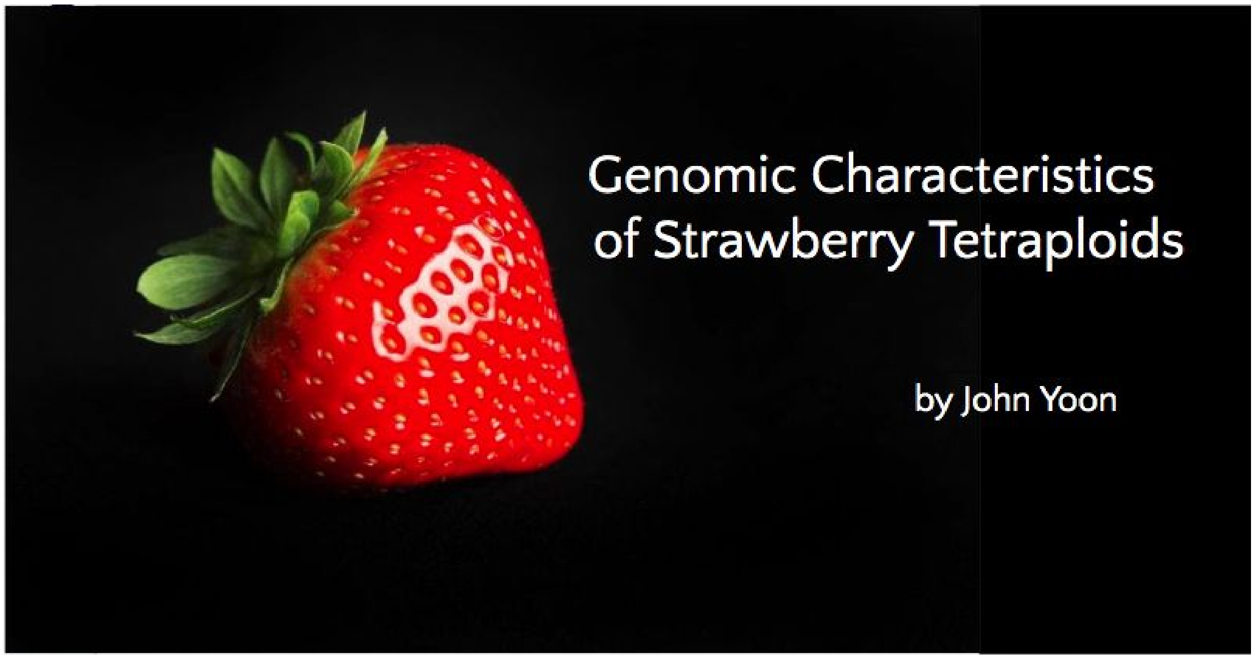
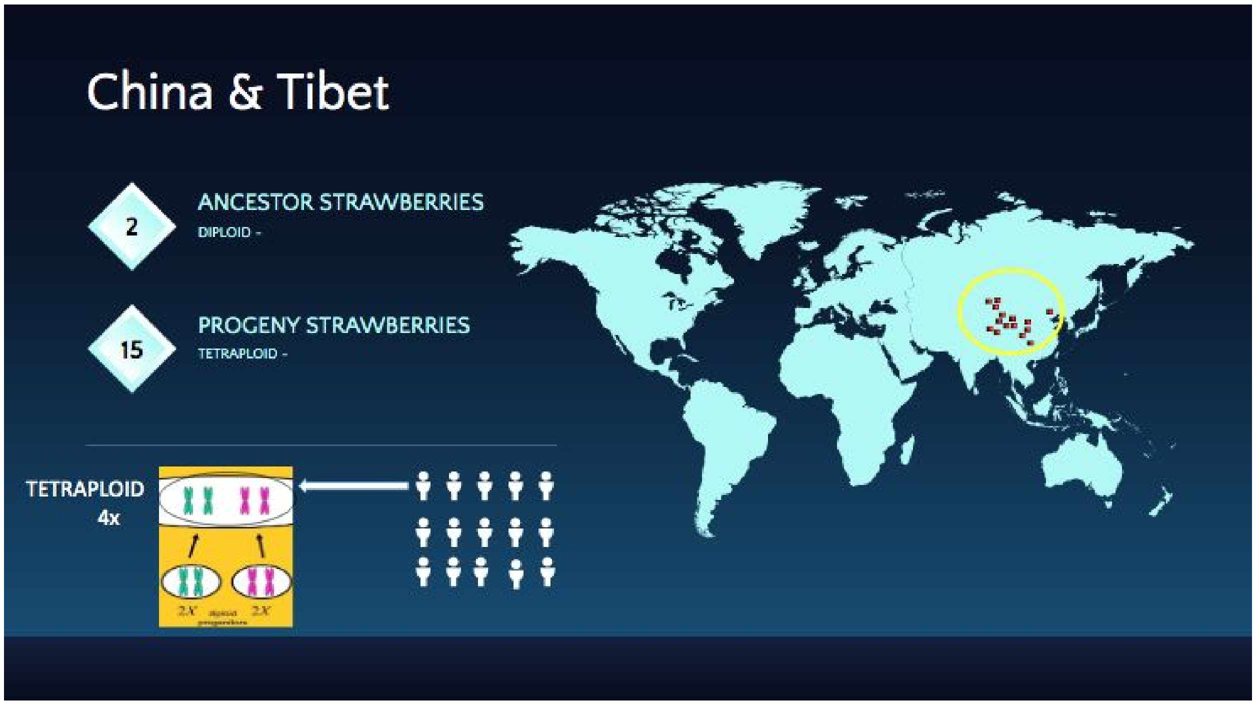


Genomic Characteristics of Strawberry Tetraploids

by John Yoon

Background



China & Tibet

2 ANCESTOR STRAWBERRIES
DIPLOID -

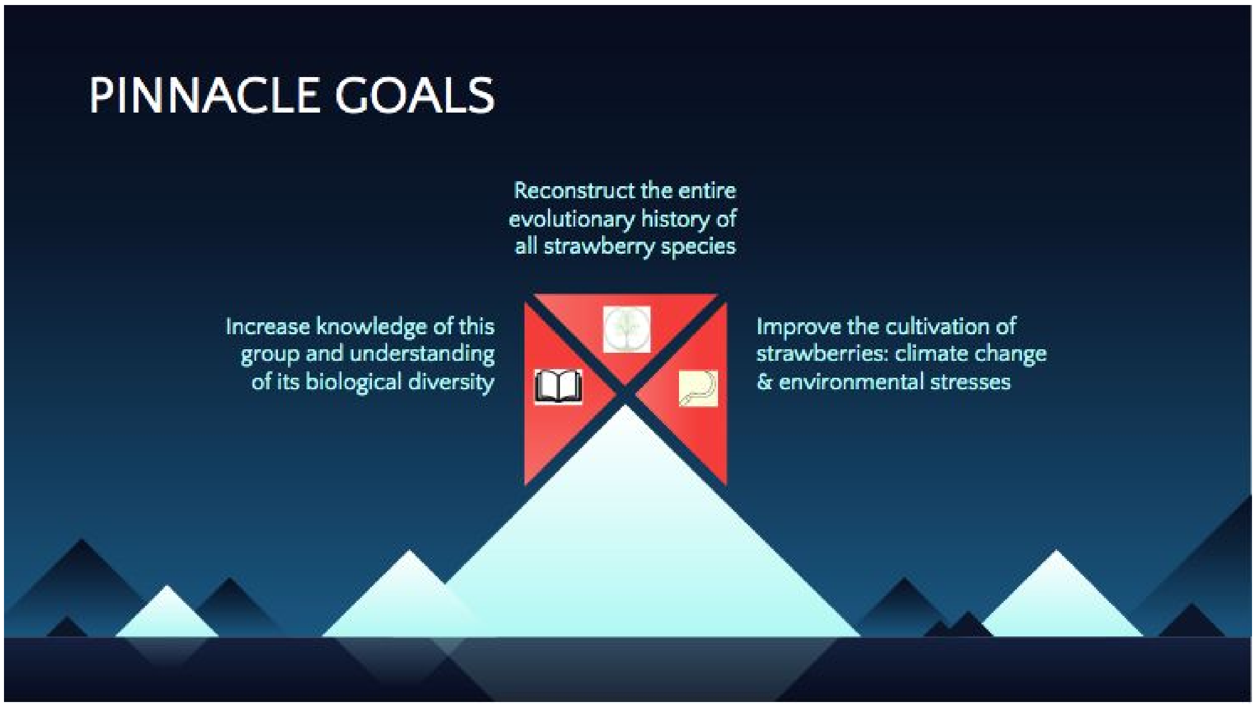
15 PROGENY STRAWBERRIES
TETRAPLOID -

The focus of my project begins several thousands of years ago in Asia, when two ancestor strawberry genotypes repeatedly gave rise to several progeny strawberries. As seen in the map, these strawberries are located in the Northwest and Southwest parts of China, as well as in the Tibet region.

In particular, our research examined 15 of these descendant strawberry genotypes, whose tetraploid nature draws distinction with their two ancestors, which were diploid; in this context, the main difference between the diploid and tetraploid strawberries is the number of chromosomes per cell. Diploid plants have two sets of chromosomes per cell while tetraploids have four. As such, these 15 genotypes were the key points of our analysis.

Big picture, the main goal of this endeavor was to estimate the sequence divergence between the progenitor diploids and their tetraploid descendants; in particular, we achieved this by using data derived from comparing the 15 sequenced genotypes, with a reference sequence from a diploid strawberry named “Hawaii 4.” This reference sequence, “Hawaii-4” was used both because of its similarity to the two ancestor strawberry genotypes, as well as the fact that is currently the only fully sequenced diploid strawberry.

Motivations



PINNACLE GOALS

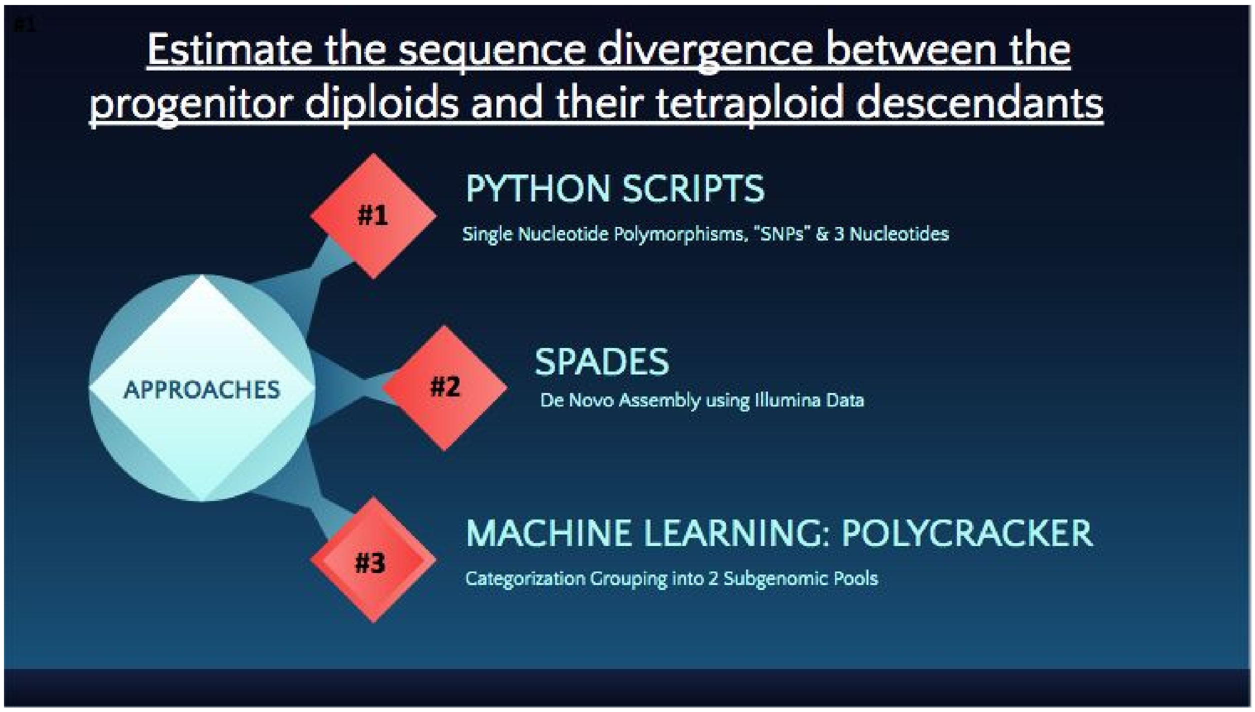
Reconstruct the entire evolutionary history of all strawberry species

Increase knowledge of this group and understanding of its biological diversity

Improve the cultivation of strawberries: climate change & environmental stresses

By analyzing the divergence among the 15 strawberry genotypes, the pinnacle goal beyond even the scope of my contribution timeline to this project, is to reconstruct the entire evolutionary history of all strawberry species, which will not only increase knowledge of this group and understanding of its biological diversity, but could also be helpful in efforts to improve the cultivation of strawberries. Correctly identifying the diploid progenitors is important for understanding and predicting the responses of polyploid plants to climate change and associated environmental stress.

Methodology


Estimate the sequence divergence between the progenitor diploids and their tetraploid descendants
APPROACHES
#1
PYTHON SCRIPTS
Single Nucleotide Polymorphisms, "SNPs" & 3 Nucleotides
#2
SPADES
De Novo Assembly using Illumina Data
#3
MACHINE LEARNING: POLYCRACKER
Categorization Grouping into 2 Subgenomic Pools

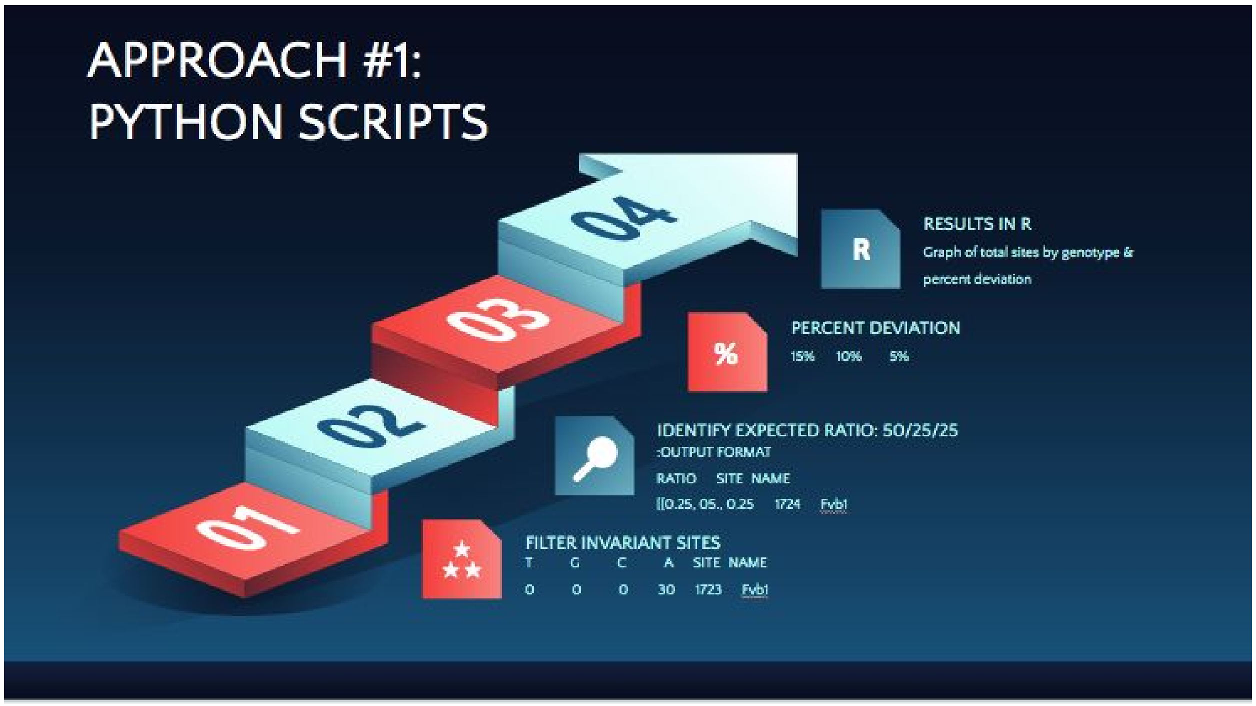
The goal to estimate the sequence divergence between the progenitor diploids and their tetraploid descendants was structured with 3 main approaches:

The first approach consisted of my writing python scripts to filter out and process data sets involving Single Nucleotide Polymorphisms, or SNPs, which is a variation in a single base pair in a DNA sequence. The goal was to find SNPs where 3 nucleotides were represented, which can only occur in tetraploids, and we were interested in documenting how many of them exist.

The second approach centered around genome assembly, in which I took raw Illumina read data files and inputted them alongside a genomic software tool called Spades; Spades is one of a number of de novo assemblers that use short read sets as input, figure out how they overlap, and stitch them together to create a much longer sequence. We did this because the length of a polyploid genome assembly can be used as a measure of subgenome divergence.

The third and final approach involved using a genomic machine learning toolkit called Polycracker; We wanted to do this because the ML would attempt to classify our assembled scaffolds into 2 subgenomic pools corresponding to the diploid ancestors of the tetraploid.

First Approach

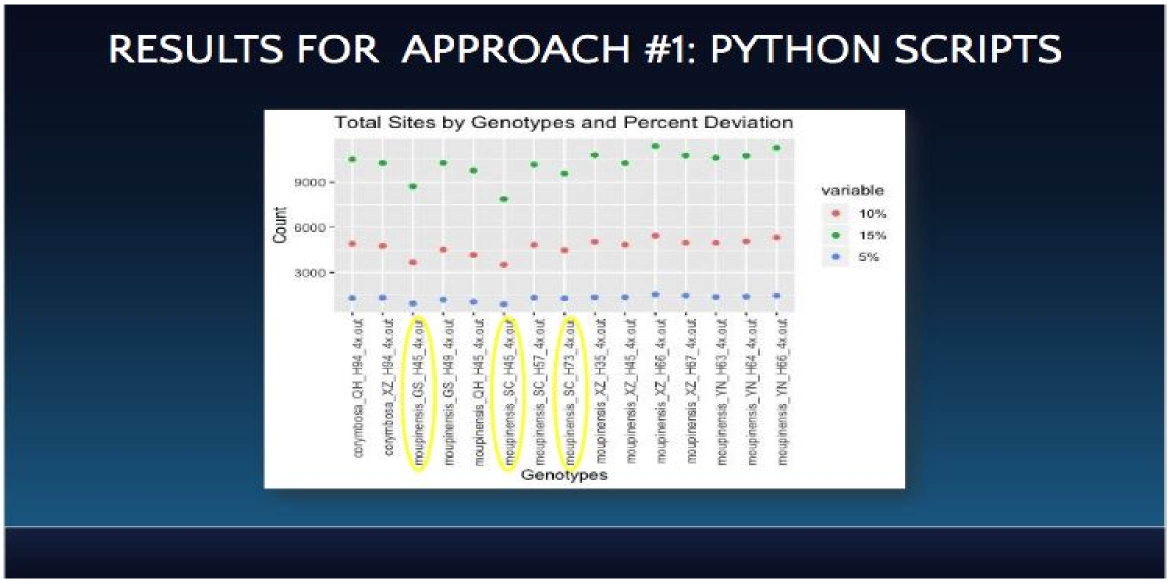


So to circle back, and more concretely dive into these approaches: the first approach relied on my creating two python scripts: the first script took data sets (as seen in the format here, in the upper left, with header columns of: the chromosome name, site, total A nucleotides, C nucleotides, G nucleotides, and T nucleotides), and removed any invariant sites; we classified these invariant sites as those with a value of 0 for 3 out of the 4 nucleotide types. For instance, if the python script came across a scenario like that in the first row of the example data in the upper left box, where there are 30 A nucleotides, but 0 for C G and T, then it would skip over it; all the other cases were recorded and outputted to new data files that would be later used. This substantially trimmed our data sets; for instance corymbosa_QH_H94_4x.pro went from about 137 million sites to just 3.7 million total sites.

After cleaning our data sets from the invariant sites, I then created another python script that focused on using our new data files to filter sites based on ratios between individual nucleotide counts and the total number of nucleotides. In other words, if we look at the second row of the example data, if there were 5 A nucleotides, 10 for C, 0 for G, and 5 for T, then there would be a sum of 20 total nucleotides, of which Adenine would make up 5/20, or 25% of the total, Cytosine would make up 10/20 or 50%, Guanine would be 0% of the total, and Thymine would be 25%. And the python script outputted the results in the bracketed format seen in Step 2. This in fact, is an example of the expected ratio that the script would look for: 50%, 25%, 25%.

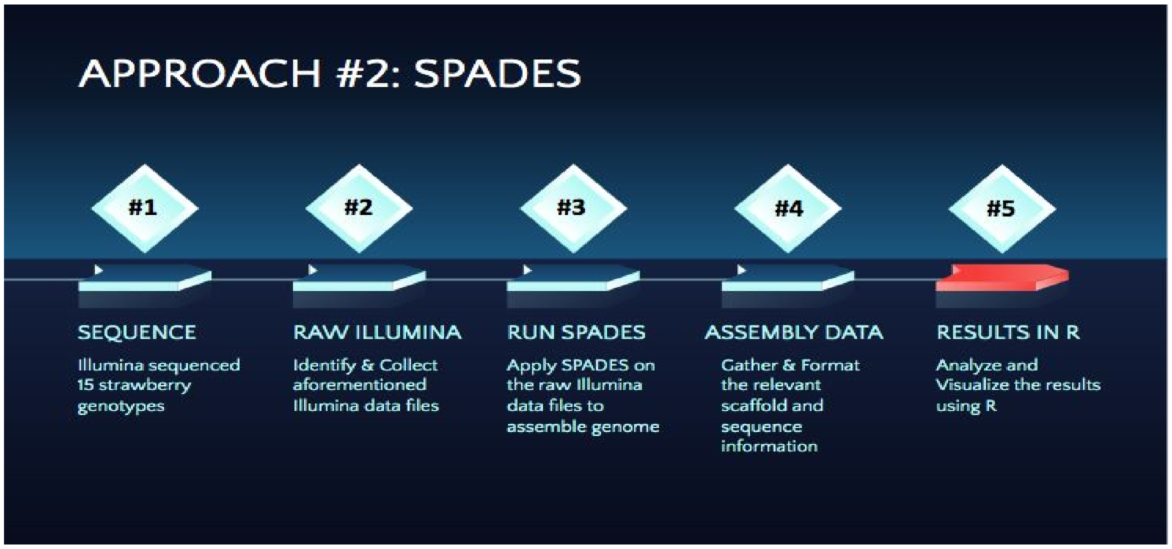
I also implemented a feature within the python script that would allow for the user to select a percent deviation from the expected ratio. For this project, we established three different percent deviation values, of 5% deviation, 10% deviation, and 15%. So after all these processes of structuring and filtering the data sets, we ultimately got the following results, which I visualized through R.

<u>First Approach Conclusions</u>


RESULTS FOR APPROACH #1: PYTHON SCRIPTS

We can see that three genotypes standout with smaller total site count values. In particular within the 15% deviation green plotted values, these 3 genotypes' relatively low site count totals were noteworthy results that indicated less divergence between the subgenomes.

<u>Second Approach</u>


APPROACH #2: SPADES

#1 SEQUENCE
Illumina sequenced 15 strawberry genotypes

#2 RAW ILLUMINA
Identify & Collect aforementioned Illumina data files

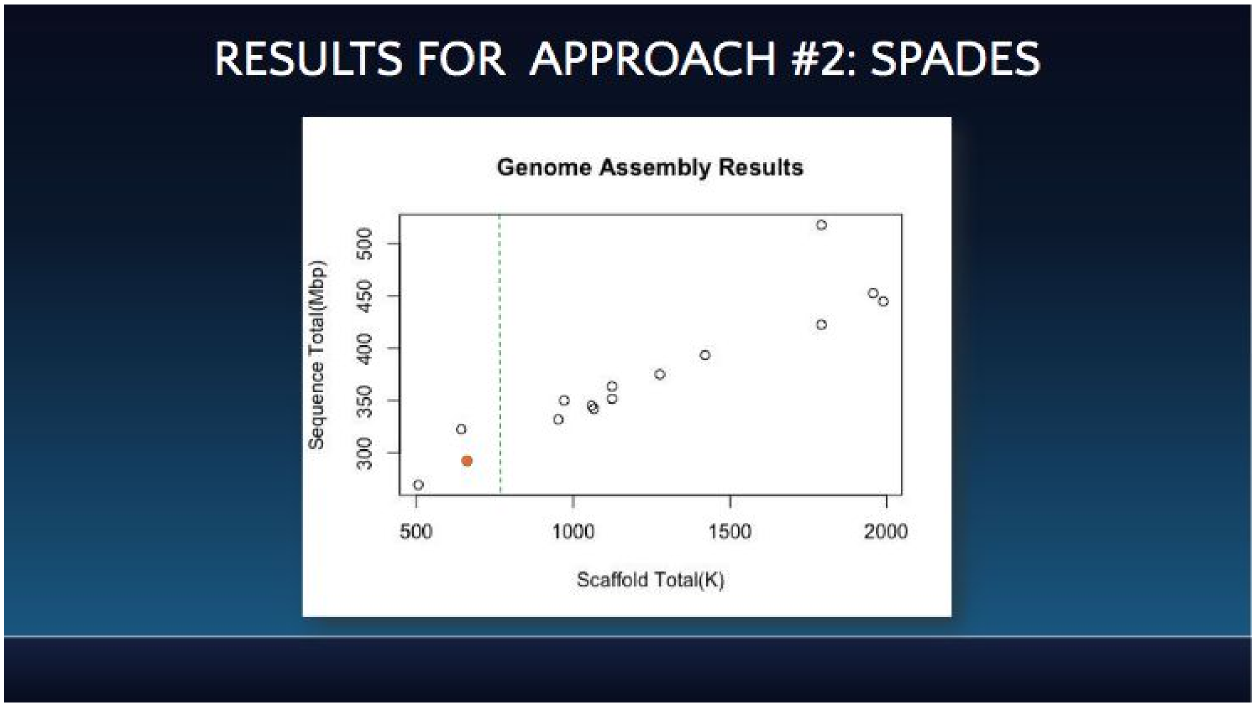
#3 RUN SPADES
Apply SPADES on the raw Illumina data files to assemble genome

#4 ASSEMBLY DATA
Gather & Format the relevant scaffold and sequence information

#5 RESULTS IN R
Analyze and Visualize the results using R

Upon completion of this first python approach of analyzing the nucleotides, the next and second approach was to assemble the genomes of the 15 strawberry genotypes using the genomic software SPADES, which allowed me to take the raw Illumina read data files and piece them together to construct much longer sequences of scaffolds.

Due to the scope of the task and the sheer size of the data sets involved, SPADES had to be run through the CGRB servers using 60 CPUs per genotype assembly run. These were submitted through bash scripts that I wrote and were used by the qsub wrapper, SGE_ARRAY, which automates the submission of a large number of jobs. We were able to accomplish this in a span of just a few days because we ran 900 (60cpusx15runs) total processors on the CGRB infrastructure. When it finally completed, we were able to use the outputted data to generate results into a graph, again with R, that appeared as such:

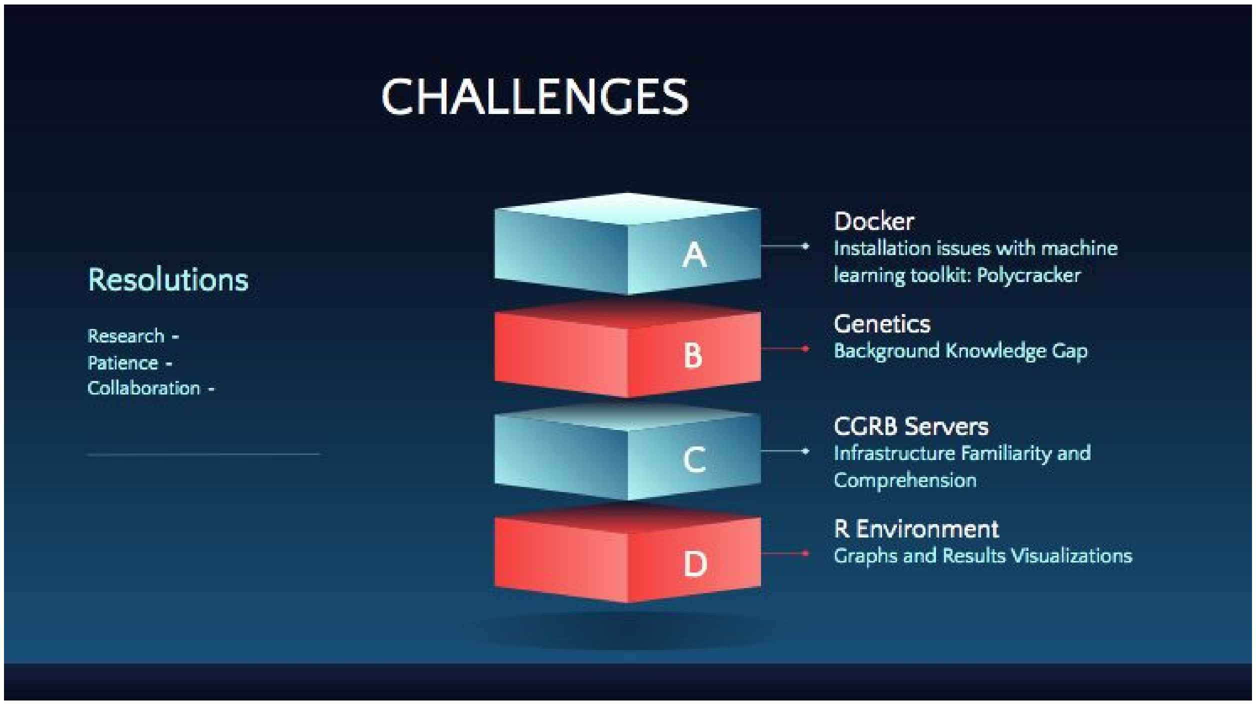


We can see that the three plotted genotypes to the left of the dotted line, seem diploid-like based on the scaffold and sequence sizes. However, through Aaron's research looking at ratios of di-nucleotides, we know that the genotype represented by the orange plot actually is diploid instead of the tetraploid we originally assumed; the other two genotypes that are still actually tetraploid but appear diploid based on the scaffold and sequence values, indicate that there is very little subgenome divergence in those two samples. These three plots were promising because they corresponded with the 3 genotypes from the results of the prior results graph.

Third Approach

The final approach involved using a machine learning software called polycracker. We hoped to classify our newly assembled scaffolds into 2 subgenomic pools corresponding to the diploid ancestors of the tetraploid. However we did not finalize this stage. We needed to use Docker to download the machine learning software and Docker is a tool used to develop and deliver software in packages called containers; while we were able to easily download Docker, there were issues loading the actual machine learning package for polycracker and ultimately I was unable to complete this third stage.

Obstacles & Conclusions



CHALLENGES

Resolutions

Research -
Patience -
Collaboration -

A — Docker: Installation issues with machine learning toolkit: Polycracker

B — Genetics: Background Knowledge Gap

C — CGRB Servers: Infrastructure Familiarity and Comprehension

D — R Environment: Graphs and Results Visualizations

In fact, much like this docker issue, throughout the process, learning opportunities arose as challenges in several ways. One was the fundamental gap in knowledge regarding genetics that stemmed largely due to my academic background of computer science; however my PI Aaron was instrumental in helping to bridge this gap, and thanks to his guidance and patience, as well as help from several of my fellow interns, I was able to navigate through the project.

Another challenge was related to the CGRB servers; while I had experience with the command line and Terminal prior to this internship, picking up the syntax and broader understanding of the servers' abstractions was a difficult but necessary aspect of the project.

Finally, using R seriously for the first time to create graphs was not the easiest, but thanks to the early immersion from Molly and Justin's lessons, I was able to ease my way into the R environment. In short, a theme drawn from all of these challenges is that they were, by and large, overcome due to a fair bit of online research, patience, and the support of the program's community.