A System for Managing Maize Research Crops

Toni Kazic

Dept. of Electrical Engineering and Computer Science, MU Plant Science Foundry, Missouri Maize Center, MU Informatics Institute
University of Missouri, Columbia

Introduction

Planning, managing, and collecting data from research crops are labor-intensive tasks. Each laboratory and experiment has its own goals and methods, and these can change with every planting. I have built a flexible system that reduces manual effort and can be easily changed as one's needs evolve. It supports maize genetics, but can be adapted to breeding programs and other crops if needed. The system has been tested and refined with multiple generations of students over the last twelve years, supporting my ongoing study of maize disease lesion mimic mutants' phenotypes and high-dimensional phenotypes. You need use only those parts you find useful.

Some years ago I suggested ten simple rules for assuring the provenance of experimental data. Those rules and their illustrations were based on this work. The code strikes a balance between manual inspection and correction of data and automated data management, since it is always true that later results can change one's interpretation of earlier data; scanners can misread barcodes; and long days in the field or laboratory can introduce silly mistakes.

Key Design Principles

- **The code should rely on open-source languages and packages, port easily, and be fast enough and flexible enough to accommodate a career's worth of changes.**
- **Only essential data correction should occur in the field**, ideally contemporaneously with their initial gathering. The rest should occur in comfort.
- **All procedures should assure provenance**, make field operations more efficient, provide redundant information for error detection and correction, and rely on readily available commodity hardware, software, cloud services, and stationery.
- As far as possible, **all components and their supporting code should be mutually independent**. Use what you need and discard the rest.
- Use familiar and well-designed interfaces whenever possible (**spreadsheets!**).
- **Exploit cloud services for transfer of the raw data from the field** to your lab; and separate and preserve raw data, corrected data, and the database data.
- **Use commodity devices with big market shares** and long lifetimes (cell phones, tablets, scanners).
- **KISS**: keep it simple, stupid.

Workflow

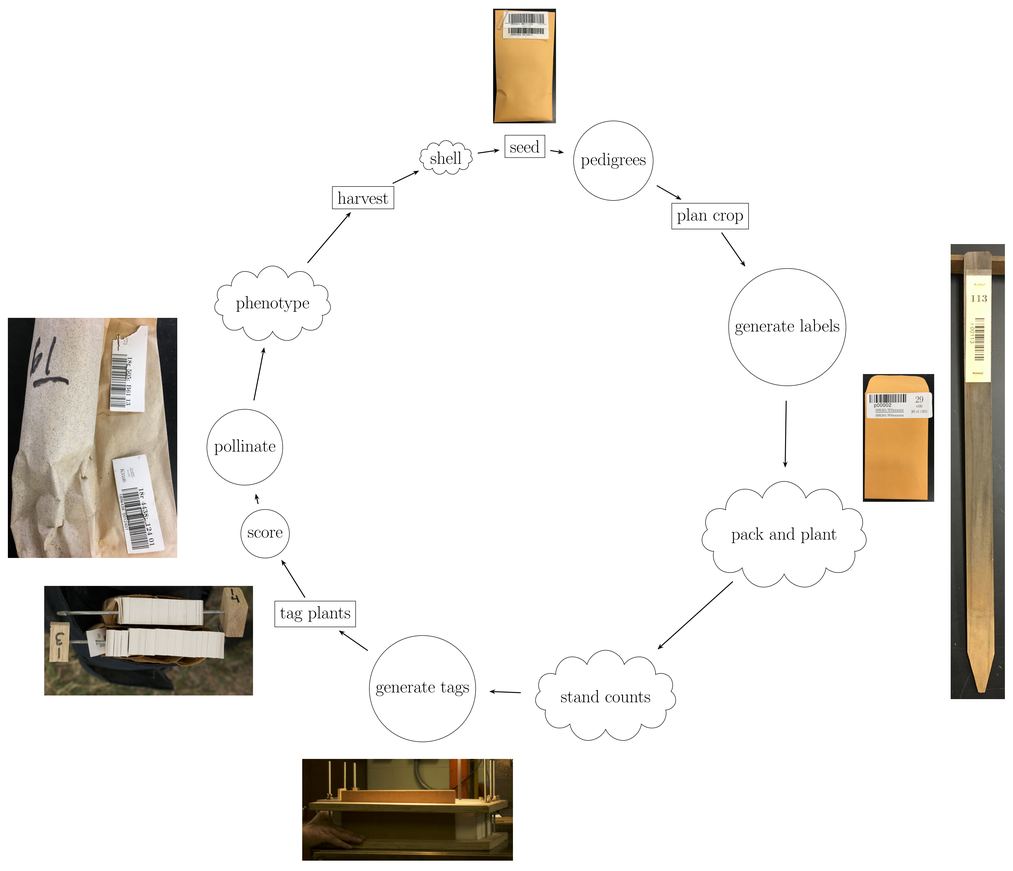

Fig. 1: The cycle of work. Manual operations are in rectangles, code-supported ones in circles, and field operations with data transfer to the cloud are in ... clouds, of course.

Demeter: Prolog Knowledge Base for Crops and Data

This captures your data *and your ideas about your data in nearly plain text.* It supports computation of pedigrees from numerical genotypes, crop planning, and management of the field work and seed inventory. If your ideas and experiment differ from mine, you'll have to modify the code, but since Prolog is a logic programming language this is pretty easy.

Chloe: Perl Scripts for Data Provenance and Processing

These scripts and their underlying modules process raw data after manual checking, deposit them in the Prolog database, and move processed files to permanent safety; generate tags, labels, and 1D barcodes using LaTeX and GNU barcode; and support miscellaneous crop management and data collection tasks. Chloe is one of the many titles of Demeter, meaning "young green shoot".

Scanners, Spreadsheets, and Smart Phones for Data Collection

We use Numbers on iPhones and iPads to collect the tabular data using matchbox-size KOAMTAC KDC200 bluetooth scanners. Setting the scanner to "wedge" dumps the scanned barcode directly into the active cell of the spreadsheet. Our data are timestamped with varying precision, depending on the data type: seed packing, planting, setups, and pollinations are timestamped within a minute or less, others have longer temporal windows. Spreadsheets are saved as csv files.

Robust Protocols and Equipment for Provenance

I prefer 1D to 2D barcodes because it's easier to see if the barcode is too damaged to scan, and how to aim the scanner to use the better parts of the barcode. We use Dropbox to transfer data from the devices to the cloud. We save a copy of the spreadsheet on the device, switch to airplane mode, and collect and save the data. We transfer the data only when we on a good connection. This prevents an app from losing data because it can't zealously sync every change.

Availability

My code and documentation are on GitHub. Comments and pull requests are welcome, *especially for the documentation.* Please use the GitHub machinery for reporting issues, rather than emailing me. I can guide you in adapting the code for your work: you don't need to know much about writing code to get started.

Fig. 2: To the code!

Acknowledgments

Many undergraduate and graduate students have field-tested this system over the years, refining my initial ideas and helpfully breaking things. Much of what I show here originated in discussions with and demonstrations by the members of the Missouri maize community. I thank Mac and Vinny for helpful discussions. This work was supported by NSF MCB-1122130, the MU Research Board, and an anonymous gift.