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

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

| | |
|---|---|
| Date: | April 9, 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 [1]. 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) [2].

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 [2]. This decrease in efficiency is attributed to climate change bringing in more hostile conditions, which these elite and thoroughly domesticated genotypes are unprepared for.

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 [3].

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 biological data used in this solution are as follows:

The population of wheat used contains samples from the following genotypes:

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

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

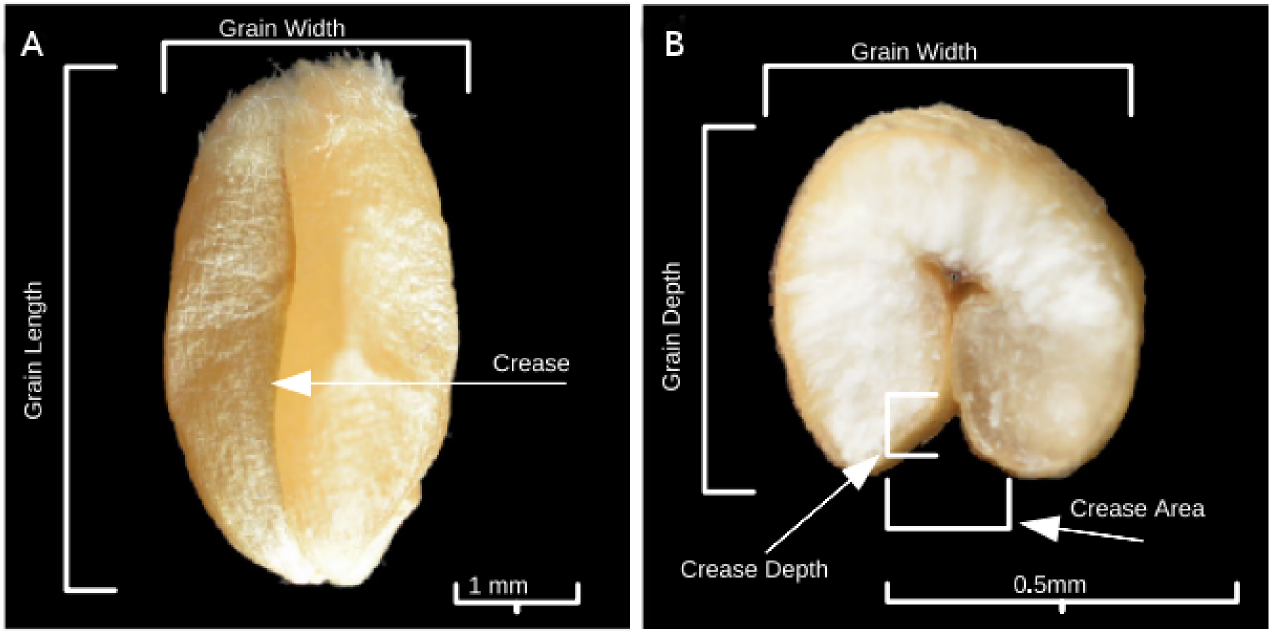


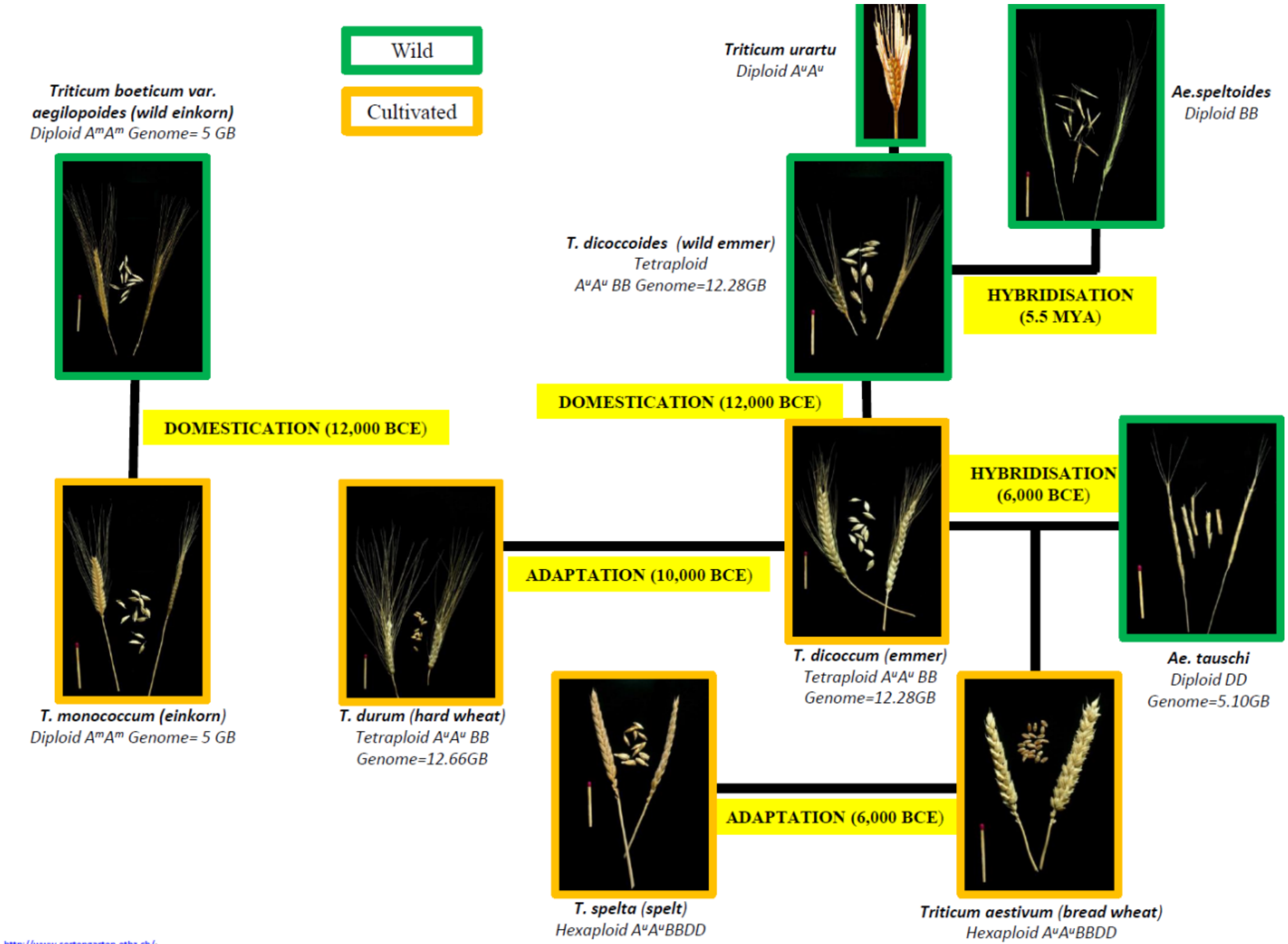
Figure 1.1: Wheat grain labelled (*left*), wheat grain cut in half (*right*)

## Significance to Current Research

The biological interest in this area has been expressed in several areas of research [4], it is proposed that the key to unlocking diversity in the wheat genus lies in these ancestor, undomesticated species [5].

This research has the potential to be useful in several areas including: crop breeding; disease resistance; environmental stress.

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.



http://www.sortengarten.ethz.ch/;
ftp://ftp.fao.org/docrep/fao/009/y5553e/y5553e.pdf

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

## Aim 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.

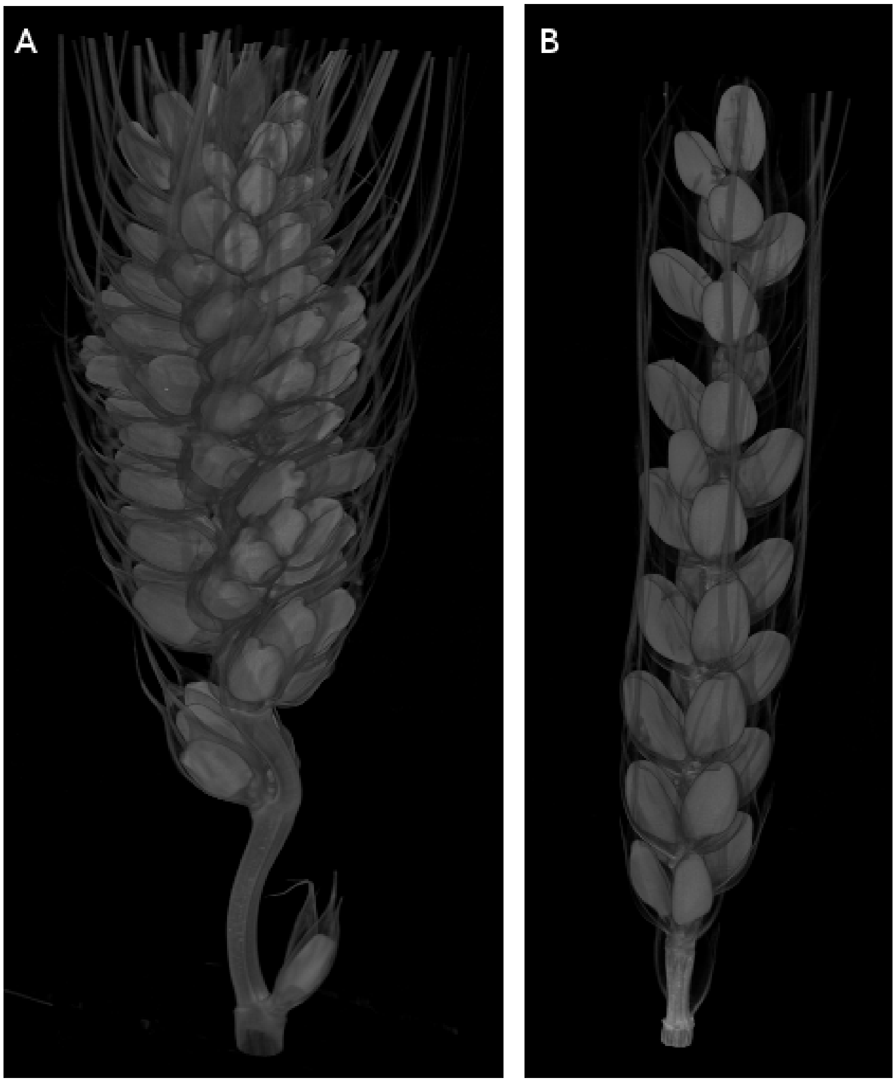


Figure 1.3: Two µ-CT scans of wheat spikes, showing diversity in Population, Compactum (6N) left, Durum right (4N)

These aims have a focus on the phenotypic attributes generated from customised image analysis software [6] 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

## Problems Overview

The problems 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 Problems

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 [7]. 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 Problems

Using µ-CT data in plant sciences is becoming more and more common [8, 9, 6, 10] 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 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.

## Software Development Methodology

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.

## Functional Requirements

### Requirements for CT Analysing Library

### Requirements for CT GUI Application

- Useable by *anyone*

## Version control

## Designing Process

## Documentation

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:2.

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:3

Similarly, the documentation for using the CT GUI Application is based on feedback provided via user testing and Google form feedback data.

## Language Choice

## Software Library Choices

## Implementation

## Testing

### Feedback Forms

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

### Unit Testing CT Analysing Library

### Unit Testing CT GUI Application

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

| **Result** | **Test** | **Duration** |
|---|---|---:|
| Passed | CTData.py::test_aggregate_spike_averages | 0.03 |
| Passed | CTData.py::test_clean_data_maximum_removed | 0.00 |
| Passed | CTData.py::test_clean_data_minimum_removed | 0.00 |
| Passed | CTData.py::test_load_additional_data | 1.30 |
| Passed | CTData.py::test_load_additional_data_no_data | 0.00 |
| Passed | CTData.py::test_load_data | 0.13 |
| Passed | CTData.py::test_NoDataFoundException | 0.00 |
| Passed | Data_transforms.py::test_box_cox_data | 0.01 |
| Passed | Data_transforms.py::test_pca_to_table | 0.01 |
| Passed | Data_transforms.py::test_perform_pca | 0.02 |
| Passed | Data_transforms.py::test_standardise_data | 0.01 |
| Passed | Graphing.py::test_plot_boxplot_as_dataframe | 0.05 |
| Passed | Graphing.py::test_plot_boxplot_as_object | 0.05 |
| Passed | Graphing.py::test_plot_difference_of_means | 0.11 |
| Passed | Graphing.py::test_plot_histogram_as_dataframe | 0.02 |
| Passed | Graphing.py::test_plot_histogram_as_object | 0.02 |
| Passed | Graphing.py::test_plot_pca | 0.17 |
| Passed | Graphing.py::test_plot_qqplot | 0.00 |
| Passed | Statistical_tests.py::test_baysian_hypothesis_test | 10.76 |
| Passed | Statistical_tests.py::test_t_test | 0.00 |
| Passed | Statistical_tests.py::test_test_normality | 0.00 |

# Chapter 3

# Methods and Solutions

## Data Pipeline

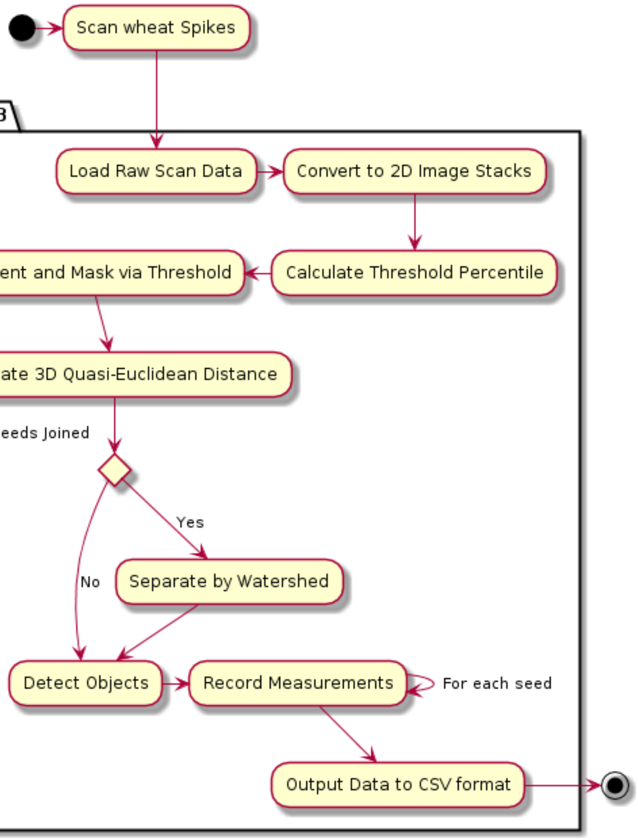

Scan wheat Spikes
MATLAB
Load Raw Scan Data
Convert to 2D Image Stacks
Segment and Mask via Threshold
Calculate Threshold Percentile
Calculate 3D Quasi-Euclidean Distance
Seeds Joined
Yes
No
Separate by Watershed
Detect Objects
Record Measurements
For each seed
Output Data to CSV format

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:3.1). This provided much better results than the previous *chessboard* transform which had been successful on more uniform data in previous studies [6].

### Quasi-Euclidean algorithm

This algorithm measures the total euclidean distance along a set of horizontal, vertical and diagonal line segments [11].

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

In order to apply this to a 3D space Kleinberg's method is used [12]. 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

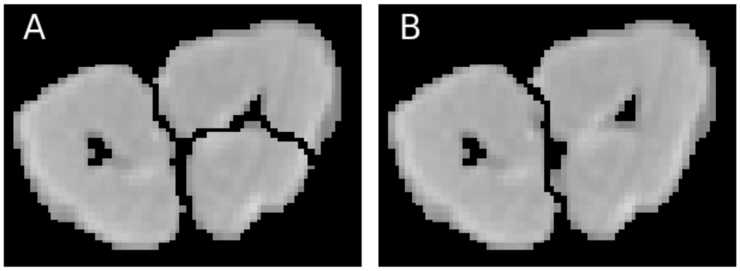


Figure 3.2: *A* showing the chessboard method, *B* improved quasi-euclidean method

## CT Analysing Library Methods

## CT GUI Application Methods

## Data Analysis Methods

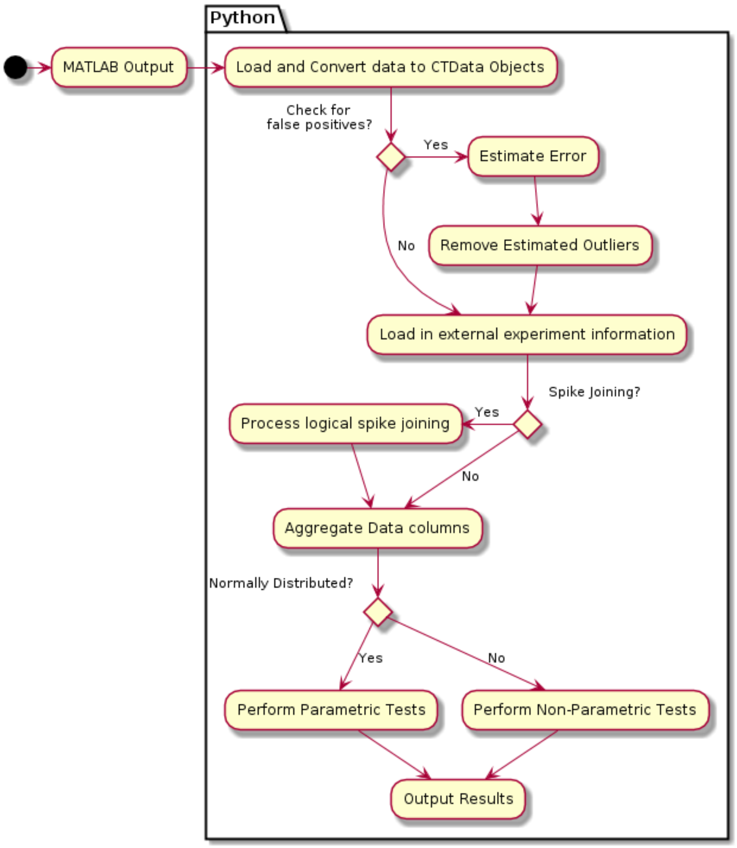

Python
MATLAB Output
Load and Convert data to CTData Objects
Check for false positives?
Yes
Estimate Error
No
Remove Estimated Outliers
Load in external experiment information
Spike Joining?
Yes
Process logical spike joining
No
Aggregate Data columns
Normally Distributed?
Yes
No
Perform Parametric Tests
Perform Non-Parametric Tests
Output Results

Figure 3.3: 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

| | | |
|---|---|---|
| MATLAB Image Processing Toolbox | Numpy | Matplotlib |
| Seaborn | Scipy | Sklearn |
| Statsmodels | Pymc3 | Xlrd |
| PyQt5 | | |

### Tools

Table 7.2: Software tools used

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

## *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 |

## *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 1: 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")))))
    (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 2: Custom lisp code for generating easy to read documentation

### Self-Documenting Code Example

```
def standarise_data(df, features, groupby):
    """
    This is to conform with the likes of PCA, the following text is borrowed from:
    https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60
    Some code also also heavily borrowed from this page and I
    take minimal credit for it.

    PCA is effected by scale so you need to scale the features in your data before
    applying PCA.

    Use StandardScaler to help you standardize the datasets features onto unit scale
    (mean = 0 and variance = 1)
    which is a requirement for the optimal performance of many machine
    learning algorithms.
    If you want to see the negative effect not scaling your data can have,
    scikit-learn
    has a section

    on the effects of not standardizing your data.

    uses: \dfrac{x_i  mean(x)}{stdev(x)}
    assumes Normal Distribution

    To try and fit a normal distribution I am applying log scales of log_2

    @param df the data to be standarised
    @param features the list of features to standardise
    @param groupby how the columns should be grouped
    @returns scaled values
    """
    # Separating out the features
    x = df.loc[:, features].values

    # Attempt to normalise through power transforms
    box_cox_data(x)

    # Standardizing the features
    x = StandardScaler().fit_transform(x)
    return x
```

Listing 3: Example of code documentation and readability from *data_ transforms.py*

# References

[1] 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

[2] 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

[3] 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

[4] 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.

[5] 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.

[6] 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.

[7] 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

[8] 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 ($\mu$CT),” *Plant Methods*, vol. 13, no. 1, pp. 1–12, 2017.

[9] 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.

[10] 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.

[11] 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

[12] 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