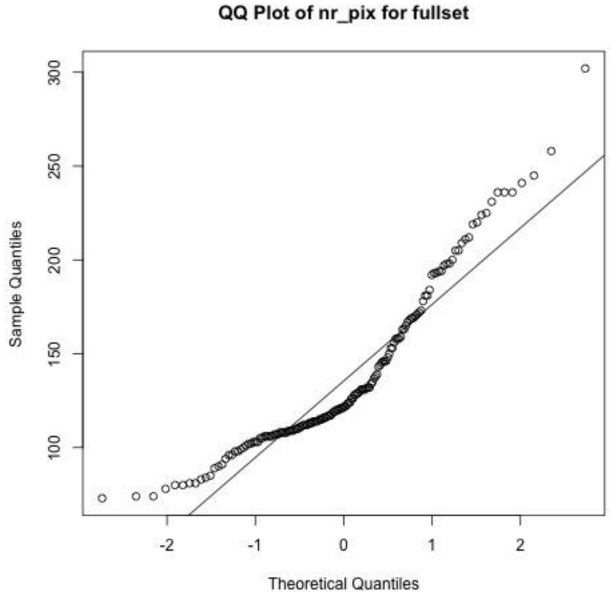


Figure 10 QQ Plot of nr_pix for fullset

For the shape of the distribution of feature *nr_pix* for non-living things, Figure 11 shows that the modality is unimodal (ignoring the small peak at the left), and the skewness is symmetric. Figure 12 illustrates that the normality of the distribution is long tails.

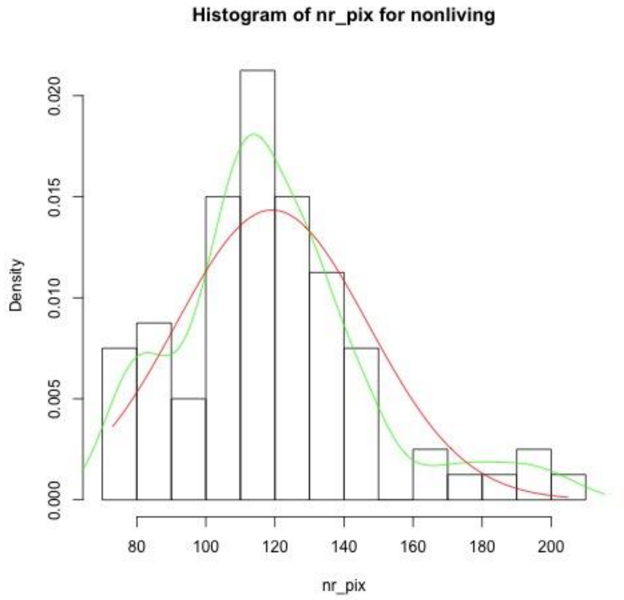


Figure 11 Histogram of the distribution of nr_pix for non-living

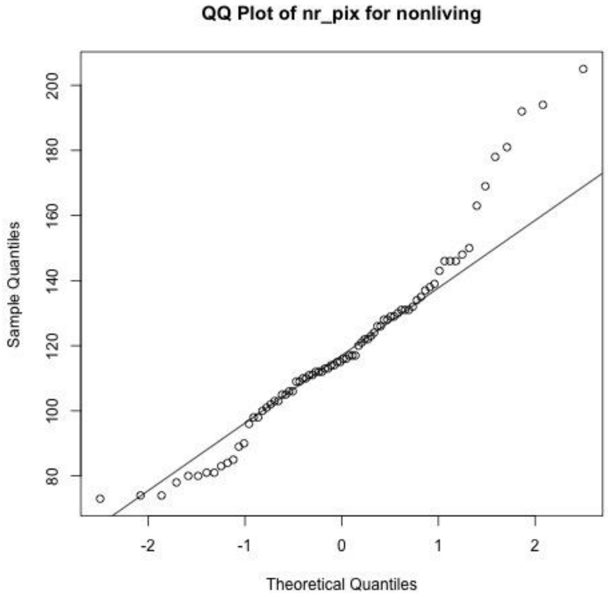


Figure 12 QQ Plot of nr_pix for nonliving

For the shape of the distribution of feature *nr_pix* for living things, Figure 13 shows that the modality is unimodal, and the skewness is right skew. Figure 14 illustrates that the normality of the distribution is short tails.

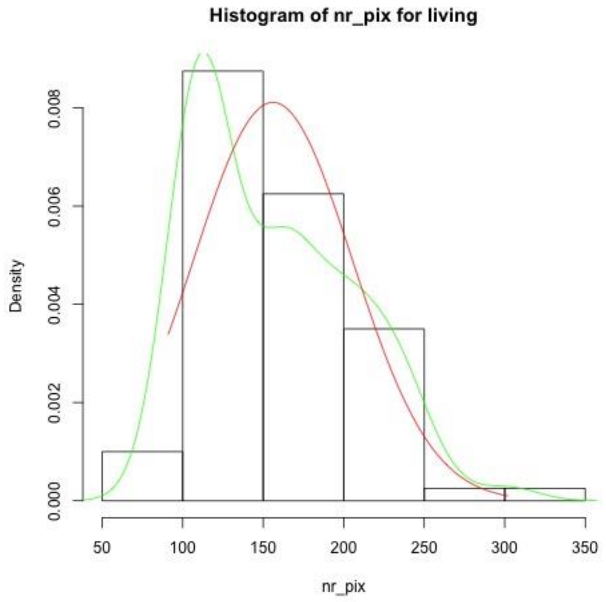


Figure 13 Histogram of the distribution of nr_pix for living

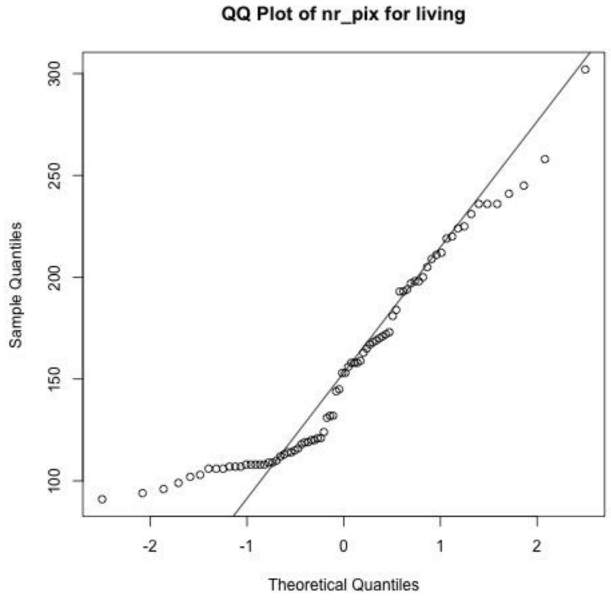


Figure 14 QQ Plot of nr_pix for living

height

For the shape of the distribution of feature *height* for fullset, Figure 15 shows that the modality is unimodal, and the skewness is left skew. However, it contains some outliers at the very left, which may be indicative of measurement error. Figure 16 illustrates that the normality of the distribution is left skew.

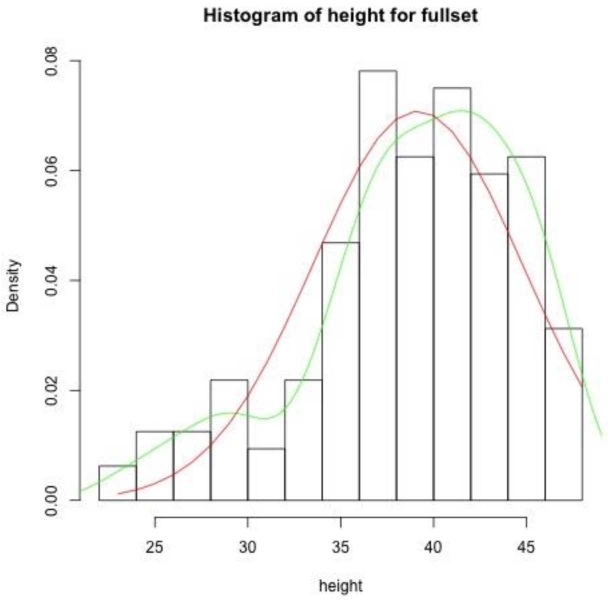


Figure 15 Histogram of the distribution of height for fullset

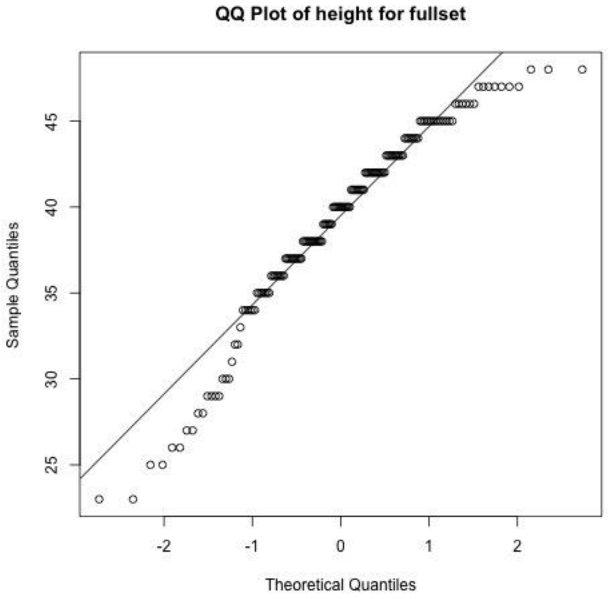


Figure 16 QQ Plot of height for fullset

For the shape of the distribution of feature *height* for non-living things, Figure 17 shows that the modality is unimodal, and the skewness symmetric. Figure 18 illustrates the normality of the distribution is left skew.

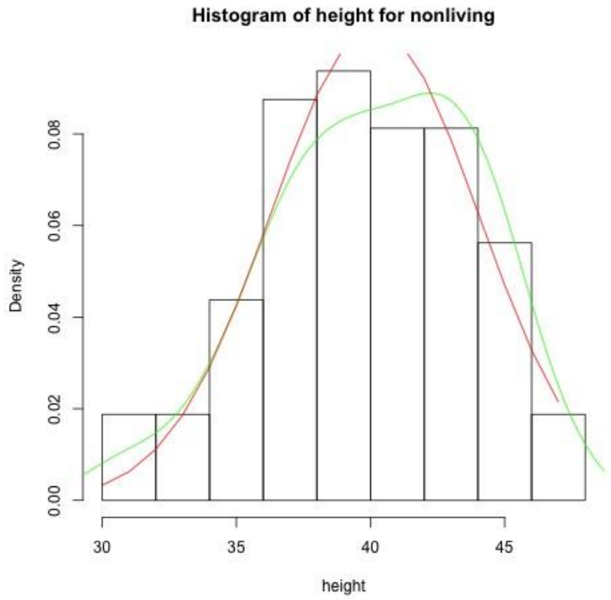


Figure 17 Historgram of the distribution of height for non-living

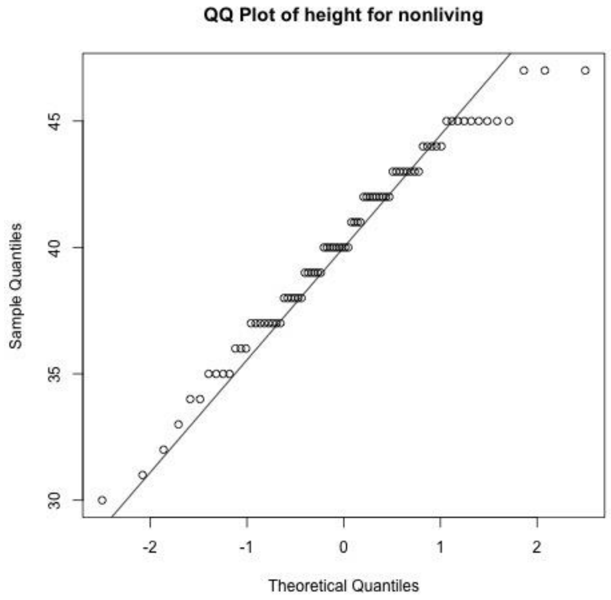


Figure 18 QQ Plot of height for non-living

For the shape of the distribution of feature *height* for living things, Figure 19 shows that the modality is bimodal, and the skewness slightly left skew. Figure 20 illustrates the normality of the distribution is short tails.

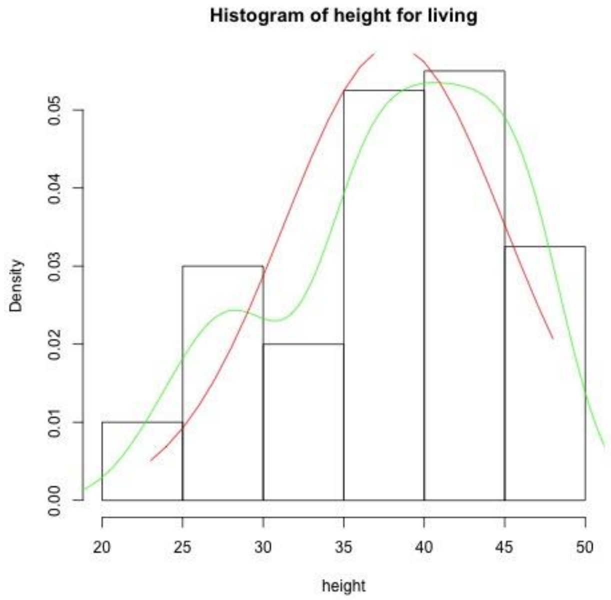


Figure 19 Histogram of the distribution of height for living

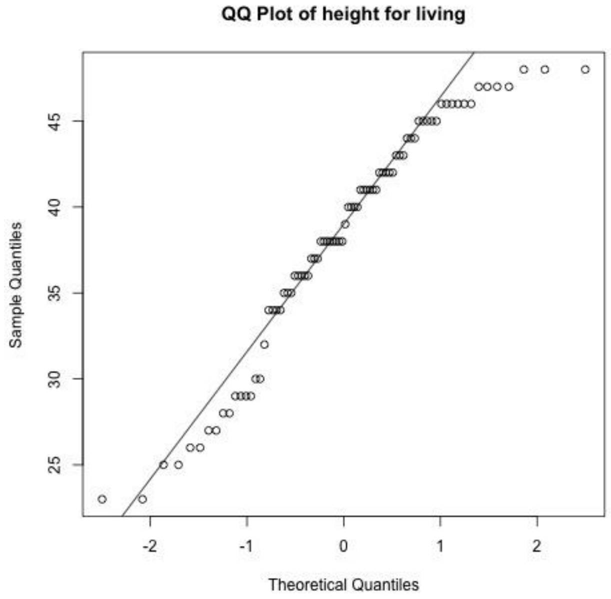


Figure 20 QQ Plot of height for living

cols_with_5

For the shape of the distribution of feature *cols_with_5* for full set, Figure 21 shows that the modality is unimodal, and the skewness slightly right skew. It contains some outliers at the very right which may be indicative of measurement error. Figure 22 illustrates the normality of the distribution is right skew.

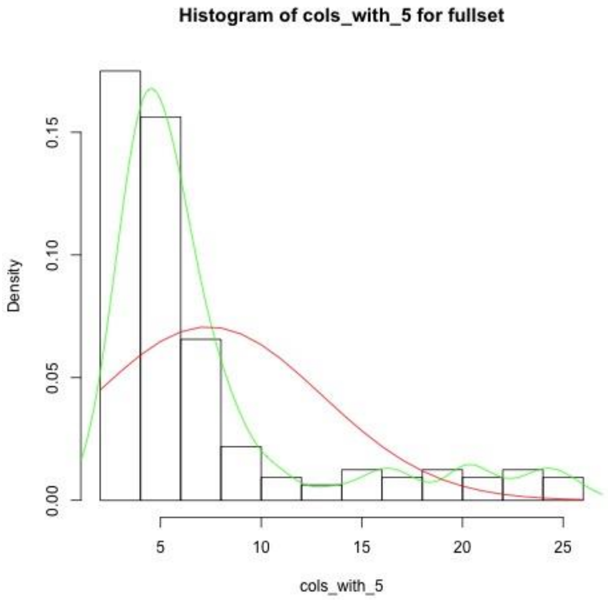


Figure 21 Histogram of the distribution of cols_with_5 for fullset

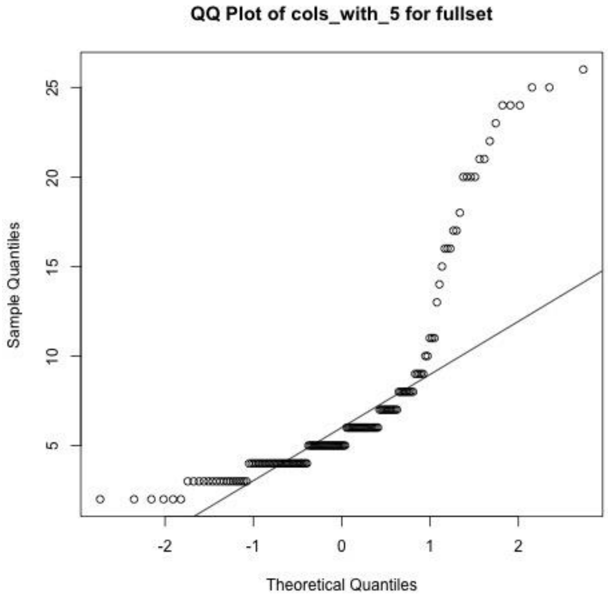


Figure 22 QQ Plot of cols_with_5 for fullset

For the shape of the distribution of feature *cols_with_5* for non-living things, Figure 23 shows that the modality is unimodal, and the skewness is slightly right skew. However, it contains some outliers at the very right, which may be indicative of measurement error. Figure 24 illustrates the normality of the distribution is right skew.

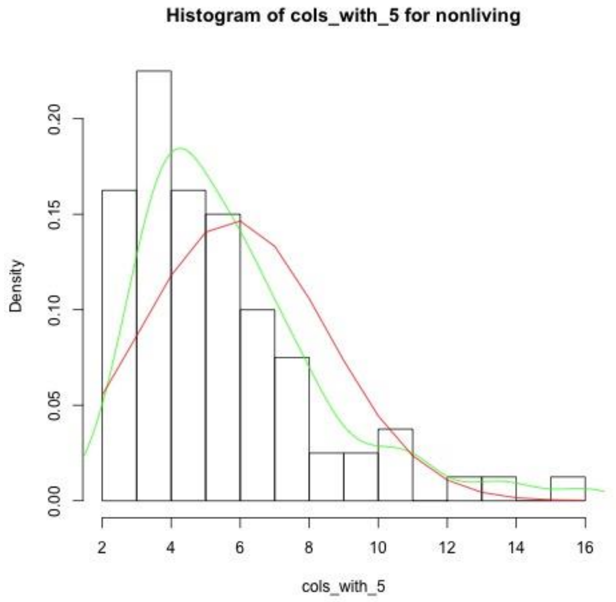


Figure 23 Histogram of the distribution of cols_with_5 for non-living

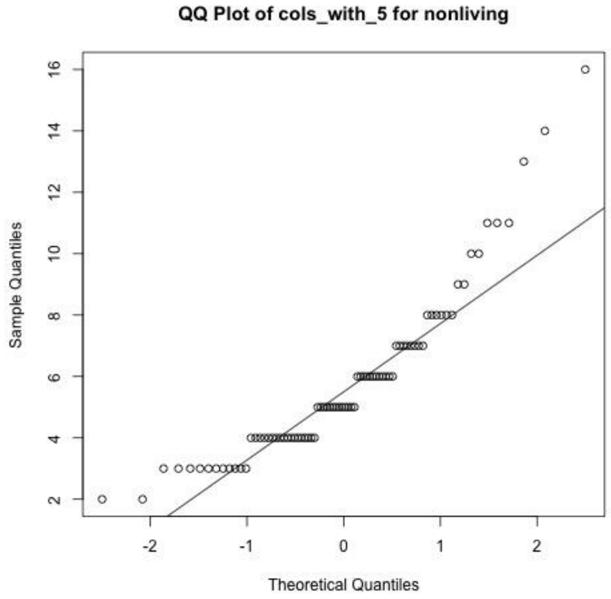


Figure 24 QQ Plot of cols_with_5 for nonliving

For the shape of the distribution of feature *cols_with_5* for living things, Figure 25 shows that the modality is bimodal, and the skewness is slightly right skew. Figure 26 illustrates that the normality of the distribution is right skew.

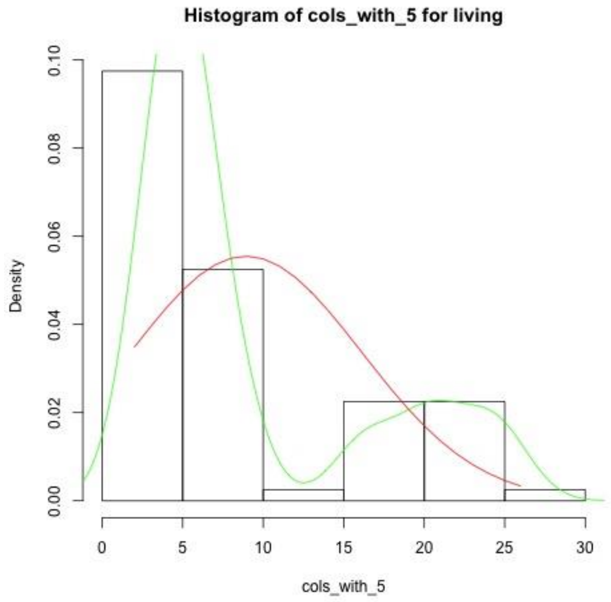


Figure 25 Histogram of the distribution of cols_with_5 for living

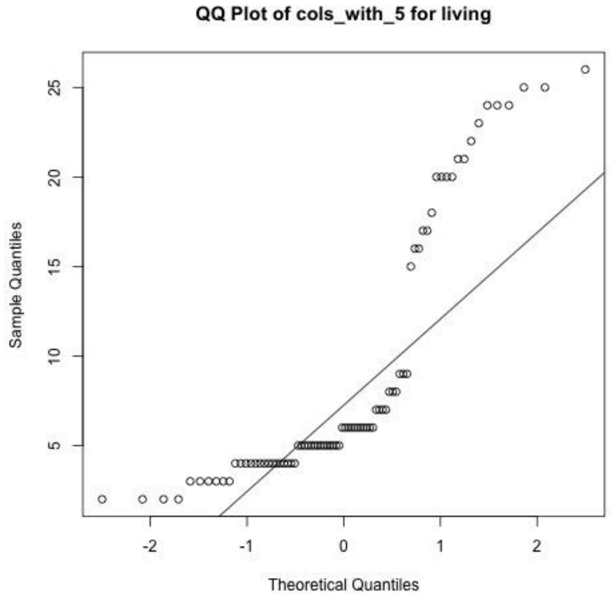


Figure 26 QQ Plot of cols_with_5 for living

Subtask 2

Requirement

We need to generate summary statics for three groups of things/objects and perform analysis to discriminate things from different groups.

Reasoning

Definition: the **STANDARD RANGE** of a feature is the range of $mean \pm (standard\ deviation)$, where mean and standard deviation are calculated from the values of the feature.

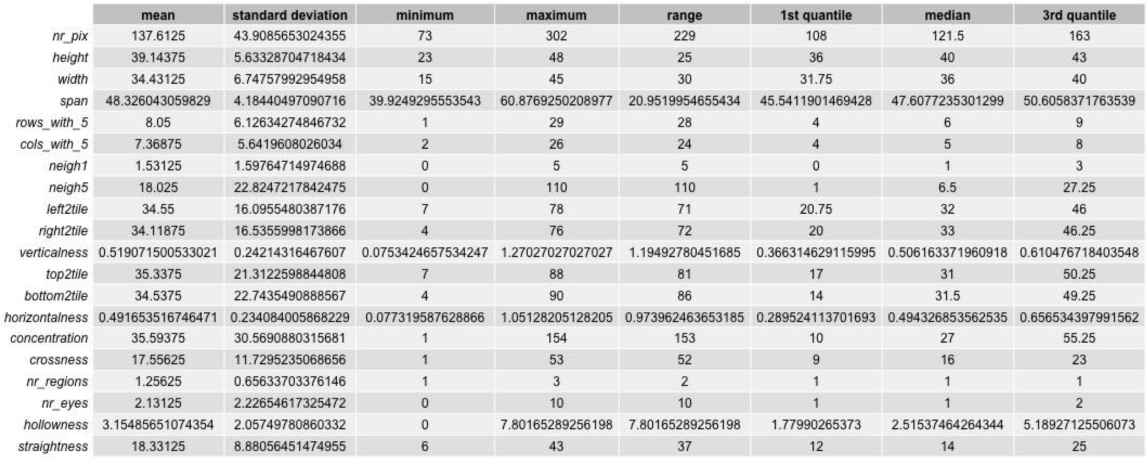
For example, if a feature has the mean of 10 and the standard deviation of 2, its STANDARD RANGE is $(10 - 2, 10 + 2)$.

Assuming the values of each feature is normally distributed, we consider the means and standard deviations of the values from any two groups. Compared to any two groups, we calculate the STANDARD RANGE of every feature from group A, and compare that range with the range of the same feature from group B. If these two ranges do not overlap too much, it means it may be a valuable feature to differentiate the things between this two groups. The reason is that there are 65% values of the feature falling into the STANDARD RANGE $(mean - (standard\ deviation)), (mean + (standard\ deviation))$, and if the STANDARD RANGEs do not overlay, we have a good chance to put the things into the right groups by analysing their feature values.

For example, the mean and the standard deviation of values from group A are 5 and 5, respectively. It is believed that 65% of values in group A are in $(0, 10)$. The mean and the standard deviation of values from group B are 20 and 5 respectively which means that 65% of values in group B are in $(15, 25)$. When getting a value of 9 which is in $(0, 10)$, we are confident that the instance that the value of 9 is corresponding to is in group A.

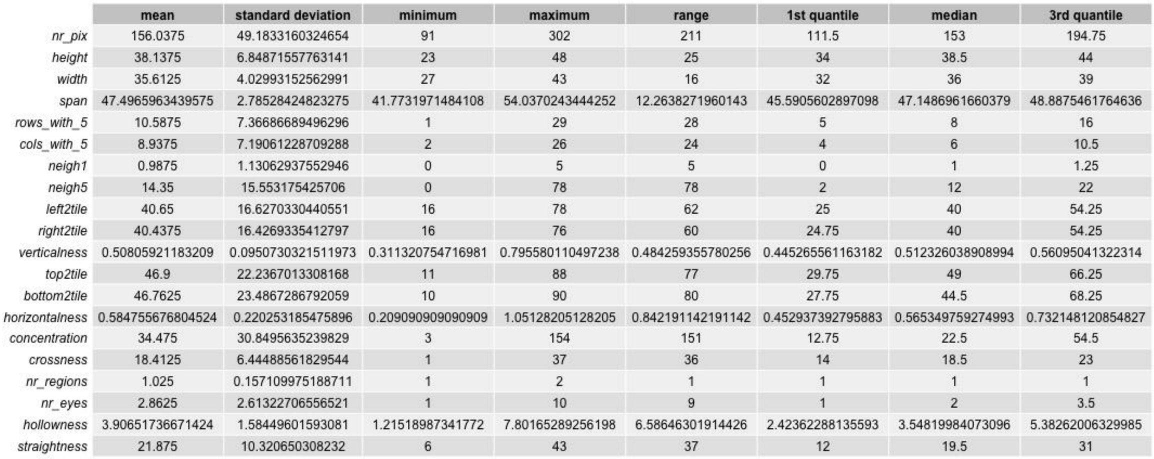
Implementation

We use the summary() function for each feature and generate the tables (Figure 27, Figure 28 and Figure 29) for three groups of things.



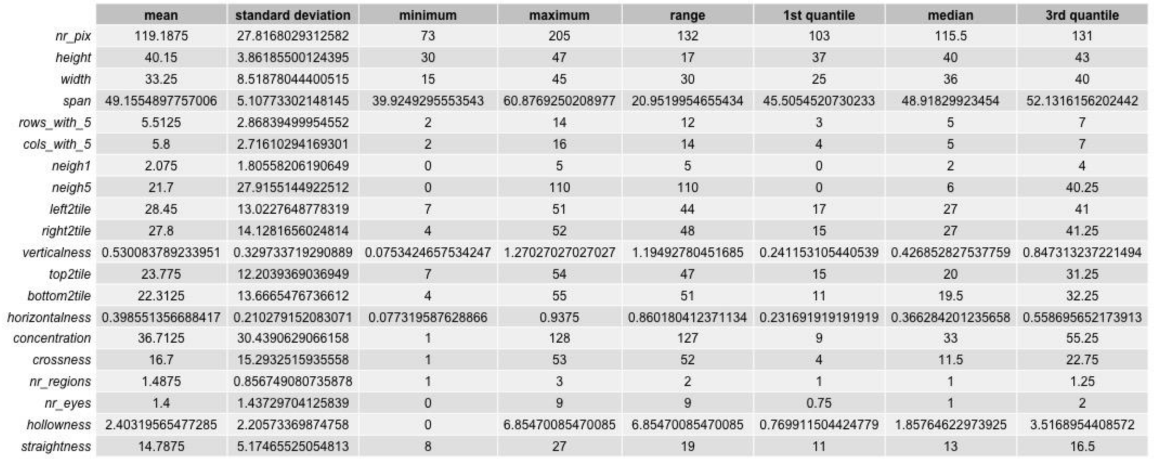
	mean	standard deviation	minimum	maximum	range	1st quantile	median	3rd quantile
nr_pix	137.6125	43.9085653024355	73	302	229	108	121.5	163
height	39.14375	5.63328704718434	23	48	25	36	40	43
width	34.43125	6.74757992954958	15	45	30	31.75	36	40
span	48.326043059829	4.18440497090716	39.9249295553543	60.8769250208977	20.9519954655434	45.5411901469428	47.6077235301299	50.6058371763539
rows_with_5	8.05	6.12634274846732	1	29	28	4	6	9
cols_with_5	7.36875	5.6419608026034	2	26	24	4	5	8
neigh1	1.53125	1.59764714974688	0	5	5	0	1	3
neigh5	18.025	22.8247217842475	0	110	110	1	6.5	27.25
left2tile	34.55	16.0955480387176	7	78	71	20.75	32	46
right2tile	34.11875	16.5355998173866	4	76	72	20	33	46.25
verticalness	0.519071500533021	0.24214316467607	0.0753424657534247	1.27027027027027	1.19492780451685	0.366314629115995	0.506163371960918	0.610476718403548
top2tile	35.3375	21.3122598844808	7	88	81	17	31	50.25
bottom2tile	34.5375	22.7435490888567	4	90	86	14	31.5	49.25
horizontalness	0.491653516746471	0.234084005868229	0.077319587628866	1.05128205128205	0.973962463653185	0.289524113701693	0.494326853562535	0.656534397991562
concentration	35.59375	30.5690880315681	1	154	153	10	27	55.25
crossness	17.55625	11.7295235068656	1	53	52	9	16	23
nr_regions	1.25625	0.65633703376146	1	3	2	1	1	1
nr_eyes	2.13125	2.22654617325472	0	10	10	1	1	2
hollowness	3.15485651074354	2.05749780860332	0	7.80165289256198	7.80165289256198	1.77990265373	2.51537464264344	5.18927125506073
straightness	18.33125	8.88056451474955	6	43	37	12	14	25

Figure 27 Summary statics for fullset



	mean	standard deviation	minimum	maximum	range	1st quantile	median	3rd quantile
nr_pix	156.0375	49.1833160324654	91	302	211	111.5	153	194.75
height	38.1375	6.84871557763141	23	48	25	34	38.5	44
width	35.6125	4.02993152562991	27	43	16	32	36	39
span	47.4965963439575	2.78528424823275	41.7731971484108	54.0370243444252	12.2638271960143	45.5905602897098	47.1486961660379	48.8875461764636
rows_with_5	10.5875	7.36686689496296	1	29	28	5	8	16
cols_with_5	8.9375	7.19061228709288	2	26	24	4	6	10.5
neigh1	0.9875	1.13062937552946	0	5	5	0	1	1.25
neigh5	14.35	15.553175425706	0	78	78	2	12	22
left2tile	40.65	16.6270330440551	16	78	62	25	40	54.25
right2tile	40.4375	16.4269335412797	16	76	60	24.75	40	54.25
verticalness	0.50805921183209	0.0950730321511973	0.311320754716981	0.795580110497238	0.484259355780256	0.445265561163182	0.512326038908994	0.56095041322314
top2tile	46.9	22.2367013308168	11	88	77	29.75	49	66.25
bottom2tile	46.7625	23.4867286792059	10	90	80	27.75	44.5	68.25
horizontalness	0.584755676804524	0.220253185475896	0.209090909090909	1.05128205128205	0.842191142191142	0.452937392795883	0.565349759274993	0.732148120854827
concentration	34.475	30.8495635239829	3	154	151	12.75	22.5	54.5
crossness	18.4125	6.44488561829544	1	37	36	14	18.5	23
nr_regions	1.025	0.157109975188711	1	2	1	1	1	1
nr_eyes	2.8625	2.61322706556521	1	10	9	1	2	3.5
hollowness	3.90651736671424	1.58449601593081	1.21518987341772	7.80165289256198	6.58646301914426	2.42362288135593	3.54819984073096	5.38262006329985
straightness	21.875	10.320650308232	6	43	37	12	19.5	31

Figure 28 Summary statics for living things



	mean	standard deviation	minimum	maximum	range	1st quantile	median	3rd quantile
nr_pix	119.1875	27.8168029312582	73	205	132	103	115.5	131
height	40.15	3.86185500124395	30	47	17	37	40	43
width	33.25	8.51878044400515	15	45	30	25	36	40
span	49.1554897757006	5.10773302148145	39.9249295553543	60.8769250208977	20.9519954655434	45.5054520730233	48.91829923454	52.1316156202442
rows_with_5	5.5125	2.86839499954552	2	14	12	3	5	7
cols_with_5	5.8	2.71610294169301	2	16	14	4	5	7
neigh1	2.075	1.80558206190649	0	5	5	0	2	4
neigh5	21.7	27.9155144922512	0	110	110	0	6	40.25
left2tile	28.45	13.0227648778319	7	51	44	17	27	41
right2tile	27.8	14.1281656024814	4	52	48	15	27	41.25
verticalness	0.530083789233951	0.329733719290889	0.0753424657534247	1.27027027027027	1.19492780451685	0.241153105440539	0.426852827537759	0.847313237221494
top2tile	23.775	12.2039369036949	7	54	47	15	20	31.25
bottom2tile	22.3125	13.6665476736612	4	55	51	11	19.5	32.25
horizontalness	0.398551356688417	0.210279152083071	0.077319587628866	0.9375	0.860180412371134	0.231691919191919	0.366284201235658	0.558695652173913
concentration	36.7125	30.4390629066158	1	128	127	9	33	55.25
crossness	16.7	15.2932515935558	1	53	52	4	11.5	22.75
nr_regions	1.4875	0.856749080735878	1	3	2	1	1	1.25
nr_eyes	1.4	1.43729704125839	0	9	9	0.75	1	2
hollowness	2.40319565477285	2.20573369874758	0	6.85470085470085	6.85470085470085	0.769911504424779	1.85764622973925	3.5168954408572
straightness	14.7875	5.17465525054813	8	27	19	11	13	16.5

Figure 29 Summary statics for nonliving things

Unfortunately, after analysing, we did not find any two features that have STANDARD RANGE not overlaying. However, we still got three features *top2tile*, *bottom2tile* and *horizontalness* that do not have too much overlaying and could be distinguishable between the groups living things and non-living things.

top2tile

The feature *top2tile* has the mean of 46.9 and the standard deviation of 22.24 in the group of living things, which has the STANDARD RANGE of $(24.66, 69.14)$. Similarly, the mean is 23.78 and the standard deviation is 12.20 in non-living things where the STANDARD RANGE is $(11.58, 35.98)$.

bottom2tile

The feature *bottom2tile* has the mean of 46.76 and the standard deviation of 23.49 in the group of living things, which has the STANDARD RANGE of $(23.27, 70.25)$. Similarly, the mean is 22.31 and the standard deviation is 13.67 in non-living things where the STANDARD RANGE is $(8.64, 35.98)$.

horizontalness

The feature *horizontalness* has the mean of 0.5848 and the standard deviation of 0.2203 in the group of living things, which has the STANDARD RANGE of $(0.3645, 0.8051)$. Similarly, the mean is 0.3986 and the standard deviation is 0.2103 in non-living things where the STANDARD RANGE is $(0.1883, 0.6089)$.

Subtask 3

Assumption

We assume the variable nr_pix is randomly sampled from a population which is normally distributed (Devereux, 2019).

We assume these n values have been sampled without replacement, and $n < 10\%$ of the population.

Reasoning

The sample data meets these two requirements.

- Independence
- The population distribution is normal

We consider it is independence, because the values are randomly sampled without replacement and the sample size is greateer than 30, and less than 10% of the population. Also, as the Assumption, the population is normal distributed.

Thus, Central Limit Theorem (CLT) applies. We can get the estimated variance and the confidence interval for the actual mean of the population.

Implementation

We calculate

- n. The length of the data
- μ. The mean of the sample data
- σ. The standard deviation of the sample data

According to CLT, the distribution of the sample mean is

$$\bar{x} \sim N\left(mean = \mu, SE = \frac{\sigma}{\sqrt{n}}\right),$$

(Devereux, 2019)

We then draw the distribution (Figure 30).

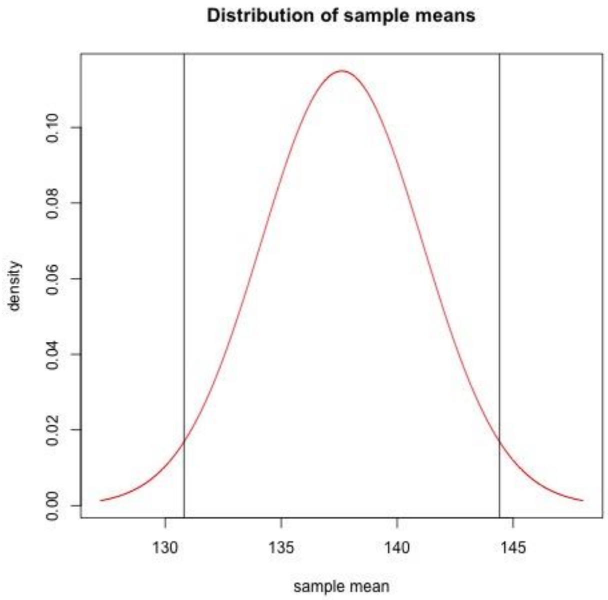


Figure 30 Normal distribution of nr_pix sample means

We assure the z-score z is 1.96 for 95% confidence.

We have the confidence interval

$$\mu \pm z * \sigma$$

We have 95% confidence that the actual mean of the population is inside this interval.

The estimated mean of the population is μ. The estimated variance of the population is $\sigma * \sigma$

With the mean() and sd() functions, the mean and standard deviation of the values of nr_pix are calculated.

We then draw Figure 30 showing the normal distribution with the mean and standard deviation of the values of nr_pix. To compare the theoretical normal distribution line with the actual distribution of nr_pix, Figure 32 is the histogram of the distribution is drawn with the same xlim and ylim values.

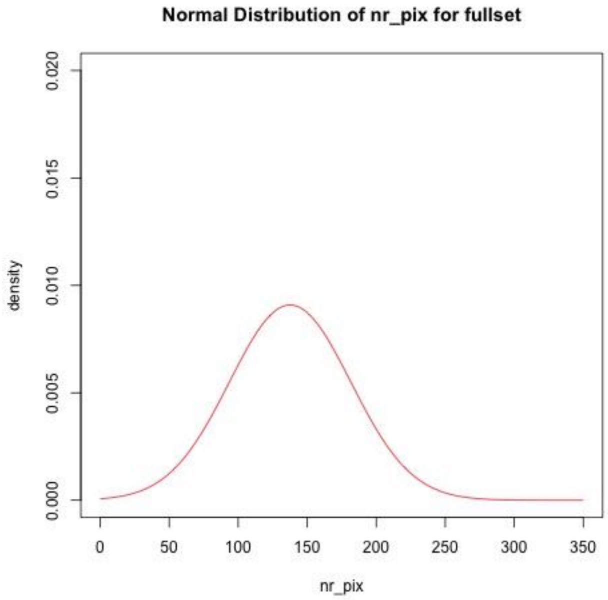


Figure 31 Normal distribution of nr_pix for fullset

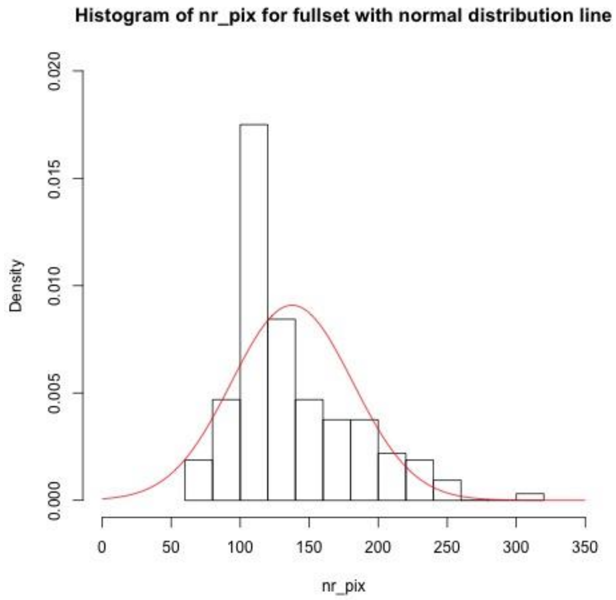


Figure 32 Histogram of nr_pix for fullset with normal distribution line

Result

The estimated mean of the population is 137.6125. We have 95% confidence that the actual mean of the population in between 130.809 and 144.416

The estimated variance of the population is 1927.962

From the normal distribution with the estimated mean and variance, we do not consider the variable nr_pix is normally distributed. It does not approximately match the normal distribution line, because of its prominent peak near to the value 100 and the almost-zero density for the values from 0 to 50.

Subtask 4

Assumption

We assume the variable nr_pix is normally distributed. The mean and the standard deviation of the normal distribution are the mean and the standard deviation of the value from the full set dataset respectively.

Implementation

With mean() and sd() functions, the mean and standard deviation of the values of nr_pix are calculated. The qnorm() function is used to find out the quantile(cut-off) value that 95% of values are observations in the normal distribution is below the cut-off value. The cut-off value is what the question is asking because it is equivalent to the cut-off value that there is 5% (1 – 95%) probability that the sample value is above it.

We then draw Figure 33 showing the normal distribution with the mean and standard deviation of the values of nr_pix and the vertical line where the cut-off value is.

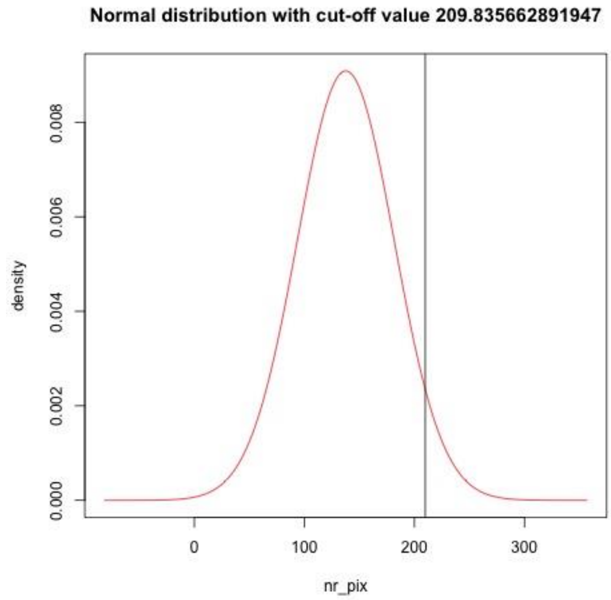


Figure 33 Normal distribution of nr_pix for fullset

Reasoning

The reason we use qnorm() is that it returns the corresponding quantile given the percentile and the normal distribution.

Result

From the qnor() function, we can get the cut-off value of 209.835662891947

Subtask 5

Implementation (part 1)

We will examine the skewness by both the statics and the histogram for each feature.

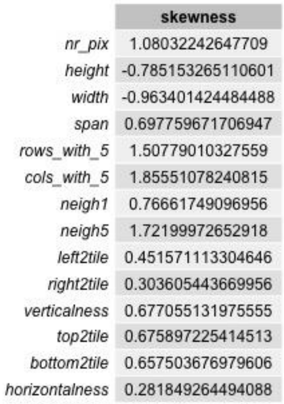
We generated a table (Figure 34) showing the skewness values for features 1 – 14 so that we can compare their absolute values. We consider the variable is (Lemos, 2018)

- Highly skewed, if the skewness value is greater than +1 or less than -1
- Moderately skewed, if the absolute skewness value is between 0.5 and 1
- Approximately symmetric, if the absolute skewness value is less than 0.5

And we will transform the highly skewed data.

Also, For each feature from 1 to 14, we generated a histogram of the distribution with a green line indicating the density of the distribution, and a red line showing the normal distribution of the sample mean and standard deviation, which also helps us to identify the extreme skewness.

Results (Part 1)



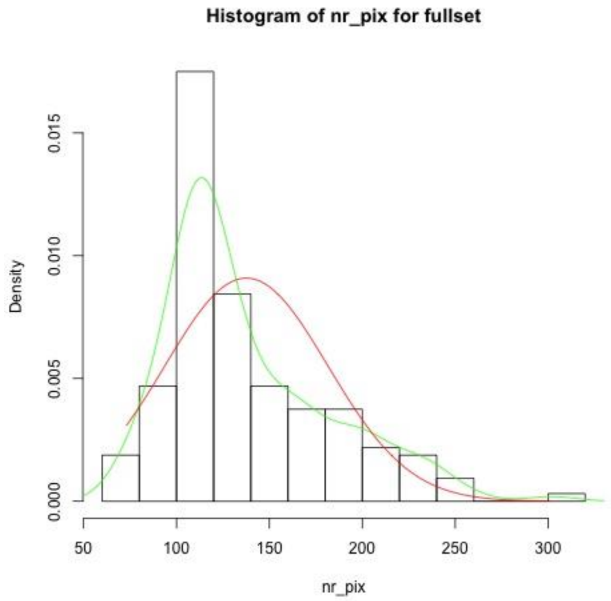
	skewness
nr_pix	1.08032242647709
height	-0.785153265110601
width	-0.963401424484488
span	0.697759671706947
rows_with_5	1.50779010327559
cols_with_5	1.85551078240815
neigh1	0.76661749096956
neigh5	1.72199972652918
left2tile	0.451571113304646
right2tile	0.303605443669956
verticalness	0.677055131975555
top2tile	0.675897225414513
bottom2tile	0.657503676979606
horizontalness	0.281849264494088

Figure 34 Skewnesses of features 1 - 14

The statics show the feature *nr_pix*, *rows_with_5*, *cols_with_5* and *neigh5* are highly skewed.

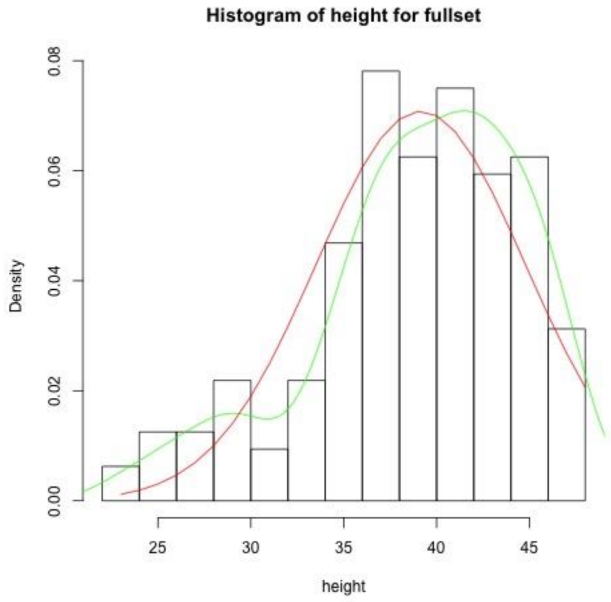
Feature 1

From the histogram, the distribution is considered to be extreme skew, and its skewness value is 1.08 Log transformation is required.



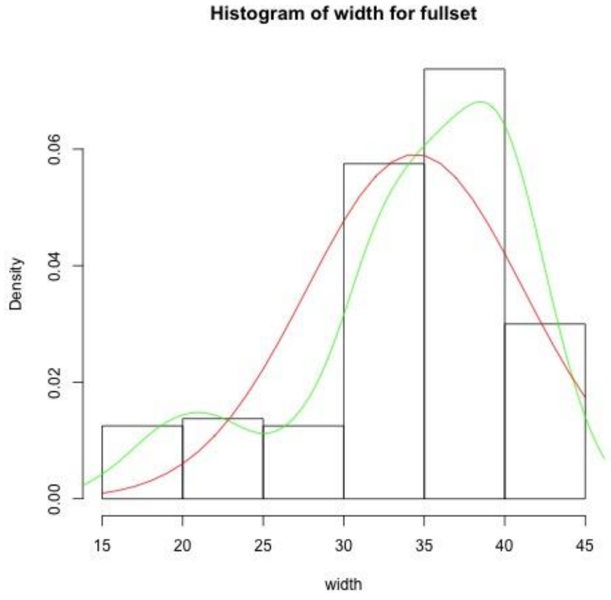
Feature 2

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



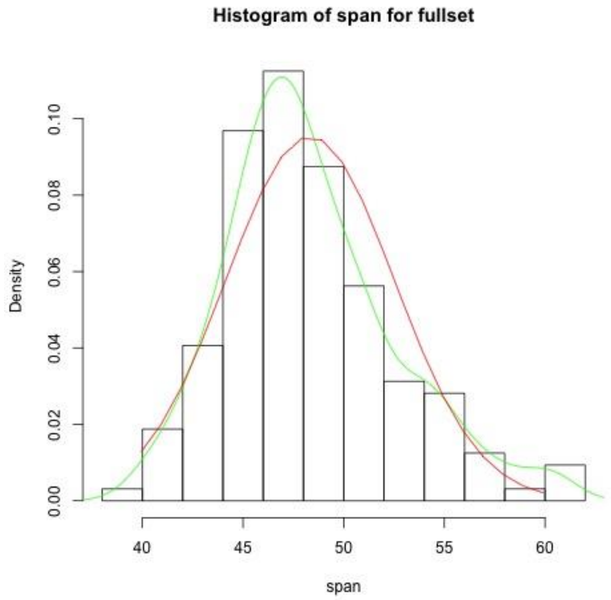
Feature 3

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



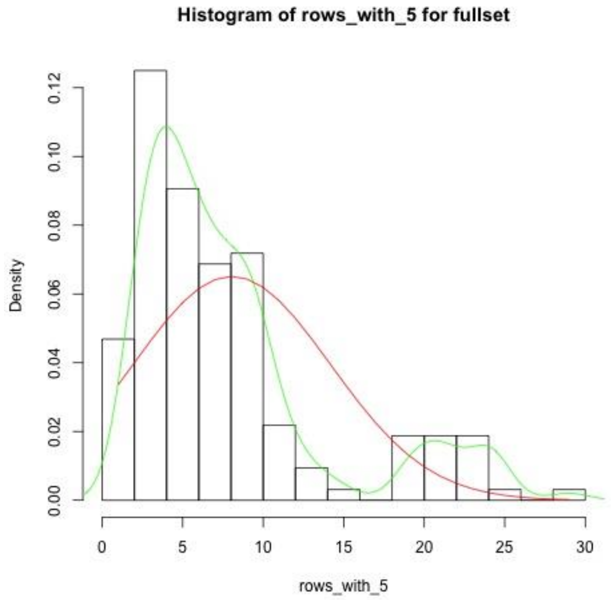
Feature 4

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



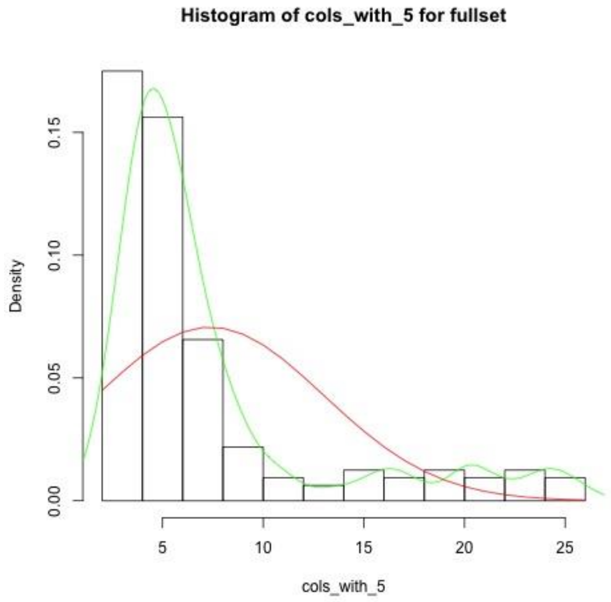
Feature 5

From the histogram, the distribution is considered to be extreme right skew, and its skewness value is 1.51. Log transformation is required.

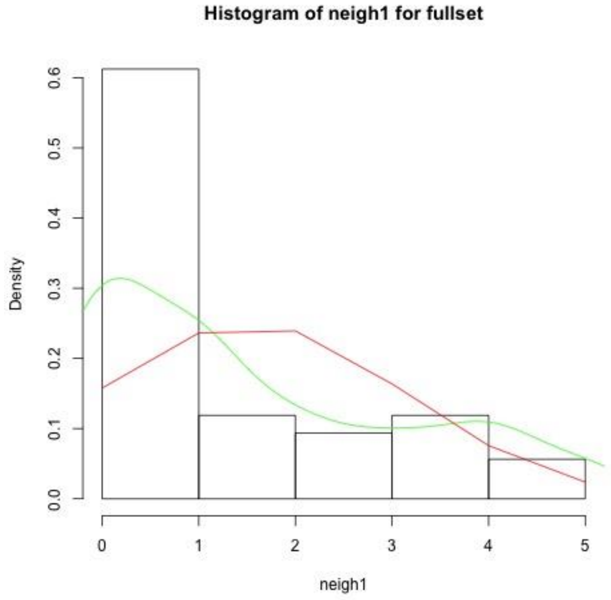


Feature 6

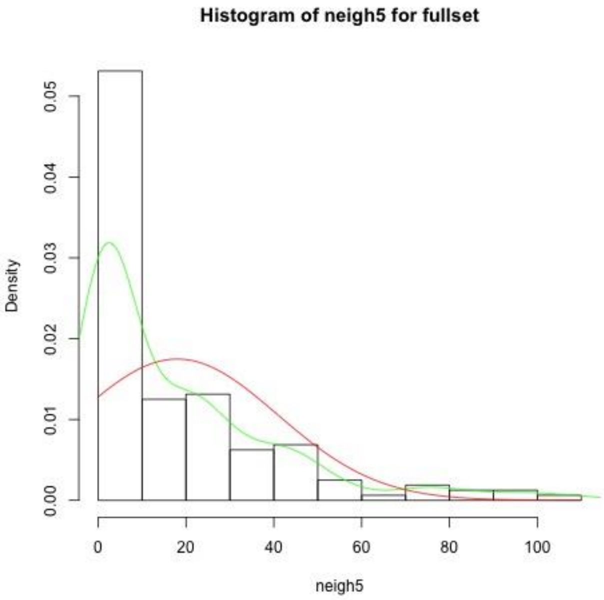
From the histogram, the distribution is extreme right skew, and its skewness value is 1.86. Log transformation is required.



Feature 7
From the histogram, the distribution is right skew, but it is now extreme. Log transformation is not required.

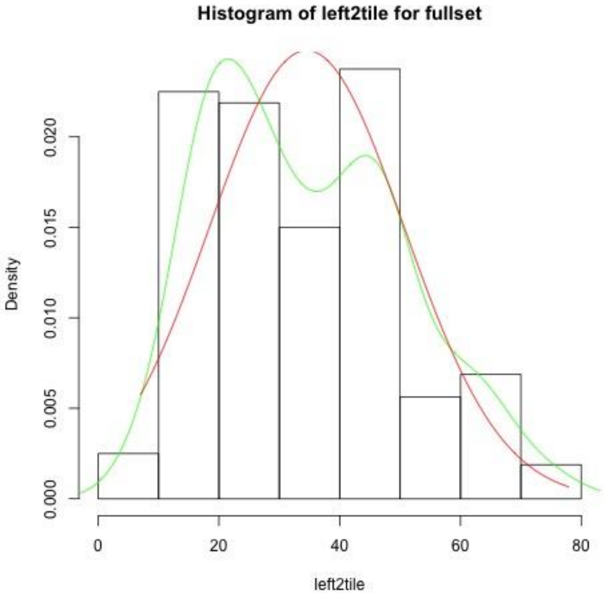


Feature 8
The histogram illustrates that the distribution is extreme right skew, and its skewness value is 1.72. Log transformation is required.



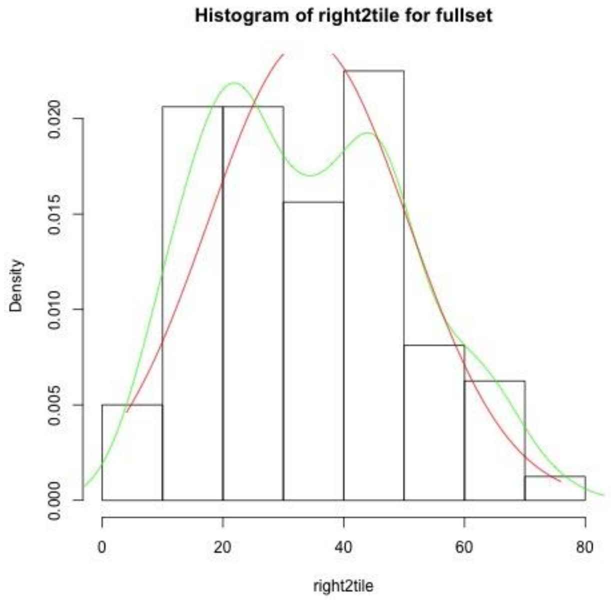
Feature 9

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



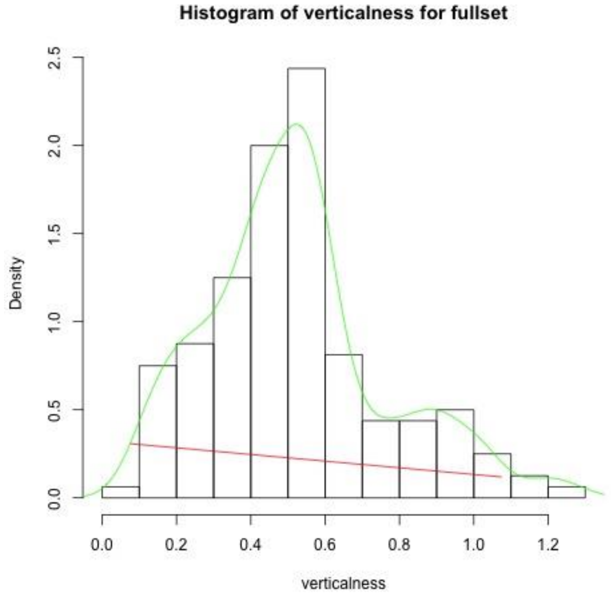
Feature 10

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



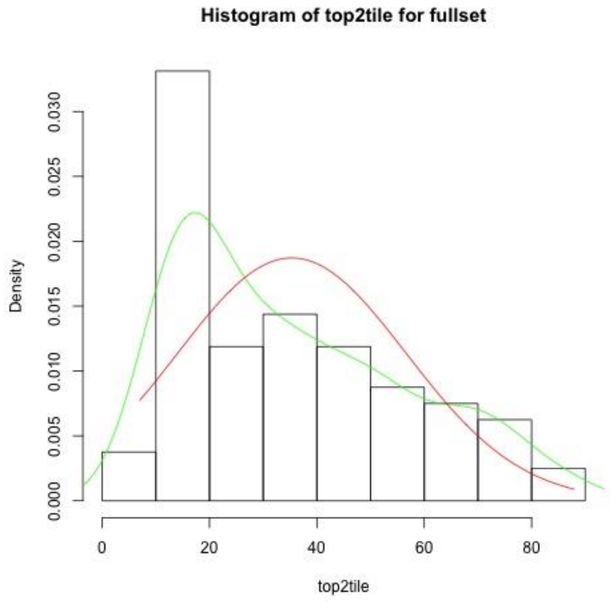
Feature 11

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



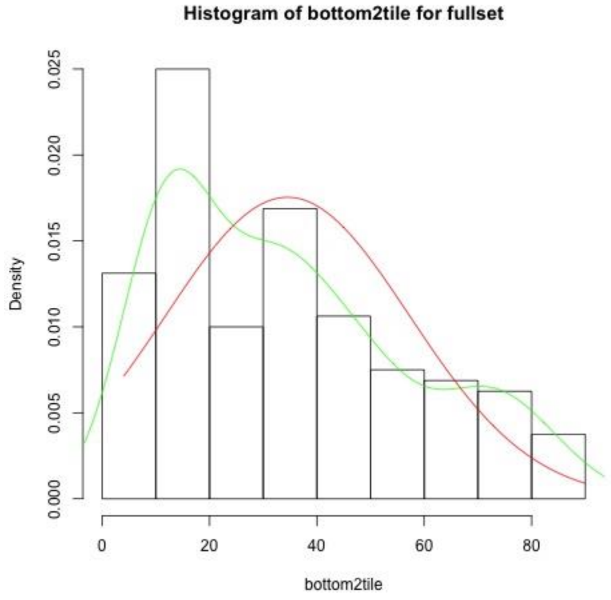
Feature 12

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



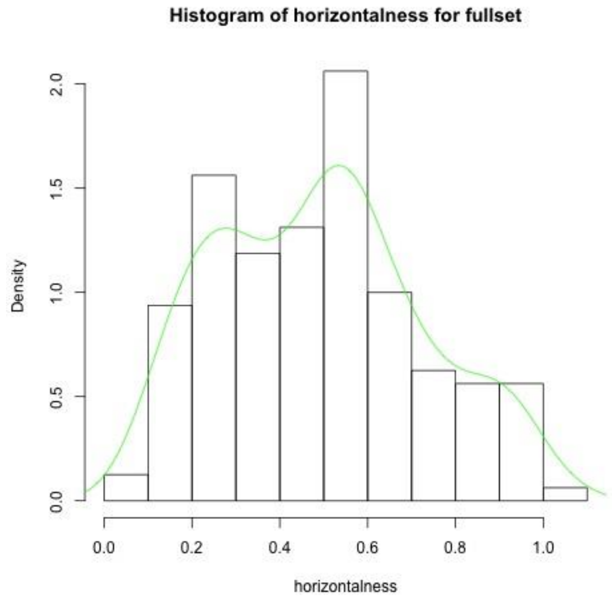
Feature 13

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



Feature 14

From the histogram, the distribution is not considered to be extreme skew. Transformation is not required.



Implementation (part 2)

For extreme skew data, logarithm function log1p() will be performed to the original data for the transformation. We then compute its new skewness value to make sure the skewness has been improved. We will redo the log1p() function until the absolute skewness value has been decreased to under 1. Finally, we redraw the histograms.

Result (part 2)

From Results (Part 1), we concluded that we need to transform Feature 1, Feature 5, Feature 6 and Feature 8.

Feature 1

The distribution of the feature *nr_pix* has extreme right skew. Log transformation will be performed.

The new distribution of the values of this feature is re-drawn (Figure 35).

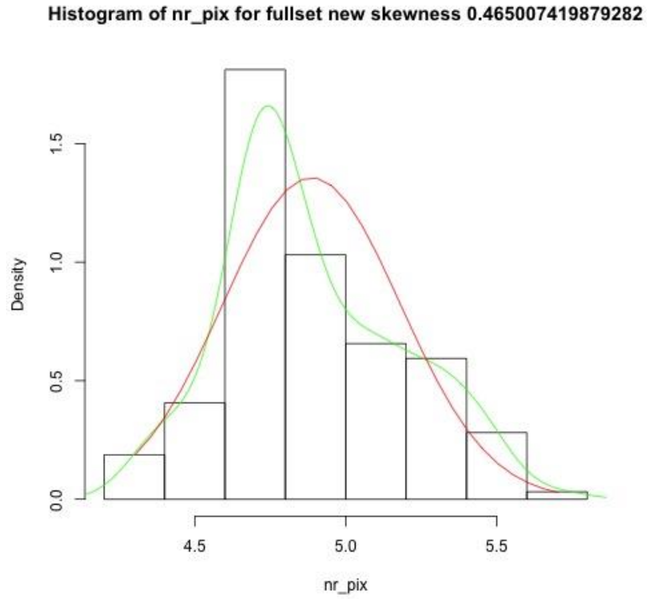


Figure 35Transformed data for feature nr_pix

The new skewness value of the transformed data is 0.47, which is considered to be approximately symmetric.

Feature 5

The distribution of the feature *rows_with_5* has extreme right skew. Log transformation is will be performed.

The new distribution of the values of this feature is re-drawn (Figure 36).

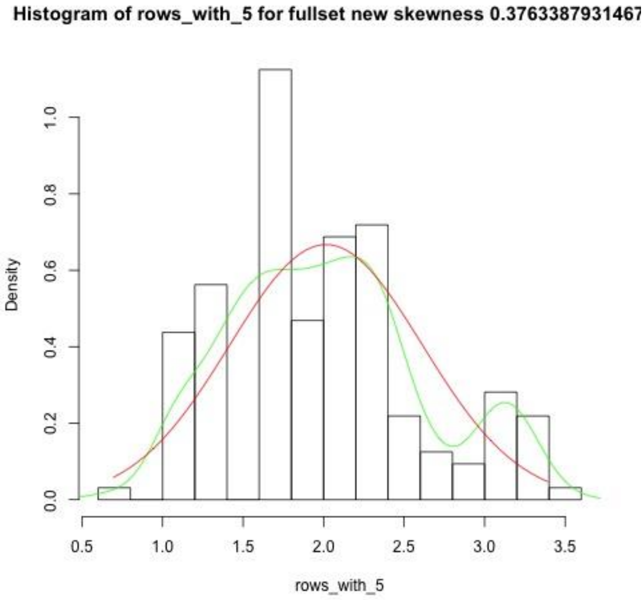


Figure 36 Transformed data for feature rows_with_5

The new skewness value of the transformed data is 0.38, which is considered to approximately symmetric.

Feature 6

The distribution of the feature *cols_with_5* has extreme right skew. Log transformation will be performed.

The new distribution of the values of this feature is re-drawn (Figure 37).

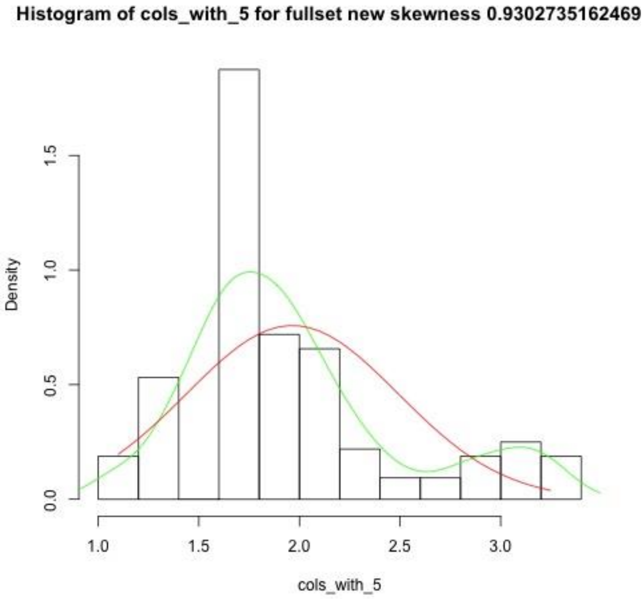


Figure 37 Transformed data for feature cols_with_5

The new skewness value of the transformed data is 0.93, which is not considered to be extreme skew.

Feature 8

The distribution of the feature *neigh5* has extreme right skew. Log transformation is will be performed.

The new distribution of the values of this feature is re-drawn (Figure 38).

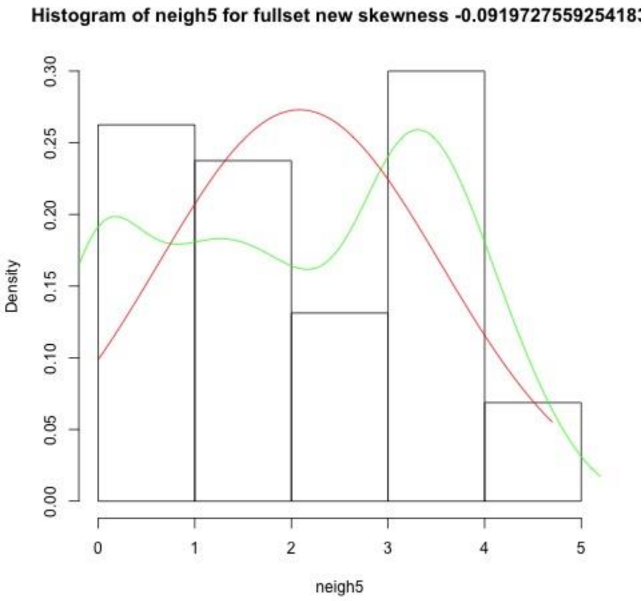


Figure 38 Transformed data for feature neigh5

The new skewness value of the transformed data is -0.09, which is almost a perfect symmetric distribution.

Result

Conclusions

References

Devereux, B., 2019. *Assignment 2 – CSC3060 "AIDA".* [Online] Available at: https://canvas.qub.ac.uk/courses/8433/files/452305/download?wrap=1 [Accessed 12 November 2019].

Lemos, M., 2018. *A hands-on tutorial about Log Transformations using R language.* [Online] Available at: http://rpubs.com/marvinlemos/log-transformation [Accessed 15 11 2019].