Modelling the effects of domestication in Wheat through novel computer vision techniques

Author:	Mr. Nathan Hughes (nah26@aber.ac.uk)
Supervisor:	Dr. Wayne Aubrey (waa2@aber.ac.uk)
Degree Scheme	G401 (Computer Science)

Date:	April 19, 2018
Revision:	0.1
Status:	Draft

This report was submitted as partial fulfilment
of a BSc degree in Computer Science (G401)

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name ....

Date ....

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name ....

Date ....

Contents

List of Tables

List of Figures

List of Listings

Chapter 1

Introduction, Analysis and Objectives

This project aims to answer a biological research question through the use of computer science, whilst also creating a software suite which will enable further studies to be carried out with ease.

Primarily the focus has been on the data science elements of my degree, creating, cleaning and discerning meaning in it.

Using a population of genetically diverse wheat, several hypothesis and questions are explored in the hopes of contributing to the scientific understanding of domestication. A mixture of image analysis through three-dimensional micro-computed tomography and computational analysis are used to provide these much needed solutions.

Additionally, as this is very much multi-disciplinary research, specific terms and definitions have been outlined in the *glossary* (table:7.3).

Background

Western society and agriculture has been dominated by the ability to create successful crops for the past 10,000 years [2]. Of these crops wheat is considered to be one of the most vital and is estimated to contribute to 20% of the total calories and proteins consumed worldwide, and accounts for roughly 53% of total harvested area (in China and Central Asia) [3].

During domestication, the main traits selected for breeding were most likely plant height and yield. This meant that important non-expressed traits such as disease resistance and drought tolerance were often neglected and lost overtime.

Whilst the choices made for selective breeding were successful, effects are now being felt as it is estimated that as much as a 5% dip is observed yearly on wheat production [3]. This decrease in efficiency is attributed to climate change bringing in more hostile conditions, which these elite and domesticated genotypes are unprepared for.

Furthermore, with increasing populations and less arable land there is an even greater pressure for the optimisation of grain and spike characteristics. With studies showing that spikelet count can be controlled by specific and sometimes recessive genes [4], which could drastically enhance overall yield, and a general public distrust towards genetically modification [5,6,7] the reliance on breeding programs for optimisation is further stressed.

Modern breeding programs have had some success in selecting primitive undomesticated genotypes and using them to breed back in useful alleles which would have been lost during domestication [8].

As such, there are questions still left open about how best to make selections for crop breeding. There is also a lack of formalised modelling of information which could be of use to these areas of research.

Biological Question and Materials

The driving question for this research asks "Can µ-CT data be used to model domestication in wheat?". Using an already grown and harvested range of genetically diverse wheat this project has generated a collection of 3D images, processed these images into raw phenotypic data and produced biologically significant information.

The genotypes used in this study are listed here, denoted by "X N" where X indicates the ploidy. 2N - Diploid; 4N - Tetraploid; 6N - Hexiploid.

- Wild Monococcum (2N)
- Domesticated Monococcum (2N)
- Tauschii (2N)
- Durum (4N)
- Dicoccoides (4N)
- Dicoccum (4N)
- Ispahanicum (4N)
- Timopheevii (4N)
- Spelta(6N)
- Aestivum (6N)
- Compactum (6N)

Full species names are found in table:7.4.

Why use µ-CT image analysis?

In the past, science has been greatly limited by the amount of data which could be processed in an experiment. In the last few decades the inclusion of computer science has reduced this bottleneck. Now, the challenge for many fields of research is producing more data and this is often cited as the major bottleneck in creating robust studies [9].

Many experiments aim to meet the demand for data by using high-throughput automated imaging systems [10, 11, 12]. These systems have, in the last decade, become a standard and accepted tool for data generation. However, they will only produce 2-dimensional data on a per-plant basis. Image processing research has had success in modifying these automated systems in order to produce a pseudo 3-dimensional structure using stero-imaging [13]. Even so, these techniques require destructive harvesting of materials and do not provide information of internal structure.

For decades medical research has found success with X-Ray imaging technology [14]. From this, plant science has been able to benefit from the wealth of prior knowledge and more and more studies are being augmented with the use of X-Ray/µ-CT imaging [1, 15, 16, 17, 18].

In this study, µ-CT has enabled the study of individual seeds of wheat, which is the product that plant breeders, commercial growers and farmers are truly interested in. Other imaging techniques could not provide as much detail, or in such a high throughput or quality.

Extracted Data

These samples come from over 70 plants and provided in excess of 2000 seeds for analysis which data was created based on. The traits recorded are labelled in figure:1.1 and are as follows:

- Length
- Width
- Depth
- Volume
- Surface Area
- Crease Depth / Volume

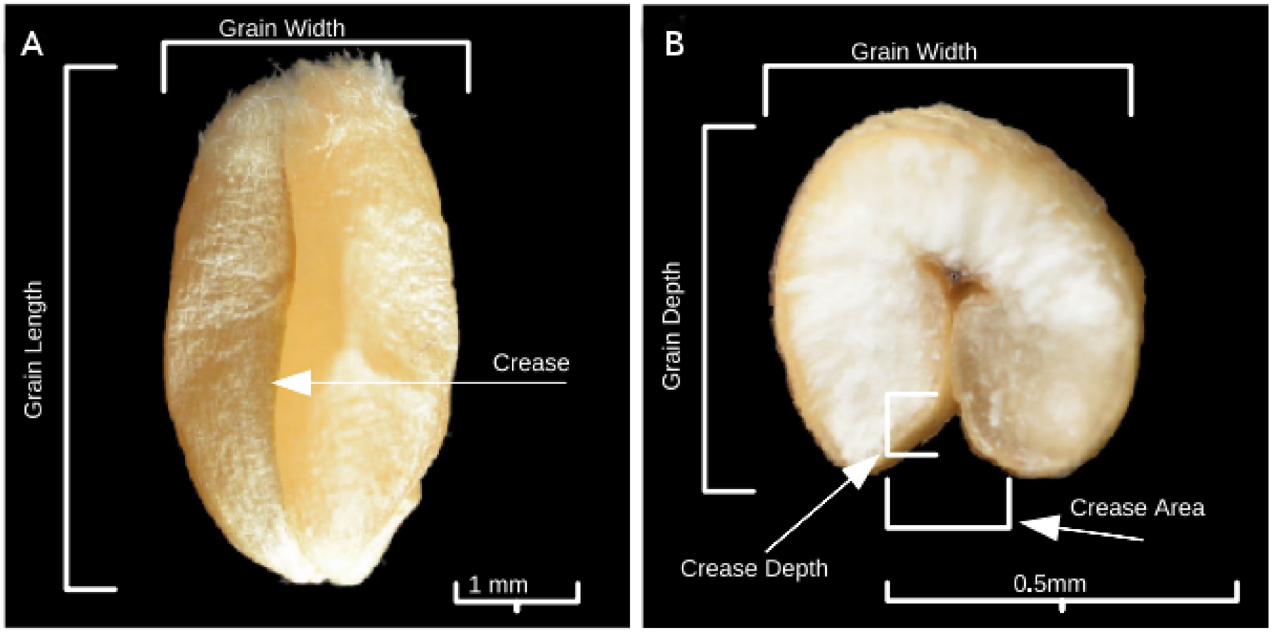


Figure 1.1: Wheat grain labelled (*left*), wheat grain cut in half (*right*), adapted from Hughes et al. [1]

Significance to Current Research

The biological interest in this area has been expressed in several areas of research [19], it is proposed that the key to unlocking diversity in the wheat genus lies in these ancestor, undomesticated species [20].

This research has the potential to be useful in several areas including: crop breeding; disease resistance; environmental stress. Each of these areas depend on making informed decisions in order to direct experiments. By producing information at an individual seed level, this study has been able to provide data that can offer suggestions of plant potential and behaviour.

Often, the most sought after traits are centred around thousand-grain-weight (TGW) as well as standard deviation of seed shapes. During harvesting, filters are used to only allow ideal shaped seeds through. This means that, potentially, despite a breed of wheat providing a high average volume of seed in reality much of it may go to waste if the shapes are not uniform. This research aims to alleviate this problem and provides low level information which is sorely required.

The individual images in figure:1.2 show, at a glance, the diversity and also the difference in the wild and cultivated (domesticated) species. This work allows for these differences to be quantified and evaluated into useful metrics for answering research based questions.

By better understanding the morphometric deviations in wheat species, more informed choices can be made when it comes to breeding wheat for the future and to fulfil ever-changing requirements.

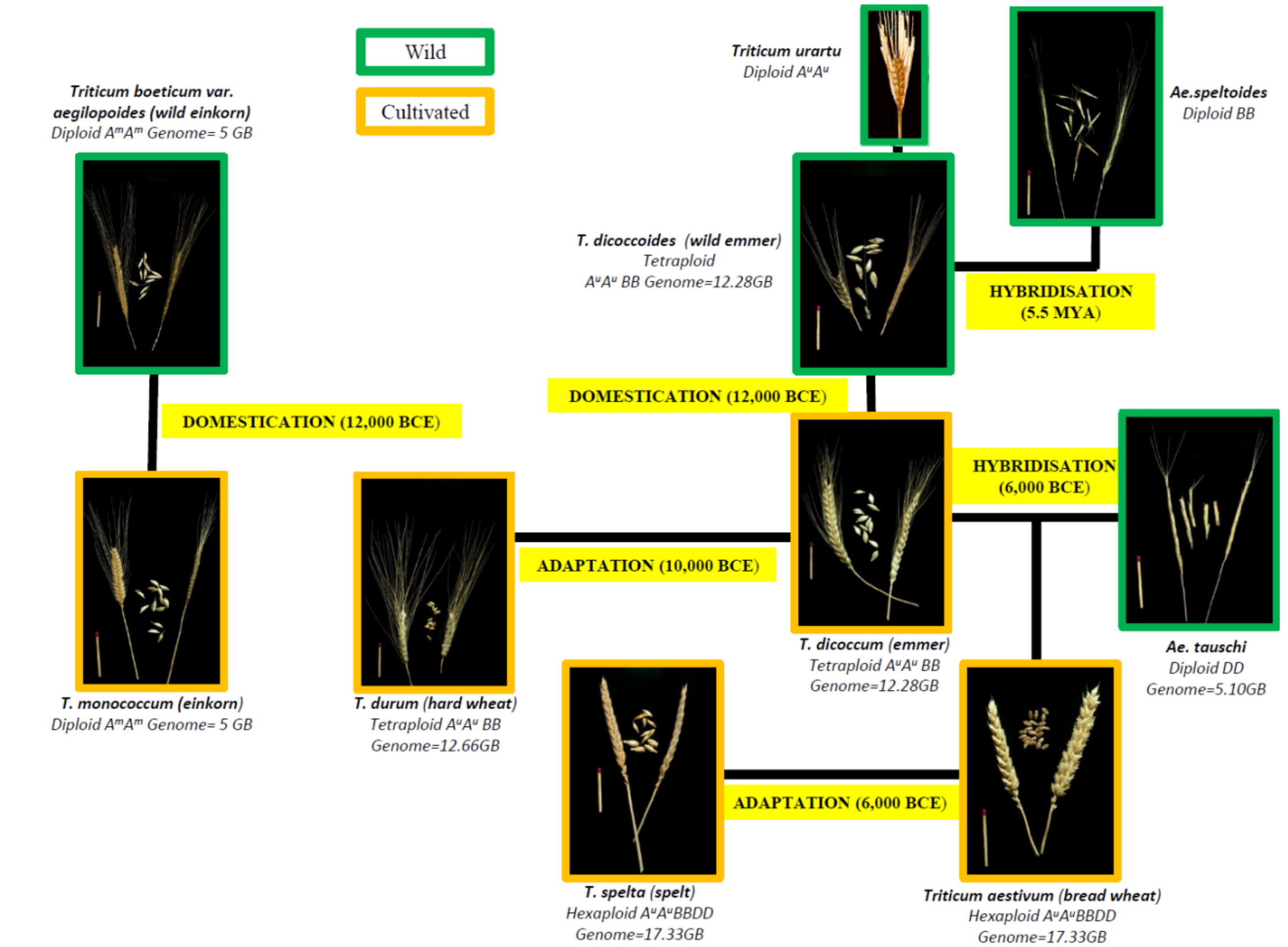


Figure 1.2: Phylogeny of wheat genotypes (Provided by Dr. Hugo Oliveira)

Aims and Objectives

The overarching aim of this project has been to create several pieces of software which aid in answering the biologically significant questions outlined. As well as to prove/disprove the hypothesis stated below.

The software created is robust in order to duplicate results and is flexible as to allow for further studies to be carried out and to use the same method.

Novel additions have been made to existing image analysis libraries in order to make them more flexible for this project. Figure:1.3 illustrates the range of diversity

Furthermore, the library written allows for easy data organisation and automation of otherwise difficult tasks such as concatenating data from multiple sources and graphing of information. Full documentation and integrated testing allows for a suite of tools which can be built upon in future and reduce the amount of effort required for similar studies to be carried out and analysed.

These aims have a focus on the phenotypic attributes generated from customised image analysis software [1] and can be seen in figure:1.1.

Hypothesis

To provide a full spectrum of analysis the null-hypothesis of this work is presented as investigating if there are morphometric differences in the seeds of several wheat varieties outlined in figure:1.2.

The comparison pairs are as follows:

1. Monococcum Wild and Monococcum Domesticate
2. Dicoccoides and Dicoccum
3. Spelta and Aestivum
4. Dicoccum and Durum
5. Monococcum Wild and Dicoccoides

Figure 1.3: Scans of wheat, showing diversity in Population, Compactum (6N) left, Durum right (4N)

Challenges Overview

The challenges which this project tackles come in two flavours: Computational and Biological. As such keen awareness of these is needed to appreciate the novelty of this work.

Biological Challenges

Previous studies have been able to demonstrate that variation in wheat grain morphology can be partially explained, in 2010 Gegas et al. demonstrated this through a 99.4% 2 component PCA [21]. However there is much left to do in terms of formal classifications and descriptions of these differences. This project deals with this problem through computational analysis.

Two effects run parallel in this study which requires acute biological knowledge of in order to make correct decisions:

1. The effects of ploidy in wheat.
2. The effects of domestication in wheat.

Hypothesis are required to take into account, both of these effects so as not to misidentify results.

Computational Challenges

Using µ-CT data in plant sciences is becoming more and more common [1, 15, 17, 22] and whilst a lot of studies focus on the traits of grains specifically no formal model has been created, no accepted data format. This is a data engineering problem and the methods described in this project address this.

Further to data organisation, proposals are made for the statistical analysis which should be used. This allows for studies to become more robust and repeatable, thus strengthening the studies overall.

The biological material used in this research is much more diverse a population than has been previously studied with µ-CT image analysis, this requires current computer vision methods to be adapted in order to be accurate.

Deliverables

This project provides three final deliveribles:

1. A flexible software suite written in *Python* that provides a standardised method for analysing and interpreting µ-CT data output.
2. A Graphical User Interface (GUI) which offers a point and click method for data gathering, graphing and manipulating µ-CT data, using the library from deliverable 1 as a backend.
3. Answers to the proposed questions (hypothesis), the *Results* and *Discussion* sections of this report provides this.

Chapter 2

Software Design, Implementation and Testing

This chapter outlines choices and methodologies employed in the software engineering aspect of this project, as well as highlighting the key functional requirements and implementation decisions.

Functional Requirements

Requirements for this project are split between software requirements for both the CT Analysing Library and the CT GUI Application and the research requirements (i.e. the answers to the proposed hypothesis). Here the requirements for the software are discussed:

Requirements for CT Analysing Library

These are the functional requirements for the Python library produced:

- Provide an OOP means to deal with data
- Make gathering of data simplified
- Handle Saving of data in a useable format
- Easily enable data transformations
- Perform hypothesis testing
- Process rejoining of split scans
- Handle Removing of erroneous data
- Enable matching data to external information
- Auto plot data (boxplots, histograms etc.)
- Allow easy filtering of data

Requirements for CT GUI Application

- Provide a intuitive user interface for working with CT data
- Allow a interaction with data without the need for programming
- Implement the Matplotlib plotting utility
- Easily join experiment data with CT data
- Use an MVC model
- Implement the CT Analysis Library
- Display data visually
- Dynamically create graphs
- Provide hypothesis testing

Software Development Methodology

This project made use of formal design methods and strict organisation whilst being flexible to change. Overall the design took a hybridised form in order to best suit the scientific environment which this domain specific software is built for.

Data analysis drove the direction of the project, as a result an agile methodology was adopted. Weekly sprints were implemented as a list of "todo's", these were written on a Monday morning based off of the previous week's list.

Critical self-evaluation was performed by means of a "one-man SCRUM" meeting, this is a technique which requires self-discipline in order to accurately find faults and areas for improvement [23].

Further to this, regular meetings with research staff, at the National Plant Phenomics Centre, allowed for a developer-client relationship which SCRUM defines as being key. During these meetings details of the research was discussed and ideas given as to how future experiments could proceed. This allowed for critical decisions to be made as to software design and overall structure.

Sprint Timeline

The implementation of this work was done in a agile method, treating each week as a sprint, for each of these a detailed organisational programme was created, discussed with supervisors and then used to formulate plans of action for following up on.

Sprint - Week 0

Initial planning was taken out, discussions with researchers at the National Plant Phenomics Centre (NPPC), to create a general set of targets and research goals.

A website was built in order to host weekly progress reports, this was used to share with supervisors and with staff at the NPPC. It also provided a list of discussion points to go through at weekly meetings.

A bug was identified in image analysis software, this was raised to be fixed a later date.

A literature review was taken out to highlight the novelty of this research, as well as current trends in the field in terms of analysis and known/accepted information.

Sprint - Week 1

Initial running of grain analysis software was performed multiple times, as per instructions in literature [1], multiple parameters for minimum and maximum expected sizes of grains needed to be tested. Data which was produced was very noisy and would require further work.

Spike work was carried out in investigating the potential of using a skeletonising method on the wheat spikes. The hope behind this was to simplify structure in a three dimensional structure, of 1 pixel thick lines. This technique is often used in plant root analysis [24, 25]. Experimentation with these methods were not trivial and a decision was made to revisit if time permitted at the end of the project.

A key function of the CT Analysis Library was created, showing in listing:1. The method enables additional experiment information to be joined with extracted seed data. This provides a way of grouping seeds into more useful groupings than just their scanning data.

An issue was identified in the choices of testing which are typically used in these studies, the use of ANOVA and Student's T-Test, for example, are best used with parametric data, that is to say data where distribution is normally distributed. Further reading into this presented the use of Box Cox transforms as a method to counter these issues.

An issue was raised; a method for visualising outliers in the data could provide greatly beneficial insight into finding errors. If time was available at the end of the project, this would be explored further.

Sprint - Week 2

Spikes of wheat are scanned in two separate imaging cycles sometimes, this is because the tube used by the µ-CT machine are 10cm tall and often a spike will exceed this. In order for full analysis to be carried out these separate scans need to be rejoined. A method for doing this was added to the python Analysis Library.

An initial Model-View-Controller (MVC) model was constructed for how a GUI might take form around. Using this several wire-frames were created

An idea for how data could be cleaned was created by using the information found in previous studies [1]. From this reported data minimum and maximum expected size could be assumed for wheat grains in terms of volume.

Decisions to move the CT Grain Analysing Library towards an object orientated model were made during this sprint, after evaluating the potential of a functionally programmed model or a object one. Handling everything in terms of classes was decidedly easier in terms of understanding how to use the library.

Sprint - Week 3

A lot of progress was made on constructing a GUI here, dynamic plotting was put together as a proof of this concept. Histograms of the data were able to be made by using the mouse to select which attribute to measure.

GUI elements such as navigation, data tabs and file menus were added to make more clear to the user how functionality should work

Sprint - Week 4

Sprint - Week 5

Sprint - Week 6

Sprint - Week 7

Sprint - Week 8

Sprint - Week 9

Sprint - Week 10

Sprint - Week 11

Sprint - Week 12

Sprint - Week 13

Language Choices

Both the CT Analysing Library and the CT Analysing GUI are implemented using the Python programming language, it has been developed and tested in versions 3.5 and 3.6 (Python 2 is not supported at all by this project).

In scientific programming three of the most commonly used languages are Python, R and MATLAB [26].

These three languages are able to provide all the features which this project requires. However Python was chosen for several reasons.

MATLAB could not be used as a potential language due to it being pay to use software, as this project aims to be accessible, the cost of software would greatly reduce the scope of access.

R is a valid candidate, it provides all of the statistical capabilities required by the project, it also provides packages for creating GUI based applications, it is fast and it is widely used in scientific computing and data science.

The main deciding factor is Python's wealth of resources, adoption rate and the developer of this project being vastly more experienced with Python's ecosystem than R's.

Designing Process

Through meetings and emails, the agile principles of communication over comprehensive documentation was used. Where conversations were decidedly much more beneficial than complex planing prior to developing a product.

Graphical elements, such as the graphing functionality of the CT Analysing Library and the CT GUI Application were sketched using wire-frames whilst in meetings where the potential users (clients) could provide their ideas.

In figure:2.1 an example of the wire-frames created during meetings is show (A), next to it is displayed the final look of the loading window (B).

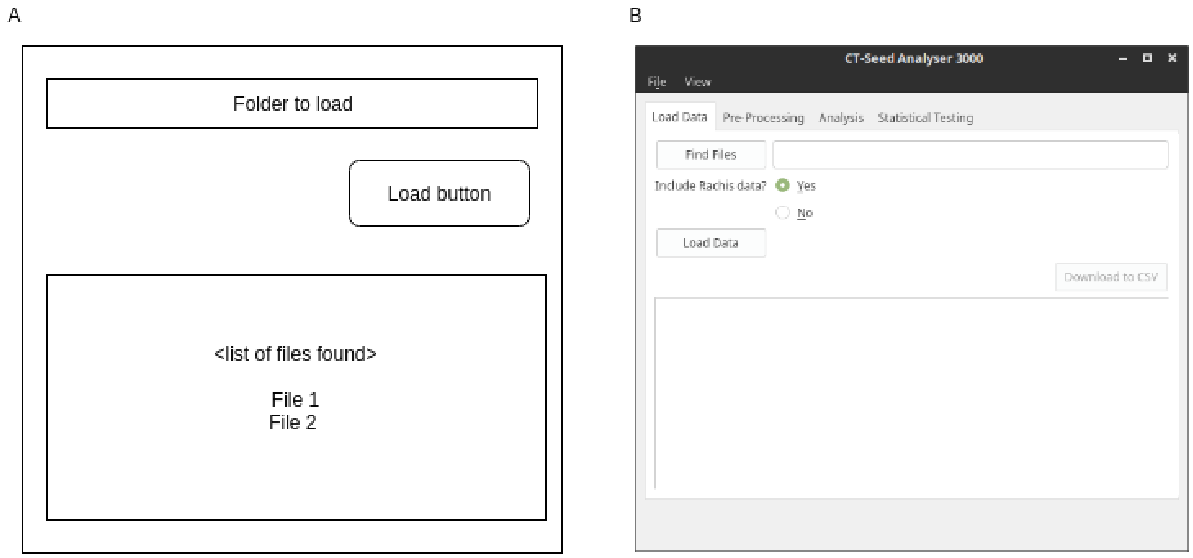


Figure 2.1: Wire-frame of the GUI loading data window

Similarly, figure:2.2 provides the initial wire-frame (A) of how the analysis window could have looked and what kind of GUI elements would be required, again, next to it is the final analysis window (B)

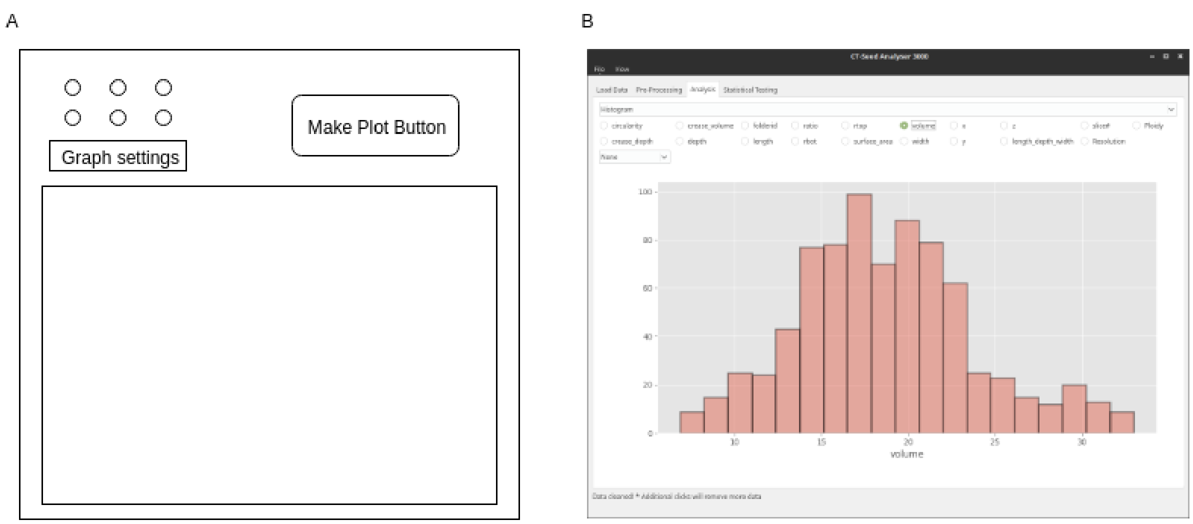


Figure 2.2: Wire-frame of the GUI analysis window

Documentation

Whilst an agile approach was used, some documentation was created for use with the CT Analysing Library.

The provided CT Analysing Library comes with "human-readable" format. Where most documentation generators (Doxygen, Pydocs, Javadocs etc.) implement very well structured and comprehensive documentation, the output is generally not very friendly and easy to read. Particularly for non-career-programmers. A core feature of these provided software implementations are that they are well suited for a biologist, researcher or statistician to use.

This documentation generator was purpose created, implemented in LISP and provided in listing:3.

Beyond this, inline commenting is provided for supplied software. Keeping in line with the agile development ethos the software is self-documented and self-evident. A brief example of this is shown in listing:4

Documentation for the CT GUI application is provided as a visual user guide, and provides sample data for the user to test with.

Software Library Choices

The software libraries used for this project focus around data manipulation, where possible core libraries of the Python language were used and only well supported, established and documented libraries were chosen. Software support is a major requirement for reproducible results.

All software packages used in the Analysing Library are required by the CT Analysing GUI as the Library is a dependency of it. The GUI has a single separate requirement *PyQT5*. Table:7.1 contains a full listing of all software used and required by this project.

Numpy

The Numpy library is one of the most commonly used additions to the Python ecosystem, it is fundamental to many data science projects. Here it is used to handle data lists, arrays and structures. There is no viable alternative to this package and it is required by Scipy and Statsmodels.

Matplotlib/Seaborn

Matplotlib acts as the plotting backend for the project. The Seaborn package acts as a porcelain for matplotlib and makes graph creation and decoration much easier.

Scipy

Data transforms such as Box Cox and PCA are dependant on the functions of the Scipy library. Alternatives are available, however this is the most well established and often used library for these functions.

Pandas

Pandas is used to read the CSV files which the raw data is stored in. This library converts and stores data in dataframes which are used throughout this project to manipulate data.

Xlrd

This extension library is required in order to read Microsoft encoded files. Extra experiment information can be provided with the "xlsx" extension.

Statsmodels

Bayesian hypothesis testing is provided through this library.

PyQT5

There were many options for creating a user interface in Python, the language provides its own core library via the *TKinter* module. However PyQT is a port of the QT framework, one of the most widely used libraries for GUIs in software development. It is cross platform, robust and has excellent documentation and user-guides.

Implementation

Strict software engineering principles were applied during creation of this project. The use of standards, design patterns and code-linters have been used throughout to minimise the possibility of errors and to create wholly extendable software. These devices enable understandable and self-documented code allowing future users to quickly start using the provided packages.

CT Analysing Library Design Pattern

The CT Analysing Library uses a Singleton style design pattern. A single data object is created from a *CTData* class.

A very functional paradigm is used by this library. By applying mapping and filter style functions data elements can be passed to the supporting modules: *data_transforms.py*; *graphing.py*, *statistical_tests.py*. These modules enable scientific functions to be applied to the *CTData* object. A UML style class diagram is shown in figure:2.3, here the interactions of the classes can be seen, as well as their internal functions.

CT GUI Application Design Pattern

The Model-View-Controller (MVC) design pattern is one of the most commonly structures for creating user interfaces. It allows for the user's view/interface code to be separated from the model, the code which changes the data. The model and the view communicate and update each other via the controller element of the design.

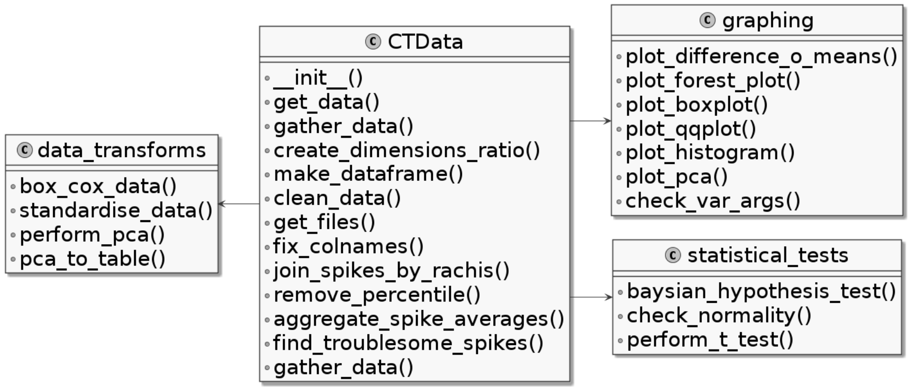


Figure 2.3: CT Analysing Library UML

The QT framework provides "connectors" which act as triggers/activations for functions, these are set off by the user providing either keyboard or mouse based input.

Standards

The main standard adhered to for software provided by this project is the PEP8 style guide [27]. The principle behind this coding style, as stated by Guido van Rossum, is "Code is read much more often than it is written". This makes this styling guide perfect for the chosen agile methodology of self-evident documentation in the software.

In addition to PEP8, a Python code linter Flake8 has been used to prevent "code smells", bad formatting, incorrect white space usage etc.

Version control

This project has used Git version 2.7.4 throughout. The structure of the project has been as submodules of a larger project.

By using submodules the CT Grain Analysing Library could be kept in sync with the GUI aspect of the project.

Additionally, *setup.py* has been used to provide installation of the library, the code for this can be seen in listing:5. Using *setup.py* provides a quick and easy way for any user to install the software, along with any dependencies.

Issue Tracking

Issues were tracked during the project, both in personal notes and in the Git interface as illustrated in figure:2.5

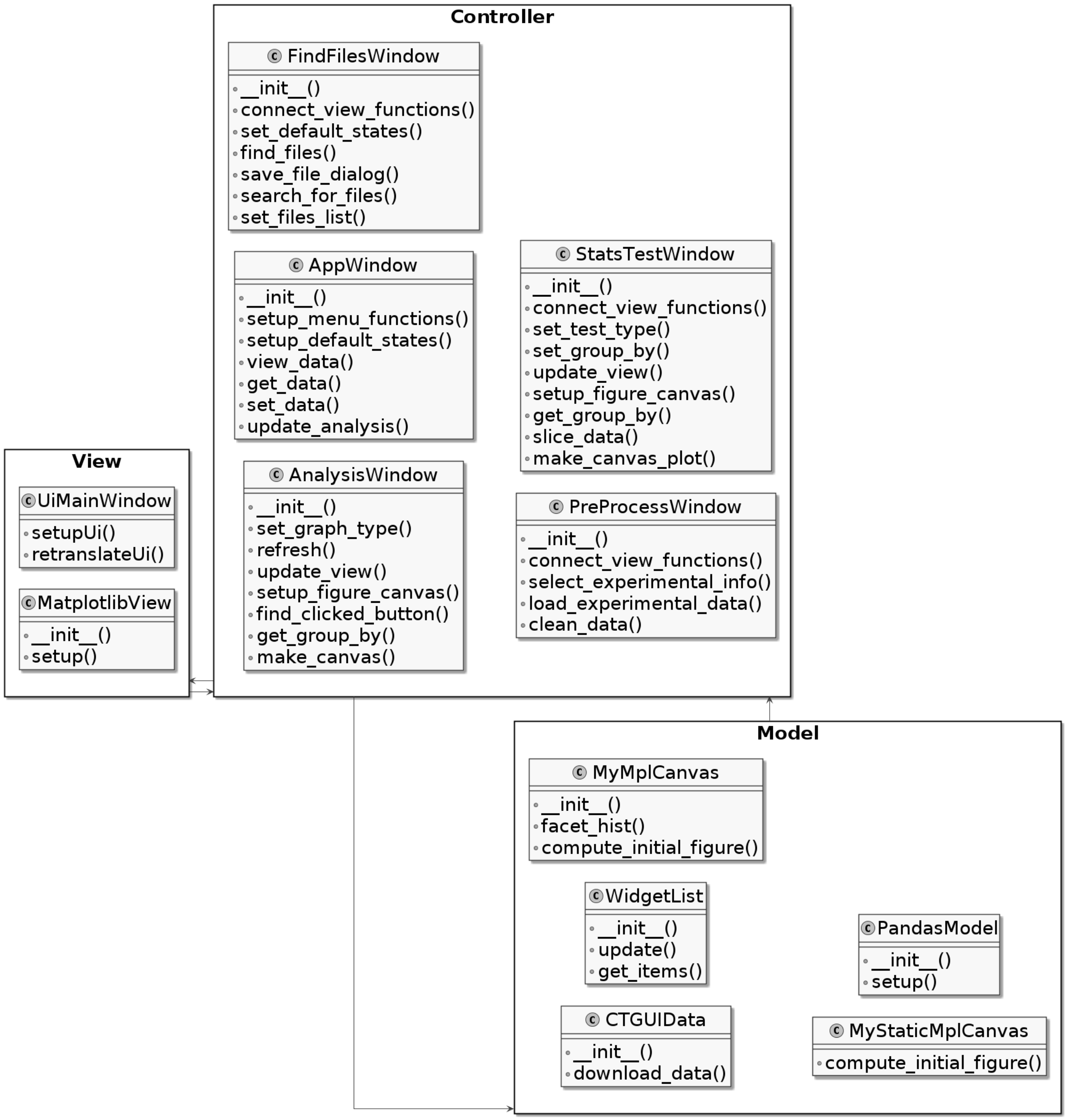


Figure 2.4: CT Analysing GUI UML

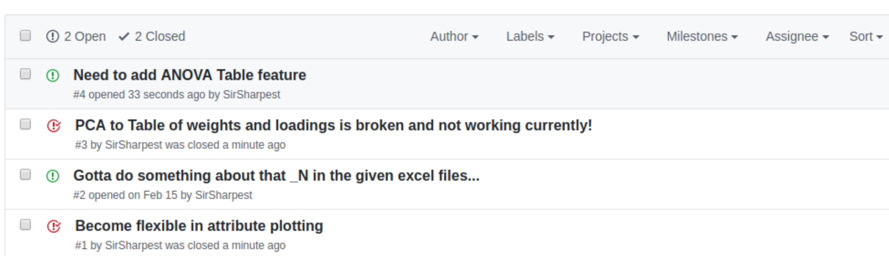


Figure 2.5: Github Issue Tracking

Library Versioning

The development of this project was performed using the Python *virtualenv.* This is a virtual environment package which Python offers, it allows for an isolated working copy of the project.

By developing in this manner, libraries were ensured to be using the correct versions required by the software.

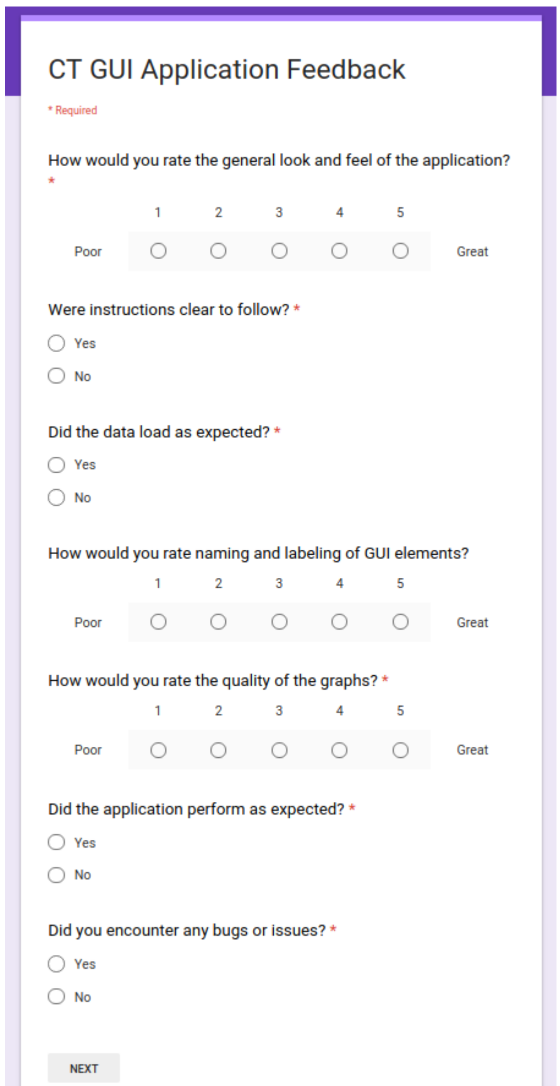
Testing

Testing was performed both in acceptance testing by using user feedback, the functional requirements and the ability to use the software to answer the hypothesis of the research elements of this project. Further to this, unit testing was performed to allow for automated testing as well as test-driven-development of features.

Feedback Forms

Feedback and constructive suggestions were made by researchers at the National Plant Phenomics Centre, these were submitted via the Google forms service.

These provided a method of acceptance testing by those who would be using the software to help with investigating data. Page 1 of the given form is shown in figure:2.6. This form was completed by 3 researchers at the National Plant Phenomics Centre, feedback was very positive overall.


CT GUI Application Feedback

* Required

How would you rate the general look and feel of the application? *

	1	2	3	4	5	
Poor	○	○	○	○	○	Great

Were instructions clear to follow? *

- ○ Yes
- ○ No

Did the data load as expected? *

- ○ Yes
- ○ No

How would you rate naming and labeling of GUI elements?

	1	2	3	4	5	
Poor	○	○	○	○	○	Great

How would you rate the quality of the graphs? *

	1	2	3	4	5	
Poor	○	○	○	○	○	Great

Did the application perform as expected? *

- ○ Yes
- ○ No

Did you encounter any bugs or issues? *

- ○ Yes
- ○ No

NEXT

Figure 2.6: CT Feedback form

Unit Testing CT Analysing Library

The unit tests for the CT Analysing Library were straightforward, using the *PyTest* framework and a subset of data from a data set, these tests assert that features are implemented correctly and that the correct results are given.

Table 2.1: Output of *pytest* Unit Tests and results for CT Analysing Library

Result	**Test**
Passed	CTData.py::test_aggregate_spike_averages
Passed	CTData.py::test_clean_data_maximum_removed
Passed	CTData.py::test_clean_data_minimum_removed
Passed	CTData.py::test_load_additional_data
Passed	CTData.py::test_load_additional_data_no_data
Passed	CTData.py::test_load_data
Passed	CTData.py::test_NoDataFoundException
Passed	Data_transforms.py::test_box_cox_data
Passed	Data_transforms.py::test_pca_to_table
Passed	Data_transforms.py::test_perform_pca
Passed	Data_transforms.py::test_standardise_data
Passed	Graphing.py::test_plot_boxplot_as_dataframe
Passed	Graphing.py::test_plot_boxplot_as_object
Passed	Graphing.py::test_plot_difference_of_means
Passed	Graphing.py::test_plot_histogram_as_dataframe
Passed	Graphing.py::test_plot_histogram_as_object
Passed	Graphing.py::test_plot_pca
Passed	Graphing.py::test_plot_qqplot
Passed	Statistical_tests.py::test_baysian_hypothesis_test
Passed	Statistical_tests.py::test_t_test
Passed	Statistical_tests.py::test_test_normality

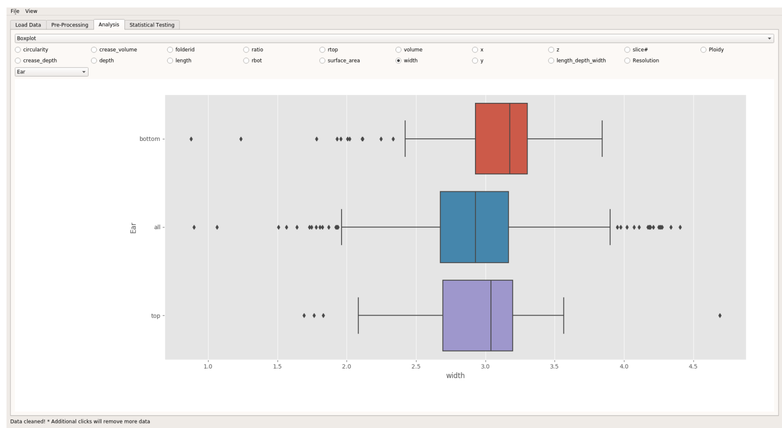
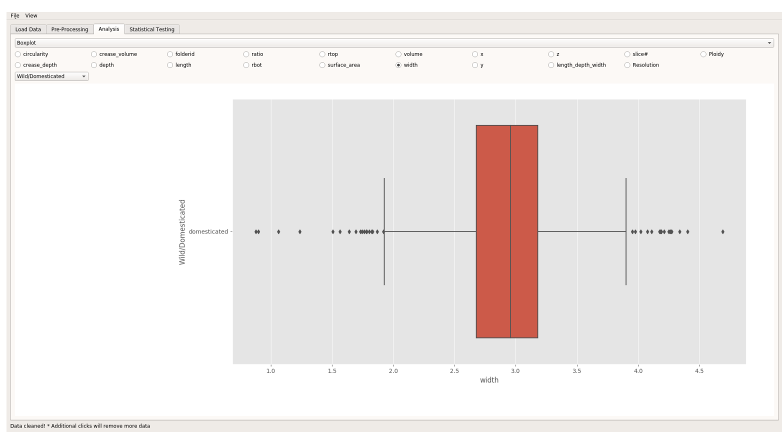
Unit Testing CT GUI Application

The unit testing used for the CT GUI Application was more sophisticated than that of the Library. This testing required visual confirmation that figures and graphs generated were displayed correctly and that they showed what the user would expect, given the data.

To do this a *PyTest* plugin was used, *QtBot* which provides simulated user input. This allows for the GUI to be thoroughly tested, automatically.

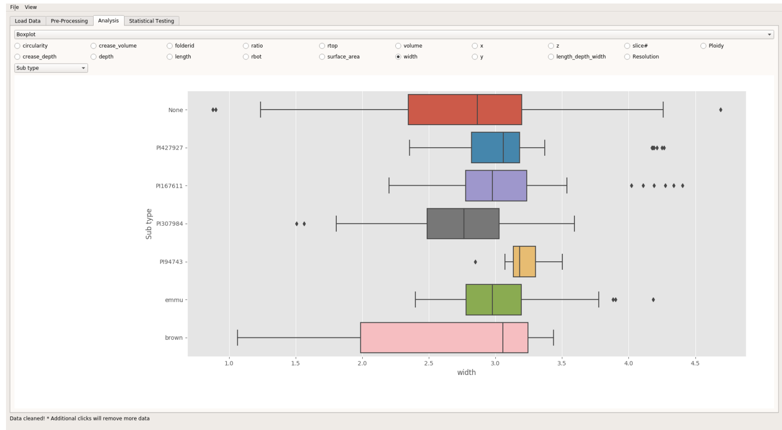
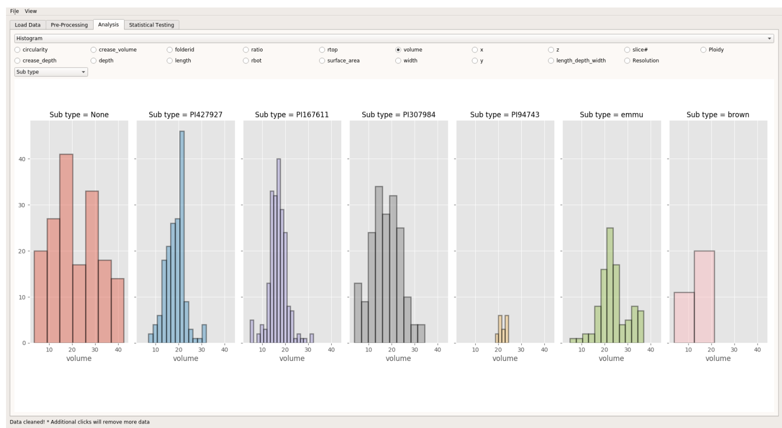
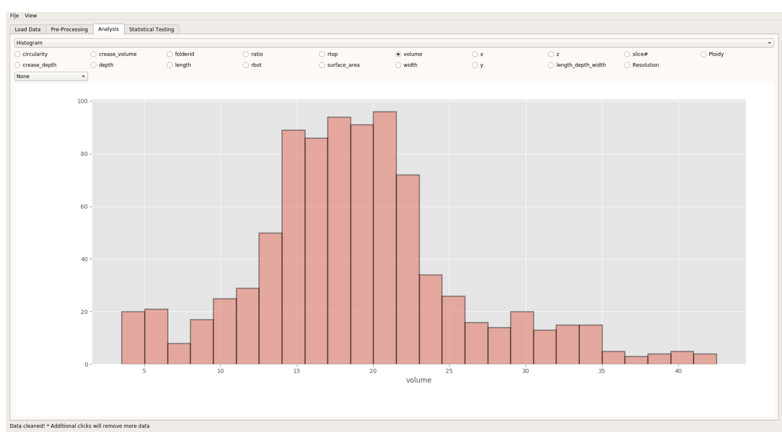
In table:2.2 the results of the automated testing is given along side an image of several of the tests, tests of the same graphs but with different parameters were also generated and manually verified and provided as supplemental data.

Table 2.2: Output of *pytest* Unit Tests and results for CT GUI Application

Result	**Test**	**Image**
Passed	analysis.py:: box_groupby_1_rb_1	
Passed	analysis.py:: box_groupby_2_rb_2	

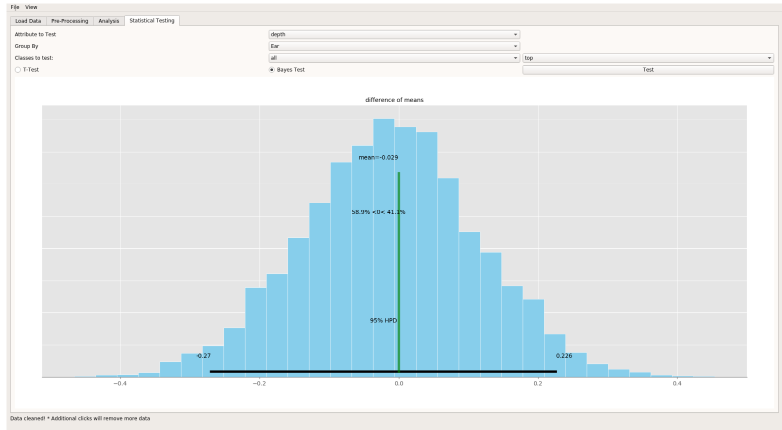
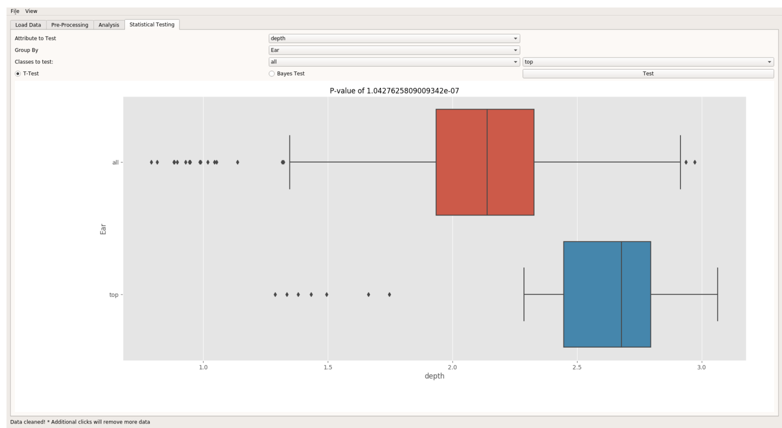
Continued on next page

Continued from previous page

Result	Test	Image
Passed	analysis.py:: box_rb_1	
Passed	analysis.py:: hist_groupby_1_rb_1	
Passed	analysis.py:: hist_rb_1	

Continued on next page

Continued from previous page

Result	**Test**	**Image**
Passed	hypothesis_tests.py:: bayesg1_att_1	
Passed	hypothesis_tests.py:: tg1_att_1	
Passed	analysis.py:: box_rb_2	N/A
Passed	analysis.py:: hist_groupby_1_rb_2	N/A
Passed	analysis.py:: hist_rb_2	N/A
Passed	analysis.py:: loads	N/A
Passed	hypothesis_tests.py:: bayesg1_att_2	N/A
Passed	hypothesis_tests.py:: bayesg2_att_1	N/A
Passed	hypothesis_tests.py:: bayesg2_att_2	N/A
Passed	hypothesis_tests.py:: tg1_att_2	N/A
Passed	hypothesis_tests.py:: tg2_att_1	N/A
Passed	hypothesis_tests.py:: tg2_att_2	N/A

Continued on next page

Continued from previous page

Result	**Test**	**Image**
Passed	hypothesis_tests.py:: loads	N/A
Passed	GUI.py:: startup	N/A
Passed	load_data.py:: load_data_with_rachis	N/A
Passed	load_data.py:: load_data_without_rachis	N/A
Passed	preprocessing.py:: clean_data_remove_large	N/A
Passed	preprocessing.py:: clean_data_remove_none	N/A
Passed	preprocessing.py:: clean_data_remove_small	N/A
Passed	preprocessing.py:: clean_data_remove _small_and_large	N/A
Passed	preprocessing.py:: load_additional_data	N/A
Passed	preprocessing.py:: load_additional_data _expected_fail	N/A

Chapter 3

Methods and Solutions

Data Pipeline

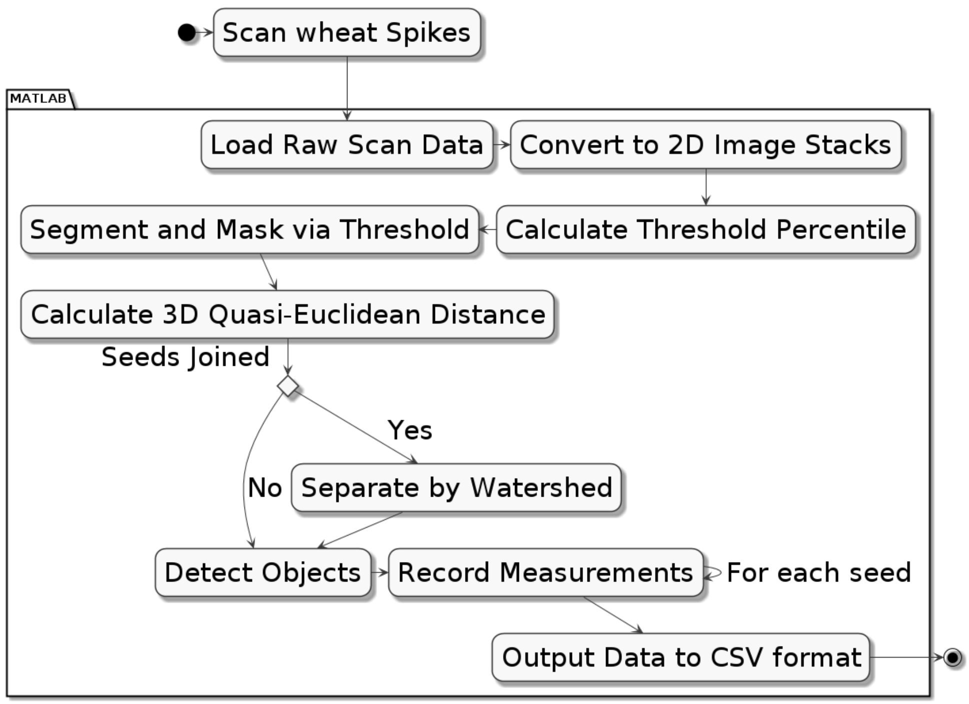


Figure 3.1: Image Processing Pipeline

Image Analysis Methods

New Watershed Algorithm

In order to solve the problem of misidentified and joint seeds, from the primitive collection, a *quasi-euclidean* distance transform was implemented into the analysis pipeline (figure:). This provided much better results than the previous *chessboard* transform which had been successful on more uniform data in previous studies [1].

Quasi-Euclidean algorithm

This algorithm measures the total euclidean distance along a set of horizontal, vertical and diagonal line segments [28].

$$|x_1 - x_2| + (\sqrt{2} - 1), |x_1 - x_2| > |y_1 - y_2| \, (\sqrt{2} - 1) \, |x_1 - x_2| \, , \text{otherwise} \tag{3.1}$$

In order to apply this to a 3D space Kleinberg's method is used [29]. This allows for nearest neighbour pixels to be sorted by k-dimensional trees and enabling fast distance transforms via Rosenfeld and Pfaltz's *quasi-euclidean* method stated in equation:3.1.

Effect of Enhanced Watershed algorithm

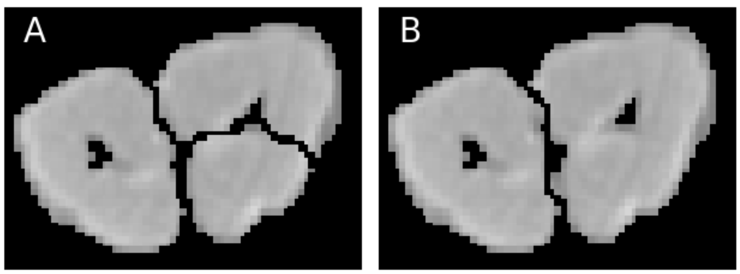


Figure 3.2: *A* showing the chessboard method, *B* improved quasi-euclidean method

Extracted Grains

CT Analysing Library Methods

Adding Spike Experiment Information

CT GUI Application Methods

Data Analysis Methods

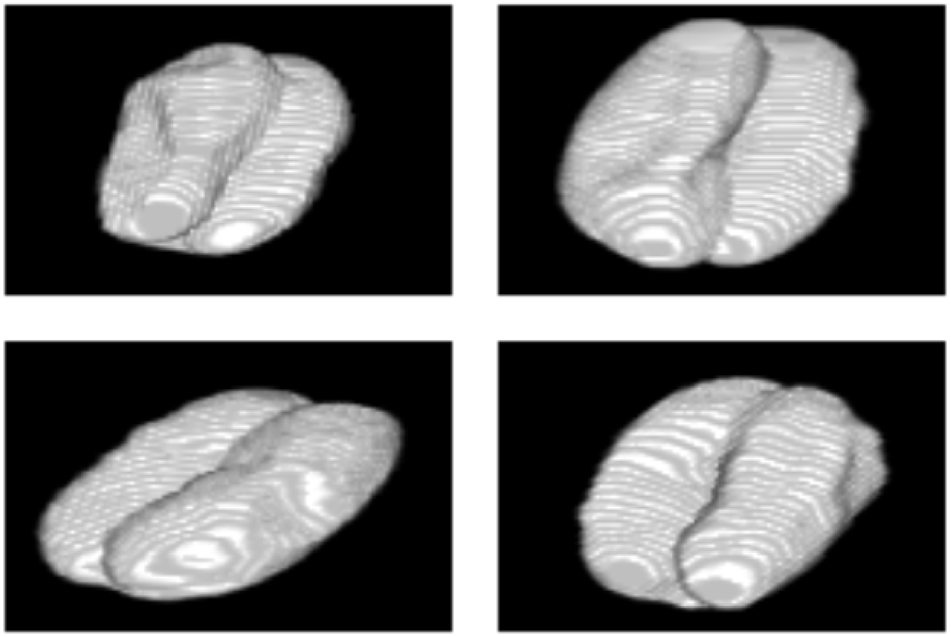


Figure 3.3: Individual Wheat grains, rendered in 3D

```
def get_spike_info(grain_df, excel_file, join_column='Folder#'):

    # Make a copy as we don't want to change the original
    df = grain_df.copy(deep=True)

    # Grab the linking excel file
    info = pd.read_excel(excel_file,
                         index_col='Folder#')

    # These are the features to grab
    features = ['Hulled/Naked', 'Common name', 'Genome', 'Ploidy',
                'Wild/Domesticated', 'Sample name', 'Sub type', 'Ear']

    # Lambda to look up the feature in excel spreadsheet
    def look_up(x, y): return info.loc[x['folderid']][y]

    # Lambda form a series (data row) and apply it to dataframe
    def gather_data(x): return pd.Series([look_up(x, y) for y in features])

    df[features] = df.apply(gather_data, axis=1)

    # Return the copy
    return df
```

Listing 1: Spike Information Joining Algorithm

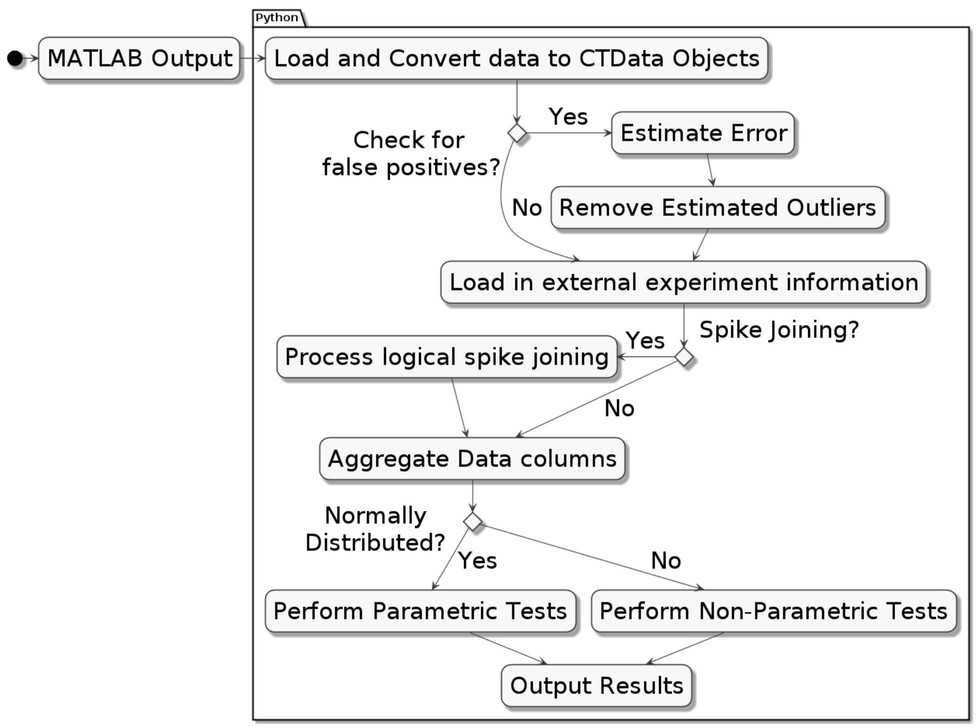


Figure 3.4: How data is integrated with the CT Analysing Library

Chapter 4

Results

Chapter 5

Discussion

Similar Research

Alternate Solutions

Chapter 6

Critical Evaluation

Organisational Methods

Relevance to Degree

Time Management

Collaborative Work

Other Issues

Chapter 7

Appendix

Software Packages Used

Libraries

Table 7.1: Software libraries used

Seaborn	Scipy	Sklearn
MATLAB Image Processing Toolbox	Numpy	Matplotlib
Statsmodels	Pymc3	Xlrd
PyQt5	gcc	Pip

Tools

Table 7.2: Software tools used

MATLAB	Python Debugger (PDB)	IPython
Emacs	git	org-mode
Tomviz	ImageJ	PlantUML

Glossary

Table 7.3: Dictionary for Terms and acronyms

Term	**Definition**
µ-CT	Micro Computed Tomography
Genotype	A genetically distinct individual or group
Phenotype	A physical/measurable trait
Alleles	A variant of a gene
Genus	Classification ranking, below the *family* grouping
Genome	The complete genetic make up of an organism, which defines its individuality
Morphometric	The shape and form of an organism
GUI	Graphical User Interface
PCA	Principal Component Analysis
Spike	A singular stalk of wheat
Spikelet	A group of seeds all forming from the same node in a spike
MVC	Model View Controller - A design pattern for GUIs
OOP	Object Orientated Programming

Wheat Varieties

Table 7.4: Dictionary for Wheat names used

Used name	**Species name**
Monococcum	*Triticum monococcum*
Monococcum Wild	Triticum Boeticum
Tauschii	Aegilops tauschii
Durum	Triticum Durum
Dicoccoides	Triticum Dicoccoides
Dicoccum	Triticum Dicoccum
Ispahanicum	Triticum Ispahanicum
Timopheevii	Triticum Timopheevii
Spelta	Triticum Spelta
Aestivum	Triticum Aestivum
Compactum	Triticum Compactum

Code Segments and Examples

MATLAB Watershedding

```
function [W] = watershedSplit3D(A)
  % Takes image stack A and splits it into stack W
  % Convert to BW
  bw = logical(A);
  % Create variable for opening and closing
  se = strel('disk', 5);
  % Minimise object missshapen-ness
  bw = imerode(bw, se);
  bw = imdilate(bw, se);
  % Fill in any left over holes
  bw = imfill(bw,4,'holes');
  % Use chessboard for distance calculation for more refined splitting
  chessboard = -bwdist(~bw, 'quasi-euclidean');
  % Modify the intensity of our bwdist to produce chessboard2
  mask = imextendedmin(chessboard, 2);
  chessboard2 = imimposemin(chessboard, mask);
  % Calculate watershed based on the modified chessboard
  Ld2 = watershed(chessboard2);
  % Take original image and add on the lines calculated for splitting
  W = A;
  W(Ld2 == 0) = 0;
end
```

Listing 2: MATLAB Watershedding function

Custom Documentation Generator

```
(defun populate-org-buffer (buffer filename root)
  (goto-char (point-min))
  (let ((to-insert (concat "* " (replace-regexp-in-string root "" filename) "\n") ))
    (while (re-search-forward
            (rx (group (or "def" "class"))
                space
                (group (+ (not (any "()"))))
                (? "(" (* nonl) "):" (+ "\n") (+ space)
                   (= 3 "\"")
                   (group (+? anything))
                   (= 3 "\"")))
            nil 'noerror)
      (setq to-insert
            (concat
             to-insert
             (if (string= "class" (match-string 1))
                 "** "
               "*** ")
             (match-string 2)
             "\n"
             (and (match-string 3)
                  (concat (match-string 3) "\n")))))helm-semantic-or-imenu
    (with-current-buffer buffer
      (insert to-insert))))
(defun org-documentation-from-dir (&optional dir)
  (interactive)
  (let* ((dir  (or dir (read-directory-name "Choose base directory: ")))
         (files (directory-files-recursively dir "\py$"))
         (doc-buf (get-buffer-create "org-docs")))
    (dolist (file files)
      (with-temp-buffer
        (insert-file-contents file)
        (populate-org-buffer doc-buf file dir)))
    (with-current-buffer doc-buf
      (org-mode))))
```

Listing 3: Custom lisp code for generating easy to read documentation

Self-Documenting Code Example

```
def get_spike_info(self, excel_file, join_column='Folder#'):
    """
    This function should do something akin to adding additional
    information to the data frame

    @note there is some confusion in the NPPC about whether to use
    folder name or file name as the unique id when this is made into
    end-user software, a toggle should be added to allow this

    @param excel_file a file to attach and read data from
    @param join_column if the column for joining data is
    different then it should be stated
    """
    try:
        # Grab the linking excel file
        info = pd.read_excel(excel_file,
                             index_col='Folder#')

        features = list(info.columns)
        # Lambda to look up the feature in excel spreadsheet
        def look_up(x, y): return info.loc[x['folderid']][y]

        # Lambda form a series (data row) and apply it to dataframe
        def gather_data(x): return pd.Series(
            [look_up(x, y) for y in features])

        self.df[features] = self.df.apply(gather_data, axis=1)
    except KeyError as e:
        print('Error matching data')
        print(e)
        raise NoDataFoundException
    except AttributeError as e:
        print(e)
        raise NoDataFoundException
```

Listing 4: Example of code documentation and readability from *data_ transforms.py*

Setup.py

```
from setuptools import setup
setup(name='CT_Analysing_Library',
      version='0.2',
      description='Library used for CT grain analysis at the NPPC',
      url='https://github.com/SirSharpest/CT_Analysing_Library',
      author='Nathan Hughes',
      author_email='nathan1hughes@gmail.com',
      license='MIT',
      packages=['ct_analysing_library'],
      install_requires=['pandas',
                        'numpy',
                        'matplotlib',
                        'seaborn',
                        'scipy',
                        'sklearn',
                        'statsmodels',
                        'pymc3',
                        'xlrd'],
      zip_safe=True)
```

Listing 5: The *setup.py* configuration for the CT Analyser Library

References

[1] N. Hughes, K. Askew, C. P. Scotson, K. Williams, C. Sauze, F. Corke, J. H. Doonan, and C. Nibau, "Non-destructive, high-content analysis of wheat grain traits using X-ray micro computed tomography," *Plant Methods*, vol. 13, 2017.

[2] H. Özkan, A. Brandolini, R. Schäfer-Pregl, and F. Salamini, "AFLP Analysis of a Collection of Tetraploid Wheats Indicates the Origin of Emmer and Hard Wheat Domestication in Southeast Turkey," *Molecular biology and evolution*, vol. 19, no. 10, pp. 1797–1801, oct 2002. [Online]. Available: http://academic.oup.com/mbe/article/19/10/1797/1259152

This paper discusses the origin of wheat domestication and provides evidence for it's origin in society

[3] B. Shiferaw, M. Smale, H.-J. Braun, E. Duveiller, M. Reynolds, and G. Muricho, "Crops that feed the world 10. Past successes and future challenges to the role played by wheat in global food security," *Food Security*, vol. 5, no. 3, pp. 291–317, jun 2013. [Online]. Available: http://link.springer.com/10.1007/s12571-013-0263-y

[4] E. J. Finnegan, B. Ford, X. Wallace, F. Pettolino, P. T. Griffin, R. J. Schmitz, P. Zhang, J. M. Barrero, M. J. Hayden, S. A. Boden, C. A. Cavanagh, S. M. Swain, and B. Trevaskis, "Zebularine treatment is associated with deletion of <i>FT</i> - <i>B1</i> leading to an increase in spikelet number in bread wheat," *Plant, Cell & Environment*, apr 2018. [Online]. Available: http://doi.wiley.com/10.1111/pce.13164

[5] I. Aleksejeva, "EU Experts' Attitude Towards Use of GMO in Food and Feed and Other Industries," *Procedia - Social and Behavioral Sciences*, vol. 110, pp. 494–501, jan 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187704281305533X

[6] T. Twardowski and A. Małyska, "Uninformed and disinformed society and the GMO market," *Trends in Biotechnology*, vol. 33, no. 1, pp. 1–3, jan 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167779914002327

[7] M. Lynas, "Opinion | With G.M.O. Policies, Europe Turns Against Science - The New York Times." [Online]. Available: https://www.nytimes.com/2015/10/25/opinion/sunday/with-gmo-policies-europe-turns-against-science.html?ref=todayspaper

[8] G. Charmet, "Wheat domestication: Lessons for the future," *Comptes Rendus - Biologies*, vol. 334, no. 3, pp. 212–220, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.crvi.2010.12.013

[9] R. T. Furbank and M. Tester, "Phenomics - technologies to relieve the phenotyping bottleneck," *Trends in Plant Science*, vol. 16, no. 12, pp. 635–644, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.tplants.2011.09.005

[10] B. Naumann, M. Eberius, and K.-J. Appenroth, "Growth rate based doseresponse relationships and EC-values of ten heavy metals using the duckweed growth inhibition test (ISO 20079) with Lemna minor L. clone St," *Journal of Plant Physiology*, vol. 164, no. 12, pp. 1656–1664, dec 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0176161706003154

[11] B. M. Prasanna, J. L. Araus, J. Crossa, J. E. Cairns, N. Palacios, B. Das, and C. Magorokosho, "High-Throughput and Precision Phenotyping for Cereal Breeding Programs," in

Cereal Genomics II. Dordrecht: Springer Netherlands, 2013, pp. 341–374. [Online]. Available: http://link.springer.com/10.1007/978-94-007-6401-9{_}13

[12] J. F. Humplík, D. Lazár, A. Husičková, and L. Spíchal, "Automated phenotyping of plant shoots using imaging methods for analysis of plant stress responses a review," *Plant Methods*, vol. 11, no. 1, p. 29, dec 2015. [Online]. Available: http://www.plantmethods.com/content/11/1/29

[13] J. Roussel, F. Geiger, A. Fischbach, S. Jahnke, and H. Scharr, "3D Surface Reconstruction of Plant Seeds by Volume Carving: Performance and Accuracies," *Frontiers in Plant Science*, vol. 7, no. June, pp. 1–13, 2016. [Online]. Available: http://journal.frontiersin.org/Article/10.3389/fpls.2016.00745/abstract

[14] G. Wang, H. Yu, and B. De Man, "An outlook on x-ray CT research and development," *Medical Physics*, vol. 35, no. 3, pp. 1051–1064, feb 2008. [Online]. Available: http://doi.wiley.com/10.1118/1.2836950

[15] V. M. Jhala and V. S. Thaker, "X-ray computed tomography to study rice (Oryza sativa L.) panicle development," *Journal of Experimental Botany*, vol. 66, no. 21, pp. 6819–6825, 2015.

[16] S. R. Tracy, C. R. Black, J. A. Roberts, C. Sturrock, S. Mairhofer, J. Craigon, and S. J. Mooney, "Quantifying the impact of soil compaction on root system architecture in tomato (Solanum lycopersicum) by X-ray micro-computed tomography." *Annals of botany*, vol. 110, no. 2, pp. 511–519, 2012.

[17] R. Metzner, A. Eggert, D. van Dusschoten, D. Pflugfelder, S. Gerth, U. Schurr, N. Uhlmann, and S. Jahnke, "Direct comparison of MRI and X-ray CT technologies for 3D imaging of root systems in soil: Potential and challenges for root trait quantification," *Plant Methods*, vol. 11, no. 1, pp. 1–11, 2015.

[18] Y. M. Staedler, D. Masson, and J. Schönenberger, "Plant Tissues in 3D via X-Ray Tomography: Simple Contrasting Methods Allow High Resolution Imaging," *PLoS ONE*, vol. 8, no. 9, 2013.

[19] F. J. Leigh, I. Mackay, H. R. Oliveira, N. E. Gosman, R. A. Horsnell, H. Jones, J. White, W. Powell, and T. A. Brown, "Using diversity of the chloroplast genome to examine evolutionary history of wheat species," *Genetic Resources and Crop Evolution*, vol. 60, no. 6, pp. 1831–1842, 2013.

[20] J. Cockram, H. Jones, F. J. Leigh, D. O'Sullivan, W. Powell, D. A. Laurie, and A. J. Greenland, "Control of flowering time in temperate cereals: Genes, domestication, and sustainable productivity," *Journal of Experimental Botany*, vol. 58, no. 6, pp. 1231–1244, 2007.

[21] V. C. Gegas, A. Nazari, S. Griffiths, J. Simmonds, L. Fish, S. Orford, L. Sayers, J. H. Doonan, and J. W. Snape, "A Genetic Framework for Grain Size and Shape Variation in Wheat," *The Plant Cell*, vol. 22, no. 4, pp. 1046–1056, 2010. [Online]. Available: http://www.plantcell.org/lookup/doi/10.1105/tpc.110.074153

[22] S. R. Tracy, J. F. Gómez, C. J. Sturrock, Z. A. Wilson, and A. C. Ferguson, "Non-destructive determination of floral staging in cereals using X-ray micro computed tomography (μCT)," *Plant Methods*, vol. 13, no. 1, pp. 1–12, 2017.

[23] A. Andrews, "Scrum Of One: How to Bring Scrum into your One-Person Operation." [Online]. Available: https://www.raywenderlich.com/162654/scrum-one-bring-scrum-one-person-operation

[24] S. Mairhofer, C. J. Sturrock, M. J. Bennett, S. J. Mooney, and T. P. Pridmore, "Extracting multiple interacting root systems using X-ray microcomputed tomography," *Plant Journal*, vol. 84, no. 5, pp. 1034–1043, 2015.

[25] K. R. Daly, S. R. Tracy, N. M. Crout, S. Mairhofer, T. P. Pridmore, S. J. Mooney, and T. Roose, "Quantification of root water uptake in soil using X-ray computed tomography and image-based modelling," *Plant, Cell & Environment*, 2017. [Online]. Available: http://doi.wiley.com/10.1111/pce.12983

[26] C. Ozgur, U. Hall, T. Colliau, G. R. J. o. D. . . . , and U. 2017, "MatLab vs. Python vs. R," *Journal of Data Science*, 2016. [Online]. Available: https://www.researchgate.net/profile/Ceyhun{_}Ozgur/publication/320475107{_}C{_}Ozgur{_}T{_}Colliau{_}G{_}Rogers{_}ZHughes{_}B{_}Myer-Tyson{_}MatLab{_}vs links/59e7be12458515c3630fab8d/C-Ozgur-T-Colliau-G-Rogers-ZHughes-

[27] G. van Rossum, "PEP 8 – Style Guide for Python Code | Python.org." [Online]. Available: https://www.python.org/dev/peps/pep-0008/

[28] J. L. Pfaltz, "Sequential Operations in Digital Picture Processing," *Journal of the ACM*, vol. 13, no. 4, pp. 471–494, oct 1966. [Online]. Available: http://portal.acm.org/citation.cfm?doid=321356.321357

[29] J. M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing - STOC '97*. New York, New York, USA: ACM Press, 1997, pp. 599–608. [Online]. Available: http://dl.acm.org/citation.cfm?id=258533.258653