**CSC3060 AIDA – Assignment 1**

Dewei Liu

40216004

15 November 2019

# Introduction

## Report Structure

This report contains three section 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) Implemenation. 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 (envelop, golf club, pencil and wineglass). Each CSV file is a 50 pixels by 50 pixels image which represents an instance of one thing, and there are 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 to CSV files using Python. With the functions and , 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 name of the thing. 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

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

# Section 2

## Requirements

The 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 and 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 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 . The type is string.

#### Implementation

It indicates the name of the thing.

### Index

#### Name & Type

The name is . The type is integer.

#### Implementation

It indicates the index of the instance of the thing.

### Feature 1

#### Name & Type

The name is . 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 . 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 . 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 . 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 . 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 . 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 . 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 . 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 . 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 . 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 . The type is float.

#### Implementation

The value of this feature is .

### Feature 12

#### Name & Type

The name is . 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 . 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 . The type is float.

#### Implementation

The value of this feature is .

### Feature 15

#### Name & Type

This is a custom feature. The name is . 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 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 . 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 in the tile are black. Other pixels are white.
* Only these three pixels 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 the things. 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 . The type is integer.

#### Implementation

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

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

For example, for Figure 6, three list ’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 . 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 ’s. each list 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 is a potential eye. A list 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 ’s.

For example, for Figure 7, three list ’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 ), 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 . 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 ’s). We count the number of white pixels in these eyes, marked as value . The value of this feature is .

### Feature 20

#### Name & Type

This is a custom feature. The name is . 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 are LINEAR because they are in the line ).

#### 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

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 the 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

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 gourps (fullset, 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 quantile and actualy quantile.

### Reasoning

We we use histogram and Q-Q line for the analysation. By using historgram 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 fullset, 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.

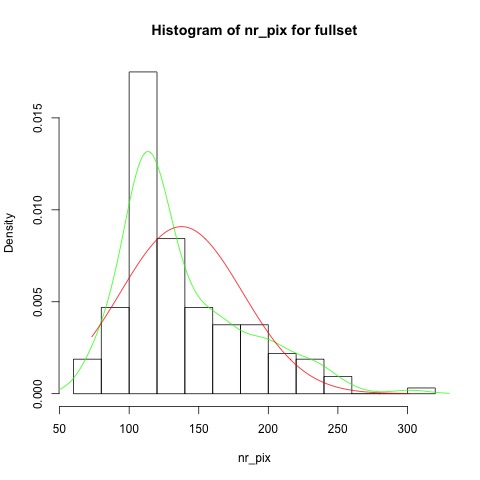


Figure 9 Histogram of the distribution of nr\_pix for fullset

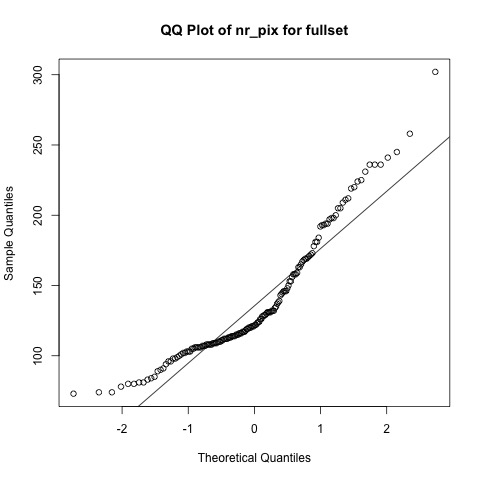


Figure 10 QQ Plot of nr\_pix for fullset

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.

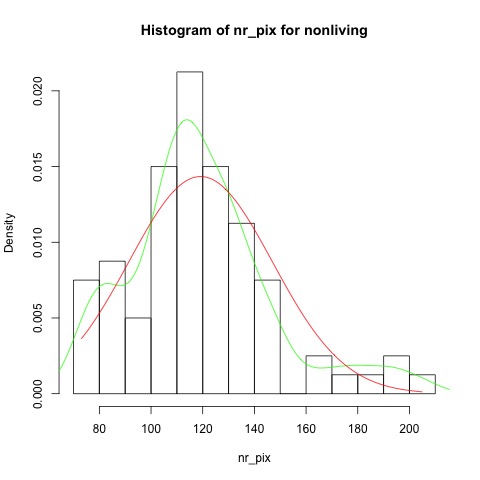


Figure 11 Histogram of the distribution of nr\_pix for non-living

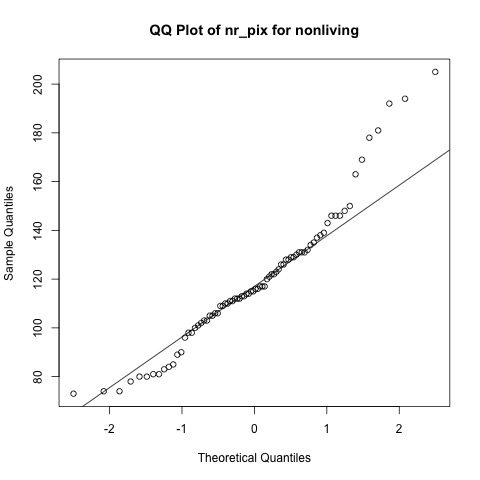


Figure 12 QQ Plot of nr\_pix for nonliving

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

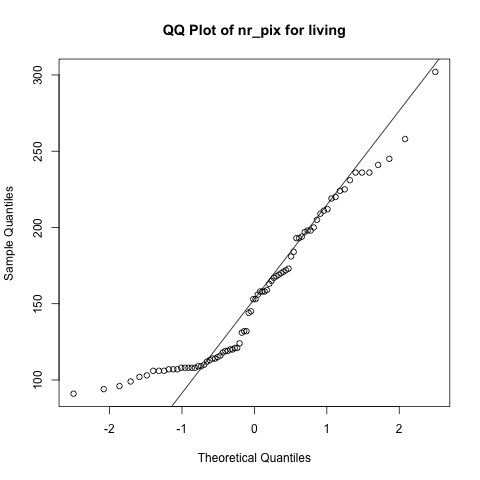


Figure 14 QQ Plot of nr\_pix for living

#### 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.

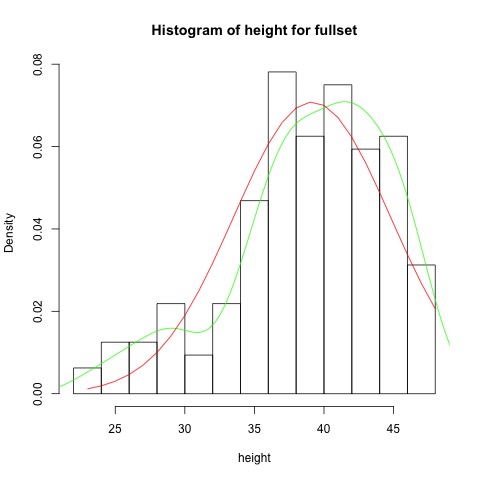


Figure 15 Histogram of the distribution of height for fullset

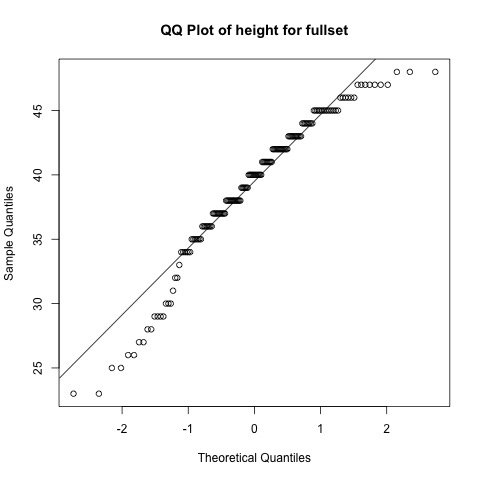


Figure 16 QQ Plot of height for fullset

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.

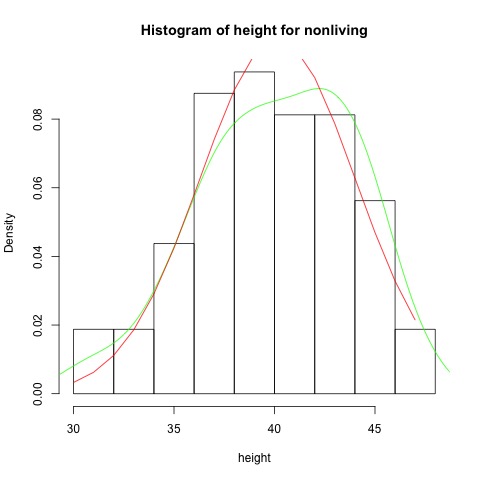


Figure 17 Historgram of the distribution of height for non-living

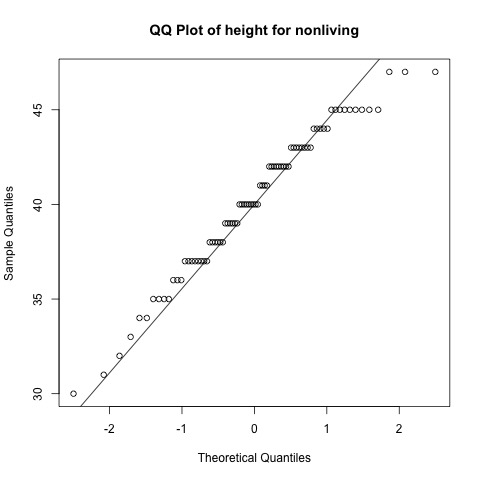


Figure 18 QQ Plot of height for non-living

For the shape of the distribution of feature *height* for living things, **Error! Reference source not found.** shows that the modality is bimodal, and the skewness slightly left skew. Figure 20 illustrates the normality of the distribution is short tails.

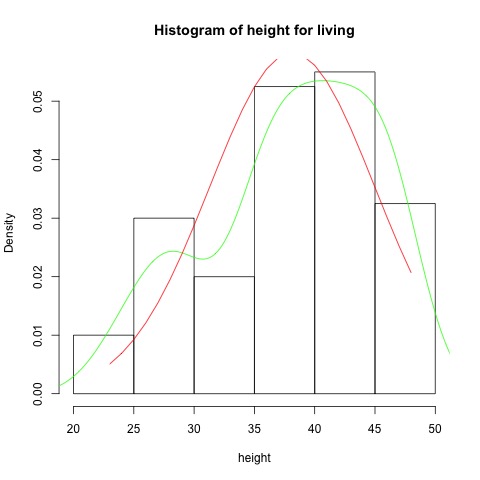


Figure 19 Histogram of the distribution of height for living

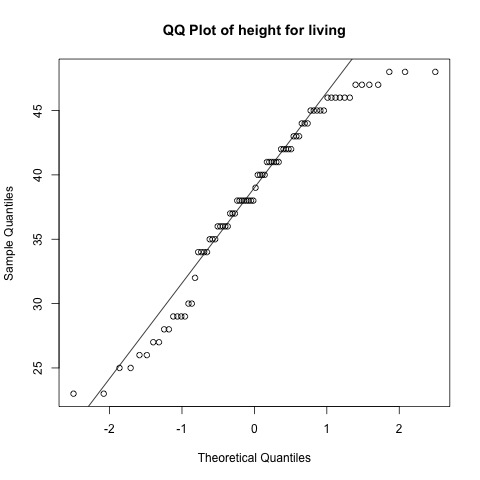


Figure 20 QQ Plot of height for living

#### cols\_with\_5

For the shape of the distribution of feature *cols\_with\_5* for fullset, 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.

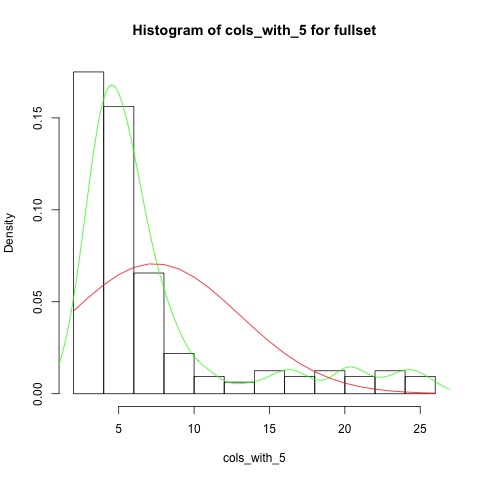


Figure 21 Histogram of the distribution of cols\_with\_5 for fullset

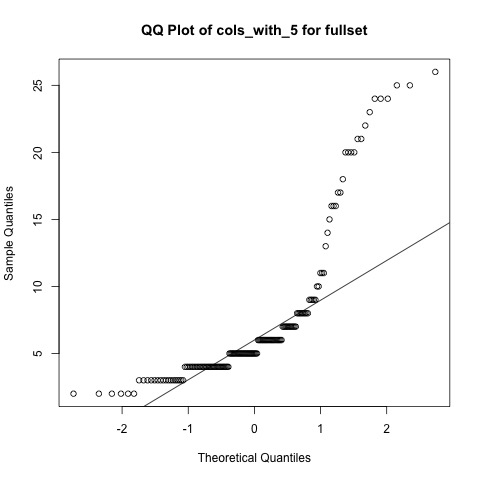


Figure 22 QQ Plot of cols\_with\_5 for fullset

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.

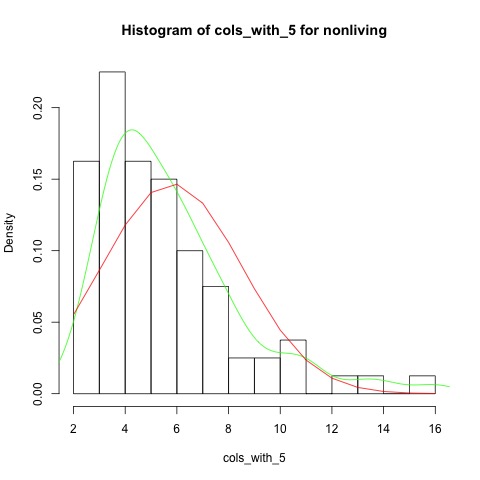


Figure 23 Histogram of the distribution of cols\_with\_5 for non-living

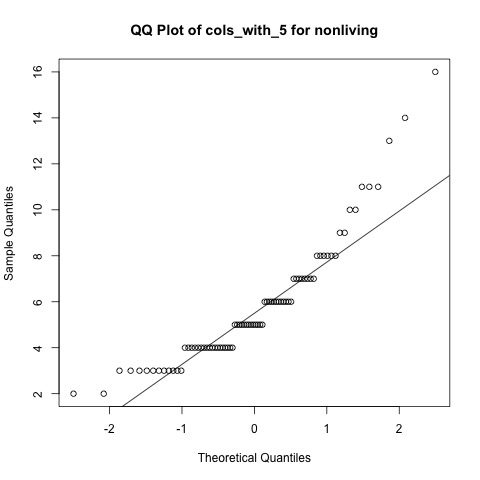


Figure 24 QQ Plot of cols\_with\_5 for nonliving

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.

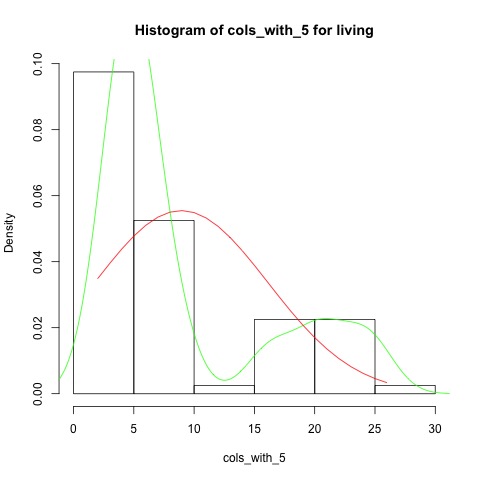


Figure 25 Histogram of the distribution of cols\_with\_5 for living

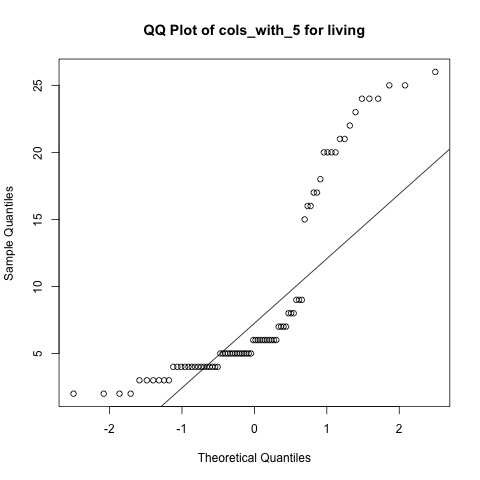


Figure 26 QQ Plot of cols\_with\_5 for living

## Subtask 2

## Subtask 3

### Assumption

We assume the variable nr\_pix is randomly sampled from a population which is normal distributed (Devereux, 2019).

We assume thesevalues have been sampled without replacement, and 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 less than of the population. Also, as the Assumption, the population is normal distributed.

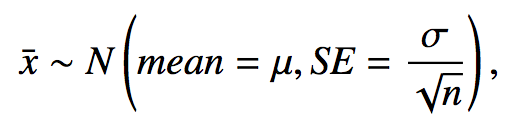
Thus, Central Limit Theorem (CLT) applies. We can get the exsimated variance and the confidence interval for the actual mean of the population.

### Implementation

We calculate

* . The length of the data
* . The mean of the sample data
* . The standard deviation of the sample data

According to CLT, the distribution of the sample mean is

 (Devereux, 2019)

We then draw the dirstribution (Figure 27).

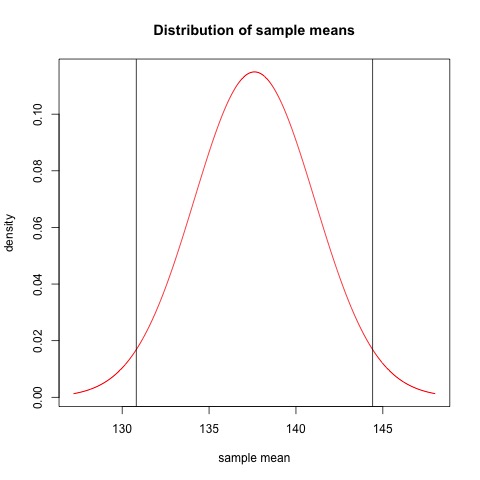


Figure 27 Normal distribution of nr\_pix sample means

We assure the z-score is 1.96 for confidence.

We have the confidence interval

The estimated variance of the population is

### Result

The estimated mean of the population is . 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

## Subtask 4

### Assumption

We assume the variable nr\_pix is normal distributed. The mean and the standard deviation of the normal distribution are the mean and the standard deviation of the value from the fullset dataset resepectively.

### Implementation

With mean() and sd() funtions, the mean and standard deviation of the values of nr\_pix are calculated. qnorm() function is used to to find out the qutatile(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 equivelant to the cut-off value that there is 5% (1 – 95%) probability that the sample value is above it.

We then draw Figure 27 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.

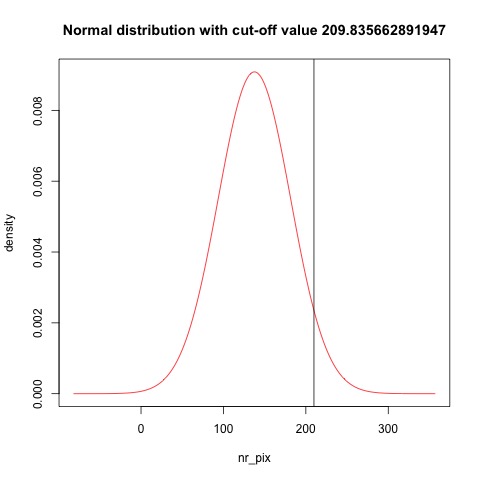


Figure 29 Normal distribution of nr\_pix for fullset

### Resoning

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

### Result

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

## Subtask 5

## 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].