2020-06-18

EMS Mutagenesis in Quinoa: Developing a Genetic Resource

Brian James Cox
Brigham Young University

EMS Mutagenesis in Quinoa: Developing a Genetic Resource

Brian James Cox

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

David Jarvis, Chair
Jeff Maughan
Rick Jellen

Department of Plant and Wildlife Sciences

Brigham Young University

ABSTRACT

EMS Mutagenesis in Quinoa: Developing a Genetic Resource

Brian James Cox
Department of Plant and Wildlife Sciences, Brigham Young University, Provo, UT
Master of Science

Chenopodium quinoa, a South American pseudocereal, has valuable agricultural traits such as salt tolerance and drought tolerance, and it has beneficial nutritional properties such as high protein content and a complete amino acid profile. However, problems including disease susceptibility, low harvest index, lodging, seed shattering, low heat tolerance, and saponin content plague quinoa. Genetic resources for quinoa are needed to fix these problems and make quinoa more available throughout the world. We used ethyl methanesulfonate (EMS) to create a mutant population of QQ74 quinoa (USDA GRIN PI 614886) of 5,030 mutant families. We did whole exome sequencing (WES) on 44 mutant families. Using the recently published quinoa reference genome and MAPS, a mutation detection pipeline, we found a mutation rate of 11.35 mutations/Mb in these families. We also used whole genome sequencing (WGS) to calculate a mutation rate of 21.67 mutations/Mb in an additional nine mutant families. To demonstrate the utility of this population as a genetic resource, we found an EMS-induced nonsense mutation in the betalain synthesis pathway that prevents red betacyanins from accumulating in the hypocotyl of quinoa. With the mutation rates in our population, we calculate that analysis of 300 mutant families will yield 3-7 mutations in any gene of interest, which will facilitate forward and reverse genetic studies in quinoa.

Keywords: *Chenopodium quinoa*, ethyl methanesulfonate, mutation, whole exome sequencing, whole genome sequencing, betalains

ACKNOWLEDGEMENTS

I could not have completed this research without the work, expertise, and love of many individuals. Thank you to Bronte Miller, Leanne Wilson, Kate Christensen, Daniel Lewis, Zach Jaramillo, Tom Kerby, and the many other undergraduates who made this research possible. My teachers at BYU gave me important knowledge and skills that were crucial to my understanding of research and genetic research in particular. Thanks go especially to Drs. David Jarvis, Jeff Maughan, and Rick Jellen for their patience, hours spent talking through aspects of the project, and willingness to be flexible with adjustments. Lastly, I am grateful to my family and friends, especially Jason, Lauren, Sam, Megan, Mom, Dad, Mike, and Andy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Introduction to Quinoa

Even though quinoa is an important crop for global food security because of its nutritional properties and abiotic stress tolerance, increasing quinoa production in diverse, worldwide environments requires more research. Quinoa's excellent nutritional properties include high protein, presence of healthy fats, and high levels of vitamins and bioactive antioxidants. The seeds of quinoa contain 16.5% protein and have all nine essential amino acids (Wu, 2015). Other cereals do not contain as much protein nor all essential amino acids, which makes quinoa stand out as a food source. The unusually high protein levels in this pseudocereal give it the potential to improve people's diets throughout the world. In addition to high protein content, quinoa has high fiber content and high starch content (Bhargava & Srivastava, 2013). Plus, quinoa seeds contain high levels of calcium, phosphorus, magnesium, iron, zinc, potassium and copper (Fuentes, 2015). These beneficial nutritional properties should make agricultural research on quinoa a high priority.

Besides being very nutritious, quinoa can grow in adverse environmental conditions. While most major agricultural crops do not tolerate salty soil conditions, quinoa tolerates saline soils very well (Bhargava, 2013 and Troise et al., 2015). This trait is becoming increasingly important as irrigation creates more highly saline soils in traditional agricultural areas, and salinity tolerance could also enable quinoa to be grown on marginal lands unsuited for other crops. Besides salinity tolerance, quinoa can grow well with little water (Bhargava & Srivastava, 2006 and Troise et al., 2013.) Drought tolerance will become increasingly important as climate change creates less predictable precipitation patterns and as populations continue to expand in drought

susceptible areas. Even in areas where water is not a problem quinoa would tolerate drought conditions that would normally devastate crops. Drought tolerance along with salinity tolerance gives quinoa an advantage in challenging agricultural settings.

Additional research could help unlock the untapped potential of this unconventional crop (Massawe, Mayes & Cheng, 2016 and Zurita-Silva, Fuentes, Zamora, Jacobsen & Schwember, 2014). Areas which need more research include quinoa's susceptibility to downy mildew, its high seed saponin levels, and its low harvest index (Zurita-Silva et al., 2014). Improving these traits will be facilitated by understanding the underlying genetic basis of each and by identifying sources of improved traits to breed into new varieties. However, quinoa germplasm that can be used for these purposes is scarce. Access to South American quinoa collections is limited due to international laws, and few mutant populations with novel traits exist. Locating or creating new variation in quinoa is an important research area that will provide genetic tools for the improvement of this crop.

Induced Mutations as Potential Sources of Variation in Quinoa

Variation in quinoa should be sought after in order to study the genetic basis of quinoa traits and provide breeding material for the creation of improved cultivars. Variation can be obtained by utilizing diversity in currently cultivated quinoa varieties, by examining wild quinoa or its relatives, or by creating variation through mutagenesis. Although perhaps less common than the other methods, mutant populations are important resources for studying gene function and obtaining valuable traits. Historically, mutagenesis has been a powerful tool for introducing useful traits into plants. In fact, from 1971-2011 2,965 crop cultivars were released that included traits obtained from mutagenesis experiments in species such as wheat, rice, alfalfa, tomato, and cabbage (Sikora, Chawade, Larsson, Olsson & Olsson, 2011). Mutations in plant genomes may

be induced a variety of ways, each with its strengths and weaknesses. Targeted mutations may be caused through genetic transformation, although quinoa has proved to be transformation recalcitrant (Imamura et al., 2018). For this reason, random mutagenesis is the most effective way to create new variation in quinoa.

EMS Mutagenesis

Many mutagens have been used in plants, including radiation and chemicals (Sikora et al., 2011). EMS is one of the most widely used chemical mutagens, and this chemical produces mainly random, point mutations by alkylating guanines (Greene et al., 2003; Comai & Henikoff, 2006 and Sidhu, Mohan, Zheng, Dhaliwal, Main & Gill, 2015). This alkylation causes the G to incorrectly pair with a T in DNA replication, thus leading to a change from a G/C pair to an A/T pair in subsequent rounds of replication. Besides these transition mutations, other types of transitions and transversions as well as insertions and deletions have been observed due to EMS treatment, though with a lower frequency (Sidhu et al., 2015), so G/C – A/T transitions are referred to as canonical EMS mutations.

EMS has been used to create mutations in both diploid and polyploid plant species. Some diploid species treated with EMS include tomato (Saba & Mirza, 2002), peppers (Arisha et al., 2015), *Arabidopsis*, barley, and many others (Sikora et al., 2011). In addition, several polyploid plant species have been treated with EMS, including alfalfa, bread wheat, and durum wheat (Sikora et al., 2011). Due to the presence of multiple copies of homologous genes, polyploids can tolerate a higher mutation frequency than diploids (Sikora et al., 2011). This can be beneficial because it requires a smaller number of plants to represent a saturated mutant population, or a population which contains a mutation in every gene in the genome.

EMS has also been used recently in quinoa. In a gene function study, Imamura et al. (2018) identified two mutant alleles of a quinoa betalain biosynthesis gene in an EMS population that caused the hypocotyl to be green instead of red. Quinoa has also been treated with EMS in an attempt to create valuable agronomic traits. Mestanza et al. (2019) found EMS quinoa mutants with mutations in AHAS genes involved in herbicide resistance, although the mutant plants did not exhibit herbicide resistance. These two studies demonstrate the effectiveness of using EMS on quinoa.

The mutation frequency in a mutant population is important because it determines minimum population size needed for saturation with mutations. EMS produces different mutation frequencies depending on treatment times, treatment concentrations, and each particular plant species. Varied EMS concentrations have been used in previous studies, including 0.2% - 0.4% in Arabidopsis (Greene et al., 2003), 0.1% - 0.3% in barley, (Caldwell, Nicola, Muehlbauer, Marshall & Robbie, 2004), 0.6% in peppers (Arisha et al., 2015), 0.7% - 0.75% in tetraploid wheat (Uauy et al., 2009), and 0.9% - 1% in hexaploid wheat. In their study with AHAS genes, Mestanza et al. (2019) used a 2% EMS solution to mutagenize quinoa seeds. This study that used EMS on quinoa provides a starting point for determining the ideal EMS concentration for quinoa mutagenesis.

Whole Exome Sequencing

Whole exome capture and sequencing (WES) is a targeted sequencing approach that offers benefits over whole genome sequencing (WGS). One benefit is that effects of variation in coding regions are better understood than the effects of variants in noncoding regions (Warr, 2015; Kaur, 2017). Probe hybridization technology also provides some coverage of regions near the targets, such as introns and regulatory regions (Kaur, 2017), which are also well-studied. The

selective capture of better-understood regions means that the consequences of variation can be used to infer the effect on gene products and phenotypes, whereas interpreting variation in noncoding regions obtained from WGS may be difficult or impossible with current knowledge.

Another benefit of WES is even more practical. Because only a small target area—not the entire genome—is being sequenced, WES experiments require fewer resources. For example, the small size of the WES target area means a sequencing run can provide greater coverage depth of target regions and/or coverage of more samples (Kaur, 2017 and Warr, 2015). This allows grant funds to be used to expand a study by collecting data on more individuals without sacrificing coverage depth or to sequence a few individuals at deeper coverage. Besides the savings on the cost of sequencing, a small target area allows data storage space to go farther because non-target regions are not present (Warr, 2015). Sequencing is becoming a standard method to study the genetics of an organism, and storing the immense amounts of data generated can be a problem. WES generates less data than WGS and therefore reduces the amount of storage space needed. In a similar way, analyzing a subset of the genome requires less computational power (Kaur, 2017 and Warr, 2015). Removing large, noncoding genomic regions reduces the computational load of mapping algorithms, SNP calling programs, and other bioinformatics tools, so research can move forward without the purchase of additional computing power. A decreased demand for computing power, data storage, and sequencing space coupled with the fact that coding regions are better studied make WES a valuable tool for discovering variation in populations.

Several methods are used to perform WES. Molecular inversion probes and primer extension capture are two methods of polymerase mediated target capture (Teer, 2010). Array based, or solid phase, capture has also been used (Kaur, 2017; Teer, 2010 and Warr, 2015). Solution based captures using biotinylated probes are the most commonly used methods of exome capture (Kaur, 2017 and Warr, 2015).

An important part of doing WES is determining whether the capture was efficient in capturing the target regions. Researchers commonly report a few metrics to assess the efficiency of targeted sequencing. The most common metric reported is the percentage of sequenced bases that align to the target regions, or the specificity of the capture. A wide range of specificities have been reported, from as low as 26% to as high as 97%. Another metric frequently used to measure the effectiveness of an exome capture is the sensitivity, or percentage of target regions that are were recovered at a desired coverage depth. It is important to know that all or nearly all of the target regions were captured. Other metrics which have been used to assess targeted sequencing efficiency include the uniformity of capture efficiency across several capture reactions or in different genotypes, the depth of coverage as one moves further away from the target space, and many other variations on these (Henry et al., 2014; King et al., 2015; Ruggieri et al., 2017; Saintenac, Jiang & Akhunov, 2011; Terraciano et al., 2017 and Zhou & Holliday, 2012). Many factors play a role in the efficiency of a WES approach, including genomic G/C content, the size of exons and introns, the total size of the capture space, and the extent of multiplexing used in the capture (Henry et al., 2014 and King et al., 2015), so assessing efficiency is important for data analysis and for improving captures in future studies.

Betalains in Quinoa

Betalains are important pigments produced by plants in the order Caryophyllales, of which quinoa is a member (Timoneda et al., 2019; Imamura et al.; 2018). Two types of betalains exist: betaxanthins, which are yellow, and betacyanins, which are red. These pigments replace the anthocyanins in the Caryophyllales. They function to attract pollinators, provide photoprotection, act as antioxidants, and improve drought and salinity tolerance (Timoneda et al., 2019). The genetic pathway to produce these pigments involves several CYP76AD genes, DODA,

*c*DOPA5GT, and betanidin 5GT/6GT (Timoneda et al., 2019). Understanding this pathway and manipulating it could lead to improved crop health and nutrition.

In this study, we used EMS mutagenesis create novel genetic diversity in *Chenopodium quinoa*. Through exome capture and whole genome sequencing, we detected mutations in 52 mutant quinoa families and predicted the consequence of these mutations. As a proof of concept, we also identified a mutation that prevents the production of red betacyanins in the hypocotyl of the plant. Finally, we calculated the mutation frequency in the 52 families to guide future research in this population.

METHODS

Mutagenesis and Population Development

Kill curve analysis was performed by treating approximately 100 seeds of the quinoa variety QQ74 (USDA-GRIN PI 614886) with 0%, 1%, 2%, 3%, or 4% EMS. Seeds were sterilized with 100% bleach for 5 minutes and then rinsed with water three times for 5 minutes each time. Seeds were then soaked in the varying concentrations of EMS for 6 hours, after which a 20% sodium thiosulfate 2% NaOH solution was added to the EMS solution to inactivate the EMS. After 20 minutes, the seeds were transferred to a 10% sodium thiosulfate 1% NaOH solution for 20 more minutes. Finally, seeds were again rinsed with water three times for 5 minutes each time. Seeds were then sown on moist filter paper in a Petri dish, and the germination rate was recorded 7 days after sowing.

Large-scale mutagenesis was performed on approximately 6,000 seeds using 2% EMS and approximately 6,000 seeds using 2.5% EMS, according to the procedure described above. Seeds were then sown in flats in the greenhouse and grown to maturity.

The first generation of plants, grown from the original mutagenized seeds, is known as the first mutant generation (M1). Over 5,000 M1 plants were individually harvested to preserve their seed. This seed was planted to produce the second mutant generation (M2). From each M1 plant, 5-10 M2 plants were grown, and distinct, visual, phenotypic mutations were noted. M2 plants were grown to maturity, after which their seed was harvested on either a pooled or on an individual basis. Seed from select M2 plants showing obvious mutant phenotypes was sown to produce the third mutant generation (M3). M3 plants were closely observed to note the occurrence of the mutant phenotype passed down from the M2 parent.

Whole Exome Sequencing (WES)

WES was performed on 44 mutant lines (hereafter referred to as families) at the M2 stage and unmutagenized QQ74 (control). For each mutant family and the control, genomic DNA from 6 plants was extracted using a DNA microprep extraction protocol described by Todd & Vodkin (1996), and DNA samples from each family were pooled. Exonic sequences were then targeted and enriched in all 45 samples in 4 multiplexed reactions using a modified version of the SeqCap EZ HyperCap Workflow Version 2.1 from Roche. Exons were targeted using probes designed by Roche using the predicted coding sequences from the quinoa genome (Jarvis et al., 2017). Multiplexed exome libraries were sent to Novogene Corporation Inc. for Illumina 150-bp paired-end sequencing. After sequencing, we trimmed the reads using trimmomatic, mapped them to the quinoa version 2 reference genome with bwa, and removed PCR duplicates using samtools rmdup (see Appendix 1). After these processing steps, we could use the sequencing data to evaluate the efficiency of the capture and detect EMS-induced mutations.

An important part of a targeted sequencing approach is estimating the efficiency of the target capture, or determining whether the capture was effective at capturing the desired target regions.

To collect information about the efficiency of the exome capture, we used CollectHSMetrics, part of Picard tools (see Appendix 1). CollectHSMetrics takes a bam file and compares it against bait interval and target interval files, which in this case were capture design files from Roche that we reformatted using the bedtools BedToInterval tool. We used four of the output fields from CollectHSMetrics to estimate capture efficiency: On_Target_Bases, PF_Bases_Aligned, Target_Territory, and the number of target bases covered at depths from 0-200+. Target_Territory is the number of bases targeted, in our case the exons of the quinoa genome. PF_Bases_Aligned is the number of bases that pass quality filters whose mapping score is >0, and On_Target_Bases measures the number of PF_Bases_Aligned that map to the target region. To calculate sensitivity, we divided the number of target positions covered 1-200+ times by Target_Territory. We calculated the specificity of the exome capture by dividing On_Target_Bases by PF_Bases_Aligned. We estimated the average coverage of target regions by calculating the expected value of target coverage (excluding target bases covered 0 times) for each line.

Whole Genome Sequencing (WGS)

WGS was performed on nine selected M3 families and a pool of unmutagenized, control plants. The mutant families were selected based on obvious, mutant phenotypes that were observed as the M3 plants were growing. DNA from multiple M3 plants from the same family was pooled for sequencing. Mutant and control DNA pools were sequenced by Novogene Corporation, Inc. using Illumina 150-bp paired-end sequencing. After receiving the sequencing data, we calculated the average depth of coverage across the whole genome by dividing the amount of sequencing data by the genome size. Then we trimmed WGS reads with trimmomatic, mapped them to the quinoa version 2 reference genome with bwa, removed suspected PCR

duplicates with samtools markdup, and passed the processed reads to the mutation detection pipeline to identify EMS-induced SNPs (see Appendix 1).

Mutation Detection

We used the Mutation and Polymorphism Survey (MAPS) program (Henry et al., 2014; http://comailab.genomecenter.ucdavis.edu/index.php/MAPS) to detect EMS-induced mutations. MAPS examines sequencing data from several lines at once and uses two rounds of filtering, MAPS 1 and MAPS 2, to identify mutations caused by EMS treatment. To accommodate differences in sequencing coverage and user preferences, each filtering criterion can be changed to adjust the stringency of mutation calling. The program is usually run several times to determine which parameter values produce the most accurate set of mutation calls.

MAPS 1, the first round of filtering, first determines which positions have sufficient coverage to be assayed for mutations. Positions must be covered in a specific number of libraries, and the coverage across these libraries must sum to a minimum cutoff but not pass a maximum cutoff. After filtering out positions that do not meet the coverage criteria, the remaining positions are examined to determine whether they are heterozygous. This is assessed on the basis of all the libraries present, while individual lines are looked at later. In order for a position to be called heterozygous, the cumulative frequency of the most common two bases at the position must meet a minimum threshold and each of the two bases must individually be present at a minimum frequency. If a position is classified as heterozygous and if both the alleles show up in two or more libraries, the position was most likely heterozygous before mutagenesis. Any positions that meet these criteria are not considered for further analysis. The last filter in MAPS 1 is the minimum coverage of the nonmutant base, which comes into play only when a

position receives low coverage and is present in only a few libraries. After this final criterion, the pared-down set of positions is passed to the second round of filtering, MAPS 2.

MAPS 2 takes any position that made it through MAPS 1 and uses additional parameters to determine whether the position is non-mutant, a heterozygous mutant, or a homozygous mutant. Each parameter in MAPS 2 looks at a position within an individual library, not amongst all libraries as MAPS 1 does. If a position in one library is polymorphic to that position in all other libraries, it is investigated for being heterozygous or homozygous. If the position passes the heterozygous coverage cutoff and two alleles at the position each make up a minimum percentage of the reads, it is called as heterozygous. If a polymorphic position does not pass both heterozygous criterion, but it does pass a homozygous coverage cutoff and varies from other libraries being analyzed, it is called as homozygous. Homozygous positions and heterozygous positions are passed to a final output file detailing every mutation found in the libraries analyzed.

Since MAPS has never been used in quinoa and because it requires optimization for different sequencing depths, we tested several levels of stringency to determine parameter values that produced the most accurate and complete set of mutations in our mutant families. Nearly all of the EMS-induced mutations should be G>A or C>T transition mutations, so we looked for parameter values that detected a high proportion of these type of mutations. To avoid parameter values that are too stringent and filter out too many true mutations, we looked for values that identified a relatively high number of mutations while still maintaining a high proportion of EMS-type mutations. We chose two different values for two parameters in MAPS 1, -i and -v, and two values for two parameters in MAPS 2, -s and -p, and we tested 16 combinations of these parameters. Appendix 1 contains more detailed information on running and optimizing MAPS.

Predicting the Effects of Mutations Using Ensembl VEP

We used Ensembl VEP (https://uswest.ensembl.org/info/docs/tools/vep/index.html) to predict the consequences of the mutations we found using MAPS. This involved creating a custom annotation using an improved quinoa reference genome and annotation (Jarvis, unpublished data) and formatting the MAPS output to interact with VEP. We then ran VEP using the default output parameters, which generated a set of predicted mutation consequences. We consolidated some consequences into larger categories. See Appendix 1 for more details.

Identifying Betalain Synthesis Pathway Mutations

Mutant family 2427 produced green hypocotyls, in contrast to the normal red hypocotyls produced in control QQ74. We searched for mutations in family 2427 in the genes involved in the betalain synthesis pathway described by Timoneda et al. in 2019. Homologs of CYP76AD (gene IDs 110733547, 110699387, 110733713, 110699385) and DODA are already annotated in the quinoa reference genome. For the other genes, we obtained the nucleotide sequence from NCBI and used CoGe BLAST to find homologous genes in the quinoa genome. We then determined whether any of these betalain pathway gene homologs contained mutations identified by MAPS and VEP. See Appendix 1 for more details.

Calculating a Mutation Rate

We used the method from Henry et al. (2014) to calculate homozygous and heterozygous mutation density in our mutant population (see Appendix 1). To obtain the most accurate mutation rate, we divided the number of mutations by only the number of bases with sufficient coverage to be assayed by the MAPS program. As reported by Henry et al., (2014), some special considerations have to be applied when calculating the mutation rate of heterozygous mutations.

For positions receiving low coverage, it is less likely that a truly heterozygous position meets the minimum coverage of the mutant allele (-d). We required 5x coverage of the mutant allele for MAPS to classify the position as heterozygous, so if a heterozygous position is covered only 6 times, there is only a 9.375% chance that 5 mutant alleles will be present in the sequencing data. Therefore, only 9.375% of positions receiving 6x coverage should be taken into consideration when calculating a mutation rate. We adjusted the number of assayed positions accordingly to calculate the heterozygous mutation rate. To calculate the homozygous mutation rate, we simply used the number of assayed bases that met the minimum cutoff for a position to be called as homozygous.

RESULTS & DISCUSSION

Mutagenesis and Population Development

The variety QQ74 (USDA-GRIN PI 614886) from Maule, Chile was selected for EMS mutagenesis. The two previously published EMS quinoa populations were based in CQ127 (Imamura et al., 2018), and Regalona-Baer (Mestanza et al., 2019). CQ127 (USDA-GRIN PI 614927) comes from La Paz, Bolivia, and is in the southern highland quinoa group (Christensen et al., 2007). Regalona-Baer is a commercial variety grown in southern Chile, but it is more similar to southern highland varieties than southern Chilean ecotypes (von Baer, Bazile & Martinez, 2009). Since QQ74 is a lowland ecotype from Chile (Christensen et al., 2007), this EMS population will likely give insights specific to the lowland group.

We determined by kill curve analysis that between 2% and 3% EMS resulted in 50% lethality (Table 1). Because of homeologous gene duplication, quinoa should show reduced lethality at high mutation densities. Therefore, although 2% EMS treatment resulted in lethality below 50%,

plants treated with this concentration should still accumulate a high mutation density. The 3% treatment led to 25% lethality, which was higher than we wanted since it killed to many seeds. A 25% lethality rate would also result in such a high number of mutations that it would be difficult to determine causative mutations for interesting phenotypes. Based on what we observed in the kill curve analysis, we used both 2% and 2.5% EMS to create the population in order to balance mutation rates and lethality. We mutagenized 12,000 quinoa seeds, 6,000 with 2% EMS and 6,000 with 2.5% EMS, and grew and harvested seed from 5,030 M1 individuals. At present, 338 families have been advanced to the M2 generation, and 17 families have been advanced to the M3 generation.

WES

To identify mutations in the EMS-treated families, we performed exome capture on 44 M2 families in four multiplexed reactions and sequenced the products (Table 2). After sequencing, we assessed the performance of the capture to determine whether we captured the desired target regions and what level of coverage these regions were sequenced at. To answer these questions, we first measured the sensitivity of the exome capture, or the proportion of target bases that were sequenced. Median sensitivity was 0.964, with a maximum of 0.969 and a minimum of 0.956 (Figure 1). Sensitivity for the unmutagenized control line was 0.956. The high sensitivity indicates that we captured nearly all the target bases; we missed only 3.6% of them.

In addition to sensitivity, we assessed specificity, or the proportion of sequenced bases that map to the target regions. Median specificity was 0.425, with a maximum of 0.452 and a minimum of 0.390 (Figure 2). Specificity in the unmutagenized control line was 0.409. The specificity of our capture is less than ideal but is similar to other capture experiments (Table 3). The low specificity could be explained by the large fragment size we used in the capture. We

sheared DNA to 300 bp, but in reality many samples had fragment sizes between 400 bp and 500 bp. These fragments could include target bases as well as nearby, off-target bases, all of which will be pulled down by the exome capture probes and included in the sequencing reactions. This would result in sequencing of off-target regions near the target regions, but not sequencing from regions far from the targets. This phenomenon was confirmed when we plotted the coverage from the WES along the quinoa chromosomes and saw that coverage followed the density of target regions quite well (Figure 3). Shearing DNA to a smaller size could alleviate this problem.

Finally, we assessed the coverage obtained for each sequenced target base. The average coverage for all target bases sequenced was calculated using the expected value of coverage for these bases, which is calculated as $\frac{\sum_1^{200} depth\ of\ coverage \cdot \#\ of positions\ at\ that\ coverage}{total\ target\ positions}$. Bases with 0 coverage were not included, and all bases covered more than 200 times were assigned a coverage of 200. On average, target bases were covered 17.6 times (Figure 4).

WGS

Many EMS studies in plants have used WES to identify induced mutations (Henry et al., 2014; King et al., 2015; Lu et al., 2016; Ruggieri et al., 2017; Saintenac, Jiang & Akhunov, 2011; Terraciano et al., 2017; Zhou & Holliday, 2012), and some studies have utilized WGS to detect mutations (Imamura et al., 2018). As an alternative approach to WES, we sequenced nine M3 families. These families were chosen because they showed distinct mutant phenotypes seen in the M2 generation (Table 4) and would be useful in linking outwardly visible phenotypes to molecular changes in the DNA. We obtained between 16.3 Gb and 21.3 Gb of data, or a mean of 18.9 Gb, for the WGS. This resulted in approximately 11.9X – 15.3X coverage of the genome,

with a mean of 13.6X (Figure 5). This data provided sufficient coverage for MAPS to detect mutations.

Mutation Detection

After investigating the WES and WGS data, we used MAPS to detect the mutations caused by the EMS treatment. To optimize the stringency of SNP detection in WES, we varied several parameters in MAPS (Figures 6 and 7) and identified values for these parameters that provided a balance between a high number of total SNPs and a high proportion of canonical EMS-type SNPs. We found that increasing the minimum coverage of a base in a single family for a homozygous SNP call (-s) from 2 to 4 increased the proportion of EMS-type SNPs that were detected, so we used a -s of 4. For the minimum percentage of wild type (WT) and mutant alleles (-p), increasing the value from 10 to 20 didn't dramatically change the proportion of EMS SNPs detected, but it did substantially decrease the total of number of SNPs. So, we used a -p of 10. Next, using a -s of 4 and -p of 10, we found that a minimum percentage of the two most common bases (-i) of 20 omitted too many EMS-type SNPs, so we chose a -i of 10. Increasing the minimum coverage across multiple libraries (-v) from 10 to 20 very slightly increased the proportion of EMS SNPs, so we chose a -v of 20. Using a -s of 4, a -p of 10, a -i of 10, and a -v of 20 (parameter set B3 in Figure 8) we detected 29,861 total SNPs, of which 29,058 (97.31%) were EMS-type SNPs. In contrast, only three SNPs were identified in the unmutagenized control plants, none of which were EMS-type (Figure 9). These SNPs in the control plants likely represent variation within QQ74 that existed before EMS treatment. Since the seeds we mutagenized were not from the originally sequenced plant used to produce the reference genome but were derived from its siblings, a small amount of variation should exist between our control plants and the reference genome. The low number of SNPs detected in the control plants

compared to the much higher numbers in the mutagenized families suggests that MAPS is effectively picking up EMS-induced mutations.

To determine whether the number of identified mutations was limited by sequencing depth, we plotted the total number of mutations against the amount of sequencing data obtained for each mutant family. We fit a line to the plot to determine whether the number of mutations correlated with the amount of sequencing data. A steep line with a statistically significant slope would indicate a correlation between these two variables, suggesting that additional sequencing would still be needed to identify most of the mutations in the lines with less sequencing data. Fitting a line to the WES data returned a very low R-squared value, and a one-way ANOVA used to test for the statistical significance of the slope against the null hypothesis of no slope gave a p-value of 0.075 (Figure 10). When the family with the highest number of mutations, family 88, is removed from the data set and a new line is fit, the R-squared is 0.113 and the p-value is 0.030 (Figure 10). Despite the statistical significance of the slope, the low R-squared value indicates very little correlation between sequencing depth and the number of identified mutations. In addition, we have no reason to eliminate family 88 from the analysis other than that it has the highest number of mutations, so keeping this point in the fit makes more sense. The poor correlation between SNPs and sequencing data suggests that the WES was sufficiently deep to identify most mutations.

We followed the same mutation detection procedure for WGS families as we did for WES families. It was necessary to optimize MAPS again with the WGS data because the coverage and number of libraries was different than in the WES. When we tested the sixteen different combinations of parameter values (Figure 7), we saw six clusters (Figure 11) grouped by -i and -s. A -i of 10 and a -s of 4 produced the highest proportion of EMS SNPs, but lowering the -s to 2

increased the total number of SNPs considerably, including more EMS-type SNPs. When using a -i of 10 and a -s of 4, higher -p and -v values also gave a larger proportion of EMS-type SNPs. Based on these results, we selected a -s of 2, a -p of 20, a -i of 10, and a -v of 20 (parameter set B2 in Figure 5) for analysis of the WGS data, which detected 170,905 total SNPs, of which 160,942 (94.17%) were EMS-type SNPs. In the control, 242 mutations were identified, of which 51 were EMS-type SNPs. This number is most likely higher than the number of mutations seen in the control plants in the WES because we sequenced more of the genome with WGS and because we sequenced non-genic areas, which generally accumulate more mutations than genic regions. In addition, the EMS-type mutations seen in the WGS control are not unexpected; these type of mutations are typical of EMS treatment but certainly may also arise from natural mutation. We likely just didn't identify a sufficienct number of SNPs in the WES control to detect these types of transition mutations.

After detecting mutations using MAPS, we did a linear regression with the WGS mutations against the amount of sequencing data. The linear regression resulted in an R squared value of 0.247, and a p-value from one-way ANOVA testing the significance of the slope of 0.144. Based on the R squared value and the p-value, the number of SNPs detected is not correlated with the amount of sequencing data, which supports our assumption that the WGS sequencing was sufficiently deep to identify most mutations.

Predicting Mutation Consequences

Using the mutations detected by MAPS, we visualized mutation density along the 18 quinoa chromosomes for the WES mutations and the WGS mutations (Figure 12). As expected, WES generally detected mutations in gene-dense regions while WGS detected mutations across the whole genome. Mutation density in the WES lines increases on the ends of the chromosomes,

where genes are located, but in the WGS families, the mutations are more concentrated towards the center of the chromosomes where gene density is lower. We would expect this to be the case not only because we targeted genes in the WES, but because mutations in non-genic regions are less likely to be lethal. Therefore, more mutations would be expected to accumulate towards the center of the chromosomes where there are fewer genes.

These patterns in mutation density were validated when we predicted the consequences of mutations with VEP. We expected WES mutations to be more concentrated in and near coding sequences than mutations from WGS. We found many more mutations in and near genes in the CDSs, UTRs and introns of WES families than we did in WGS families, with 52% of WES mutations but only 12% of WGS mutations in these regions. These results are similar to an exome capture and sequencing study in hexaploid wheat in which 86% of all mutations were located in the CDSs, UTRs, and introns (King et al., 2015). When we broke down the mutations within the CDSs, we saw the same pattern. Mutations in the CDSs include missense, start codon, and stop codon mutations. Out of all the mutations, 43.6% and 3.03% were classified as missense in the WES and WGS lines, respectively (Figure 13). Start lost mutations were similar, with 0.06% start lost mutations in the WES and 0.007% in the WGS. Premature stop codons continue the trend as 2.6% of WES mutations and 0.20% of WGS mutations.

Despite the differences in the proportion of total mutations that fall in coding regions, the proportion of different types of mutations within CDSs is remarkably similar between the WES and WGS (Figure 14). Missense mutations account for 67.5% of WES CDS mutations and 68.2% of WGS CDS mutations, start lost mutations are 0.10% of WES CDS mutations and 0.16% of WGS CDS mutations, and stop gained mutations make up 4.09% of WES CDS

mutations and 4.4% of WGS CDS mutations. Since we expect EMS to act consistently, this similarity confirms the accuracy of mutation detection by MAPS in WES and WGS approaches.

It is important to note that our pipeline does not adequately address deletions because MAPS reports all mutations as SNPs. For example, a 4-bp deletion would be listed as four, 1-bp deletions by MAPS. VEP will therefore not predict the effect of a 4-bp deletion, but the individual effects of four, 1-bp deletions. Multiple-bp deletions are uncommon in the families we sequenced, but they are much more likely to have effects if they occur in genes. When looking for a causative mutation, multi-bp deletions should be searched for manually.

A Mutation in the Betalain Synthesis Pathway

To validate the mutation calling done with MAPS and the consequence prediction done with VEP, we investigated mutations in genes known to function in betalain biosynthesis in M2 family 2427. In this family, we observed a change in the color of the hypocotyl, which is red in WT QQ74 plants but was green in this mutant family. We identified quinoa genes involved in the betalain synthesis pathway (Timoneda et al., 2019) and checked whether MAPS detected mutations within them in family 2427 (Table 5). We identified only one mutation in the betalain synthesis pathway, a G>A transition in the last base of codon 124 of the CYP76AD1-1 gene (Figure 15). The mutation is identical to one of the two mutations discovered by Imamura et al. (2018) in their green hypocotyl mutant, and it causes a premature stop codon. Because this premature stop is early in the gene, is the only mutation we found in genes of the betalain synthesis pathway, and is identical to an independently identified mutation that causes the same phenotype, it is very likely that this mutation caused the mutant phenotype. Since this mutation was detected successfully by MAPS, and its result, a premature stop codon, was accurately

predicted by VEP we have good evidence that MAPS is finding real mutations and VEP is predicting accurate consequences from these mutations.

Calculating a Mutation Rate

Knowing that MAPS was working correctly, we were able to use the detected mutations to calculate a mutation rate and estimate the number of mutant families needed to identify a given number of mutant alleles for any gene. For the WES, we calculated a median heterozygous mutation rate of 11.31 mutations/Mb, and a median homozygous mutation rate of 0.035 mutations/Mb, for a total of 11.35 mutations/Mb (Figure 16 and Table 6). For the WGS, we calculated a median heterozygous mutation rate of 21.67 mutations/Mb, and a homozygous rate of 3.93 mutations/Mb for a total of 25.6 mutations/Mb (Figure 16 and Table 7). Other studies have reported a wide range of mutation rates, such as 20.1 mutations/Mb in tetraploid wheat (Henry et al., 2014), 35 mutations/Mb in hexaploid wheat (King et al., 2015), 5.2 mutations/Mb in rice (Henry et al., 2014), and 4.9 mutations/Mb in quinoa (Mestanza et al., 2019). Since quinoa is a tetraploid, we expected our mutation rate to be similar to the rate in tetraploid wheat, which it was.

However, we detected a much higher mutation rate in quinoa than Mestanza et al. (2019), although they used a 2% EMS solution and a longer treatment time, 8 hrs. The discrepancy could be attributed to differences in permeability of the seed coat in the variety Regalona-Baer used by Mestanza et al. and the QQ74 that we used, or it could be caused by the way the seeds were distributed in the EMS solution. Additionally, Mestanza et al. analyzed mutations in only six genic regions. Since our WES contained many more genic regions plus some non-genic regions and the WGS captured the entire genome, it is not surprising that we obtained a much higher mutation rate.

The differences in our WES and WGS mutation rates can be explained by the regions we targeted with the sequencing. The higher rate found in WGS data is consistent with our expectation that mutations in non-genic regions are less likely to be lethal. Since we did not target non-genic regions in the WES, we would expect to see a lower mutation rate in those sequenced families. We also expected to see many more heterozygous mutations than homozygous mutations in both the WGS and WES families, since we used early generations (M2 and M3), and since some mutations may be lethal when homozygous.

We used the calculated mutation rate to estimate the number of mutant families necessary to find mutants in any specific gene. Using the WES as a minimum mutation rate and the WGS as a maximum mutation rate, and given the quinoa genome size of 1,450 Mb, of which 57 Mb make up the 44,776 genes, we estimated that in 300 mutant families we will discover 4-10 mutant alleles of any given gene. Of course, these mutant alleles may not significantly alter the gene product. The CDS mutations that are most likely to cause changes in gene products (missense, start lost, stop gained, and stop lost), were 72% of CDS mutations in the WES and 73% of CDS mutation in the WGS. This means that it is more realistic to estimate that 300 mutant families would include 3-7 mutations with phenotypic effects in any given gene.

Cost Analysis

We conducted a cost analysis of the WES and WGS approaches to inform future sequencing work in this project and similar projects. In both sequencing approaches, sample preparation and sequencing were the major costs, since we used free software packages, MAPS and VEP, for the mutation analysis. For the WES, we did the wet lab work for exome capture and library prep of 44 mutant families and a control, which cost $6,000. Sequencing these 45 samples cost $1, 400, putting the total cost at $7,400. It should be noted that either the capture reaction or library prep

failed in one mutant family, so the sequencing data we got back really represented only 44 samples. For those 44 samples, we received a total of 120 Gb of sequencing data, which amounts to $61.67/Gb. We detected 29,858 mutations, which results in $0.25/mutation. This cost does include the three mutations from the control sample, so the cost per mutation would be less if a control was not included in future sequencing runs (Table 8).

For the WGS, library prep and sequencing were done by Novogene, Inc. for $2,800. We received 169 Gb of sequencing data, for a cost of $16.57/Gb. We detected 170,663 mutations in the 10 WGS samples, which results in $0.02/mutation (Table 8). Again, this cost includes the cost of running a control sample, which contributed 242 mutations, so not using a control would lower the cost per mutation. If we had done the WGS for 44 samples, assuming the same cost per Gb and per mutation, the total would have been $12,320 (Table 8).

Obviously, the WES approach is less expensive on a per-sample basis. However, excluding non-genic regions may not be desirable, since many promoters and enhancers can extend great distances from genes. Mutations in these regulatory elements as well as mutations in intergenic regions of unknown function could have important phenotypic effects. Determining the value of capturing intergenic mutations will be an important consideration for additional sequencing work in this project and for pursuing similar work in other projects.

CONCLUSIONS

We have created an EMS quinoa mutant population of 5,030 families. Using both targeted and whole genome sequencing approaches, we have characterized mutations in 52 mutant families. We estimate that the mutation frequency in this population ranges from 11.35 - 25.6 mutations/Mb. We have characterized the predicted effects of the identified mutations in the 52

sequenced lines, enabling the identification of candidate mutations that might underlie observed mutant phenotypes. Through a betalain synthesis mutant, we have demonstrated the ability to discover causative mutations for mutant phenotypes. Based on the mutation rate, 300 hundred families should be sufficient to discover 3-7 mutations in any gene of interest and conduct forward and/or reverse genetics studies. In addition, valuable breeding traits may be discovered as more mutant families are phenotyped. We expect this population to be a valuable resource for the genetic improvement of quinoa.

LITERATURE CITED

Arisha, M. H., Shah, S. N. M., Gong, Z., Jing, H., Li, C., & Zhang, H. (2015). Ethyl methane sulfonate induced mutations in M2 generation and physiological variations in M1 generation of peppers (*Capsicum annuum L.*). *Frontiers in Plant Science, 6*, 399. doi:10.3389/fpls.2015.00399

Bhargava, A., & Srivastava, S. (2013). *Quinoa: Botany, production, and uses*. Boston, MA: CABI. Retrieved from https://search.lib.byu.edu/byu/record/lee.6913980

Caldwell, D. G., Nicola, M., Paul, S., Muehlbauer, G. J., Marshall, D. F., & Robbie, W. (2004). A structured mutant population for forward and reverse genetics in barley (*Hordeum vulgare* L.). *The Plant Journal, 40*(1), 143-150. doi:10.1111/j.1365-313X.2004.02190.x

Christensen, S. A., Pratt, D. B., Pratt, C., Nelson, P. T., Stevens, M. R., Jellen, E. N., . . . Maughan, P. J. (2007). Assessment of genetic diversity in the USDA and CIP-FAO international nursery collections of quinoa (*Chenopodium quinoa* willd.) using microsatellite markers. *Plant Genetic Resources, 5*(2), 82-95. doi:10.1017/S1479262107672293

Comai, L., & Henikoff, S. (2006). TILLING: Practical single-nucleotide mutation discovery. *The Plant Journal, 45*(4), 684-694. doi:10.1111/j.1365-313X.2006.02670.x

Francisco F. Fuentes, & Ximena Paredes-Gonzalez. (2015). Nutraceutical perspectives of quinoa: Biological properties and functional applications. In Rome (Ed.), *State of the art report of quinoa in the world in 2013* (pp. 286-299). Santiago, Chile: FAO & CIRAD. doi:10.13140/RG.2.1.4294.2565 Retrieved from https://search.datacite.org/works/10.13140/RG.2.1.4294.2565

Greene, E. A., Codomo, C. A., Taylor, N. E., Henikoff, J. G., Till, B. J., Reynolds, S. H., . . . Henikoff, S. (2003). Spectrum of chemically induced mutations from a large-scale reverse-genetic screen in *Arabidopsis. Genetics, 164*(2), 731-740. Retrieved from http://www.genetics.org/cgi/content/abstract/164/2/731

Henry, I. M., Nagalakshmi, U., Lieberman, M. C., Ngo, K. J., Krasileva, K. V., Vasquez-Gross, H., . . . Comai, L. (2014). Efficient genome-wide detection and cataloging of EMS-induced mutations using exome capture and next-generation sequencing. *The Plant Cell, 26*(4), 1382-1397. doi:10.1105/tpc.113.121590

Imamura, T., Takagi, H., Miyazato, A., Ohki, S., Mizukoshi, H., & Mori, M. (2018). Isolation and characterization of the betalain biosynthesis gene involved in hypocotyl pigmentation of the allotetraploid *Chenopodium quinoa*. *Biochemical and Biophysical Research Communications, 496*(2), 280-286. doi://doi.org/10.1016/j.bbrc.2018.01.041

Kaur, P., & Gaikwad, K. (2017). From genomes to GENE-omes: Exome sequencing concept and applications in crop improvement. *Frontiers in Plant Science, 8*, 2164. doi:10.3389/fpls.2017.02164

King, R., Bird, N., Ramirez-Gonzalez, R., Coghill, J. A., Patil, A., Hassani-Pak, K., . . . Phillips, A. L. (2015). Mutation scanning in wheat by exon capture and next-generation sequencing. *PloS One, 10*(9), e0137549. doi:10.1371/journal.pone.0137549

Lopez-Nieves, S., Yang, Y., Timoneda, A., Wang, M., Feng, T., Smith, S. A., . . . Maeda, H. A. (2018). Relaxation of tyrosine pathway regulation underlies the evolution of betalain pigmentation in Caryophyllales. *New Phytologist, 217*(2), 896-908. doi:10.1111/nph.14822

Massawe, F., Mayes, S., & Cheng, A. (2016). *Crop diversity: An unexploited treasure trove for food security.* doi://doi.org/10.1016/j.tplants.2016.02.006

Mestanza, C., Riegel, R., Vásquez, S., Veliz, D., Cruz Rosero, N., Canchignia, H., & Silva, H. (2019). Discovery of mutations in *Chenopodium quinoa* Willd through EMS mutagenesis and mutation screening using pre-selection phenotypic data and next-generation sequencing. *Journal of Agricultural Science,* , 1-9. doi:10.1017/S0021859619000182

Ruggieri, V., Anzar, I., Paytuvi, A., Calafiore, R., Cigliano, R. A., Sanseverino, W., & Barone, A. (2017). Exploiting the great potential of sequence capture data by a new tool, SUPER-CAP. *DNA Research, 24*(1), 81-91. doi:10.1093/dnares/dsw050

Saba, N., & Mirza, B. (2002). Ethyl methane sulfonate induced genetic variability in *Lycopersicon esculentum*. *International Journal of Agriculture and Biology, 4*(1)

Saintenac, C., Jiang, D., & Akhunov, E. D. (2011). Targeted analysis of nucleotide and copy number variation by exon capture in allotetraploid wheat genome. *Genome Biology, 12*(9), R88. doi:10.1186/gb-2011-12-9-r88

Sepulveda-Jimenez, G., Rueda-Benitez, P., Porta, H., & Rocha-Sosa, M. (2005). A red beet (*Beta vulgaris*) UDP-glucosyltransferase gene induced by wounding, bacterial infiltration and oxidative stress. *Journal of Experimental Botany, 56*(412), 605-611. doi:10.1093/jxb/eri036

Sidhu, G., Mohan, A., Zheng, P., Dhaliwal, A. K., Main, D., & Gill, K. S. (2015). Sequencing-based high throughput mutation detection in bread wheat. *BMC Genomics, 16*, 962. doi:10.1186/s12864-015-2112-1

Sikora, P., Chawade, A., Larsson, M., Olsson, J., & Olsson, O. (2011). Mutagenesis as a tool in plant genetics, functional genomics, and breeding. *International Journal of Plant Genomics, 2011*, 314829-13. doi:10.1155/2011/314829

Teer, J. K., & Mullikin, J. C. (2010). Exome sequencing: The sweet spot before whole genomes. *Human Molecular Genetics, 19*(R2), R145-R151. doi:10.1093/hmg/ddq333

Terracciano, I., Cantarella, C., Fasano, C., Cardi, T., Mennella, G., & D'Agostino, N. (2017). Liquid-phase sequence capture and targeted re-sequencing revealed novel polymorphisms in tomato genes belonging to the MEP carotenoid pathway. *Scientific Reports, 7*, 5616. doi:10.1038/s41598-017-06120-3

Timoneda, A., Feng, T., Sheehan, H., Walker-Hale, N., Pucker, B., Lopez-Nieves, S., . . . Brockington, S. (2019). The evolution of betalain biosynthesis in Caryophyllales. *New Phytologist, 224*(1), 71-85. doi:10.1111/nph.15980

Timoneda, A., Sheehan, H., Feng, T., Lopez-Nieves, S., Maeda, H. A., & Brockington, S. (2018). Redirecting primary metabolism to boost production of tyrosine-derived specialised metabolites *in planta*. *Scientific Reports, 8*(1), 17256-8. doi:10.1038/s41598-018-33742-y

Troisi, J., Di Fiore, R., Pulvento, C., D'andria, R., Vega-Gálvez, A., Miranda, M., . . . Lavini, A. (2015). Saponins. In Rome (Ed.), *State of the art repot of quinoa in the world in 2013* (pp. 267-277). Santiago, Chile: FAO & CIRAD.

Uauy, C., Paraiso, F., Colasuonno, P., Tran, R. K., Tsai, H., Berardi, S., . . . Dubcovsky, J. (2009). A modified TILLING approach to detect induced mutations in tetraploid and hexaploid wheat. *BMC Plant Biology, 9*(1), 115. doi:10.1186/1471-2229-9-115

Vogt, T. (2002). Substrate specificity and sequence analysis define a polyphyletic origin of betanidin 5- and 6-O-glucosyltransferase from *Dorotheanthus bellidiformis*. *Planta, 214*(3), 492-495. doi:10.1007/s00425-001-0685-1

von Baer, I., Bazile, D., & Martinez, E. A. (2009). Cuarenta años de mejoramiento de quinoa *(Chenopodium quinoa* Willd) en la Araucania: Origen de "la Regalona-B". *Revista Geográfica De Valparaíso, 42*, 34-44. Retrieved from https://www.openaire.eu/search/publication?articleId=od_____3631::86dc404976a32ebfa3aa2194d8242d70

Warr, A., Robert, C., Hume, D., Archibald, A., Deeb, N., & Watson, M. (2015). Exome sequencing: Current and future perspectives. *G3 (Bethesda, Md.), 5*(8), 1543-1550. doi:10.1534/g3.115.018564

Wu, G. (2015). Nutritional properties of quinoa. In K. Murphy, & J. Matanguihan (Eds.), *Quinoa: Improvement and sustainable production* (pp. 193-210). Hoboken, New Jersey: John Wiley & Sons, Inc.

Zhou, L., & Holliday, J. A. (2012). Targeted enrichment of the black cottonwood (*Populus trichocarpa*) gene space using sequence capture. *BMC Genomics, 13*(1), 703. doi:10.1186/1471-2164-13-703

Zurita-Silva, A., Fuentes, F., Zamora, P., Jacobsen, S. -., & Schwember, A. R. (2014). Breeding quinoa (*Chenopodium quinoa* Willd.): Potential and perspectives. *Molecular Breeding, 34*(1), 13-30. doi:https://doi.org/10.1007/s11032-014-0023-5

FIGURES

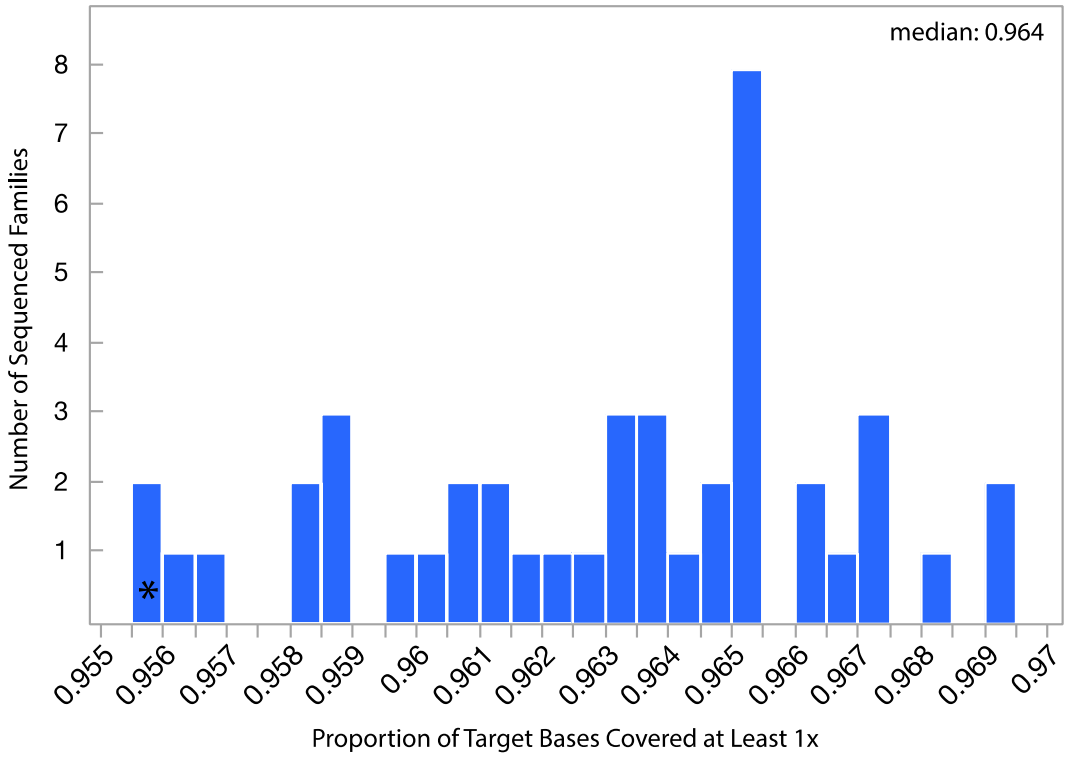

Figure 1. Sensitivity, or proportion of target bases sequenced. WES does not capture 100% of the targeted sequence; some targets will not be captured. This histogram shows the distribution of the proportion of targeted bases with at least 1x coverage from the Illumina sequencing. Among all the lines sequenced, the median proportion of target bases captured was 0.964, or the exome capture successfully pulled out 96.4% of the bases it was designed to capture. The asterisk represents the unmutagenized control, in which 95.6% of the target bases were captured.

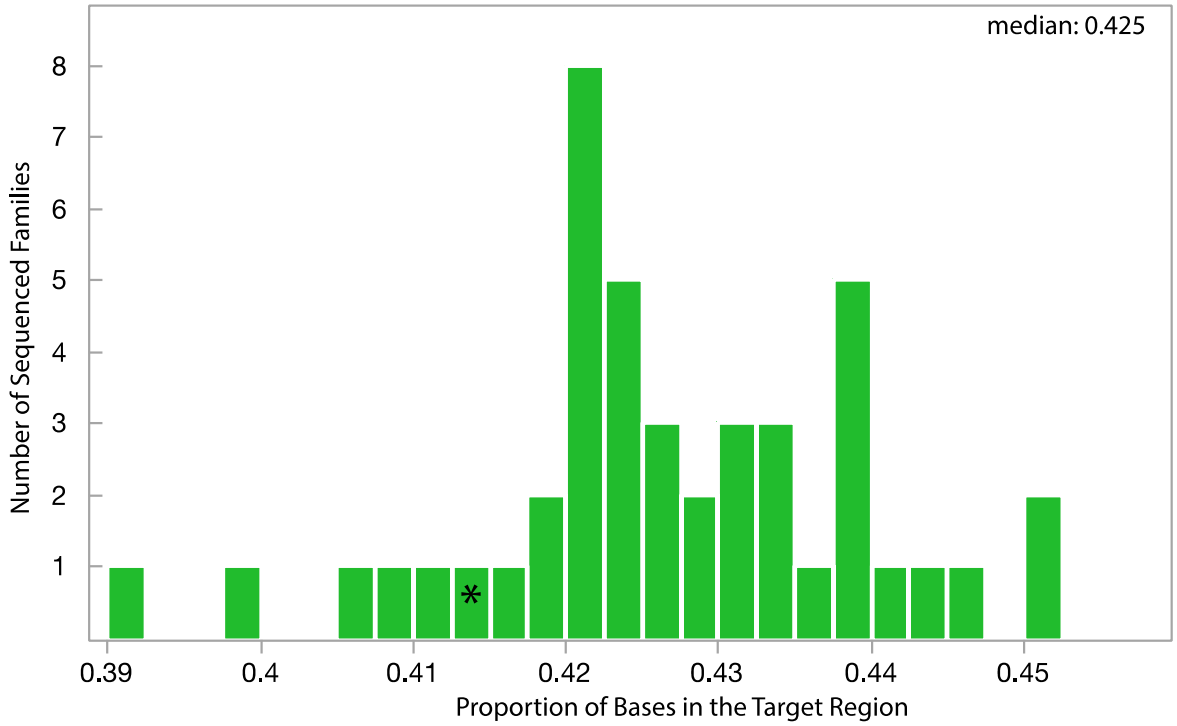

Figure 2. Specificity, or proportion of bases from the exome capture sequencing that aligned to target sequence (exons). The median proportion of on-target bases is 0.425, or in other words, the exome capture was 42.5% efficient at isolating exonic sequences from the genome. The bar with the asterisk indicates the unmutagenized control, in which 41.4% of the bases captured aligned to target regions (exons).

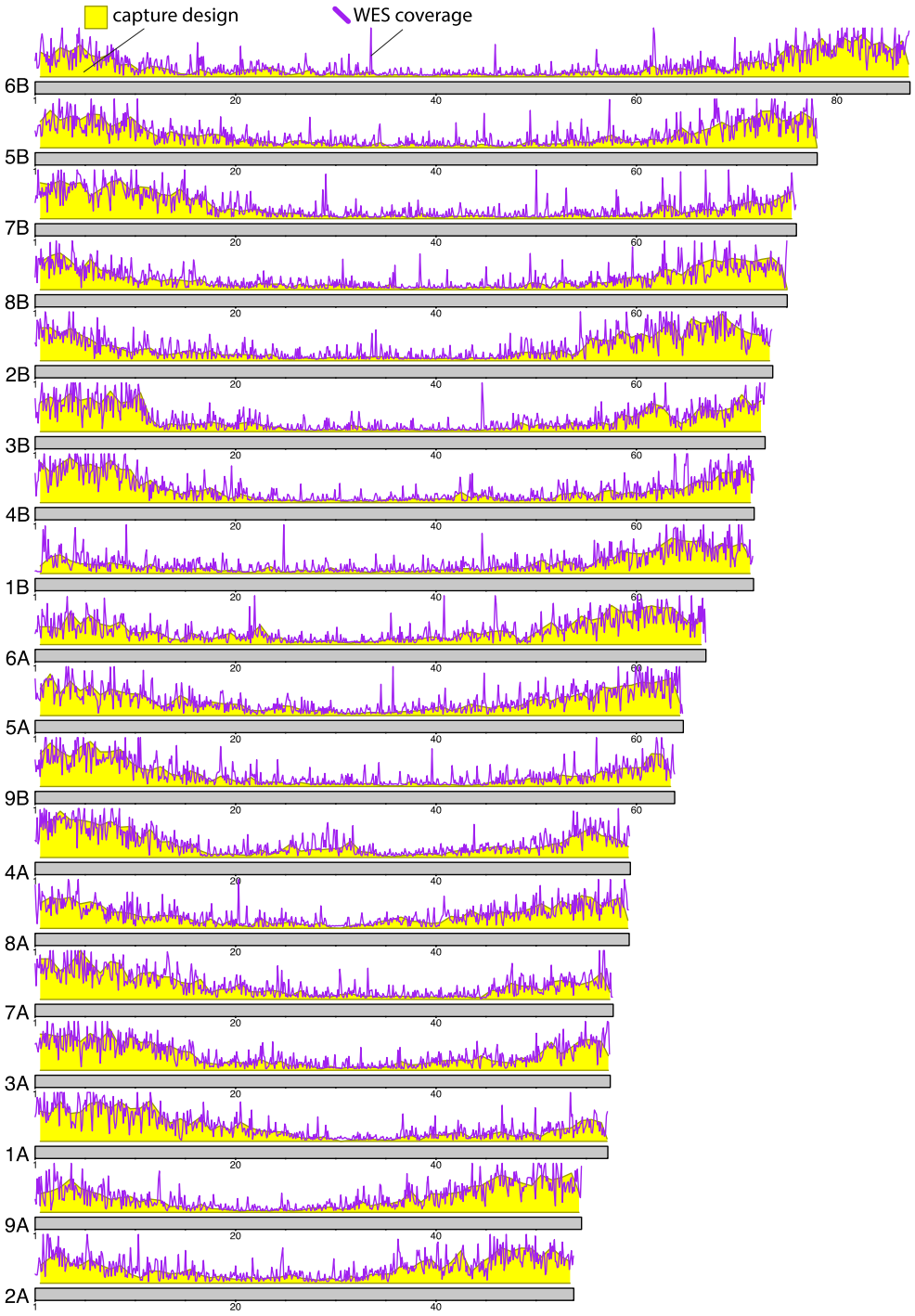

Figure 3. Specificity visualized across all 18 quinoa chromosomes. The yellow density plot shows regions targeted by the exome capture design. The purple line shows the coverage across the chromosome obtained from WES. The purple line follows the yellow plot, which indicates that most off-target sequencing was nearby targeted regions.

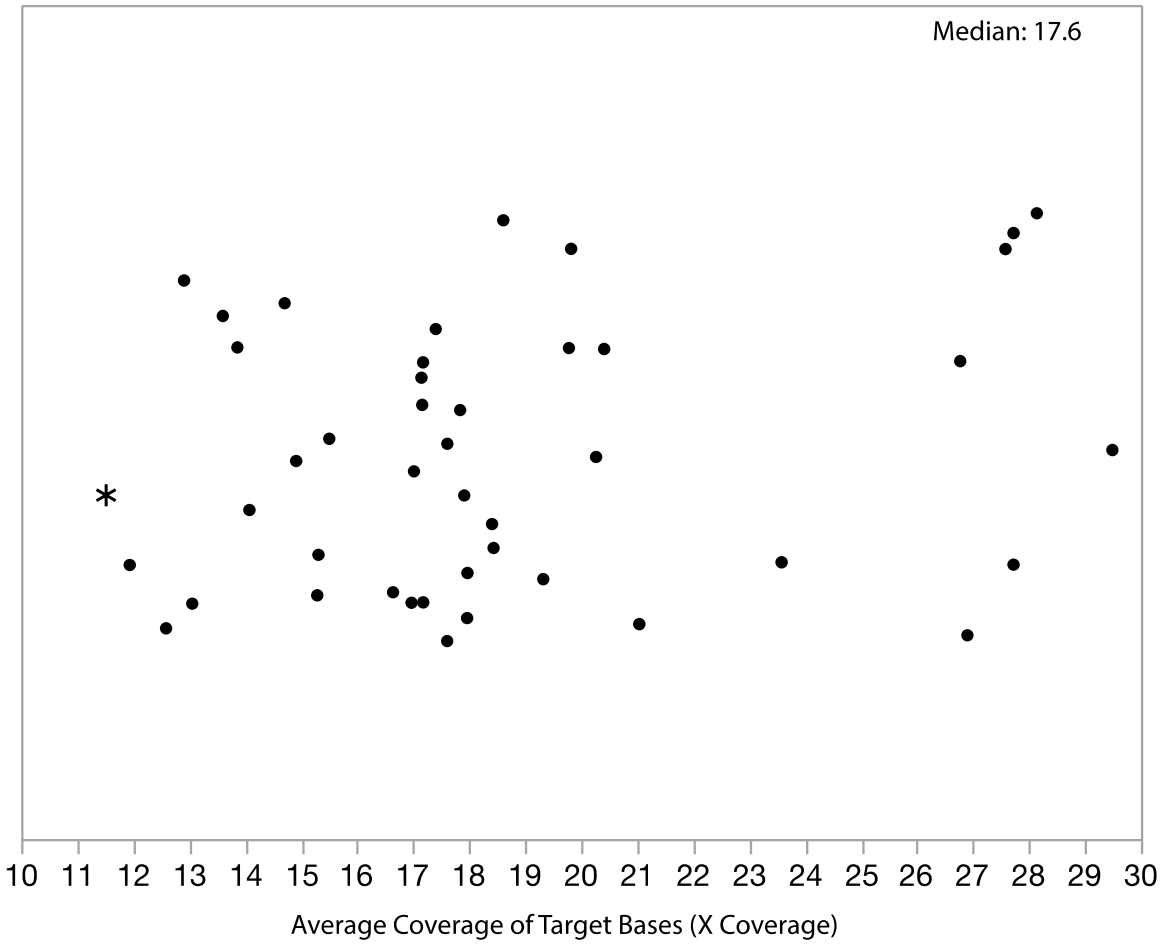

Figure 4. Average coverage of target bases covered at least once in all 44 WES families. Each point represents one sequenced family. The asterisk indicates the unmutagenized, control line.

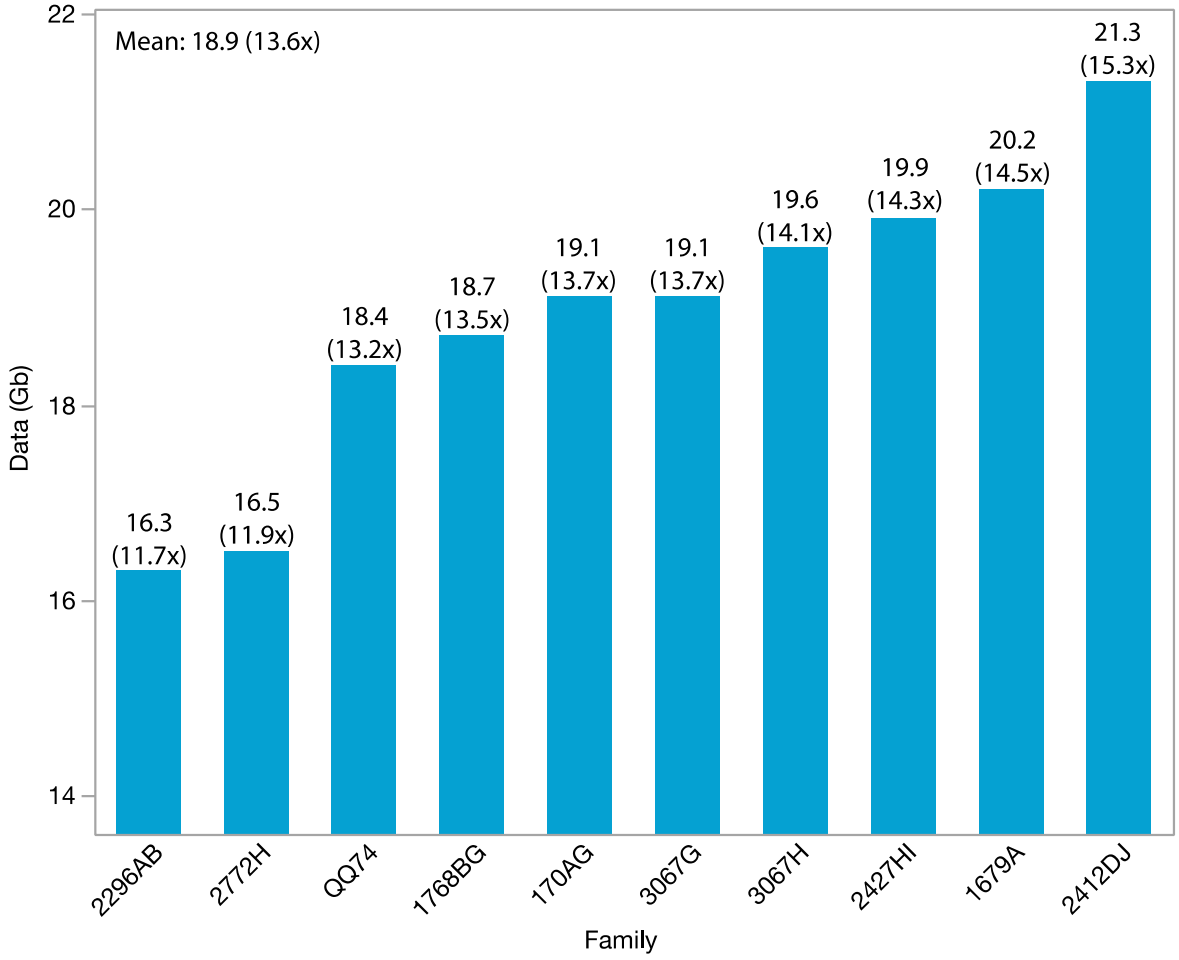

Figure 5. WGS data and coverage. Between 16.3 Gb and 21.3 Gb of data (mean 18.9Gb) was obtained from WGS, which results in 11.7X -15.3X coverage (mean 13.6X) of the genome.

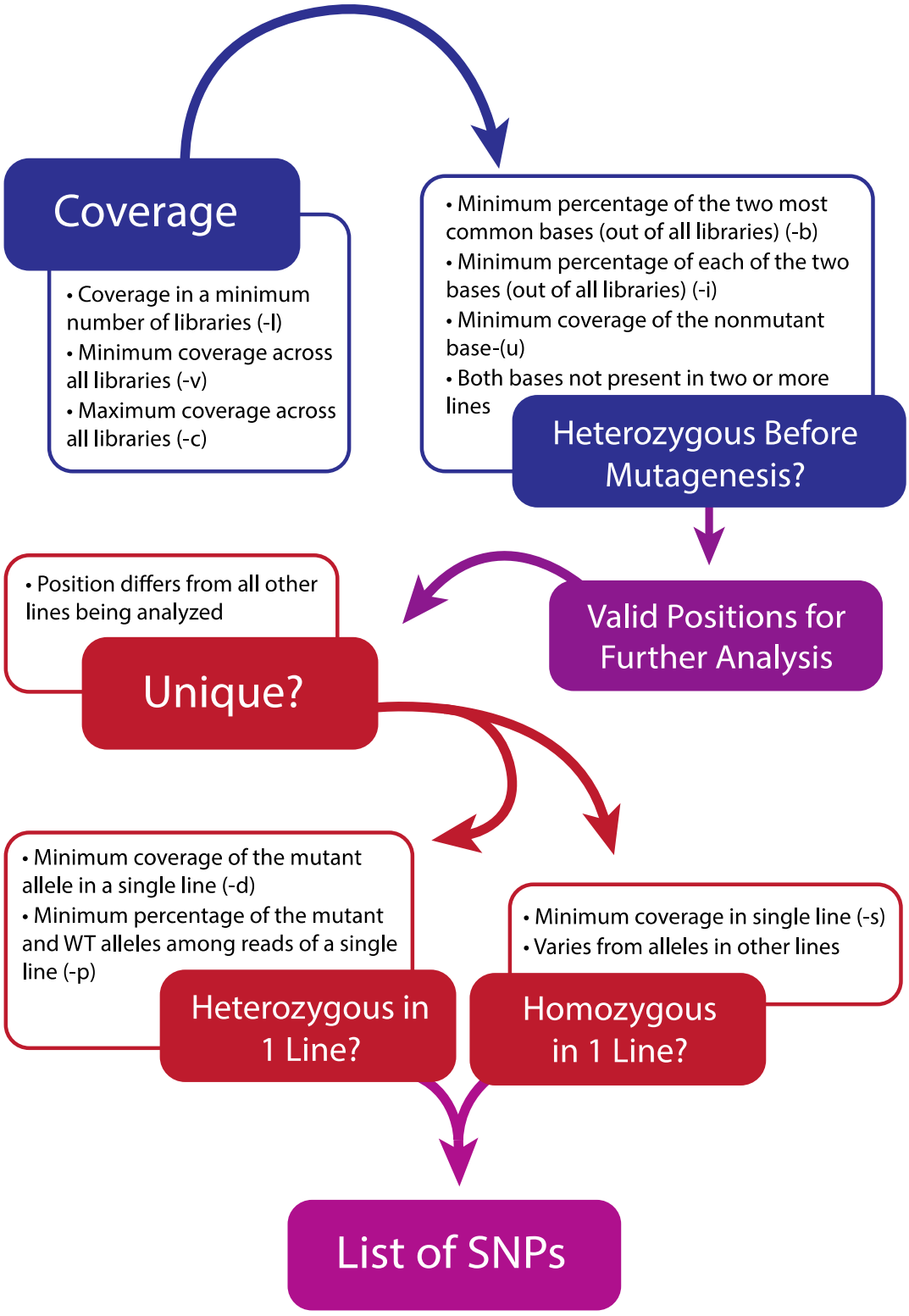

Figure 6. MAPS criteria. The program uses several criteria to determine whether a position may be an EMS-induced mutation.

Parameter	Explanation
-v	MAPS 1; the minimum total coverage be considered a valid position (across all libraries)
-i	MAPS 1; the minimum percentage of each of the two most common bases (across all libraries)
-p	MAPS 2; the minimum percentage of the mutant and WT alleles in a single library for the position to be called as heterozygous (for that library).
-s	MAPS 2; the minimum coverage for a position to be called as homozygous for a library.

		MAPS 2 Parameters			
		-s 2 -p 10	-s 2 -p 20	-s 4 -p 10	-s 4 -p 20
MAPS 1 Parameters	-i 10 -v 10	A1	A2	A3	A4
	-i 10 -v 20	B1	B2 WGS	B3 WES	B4
	-i 20 -v 10	C1	C2	C3	C4
	-i 20 -v 20	D1	D2	D3	D4

Figure 7. MAPS parameter sets. Four MAPS parameters, -v, -i, -p, and -s were varied in 16 combinations to find the optimum stringency for detecting real, EMS induced SNPs. The interior boxes are the labels for each parameter combination. Parameter sets selected for mutation detection in WES and WGS are labelled in ellipses.

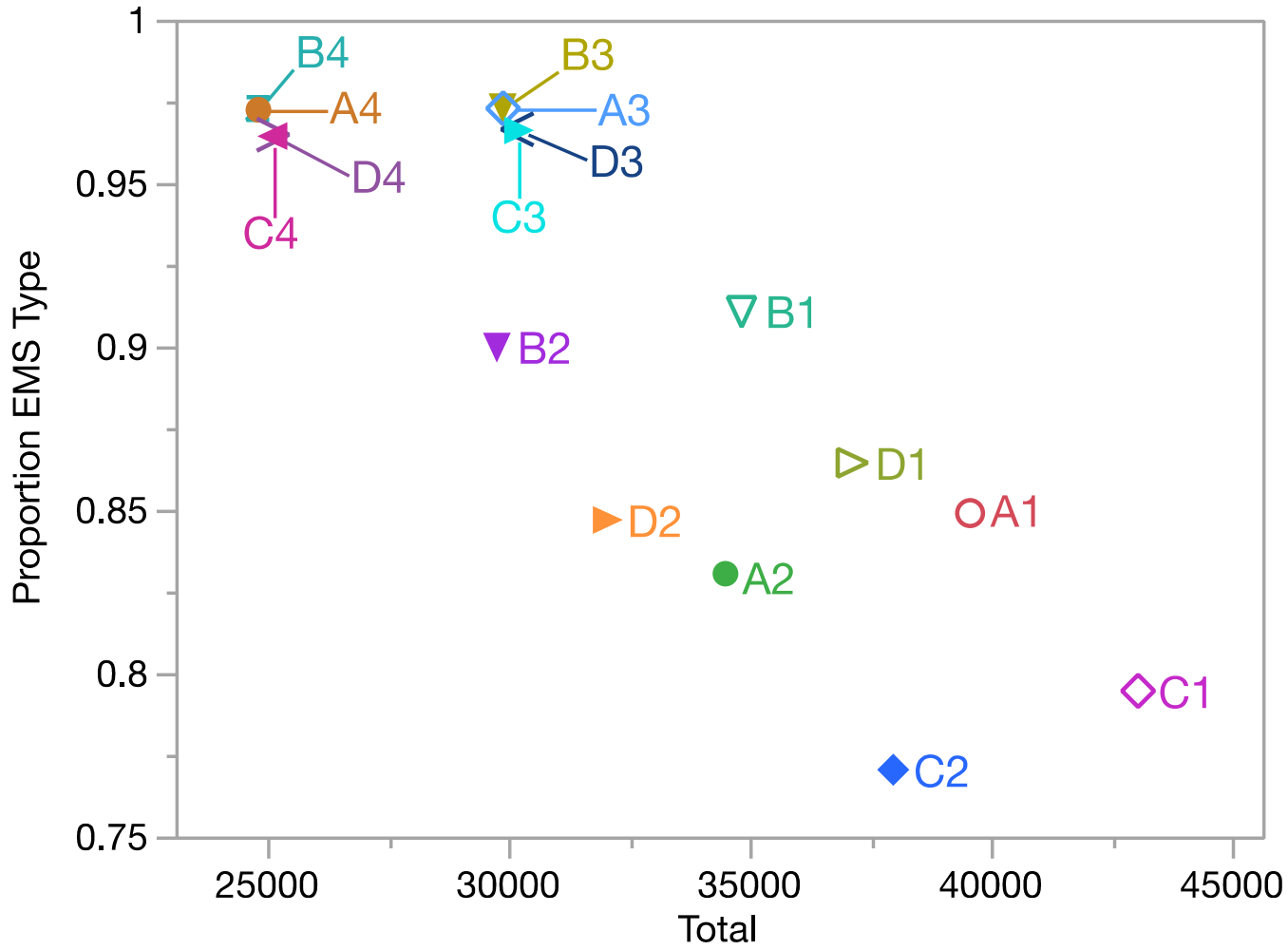

Figure 8. Proportion EMS type mutations versus total mutations for the 16 MAPS parameter sets in WES. We chose set B3 to produce our final set of mutations in the WES.

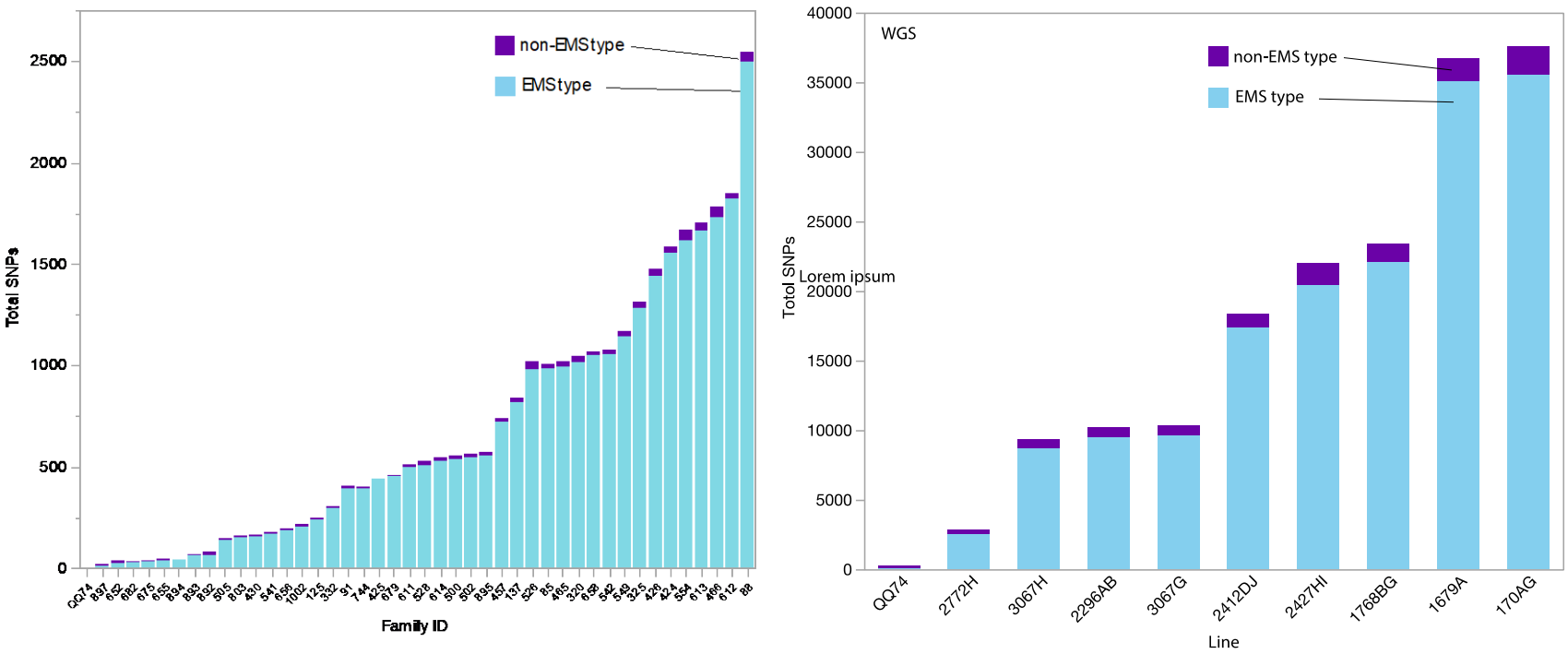

Figure 9. Types of SNPs in WES and WGS lines. In both WES and WGS lines, the number of SNPS varies greatly, but nearly all SNPs are canonical EMS type G/C > A/T transitions. The unmutagenized control, QQ74, has 3 SNPS, none of which are EMS type, in the WES, and 242 SNPs, 51 of which are EMS type, in the WGS.

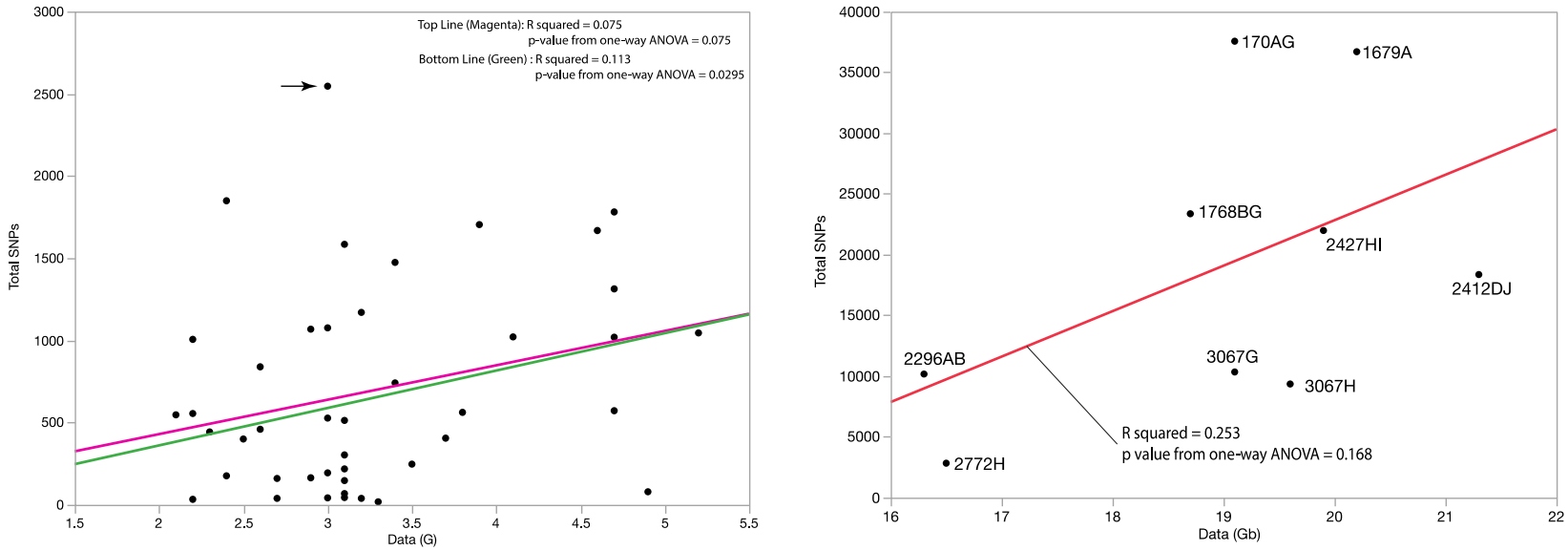

Figure 10. SNPs versus data fit lines. To test whether we captured all the SNPs in each family, we fitted a line to the total SNPs versus sequencing data for WES and WGS families. When family 88 (indicated by the arrow) is removed from the WES data, the line fits better. The p-value from a one-way ANOVA shifts from 0.075 to 0.030, a significant value. Although family 88 does appear to have an unusually high number of SNPs, there is no other to reason to remove the point from the analysis. The slope of the line shifts only slightly and remains shallow, so we are confident we did indeed capture a very high percentage of SNPs induced by EMS treatment. WGS data did not show a correlation between SNPs and sequencing data based on the R-squared values and a p-value from one-way ANOVA.

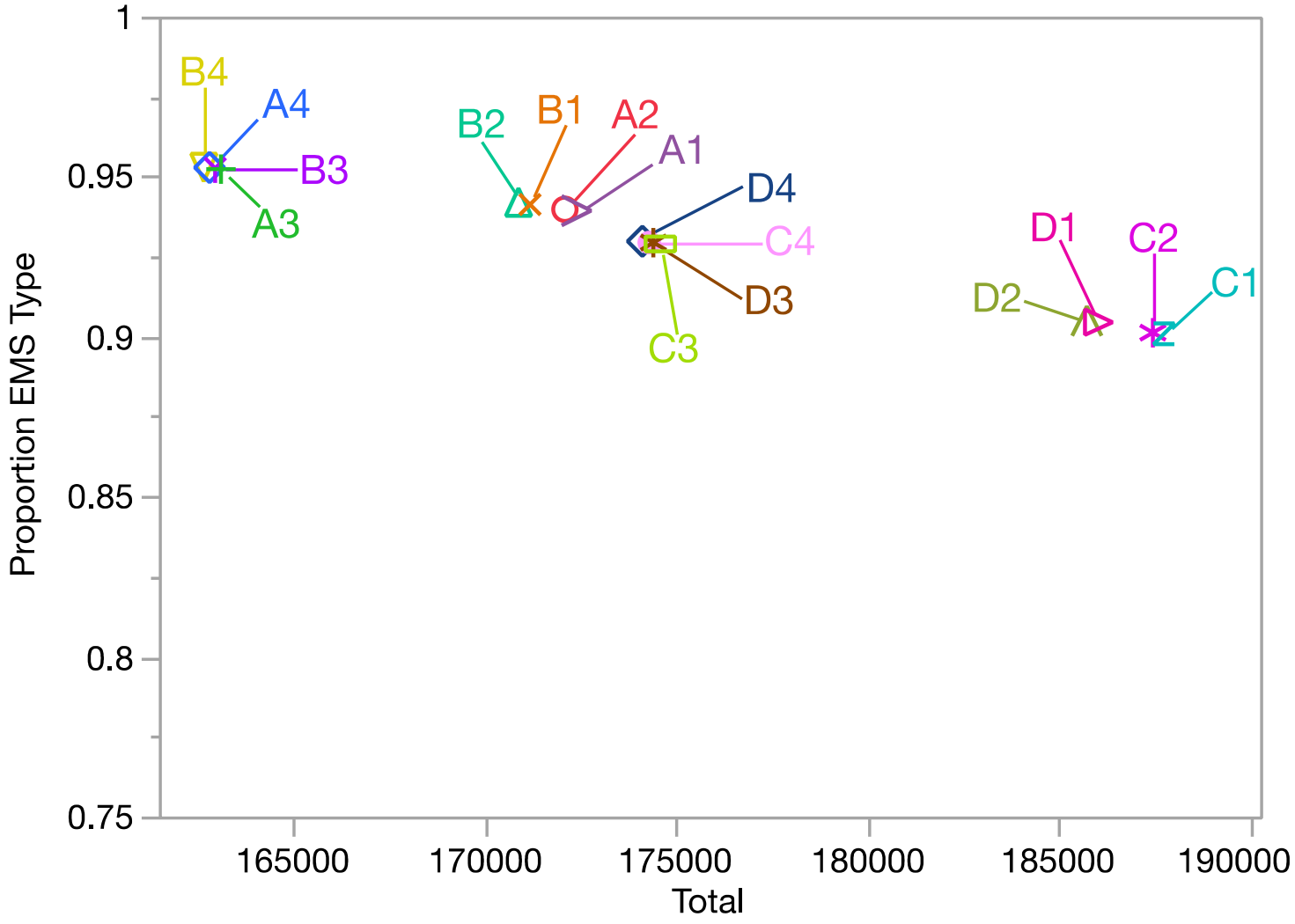

Figure 11. WGS parameter sets. We chose set B2 to produce the final set of mutations for the WGS.

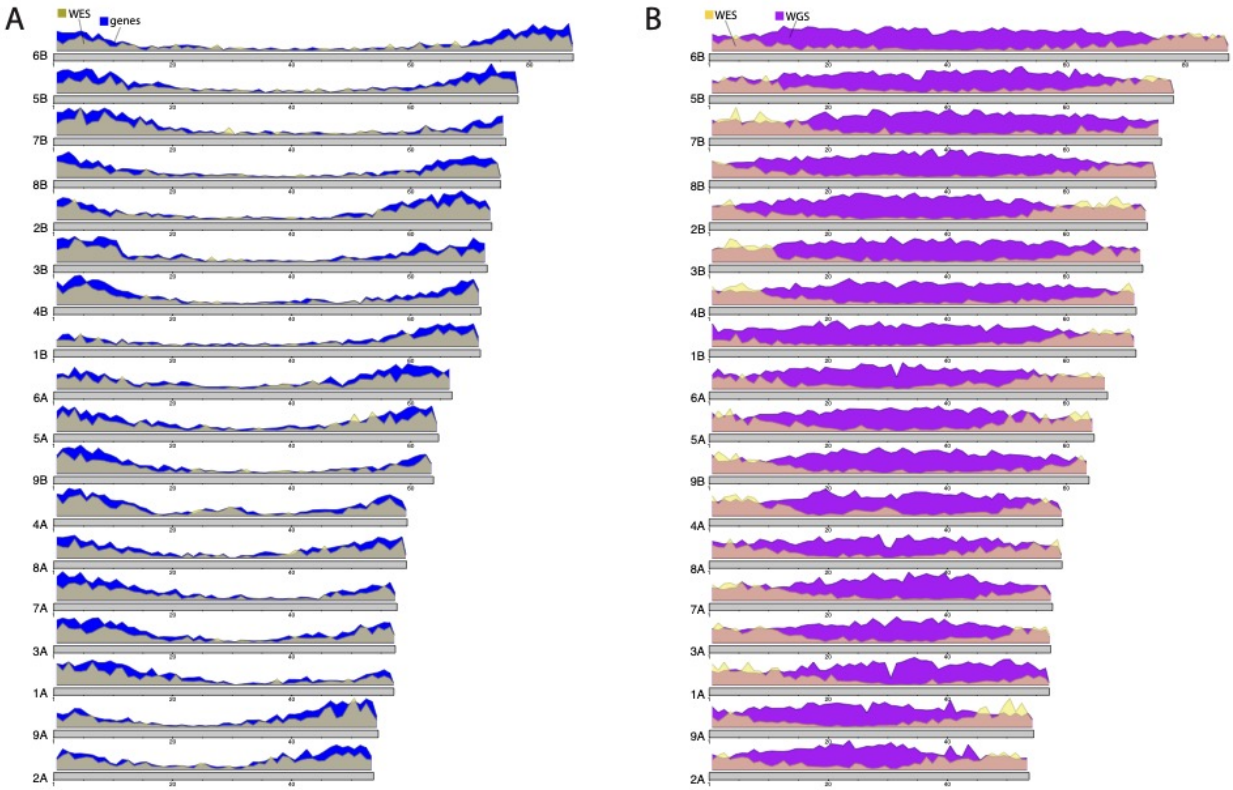

Figure 12. Mutation density along the 18 quinoa chromosomes. Panel A shows WES mutation density and gene density. WES mutations follow gene density. Panel B shows WES mutation density and WGS mutation density. WGS mutations are distributed more evenly across chromosomes than WES mutations.

Consequence	% of WES Mutations	% of WGS Mutations
upstream	32.249	17.139
5' UTR	4.022	0.654
start lost	0.064	0.007
frameshift	0.054	0.004
inframe insertion	0.000	0.001
synonymous	18.194	1.200
splice region	2.039	0.348
intron	10.314	7.432
missense	43.639	3.030
stop gained	2.646	0.196
stop lost	0.003	0.000
stop retained	0.047	0.004
3' UTR	6.838	0.965
downstream	43.502	17.287
intergenic	5.653	66.954

Figure 13. The sequencing approach affected the mutation distribution. WES revealed mutations mostly in or near genes, such as missense, synonymous, splice region, upstream, and downstream mutations. In contrast, the WGS approach revealed mainly intergenic mutations, which are located far from genes.

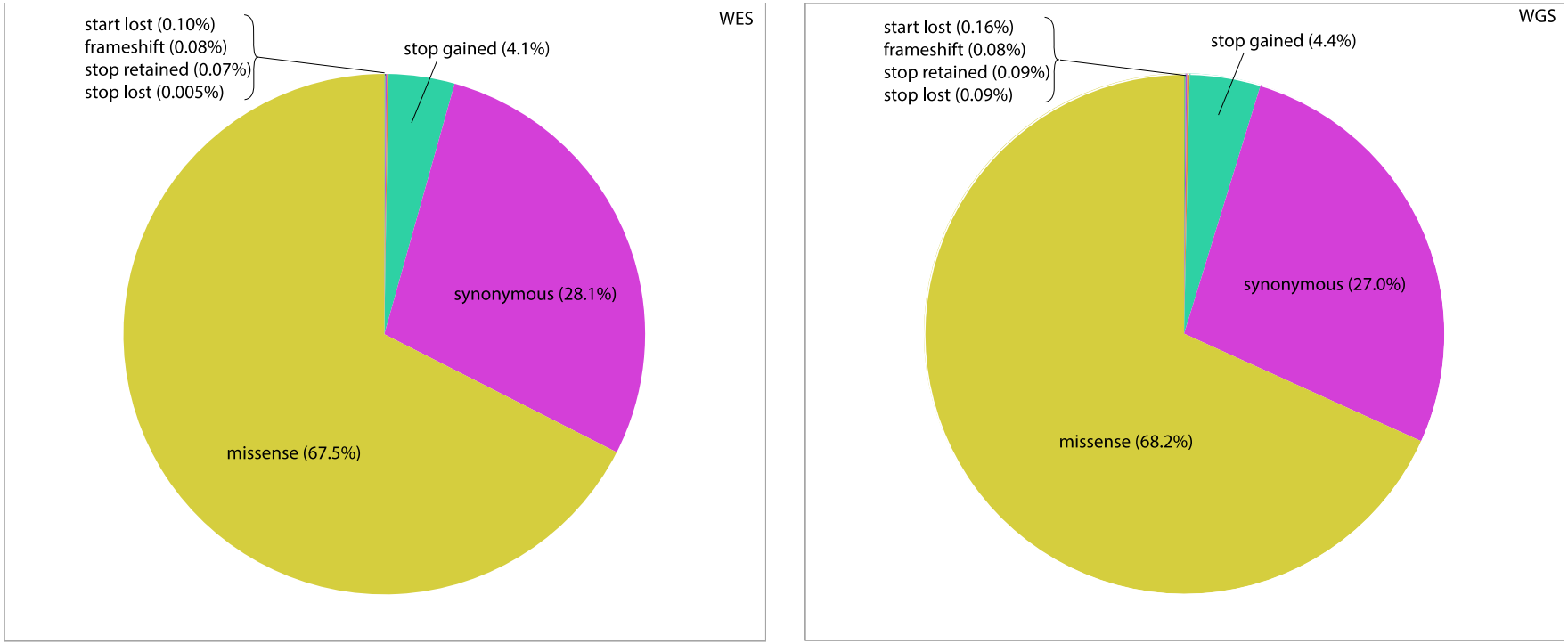

Figure 14. Mutations within CDSs. Although the proportion of total mutations that are in CDSs is very different between the WES and WGS, the distribution of mutations within CDSs is nearly identical between the two sequencing approaches.

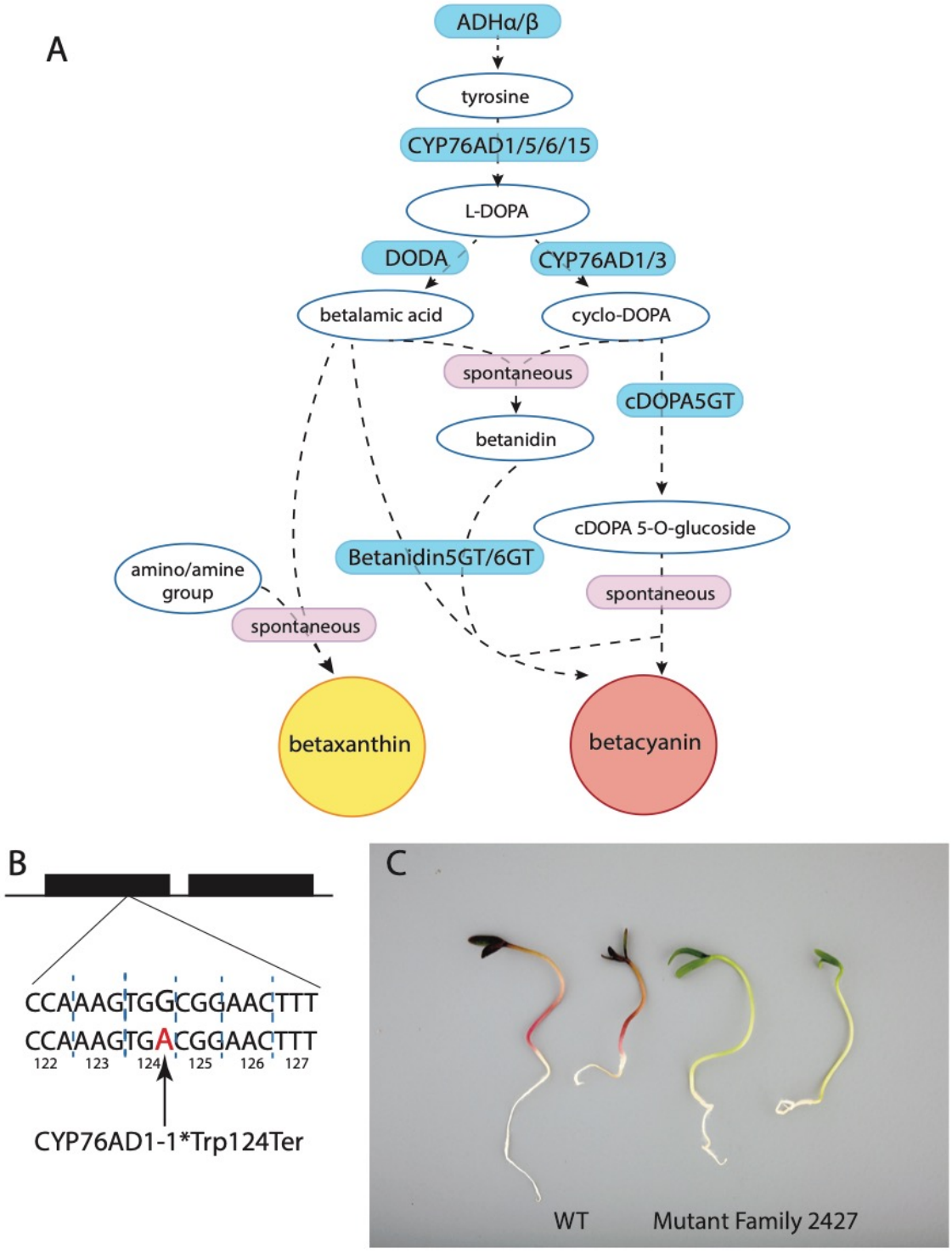

Figure 15. Betalain synthesis pathway and mutation. Panel A shows the betalain synthesis pathway. Panel B shows the EMS-induced, premature stop codon we found in CYP76AD1-1. Panel C shows the WT, red seedlings and the mutant, green seedlings with the mutation in CYP76AD1-1.

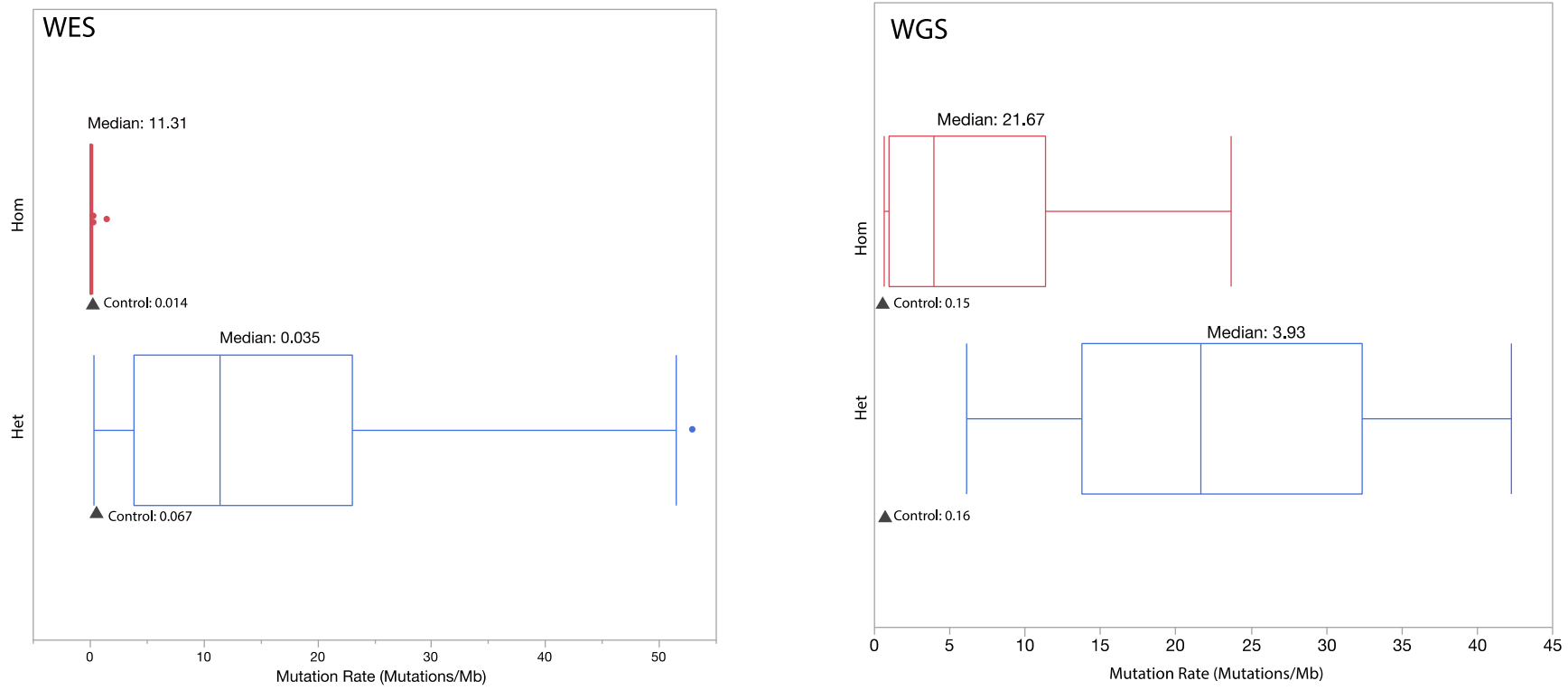

Figure 16. Mutation rates. We observed a median heterozygous mutation rate of 11.31 mutations/Mb and a median homozygous mutation rate of 0.035 mutations/Mb for the WES lines. The unmutagenized control had a homozygous mutation rate of 0.014 mutations/Mb and a heterozygous mutation rate of 0.067 mutations/Mb. For WGS, we observed a median heterozygous mutation rate of 21.67 mutations/Mb and median homozygous rate of 3.93 mutations/Mb. The unmutagenized control showed a heterozygous rate of 0.16 mutations/Mb and a homozygous rate of 0.15 mutations/Mb.

TABLES

Table 1.

EMS Concentration (% EMS)	Percent Germination
0 (Control)	~100
1	91
2	82
3	25
4	0

Table 2.

Capture Reaction	Family ID	Capture Reaction	Family ID
1	320	3	892
1	457	3	682
1	542	3	656
1	85	3	893
1	88	3	803
1	541	3	675
1	426	3	894
1	425	3	502
1	526	3	505
1	658	3	500
		3	614
2	424	3	895
2	325		
2	466	4	679
2	332	4	1002
2	465	4	137
2	430	4	744
2	549	4	125
2	554	4	612
2	528	4	652
2	91	4	613
2	345	4	611
		4	655
		4	897
		4	QQ74 Control

Table 3.

Percent of Bases Aligning to Target	Species	Paper
26	hexaploid wheat	(King et al., 2015)
40	tomato	(Terraciano et al., 2017)
43	quinoa	this study
49	durum wheat	(Henry et al., 2014)
60	tetraploid wheat	(Saintenac, Jiang, & Akhunov, 2011)
24-65	rice	(Henry et al., 2014)
67	pine	(Lu et al., 2016)
75	tomato	(Ruggieri et al., 2017)
97	poplar	(Zhou & Holliday, 2012)

Table 4.

M2 Family	M2 Parent(s)	M2 Phenotypes	M3 Phenotypes
3067	G	long leaves, lax panicle	long, narrow leaves
1768	BG	smooth leaves, branched	branched/bushy
1679	A	fasciated inflorescence	small, dark, round leaves; 3 cotyledons and 3 first true leaves; short plants
2427	H	green hypocotyls	green hypocotyls
2296	AB	branched	interveinal chlorosis on first true leaves
2412	DJ	narrow leaves; branched	narrow leaves, two-headed
3067	H	variegated	variegated
170	AG	branched	long cotyledons; round, ruffly leaves; small first true leaves
2772	H	irregular leaf shape; chlorotic spots	ruffly, irregularly shaped leaves
QQ74 (control)	-	-	-

Table 5.

Gene in Pathway	Method	Hits from CoGe BLAST*	Reference	Gene ID from Reference	Species from Reference	Notes from References	Quinoa Genome Annotation ID
ADHb	2	top 2	Lopez-Nieves et al, 2018	KY207366	beet		AUR62013276
			Lopez-Nieves et al, 2018	KY207366			AUR62011211
ADHa	2	top 2	Lopez-Nieves et al, 2018	KY207372	beet		AUR62011210
			Lopez-Nieves et al, 2018	KY207372			AUR62013277
CYP76AD	1,2	top 1	Imamura et al., 2018	-	quinoa	CYP76AD1-1	AUR62012346
	1,2	top1	Imamura et al., 2018	-	quinoa	CYP76AD1-2	AUR62022995
	1	-		-	-		AUR62010549
		-		-	-		AUR62013601
		-		-	-		AUR62022710
		-		-	-		AUR62027045
		-		-	-		AUR62027062
		-		-	-		AUR62034331
		-		-	-		AUR62030889
		-		-	-		AUR62030903
		-		-	-		AUR62018520
		-		-	-		AUR62004626
		-		-	-		AUR62004627
		-		-	-		AUR62027426

Table 5 (continued).

Gene in Pathway	Method	Hits from CoGe BLAST*	Reference	Gene ID from Reference	Species from Reference	Notes from References	Quinoa Genome Annotation ID
CYP76AD1	1,2	top 1	Imamura et al., 2018	-	quinoa	pseudogene	AUR62022993
CYP76AD1	1,2	top 1	Imamura et al., 2018	-	quinoa	inactive CYP76AD1	AUR62012348
DODA	1	-	-	-	-	-	AUR62012187
		-	-	-	-	-	AUR62012347
		-	-	-	-	-	AUR62000604
		-	-	-	-	-	AUR62000600
		-	-	-	-	-	AUR62000602
		-	-	-	-	-	AUR62022994
		-	-	-	-	-	AUR62006948
		-	-	-	-	-	AUR62006953
cDOPA5GT	2	top 4	Timoneda et al, 2018	MH836618	*Mirabilis jalapa*		AUR62022641
Betanidin 5GT	2	top 8	Timoneda et al, 2018 Sepulveda-Jimenez et al, 2005	Y18871	*Dorotheanthus bellidiformis*		AUR62004620 AUR62010259 AUR62010258 AUR62010257 AUR62013243 AUR62010260 AUR62013242 AUR62013239

Table 5 (continued).

Gene in Pathway	Method	Hits from CoGe BLAST*	Reference	Gene ID from Reference	Species from Reference	Notes from References	Quinoa Genome Annotation ID
Betanidin 6GT	2	top 4	Vogt, 2002	AF374004	*Dorotheanthus bellidiformis*		AUR62027236 AUR62003234 AUR62027238 AUR62003236

Methods of searching for genes are: 1) Search quinoa annotation for gene name; 2) CoGe BLAST nucleotide sequence from ID given in reference against quinoa annotation and pick top results

*hits from CoGe BLAST were sorted by HSP#

Table 6.

Family	Het Mutation Rate (SNPs/Mb)	Hom Mutation Rate (SNPs/Mb)	Family	Het Mutation Rate (SNPs/Mb)	Hom Mutation Rate (SNPs/Mb)
658	23	0.07	895	11.37	0.02
320	20.33	0.14	892	1.5	0.02
457	15.35	0.03	682	0.9	0.01
542	22.75	0.06	656	4.34	0.04
85	28.26	0.12	893	1.41	0.02
88	52.95	0.2	803	3.87	0
541	4.3	0	675	0.91	0
426	29.01	0.11	894	0.93	0
425	11.25	0.02	502	10.95	0.02
526	19.68	0.23	505	3.09	0.03
91	8.42	0.1	655	0.95	0.01
424	33.26	0.09	897	0.32	0.03
325	25.54	0.07	679	12.72	0.01
466	34.92	0.06	1002	5.31	0.01
332	6.41	0.02	137	22.27	0.2
465	20.3	0.03	744	11.05	0
430	3.46	0.05	125	4.96	0.02
549	24.08	0.06	612	51.53	1.5
554	31.9	0.12	652	0.6	0.11
528	10.92	0.04	613	36.09	0.12
500	14.75	0.38	611	11.79	0.01
614	16.21	0.27	QQ74	0.07	0.01

Table 7.

Family	Het Mutation Rate (SNPs/Mb)	Hom Mutation Rate (SNPs/Mb)
2427HI	21.67	7.79
170AG	25.67	23.66
3067G	11.98	3.93
2296AB	15.48	6.1
2412DJ	27.09	1.35
1768BG	42.25	2.86
2772H	6.09	0.63
1679A	37.68	14.98
3067H	18.08	0.66
QQ74	0.16	0.15

Table 8.

WES		WGS		WGS for 44 Families	
Exome Capture + Library Prep ($)	6,000	Library Prep + Sequencing ($)	2,800	Library Prep + Sequencing ($)	12,320
Sequencing ($)	1,400	Amount of Data (gigabases)	169	Amount of Data (gigabases)	744
Total Cost ($)	7,400	Mutations	170,663	Mutations	750,917
Amount of Data (gigabases)	120	Cost per Gb ($)	16.57	Cost per Gb ($)	16.57
Mutations	29,858	Cost per Mutation ($)	0.02	Cost per Mutation ($)	0.02
Cost per Gb ($)	61.67				
Cost per Mutation ($)	0.25				

APPENDIX

WES Read Processing

Trimming

All reads were 150 bp paired-end Illumina type. We used Trimmomatic to trim the raw reads using the following parameters: PE, -threads 4; leading:20; trailing:20; sliding window:5:20; minlen:75. We used the following script:

```
#!/bin/bash
#SBATCH --qos=pws -c 2 --mem=128gb -t 03:00:00
#for i in QQ74mut*_1.fq.gz; do prefix=${i/1.fq.gz}; echo "processing $prefix"; sbatch trim.sh
    $prefix; done

module purge
module load conda-pws
module load conda/trimmomatic

trimmomatic PE -threads 4 ${1}1.fq.gz ${1}2.fq.gz ../trimmed_reads/$1p_1.fq.gz
    ../trimmed_reads/$1up_1.fq.gz ../trimmed_reads/$1p_2.fq.gz ../trimmed_reads/$1up_2.fq.gz
    LEADING:20 TRAILING:20 SLIDINGWINDOW:5:20 MINLEN:75
```

Mapping

We mapped the trimmed reads to the quinoa version 2 reference genome using bwa mem with these parameters: -M; -t 8. We don't have the original script for this. I think we edited it to map the reads back to the version 1 reference genome, so we could get exome capture efficiency stats with CollectHSMetrics.

After mapping, we converted the sam files to bam format using samtools view (-Sb; --threads 8). Then, we sorted the bam files by coordinate using samtools sort and created index files with samtools index. We used the following script to run samtools view, sort, and index:

```
#!/bin/bash
#SBATCH --time=36:00:00   # walltime
```

```
#SBATCH --nodes=1
#SBATCH --ntasks=8   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=4096M   # memory per CPU core

module purge
module load conda-pws
module load conda/samtools_1.9

samtools view -Sb --threads 8 $1 > $1.bam
samtools sort $1.bam -o $1.sorted.bam
samtools index $1.sorted.bam
```

Removing PCR Duplicates

We removed duplicate reads arising from PCR using samtools rmdup using the script below. After processing the reads, we prepared them for the mutation detection pipeline.

```
   #!/bin/bash
#SBATCH --time=36:00:00   # walltime
#SBATCH --nodes=1
#SBATCH --ntasks=8   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=4096M   # memory per CPU core

module purge
module load conda-pws
module load conda/samtools_1.9

samtools rmdup $1 $1.unique.bam
```

WGS Read Processing

Trimming

All reads were 150 bp paired-end Illumina type. We used Trimmomatic to trim the raw reads using the following parameters: PE, -threads 4; leading:20; trailing:20; sliding window:5:20; minlen:7. The following script was used:

```
#!/bin/bash

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=4   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --mem-per-cpu=4096M   # memory per CPU core
```

```
#for i in ../Rawdata/S*/*_1.fq.gz; do prefix=${i/_1.fq.gz}; echo "processing $prefix"; done

module purge
module load conda-pws
module load trimmomatic

trimmomatic PE -threads 4 ${1}_1.fq.gz ${1}_2.fq.gz ./$1p_1.fq.gz ./$1up_1.fq.gz ./$1p_2.fq.gz
    ./$1up_2.fq.gz LEADING:20 TRAILING:20 SLIDINGWINDOW:5:20 MINLEN:7
```

Mapping

We used bwa mem to map reads. After mapping, we used samtools view to convert the sam files to bam files.

Remove PCR Duplicates

To remove PCR duplicates, we first used samtools sort -n to sort the sam files by read name. Then, we used samtools fixmate -m to add mate coordinates and size fields. Next, we used samtools sort to sort the reads by coordinate. These files were then sent through samtools markdup -r to mark and remove suspected PCR duplicate reads. Finally, we once again sorted the reads by coordinate. The following script is the general idea of all these steps, although piping them all together didn't work on all the read files. Splitting the steps up or having more UNIX skills than Brian is recommended.

```
#!/bin/bash

#SBATCH --time=72:00:00   # walltime
#SBATCH --ntasks=8   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --mem-per-cpu=64G   # memory per CPU core

module purge
module load bwa
module load samtools/1.9

REFERENCE=/panfs/pan.fsl.byu.edu/scr/grp/fslg_jarvis/EMS_exome/mapping_to_gDNA/reference_genome/q
    uinoa_pb_chicago-2-final_PBJELLY_pilon.fasta
FORWARD_TRIMMED_READS=$1_p_1.fq.gz
REVERSE_TRIMMED_READS=$1_p_2.fq.gz
```

```
alignments_directory=../mapping

bwa mem -M -t 16 $REFERENCE $FORWARD_TRIMMED_READS $REVERSE_TRIMMED_READS | samtools view -b --
    threads 8 -o $alignments_directory/$1.bam && samtools sort -n --threads 8
    $alignments_directory/$1.bam | samtools fixmate -m | samtools sort  | samtools markdup -s -r
    | samtools sort -o $alignments_directory/$1.markdup.sort.bam
```

Evaluating the Exome Capture

To evaluate the efficiency of the exome capture, we used Picard Tools CollectHSMetrics to compare the sequencing data to the exome capture design. CollectHSMetrics takes an alignment file (SAM or BAM) and a target interval file as input and outputs metrics that describe the performance of the targeted capture. For the alignment files, we remapped the sequencing data to the version 1 quinoa genome since the exome capture was designed using that version. This enabled an accurate comparison of the alignment files and the target regions. The target interval files must be in Picard interval_list format, so we used the bedtools BedToIntervalList tool to convert the bed file of capture targets provided by Roche into this format.

Remapping

Because the exome capture was designed using version 1 of the quinoa genome, we remapped the reads to the version 1 genome using bwa with the following parameters: -M, -t 8. The script is below.

```
#!/bin/bash
#SBATCH --time=72:00:00   # walltime
#SBATCH --nodes=1
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=32768M # memory per CPU core
#SBATCH -C rhel7

#for i in QQ74*_p_1.fq; do prefix=${i/_1.fq/}; echo "processing $prefix"; sbatch bwa2.sh $prefix;
    done

module load bwa/0.7.17
```

```
bwa mem -M -t 8
    ../mapping_to_gDNA/reference_genome_v1/C_Quinoa.V3.1.BioNanoHybridAssembly_Dovetail_Contamina
    nt_free.fa $1_1.fq $1_2.fq >../mapping_to_gDNA/mapped_reads_v1/$1.sam
```

Next, we converted the sam files to bam files with samtools view (-Sb), sorted them with samtools sort, and indexed the bam files with samtools index.

```
#!/bin/bash
#SBATCH --time=36:00:00   # walltime
#SBATCH --nodes=1
#SBATCH --ntasks=8   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=4096M   # memory per CPU core
#SBATCH -C rhel7

module load samtools/1.9

samtools view -Sb --threads 8 $1 > $1.bam
samtools sort $1.bam -o $1.sorted.bam
samtools index $1.sorted.bam
```

BedToInterval

We converted the bed file of capture targets provided by Roche to Picard interval_list format using the bedtools BedToIntervalList tool.

```
#!/bin/bash

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --mem-per-cpu=4096M   # memory per CPU core


java -jar /fslhome/bjcox21/fsl_groups/fslg_jarvis/compute/bin/picard.jar BedToIntervalList ¥

    I=/fslhome/bjcox21/fsl_groups/fslg_jarvis/compute/EMS_exome/roche_capture_design/capture_targ
    ets.bed ¥
      O=Roche_Exome_Capture.interval_list ¥

    SD=/fslhome/bjcox21/fsl_groups/fslg_jarvis/compute/EMS_exome/mapping_to_gDNA/reference_genome
    /quinoa_pb_chicago-2-final_PBJELLY_pilon.dict
```

CollectHSMetrics

We used Picard tools CollectHSMetrics to calculate exome capture efficiency statistics for each captured family. See the script below:

```
#!/bin/sh
#SBATCH --time=03:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --mem-per-cpu=16384M   # memory per CPU core
#SBATCH --qos=paulbryf
#SBATCH -C rhel7

java -jar /fslhome/dj58/fsl_groups/fslg_jarvis/compute/bin/picard.jar CollectHsMetrics I=$1
    O=$1.HsMetrics
    R=/fslhome/dj58/fsl_groups/fslg_jarvis/compute/EMS_exome/mapping_to_gDNA/reference_genome_v1/
    C_Quinoa.V3.1.BioNanoHybridAssembly_Dovetail_Contaminant_free.fa
    BAIT_INTERVALS=/fslhome/dj58/fsl_groups/fslg_jarvis/compute/EMS_exome/roche_capture_design/ca
    pture_targets.interval_list
    TARGET_INTERVALS=/fslhome/dj58/fsl_groups/fslg_jarvis/compute/EMS_exome/roche_capture_design/
    primary_targets.interval_list
```

Documentation and Resources

https://gatk.broadinstitute.org/hc/en-us/articles/360036856051-CollectHsMetrics-Picard-

http://broadinstitute.github.io/picard/picard-metric-definitions.html#HsMetrics

http://broadinstitute.github.io/picard/command-line-overview.html#BedToIntervalList

Mutation Detection

We used the Mutation and Polymorphism Survey (MAPS) developed by Henry et al (2014) to detect EMS induced mutations in mutant families sequenced with WES and mutant families sequenced with WGS. WES families were analyzed together, and WGS families were analyzed together but separate from the WES families. First, we combined all the bam files using a customized samtools mpileup provided by the authors of MAPS. Using the version from the samtools package without any modification should work as well. We then parsed this file and sent it through the two rounds of mutation detection, MAPS1 an MAPS2.

Mpileup

We used a customized version of samtools mpileup to create the mpileup files. This version is no longer available from the authors of MAPS. Using samtools mpileup without these modifications from the authors may work in place of this customized version. The following script is the version we downloaded from the authors:

```
#! /usr/bin/env python
import os, sys, math
from optparse import OptionParser

#Comai Lab, Ucdavis Genome Center
#Meric Lieberman, 2011
# This work is the property of UC Davis Genome Center - Comai Lab

# Use at your own risk.
# We cannot provide support.
# All information obtained/inferred with this script is without any
# implied warranty of fitness for any purpose or use whatsoever.
#-------------------------------------------------------------------------

#Part 1: run-mpileup.py

#This program is meant to be run on a directory of sorted.bam files. It will generate a mpileup
    file with columns for each library.

#INPUT:
#This program is run in a folder full of .sorted.bam files as input

#OUTPUT:
#This program outputs a mpileup file.

#NOTE:
#If the program samtools is not in /usr/bin, then the path to samtools must be specified using
    the command line parameters

#PARAMETERS, default value in []:
#1. REQUIRED:
#-r or reference_file, The alignment reference (fasta format) [required]
#-o or--output_file, The output mpileup.txt filename [required]
#2. OPTIONAL:
#-q or --mapqual, Minimum mapping quality for an alignment to be used [20]
#-Q or --basequal, Minimum base quality for a base to be considered [20]
#-d or --maxdepth, Max per-BAM depth coverage to avoid excessive memory usage [8000]
#-s or --samtools, File path to Samtools [/usr/bin/samtools]
```

```
usage = "¥n¥n%prog -r reference.fa -o output.txt [-q y] [-Q x] [-s path to Samtools}"
usage += "¥nRun in a directory only full of sorted.bam files, will generate a mplieup from all
    bams."
parser = OptionParser(usage=usage)
parser.add_option("-r", "--reference_file", dest="ref", help="Alignment reference file.")
parser.add_option("-o", "--output_file", dest="dest", help="Output file name.")
parser.add_option("-q", "--mapqual", dest="mapqual", default="20", help="(OPTIONAL, default = 20)
    Minimum mapping quality for an alignment to be used")
parser.add_option("-Q", "--basequal", dest="basequal", default="20", help="(OPTIONAL, default =
    20) Minimum base quality for a base to be considered")
parser.add_option("-d", "--maxdepth", dest="maxdepth", default="8000", help="(OPTIONAL, default =
    8000) Max per-BAM depth to avoid excessive memory usage")
parser.add_option("--samtools", "-s", dest="pathSAM",  type = "str",
    default='/share/apps/samtools-github-1.18/samtools', help="File path to Samtools")

(opt, args) = parser.parse_args()
mapqual = opt.mapqual
basequal = opt.basequal
maxdepth = opt.maxdepth

try:
   file = opt.ref
   o = open(opt.dest, 'w')
except:
   parser.error("Please check your command line paramters with -h or --help")

li = os.listdir(os.getcwd())
ind = filter(lambda x: ".sorted.bam" in x, li)
ind.sort()
a = map(lambda x: ["Cov-"+'-'.join(x.split('_')[:-1]).replace('lib','').replace('Lib',''),'Call-
    '+'-'.join(x.split('_')[:-1]).replace('lib','').replace('Lib',''),'Qual-'+'-
    '.join(x.split('_')[:-1]).replace('lib','').replace('Lib','')], ind)
b = [item for sublist in a for item in sublist]
header = '¥t'.join(['Chrom', 'Pos', 'Ref']+b)+'¥n'

runline = ' '.join(ind)
o.write(header)
o.close()

line = opt.pathSAM + " mpileup -d "+ maxdepth +" -Q "+basequal+" -q "+mapqual+" -f "+file+"
    "+runline+" >> "+opt.dest
print line
os.system(line)
```

We ran the mpileup program using the following script:

```
#!/bin/bash
#SBATCH --time=72:00:00   # walltime
```

```
#SBATCH --nodes=1
#SBATCH --ntasks=2   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=32768M   # memory per CPU core

module purge
module load conda-pws
module load conda/samtools_1.9

python2 ./run-mpileup.py --samtools
    /fslgroup/fslg_pws_module/compute/software/.conda/envs/samtools_1.9/bin/samtools --
    reference_file
    /fslhome/dj58/fsl_groups/fslg_jarvis/compute/EMS_exome/mapping_to_gDNA/reference_genome/quino
    a_pb_chicago-2-final_PBJELLY_pilon.fasta --output_file EMS_mpileup_gDNA.txt --mapqual 21 --
    basequal 21 --maxdepth 4000
mpileup parser
We parsed the mpileup file using a program from Henry et al (2014). This program is also no
    longer available on their website, but the script is below.
#! /usr/bin/env python2.6

import os, sys, math, datetime, gc, time
import threading, multiprocessing
from optparse import OptionParser
import subprocess

#Comai Lab, Ucdavis Genome Center
#Meric Lieberman, 2012
# This work is the property of UC Davis Genome Center - Comai Lab

# Use at your own risk.
# We cannot provide support.
# All information obtained/inferred with this script is without any
# implied warranty of fitness for any purpose or use whatsoever.
#-------------------------------------------------------------------------------

#Part 2: mpileup-parser.py
#
#This program parses a mpileup file to a simplified format to be used with our MAPS
#mutation and genotyping package.
#
#INPUT:
#This program takes an mpileup.txt file as input
#
#OUTPUT:
#This program outputs a parsed mpileup.txt file.
#
#NOTE:
#This program reads the entire file into memory before parsing, so it is recommended not to be
    run on systems with limited memory. It typically requires 1.5 times the size of the mpileup
    file in RAM to run. Many machines will not have this, and in this case it is recommended to
```

```
    break the mpileup file into smaller chunks to be processed separately. (i. e. by chromosome
    or scaffold)
#If being used with the MAPS package: the first step in MAPS is also threaded so it may be best
    to leave the chunks separate and process them individually in MAPS as well. Results can be
    combined at the end without compromising the results.
#This program is threaded, so it can be used with the -t flag to specify the number of cores to
    be used
#
#PARAMETERS:
#1. REQUIRED, default value in []:
#-f or --mpileup_file, The input mpileup file. [required]
#2. OPTIONAL:
#-t or --thread, Number of cores to be used while processing. [1]

start = time.time()

usage = "¥nUSAGE: %prog [-t #threads] -f mpileup_file.txt"
parser = OptionParser(usage=usage)
parser.add_option("-t", "--thread", dest="threads", default="1", help="How many threads to use
    during processing. (DEFAULT == 1")
parser.add_option("-f", "--mpileup_file", dest="file", help="Input mpileup file file.")
(opt, args) = parser.parse_args()

numThreads = int(opt.threads)

path = "/share/scripts/"
path = "./"

#split file into chunks
def splitter(l, n):
    i = 0
    chunk = l[:n]
    while chunk:
        yield chunk
        i += n
        chunk = l[i:i+n]

#text formatting
def form(flo):
   return str(flo).split('.')[0]+'.'+str(flo).split('.')[1][:2]
#accepted base
def test(s):
   valid = ['a','t','c','g','A','T','C','G','.',',','*','n','N']
   test = 0
   for x in valid:
      if x+'+' in s:
         test = 1
         break
   if test == 1:
```

```
      return 1
   else:
      return 0

#parse one pileup line
#based on parsing script from Joeseph Fass

class MyThread (multiprocessing.Process):

   def __init__ (self, filen, res, startpos, endpos):
      self.filen = filen
      self.res = res
      self.startpos = startpos
      self.endpos = endpos
      multiprocessing.Process.__init__ (self)

   def run (self):
      counter = 1
      ctgood = 0
      print self.startpos, self.endpos
      all = self.endpos - self.startpos
      ot = open("temp-parse-"+str(self.res)+".txt",'w')
      f = open(self.filen)
      f.readline()
      for l in f:
         if counter < self.startpos or counter > self.endpos:
            counter +=1
            continue

         if ctgood % 100008 == 0:
            print self.res, ctgood,'/',all
         ctgood +=1
         counter +=1
         #doOneLine(self.lines[k], ot)
         #y = self.lines[c]
         ######
         #def doOneLine(y, fh):
         k = l[:-1].split('¥t')
         refseq = k[0]
         position = k[1]
         refbase = k[2]
         div = list(splitter(k,3))
         result = div[0]
         #for a lib
         for sub in div[1:]:
            depth = sub[0]
            changes = sub[1]
            qualities = sub[2]
            try:
```

```
            mean_SQ = form(sum(map(lambda x: ord(x)-33,
list(qualities)))/float(len(qualities)))
         except ZeroDivisionError:
            mean_SQ = "0.0"
         if float(mean_SQ) < 20.00:
            result += ['.','.','.','.']
            continue
         else:
            inserts = {}
            quals = {'a':0,'A':0,'c':0,'C':0,'g':0,'G':0,'t':0,'T':0,'.':0,',':0,'*':0}
            valid = ['a','A','c','C','g','G','t','T','.',',','*']

            #depth = t[3]
            #changes = t[4]
            #qualities = t[5][:]
            #mappingqual = t[6][:-1]
            count = 0
            index = 0
            x = 0
            temp = sub[1]
            temp = temp.replace('^+','')
            while test(temp) == 1:
               pin = temp.index('+')
               numb = ""
               i = 0
               while 1:
                  if temp[pin+1+i].isdigit():
                     numb+= temp[pin+1+i]
                     i+=1
                  else:
                     break

               #take = int(temp[pin+1])
               take = int(numb)
               #print take
               total = temp[pin-1:pin+2+take]
               cleantotal = '.'+str.upper(total[1:])
               try:
                  inserts[cleantotal] +=1
               except:
                  inserts[cleantotal] = 1

               temp = temp.replace(total, '')
               sub[1] = sub[1].replace(total, '')
            if '+' in sub[1]:
               pin = sub[1].index('+')
               if sub[1][pin-1] != '^':
                  print "error, +/- found"
                  print sub[1]
```

```
while 1:
   if x >= len(sub[1]):
      break
   elif sub[1][x] in valid:
      quals[sub[1][x]] +=1
      index+=1
   elif sub[1][x] == "^":
      x+=1
   elif sub[1][x] == '+' or sub[1][x] == '-':
      temp = ""
      i = 0
      while 1:
         if sub[1][x+1+i].isdigit():
            temp+= sub[1][x+1+i]
            i+=1
         else:
            break
      x+= int(temp)+len(temp)

   x+=1

total_HQ = sum(quals.values())
Aa_HQ = quals['A']+quals['a']
Tt_HQ = quals['T']+quals['t']
Cc_HQ = quals['C']+quals['c']
Gg_HQ = quals['G']+quals['g']
match_HQ = quals['.']+quals[',']
dels = quals['*']


try:
   if len(inserts) >0:
      #print mean_SQ, qualities
      inlist =  map(lambda x: [x, inserts[x]], inserts.keys())
      inlist.sort(lambda x, y: cmp(y[1], x[1]))
      inname = inlist[0][0]
      incount = inlist[0][1]
      inper = incount/float(incount + total_HQ)*100
      delper = dels/float(incount + total_HQ)*100
      oin = str(sum(map(lambda x: x[1], inlist)))
      total_HQ += int(oin)
      scan = [[inname,inper]]
   else:
      inname = '.'
      inper = '.'
      oin = '.'
      delper = dels/float(total_HQ)*100
      scan = []
```

```
                aPer = Aa_HQ/float(total_HQ)*100
                tPer = Tt_HQ/float(total_HQ)*100
                cPer = Cc_HQ/float(total_HQ)*100
                gPer = Gg_HQ/float(total_HQ)*100
                matchPer = match_HQ/float(total_HQ)*100
                scan += [['A',aPer],['T',tPer],['C',cPer],['G',gPer],['*',delper]]
                for w in scan:
                   if w[0] == refbase:
                      w[1] += matchPer
                scan.sort(lambda x, y: cmp(float(y[1]),float(x[1])))
                scan2 = []
                for w in scan:
                   if w[1] == 0.0:
                      scan2.append('.')
                   else:
                      scan2.append(w[0]+'_'+form(w[1]))

                result+= [scan2[0], scan2[1], scan2[2], str(total_HQ)]
             except ZeroDivisionError:
                result+=['.','.','.','.','.']

          ot.write('¥t'.join(result)+'¥n')
          #######
       ot.close()
       f.close()


# Uses wc to get te number of lines in the file
def file_len(fname):
    p = subprocess.Popen(['wc', '-l', fname], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    result, err = p.communicate()
    if p.returncode != 0:
        raise IOError(err)
    return int(result.strip().split()[0])







t1 =  datetime.datetime.now()
#print t1
try:
   filename = opt.file
   f = open(filename)
   o = open("parsed_"+filename.split('/')[-1],'w')
except:
```

```
   parser.error("Please check your command line paramters with -h or --help")
flen = file_len(filename)
cutnum = (flen-1)/numThreads+1
cutset = []
for i in range(numThreads):
   cutset.append([i*cutnum+1, min((i+1)*cutnum, flen)])

t = f.readline()
if t == '':
   sys.exit("Empty pileup file")
f.seek(0)
header = f.readline()
header =header[:-1].split('¥t')
h2 = list(splitter(header[3:],3))
newhead = header[:3]
libs = []
for lab in h2:
   lname = ('-'.join(lab[0].split('-')[1:]))
   libs.append(lname)
   newhead += ["SNP1-"+lname,"SNP2-"+lname,"SNP3-"+lname,"Cov-"+lname]

o.write('¥t'.join(newhead)+'¥n')
o.close()
f.close()
#
#a = []
#for l in f:
#   a.append(l)
#
#
#gc.collect()
counter = 0
threads = {}
cat = "cat"
print cutset
for x in cutset:
   counter+=1
   print counter
   threads[counter] = MyThread(filename, counter, x[0], x[1])
   cat += " temp-parse-"+str(counter)+".txt"
   threads[counter].start()
   #del(threads[counter].lines)

#del(a)
#gc.collect()

#f.close()
cat += " >> parsed_"+filename.split('/')[-1]
```

```
all = []

for x in range(1, counter+1):
   threads[x].join()
   print x, "joined"
os.system(cat)
os.system("rm -f temp-parse-*")

fin = open("parsed_"+filename.split('/')[-1])
fin.readline()
bases = 0

for l in fin:
   bases+=1

fin.close()

print "parsed_"+filename.split('/')[-1]
print "Bases:¥t"+str(bases)
now = time.time()-start
print int(now/60), int(now%60)
```

We ran this program using the following script:

```
#!/bin/bash
#SBATCH --time=72:00:00   # walltime
#SBATCH --nodes=1
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=716800M   # memory per CPU core
#SBATCH --qos=paulbryf
#SBATCH -C rhel7

python2 ./mpileup-parser-v2.py --thread 2 --mpileup_file EMS_mpileup_gDNA.txt
```

MAPS1

We sent the parsed-mpileup file through the MAPS1, the initial mutation detection criteria. MAPS1 reads the whole parsed mpileup file into memory before running, so you need to give it at least 1.5 times as much memory as the size of the file to run. The first time we ran MAPS1 we used the following script:

```
#!/bin/bash
#SBATCH --time=72:00:00   # walltime
#SBATCH --nodes=1
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=358400M   # memory per CPU core

python2 ./maps-part1-v2.py -f parsed_EMS_mpileup_gDNA.txt -o maps1_EMS_mpileup_gDNA.txt -t 10 -c
    10000 -b 80 -i 10 -m m -H
```

We altered -i and -v to optimize the stringency of mutation calling for our data, so we altered the script to change the stringency of -v and -i. The script above uses the default -v of 10, and a specified-i of 10. When using the default values of the parameters they do not need to be listed in the script.

MAPS2

The next step is to run MAPS2, the final criteria for mutation detection. We used the following script to run MAPS 2:

```
#!/bin/bash
#SBATCH --time=72:00:00   # walltime
#SBATCH --nodes=1
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=358400M   # memory per CPU core
#SBATCH -C rhel7

python2 ./maps-part2-v2.py -f maps1_EMS_mpileup_gDNA.txt -o maps2_EMS_mpileup_gDNA.txt -p 10 -m m
```

We changed -p and -s in MAPS 2 to optimize the stringency of mutation calling. The script above uses a -p of 10 and a -s of 2. The -p parameter is specified because it is different than the default value, but -s is not listed because the default value is 2.

Interpreting the MAPS 2 Output File

MAPS 2 outputs a large .txt file with mutations from all the analysed samples. Descriptions for the columns we used in our analysis are as follows:

- Chrom/Scaffold – The scaffold the mutation is located on
- Pos – The position within the scaffold
- Ref – The allele in the reference genome
- WT – The allele in the sequenced samples
- MA – The allele in the only the mutant line
- Lib – The ID of the mutant line
- Ho/He – The type of mutation, either homozygous or heterozygous
- Type – The type of mutation listed as [WT][MA]

Deletions, either in the WT or MA, are notated with a *. Insertions are notated as .+[number of bases inserted compared to the reference][which bases were inserted]. For example, .+4GTGT in the mutant allele column (MA) indicates that the mutant allele is an insertion of 4 nucleotides, GTGT.

Pulling Out Specific Mutations

MAPS 2 outputs a file with the mutations it calls from all the samples, or families, that went into the mpileup file. We separated the mutations by family to enable analysis of mutations from individual families. We used the following script to separate the combined output file into separate files for each family:

```
#!/bin/bash

#SBATCH --time=01:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
```

```
#SBATCH --nodes=1   # number of nodes
#SBATCH --mem-per-cpu=4G  # memory per CPU core

LINES=`awk '{print $7}' "$1" | grep -v "Lib" |sort | uniq`

grep -E "$LINES" "$1" | while read line;do fileName=`echo "$line" | grep -oE "$LINES" | head -1`;
    echo "$line" >> "$fileName".txt; done
```

We evaluated the performance of each stringency level by counting the number of canonical EMS mutations, G/C > A/T transitions, at each level of stringency we tested.

```
#!/bin/sh

#for i in maps2_*_EMS_mpileup_gDNA.txt;do echo "processing $i"; sh count_EMS_type.sh $i; done
    .sh $i; done

MAPS_parameter_set=`echo "$1" | sed 's/_EMS_mpileup_gDNA.txt//'`
count_total=`grep -v "Chrom/Scaffold" $1 | wc -l`
count_EMS_type=`awk '{if ($11=="GA" || $11=="CT") print $0}' $1 | wc -l`

echo "$MAPS_parameter_set" "$count_EMS_type" "$count_total" >> count_EMS_total.txt
We used the following script to pull out mutations from each family into separate files:
#!/bin/sh

#for i in maps2_11_EMS_reseq_*p.txt;do echo "processing $i"; sh count_EMS_type_each_line.sh $i;
    done

line=`echo "$1" | sed 's/maps2_11_EMS_reseq//'`
count_total=`grep -v "Chrom/Scaffold" $1 | wc -l`
count_EMS_type=`awk '{if ($11=="GA" || $11=="CT") print $0}' $1 | wc -l`

echo "$line" "$count_EMS_type" "$count_total" >> count_EMS_total_each_line_reseq.txt
```

We used the following script to count the number of homozygous and heterozygous mutations in each family in the WGS. The same method should work for any number of families using WGS or WES.

```
#!/bin/sh

#for i in maps_output_file;do stuff; done

HET=`grep "het" $1 | wc -l`
HOM=`grep "hom" $1 | wc -l`
MUTANT_LINE=`echo "$1"`
```

```
echo -e "$MUTANT_LINE""¥t""$HET""¥t""$HOM">> het_hom_counts.txt
```

Documentation and Resources

http://comailab.genomecenter.ucdavis.edu/index.php/Mpileup

http://comailab.genomecenter.ucdavis.edu/index.php/MAPS

Calculating the Mutation Rate

To calculate the mutation rate, we used a spreadsheet from the Comai lab. It calculates the homozygous mutation rate and contains coefficients that are used to adjust the heterozygous mutation rate for positions with low coverage. We had to enter the number of positions that were assayed by MAPS 1 as well as their coverage to calculate the mutation rates. The information about the assayed positions is in the MAPS 1 output file assay-[input file name]. We downloaded this text file and imported the data into the spreadsheet. A blank spreadsheet is available for new data sets.

Formatting WES Coverage Files for KaryoploteR

To plot the coverage from the WES, we used bedtools coverage to calculate coverage across the genome. First, we made a windows file containing the coordinates of 100 kb windows spanning the entire genome using the following command:

```
grep -v "start" quinoa_pb_chicago-2-final_PBJELLY_pilon.fasta.chromosomes.coords | bedtools
    makewindows -b - -w 100000 >
    ../maps/remove_dups_renamed/merged_bams/quinoa_genome_100kb_no_overlap.txt
```

Next, we calculated the coverage within these windows from the bam files. To do this, we merged all 44 bam files, split them by chromosome, calculated the coverage, and then concatenated the coverage files back together.

We used this script to merge the bam files:

```
#!/bin/bash

#SBATCH --time=36:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --mem-per-cpu=16G   # memory per CPU core

module load samtools

samtools merge merged_4.bam ../*.samunique.sorted.bam
```

We used this script to split the merged bam file by chromosome:

```
#!/bin/bash

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=4   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=32768M   # memory per CPU core

# while read chromosome ;do echo "$chromosome"; sbatch samtools_view.sh $chromosome;done
    </fslhome/bjcox21/fsl_groups/fslg_jarvis/compute/EMS_project/EMS_exome/mapping_to_gDNA/reference_
    genome/quinoa_pb_chicago-2-final_PBJELLY_pilon.fasta.chromosomes.names

module load samtools

samtools view --threads 3 -b merged_4.bam $1 > "$1".bam
```

We also split the windows file by chromosome with this script:

```
#!/bin/bash

#SBATCH --time=3:00:00   # walltime
#SBATCH --ntasks=2   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=32768M   # memory per CPU core

while read chromosome;do grep "$chromosome" quinoa_chrom_windows_file.txt >
    "$chromosome"_windows_file.txt; done
    </fslhome/bjcox21/fsl_groups/fslg_jarvis/compute/EMS_project/EMS_exome/mapping_to_gDNA/reference_
    genome/quinoa_pb_chicago-2-final_PBJELLY_pilon.fasta.chromosomes.names
```

Then, we were able to use bedtools coverage to determine the coverage across the 18 bam files using this script:

```
#!/bin/bash
#SBATCH --time=36:00:00   # walltime
#SBATCH --nodes=1
```

```
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --mem-per-cpu=128G   # memory per CPU core
#SBATCH -C rhel7

#while read chromosome;do echo "$chromosome";sbatch
    bedtools_coverage_chromosome_windows_no_overlap.sh $chromosome; done
    </fslhome/bjcox21/fsl_groups/fslg_jarvis/compute/EMS_project/EMS_exome/mapping_to_gDNA/reference_
    genome/quinoa_pb_chicago-2-final_PBJELLY_pilon.fasta.chromosomes.names

module load bedtools

bedtools coverage -mean -a "$1"_windows_file.txt -b ../"$1".bam >
    "$1"_coverage_windows_100kb_no_overlap.txt
Next, we combined all the coverage files into one file with this command:
for plotting_file in Contig*plotting.txt;do echo $plotting_file; cat $plotting_file >>
    all_chromosomes_100kb_no_overlap_plotting.txt; done
Then, we reformatted the file, so we could plot in in KaryoploteR. The header has to have three, tab
    separated columns, “chr”, “start”, “stop”, and “y.” We used a command similar to this
    one:
awk 'BEGIN{print "chr" "¥t" "start" "¥t" "stop" "¥t" "y"};{if ($4>300)print $1 "¥t" $2 "¥t" $3 "¥t"
    "300"; else print}' all_chromosomes_100kb_no_overlap_plotting.txt >
    all_chromosomes_100kb_no_overlap_plotting_maxcov300.txt
```

Finally, we plotted the file in KaryoploteR to visualize the coverage across the 18 quinoa chromosomes. We ran these commands sequentially to plot the WES design and WES coverage:

```
library(karyoploteR)
library(rtracklayer)
kp_quinoa <- plotKaryotype(genome = quinoa_genome)
kpAddBaseNumbers(kp_quinoa)
kpPlotDensity(kp_quinoa,data=exome_capture,col="yellow")
kpLines(kp_quinoa, chr=WES_coverage_all_chromosomes_maxcov300 $chr, x=
    WES_coverage_all_chromosomes_maxcov300 $start, y= WES_coverage_all_chromosomes_maxcov300 $y, ymax
    = 300, col = "purple")
WES_coverage_all_chromosomes_maxcov300 <- read.table("Box/Jarvis_lab/EMS Project/Exome
    Capture/plotting_files/all_chromosomes_100kb_no_overlap_plotting_maxcov300.txt", header = TRUE)
```

Predicting Mutation Consequences

We used Ensembl Variant Effect Predictor (VEP) to predict the consequences of the mutations found in the WES and WGS. Go to https://uswest.ensembl.org/info/docs/tools/vep/index.html to learn more about VEP and download it. There are three VEP interfaces, the web interface, the command line tool,

and the REST API, and we used the command line tool to do our analysis. The version we downloaded is located in /lustre/scratch/grp/fslg_jarvis/bin/ensembl-vep.

Formatting the Genome Annotation for VEP

To use a user specified, custom annotation, VEP needs an annotation file and a reference file with its respective index files. For our annotation file, we used quinoa_pb_chicago-2-final_PBJELLY_pilon_renamed_sorted_allscaffolds.functional_blast.gff. As our reference file, we used quinoa_pb_chicago-2-final_PBJELLY_pilon.fasta.

The names of the genomic scaffolds in the reference file and annotation file must match. Scaffolds in our reference file were named as >Contig0_pilon, and scaffolds in the annotation file were labelled as Contig0_pilon. The > symbol in the reference scaffold names did not cause any errors from VEP.

VEP also requires annotation files to be sorted, zipped, and indexed. To accomplish this, we followed the instructions at *https://uswest.ensembl.org/info/docs/tools/vep/script/vep_custom.html*. Note that the htslib module must be loaded to access bgzip and tabix, which respectively zip and index the file.

Formatting the Mutation File for VEP

We also needed to reformat our mutation file to be compatible with VEP. We used the default VEP input format specified at https://uswest.ensembl.org/info/docs/tools/vep/vep_formats.html. This pretty straightforward, except for reformatting the indels. Be sure to follow the formatting that VEP needs for these. Running the following 3 commands sequentially will give you a VEP input file with the correct formatting. These are not the exact commands we used to generate the VEP input files, but they were tested on the WES data and produced a file with the same number of lines.

```
grep -v "Chrom/Scaffold" mutation_file.txt | grep -v "+" | grep -v "¥*" | awk '{print
    $1"¥t"$2"¥t"$2"¥t"$3"/"$6"¥t+"}' > output_file.txt



grep "+" mutation_file.txt | awk '{print $1"¥t"$2+1"¥t"$2"¥t""-/"$6"¥t""+"}' | sed 's/¥.+.//g' >>
    output_file.txt



grep "¥*" mutation_file.txt| awk '{print $1"¥t"$2"¥t"$2"¥t"$3"/-¥t+"}' >> output_file.txt
```

Note that MAPS reports only single base-pair deletions. So, if 3 consecutive bases were deleted, MAPS would report 3 separate single base-pair deletions, one at each of the 3 consecutive positions. VEP may not be accurately predicting the consequences of such multi-base pair deletions. While this is problematic, multi-base pair deletions are rare among the mutations we detected.

Running VEP

Once we had all the files in the correct format, we ran VEP on the WES mutations using the following command line options:

```
--custom
    /panfs/pan.fsl.byu.edu/scr/grp/fslg_jarvis/EMS_project/EMS_exome/mapping_to_gDNA/reference_ge
    nome/quinoa_pb_chicago-2-
    final_PBJELLY_pilon_renamed_sorted_allscaffolds.functional_blast_sorted_2.gff.gz,quinoa,gff -
    -fasta
    /panfs/pan.fsl.byu.edu/scr/grp/fslg_jarvis/EMS_project/EMS_exome/mapping_to_gDNA/reference_ge
    nome/quinoa_pb_chicago-2-final_PBJELLY_pilon.fasta --force_overwrite --input_file
    ../maps2_12_EMS_mpileup_gDNA_VEP_ref_MA_INDELS_2.txt --output_file VEP_WGS_INDELS --
    stats_file VEP_WGS_INDELS_stats
```

To run VEP on the WGS mutations, we substituted the WES mutation file for the WGS mutation file.

Reporting the Results from VEP

The descriptions of the consequences predicted by VEP can be found here: https://m.ensembl.org/info/genome/variation/prediction/predicted_data.html. When we reported

consequences, we grouped all the “splice acceptor variant”, “splice donor variant”, and all other consequences that that included the description, “splice region variant” into one group, which we called “splice region.”

Looking for Mutations in Betalain Pathway Genes

To search for mutations in genes involved in the betalain synthesis pathway, we created a .bed file of the genes in the pathway and used bedtools intersect to search for overlaps with genes in the quinoa genome annotation. The file of genes is called betalain_pathways_genes.bed, and the annotation we used was lifted_quinoa_pb_chicago-2-final_PBJELLY_pilon.gff3. This returned an output file called betalain_pathway_genes_MAPS2427H_bedtoolsintersect.bed.