Genomic Mutation Identification in Mice Using Illumina Sequencing and Linux-Based Computational Methods

John A. Williams,[1,2,3] George Powell,[1,4] Ann-Marie Mallon,[1] and Michelle M. Simon[1,5]

[1]MRC Harwell Institute, Mammalian Genetics Unit, Harwell Campus, Oxfordshire, United Kingdom
[2]Institute of Translational Medicine, University Hospitals Birmingham NHS Foundation Trust, Birmingham, United Kingdom
[3]Institute of Cancer and Genomic Sciences, University of Birmingham, Birmingham, United Kingdom
[4]Big Data Institute, Li Ka Shing Centre for Health Information and Discovery, Nuffield Department of Population Health, University of Oxford, Oxford, United Kingdom
[5]Corresponding author: *m.Simon@har.mrc.ac.uk*

Genetically modified mice are an essential tool for modeling disease-causing mechanisms and discovering gene function. SNP genotyping was traditionally used to associate candidate regions with traits in the mouse, but failed to reveal novel variants without further targeted sequencing. Using a robust set of computational protocols, we present a platform to enable scientists to detect variants arising from whole-genome and exome sequencing experiments. This article guides researchers on aligning reads to the mouse genome, quality-assurance strategies, mutation discovery, comparing mutations to previously discovered mouse SNPs, and the annotation of novel variants, in order to predict mutation consequences on the protein level. Challenges unique to the mouse are discussed, and two protocols use self-contained containers to maintain version control and allow users to adapt our approach to new techniques by upgrading container versions. Our protocols are suited for servers or office workstations and are usable by non-bioinformatics specialists. © 2019 by John Wiley & Sons, Inc.

Keywords: mouse genomics • mouse mutagenesis screens • mutation detection • single-nucleotide detection • variant annotation • whole genome sequencing

How to cite this article:
Williams, J.A., Powell, G., Mallon, A.-M., & Simon, M.M. (2019). Genomic mutation identification in mice using illumina sequencing and linux-based computational methods. *Current Protocols in Mouse Biology*, *9,* e64. doi: 10.1002/cpmo.64

INTRODUCTION

The first-ever completed human genome originally took around 12 years to sequence, and was completed in 2002; sequencing of the mouse genome was completed 1 year later. Both species were initially sequenced using Sanger sequencing, which was developed in the 1970s. Even though the technology was fast and efficient for its time, it was wholly unsuitable and not cost effective for whole-genome sequencing of multiple individuals. Nowadays, with the advent of next-generation sequencing (NGS), it is possible to complete whole-genome sequencing of an individual within 1 day in a very cost-effective and accurate manner. This revolutionary innovation has allowed the efficient detection

CURRENT PROTOCOLS
A Wiley Brand
Current Protocols in Mouse Biology e64, Volume 9
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpmo.64

of mutations in humans, mice, and a plethora of other species to aid the development of personal medicines, which are emerging as a reality in today's world.

One of the primary reasons the mouse is used as a model organism of human disease is because the mouse genome shares a greater proportion of similarity with the human genome than one might expect given the species' outward appearance; approximately 99% of functional genes in the human genome have an orthologous counterpart gene in mice (Mouse Genome Sequencing Consortium, 2002). The sequence of such a gene is likely to be evolutionarily conserved (to an extent, particularly in protein-coding regions) between humans and mice. Thus, it is possible to use the mouse as a model to infer the functional effects that specific genes might have on the human phenotype. Of interest is: how are disease phenotypes manifested, and what genes affect them?

Two major approaches are adopted in the effort toward understanding the genetic components of disease, and these are the genotype-driven (reverse genetics) and phenotype-driven (forward genetics) methodologies. With the former, the idea is to target a specific gene of interest and assess its effect on the resultant phenotype of an organism, in this case the mouse. With the latter, it is the phenotype of the organism that is initially observed, and the aim is to deduce the causative gene. Both approaches require genetic mutation/manipulation, either spontaneous in nature or by intervention.

N-Ethyl-*N*-nitrosourea (ENU) is a common chemical mutagen used to confer single point mutations in mouse spermatogonia. These mutations are supposedly distributed at random across the genome and occur at an approximately 1,000-fold higher rate than spontaneous mutations (Concepcion, Seburn, Wen, Frankel, & Hamilton, 2004). The fact that ENU preferentially causes point (single-nucleotide) mutations means that the effects of small variations—such as minimally altered coding sequences—can often be discovered. Such variations might confer altered or reduced function of a gene's product. This enables a broad range of alleles (multiple allelic series) and phenotypes to be

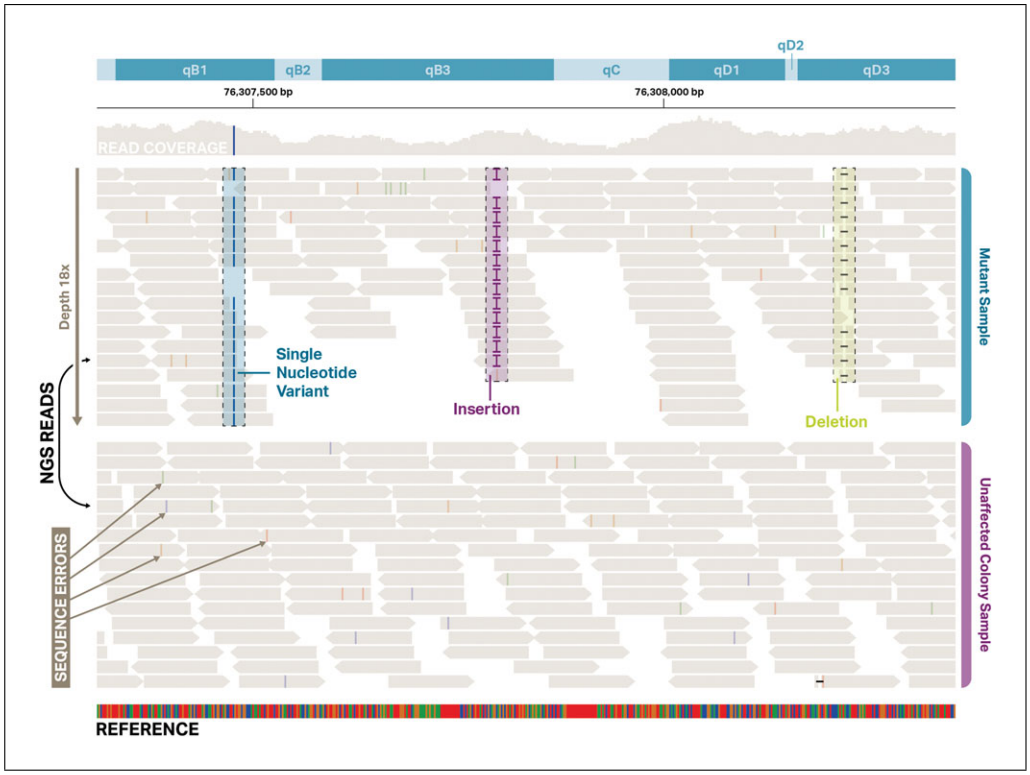


Figure 1 Graphical representation of NGS data including SNVs and small indels in a wild-type and mutant mouse sample.

observed and characterized (Qian, Mahaffey, Alcorn, & Anderson, 2011). These alleles include hypermorphic, amorphic, antimorphic, and neomorphic mutations (O'Brien & Frankel, 2004), resulting in a wide spectrum of manipulations and phenotypes.

Gene-driven screens are used to assess the phenotypes of mice after a known genetic lesion is produced. These screens can generate different mutation types using an array of different technologies. For example, the International Mouse Phenotyping Consortium (IMPC) originally used LacZ reporter methods to produce gene knock-outs in mice, while Panda et al. (2013) used transcription activator–like effector nucleases (TALENs) to produce mutations in mice. Increasingly more laboratories and large phenotype screens are using clustered regularly interspaced short palindromic repeats and the Cas-9 protein (CRISPR/Cas9) to generate direct mutations in the zygote. Due to the prior knowledge of the mutation type and genomic location, the use of NGS to map and identify novel causative mutations is usually not required. However, NGS still facilitates modern gene-driven screens, typically by quantifying the accuracy of the method (Mianné et al., 2016). For instance, in addition to the use of NGS to identify ad hoc mutations that may modify the phenotypes observed, NGS is typically used to verify any off-target genomic modifications widely seen in experiments that use CRISPR to induce mutations in mice. This protocol focuses on single nucleotide variations (SNVs) primarily but not exclusively produced by ENU mutagenesis screens, as shown in Figure 1.

STRATEGIC PLANNING OF MUTATION-DETECTION EXPERIMENTS

Forward genetic screens such as ENU mutagenesis can be either dominant or recessive. The dominant screen refers to mutations that show a dominant or semidominant phenotype in the first generation from the mutagenized male. The advantage here is that only one round of breeding is required before phenodeviants are identified; however, a disadvantage is that the causative mutations are identified less frequently, reflecting the high frequency of mutations in the first generation. To characterize recessive mutations, two more generations of breeding are required before a phenodeviant can be detected. Here, the G1 progeny are mated to a wild-type mouse, making the mutations in the G2 progeny heterozygous. To make these ENU-induced mutations homozygous and easier to characterize phenotypically, the G2 mice are intercrossed to produce G3 mice. Alternatively, female G2 mice are backcrossed to G1 males, again producing homozygous mutations (Balling, 2001). G1 dominant and G3 recessive breeding schemes are outlined in Figure 2. Typically, investigators will sequence the G3 mice to get the candidate SNVs for a particular phenotype. However, to obtain all the SNVs for the array of phenotypes generated from one mating, the G1 can be sequenced; then, all the SNVs will be captured and the G3 genotyped for particular SNVs or SNVs within a candidate region (Potter et al., 2016). Traditionally, candidate regions for a particular phenotype were obtained via positional cloning and fine mapping strategies (Justice, Noveroske, Weber, Zheng, & Bradley, 1999) to determine regions of homozygosity. With NGS and the protocols here, only gross mapping or indeed no mapping is required because sufficient low-confidence SNVs are filtered from further consideration to generate a concise list of plausible SNVs for validation. For this reason, it is advisable to have sufficient variant information from your mouse colony to use for comparison and filtering purposes. While a discussion of experimental validation techniques is beyond the scope of this article, validating genetic variants through Sanger sequencing assays remains a valuable tool to confirm putative mutations characterized using techniques covered here.

There are many common experimental platforms for performing NGS, each with their own chemistry, such as Illumina (Bennett, 2004), PacBio (Korlach et al., 2017), and others. A data variation that affects analysis is sequence read length. This article assumes that sequencing was performed with a version of the Illumina HiSeq platform to produce

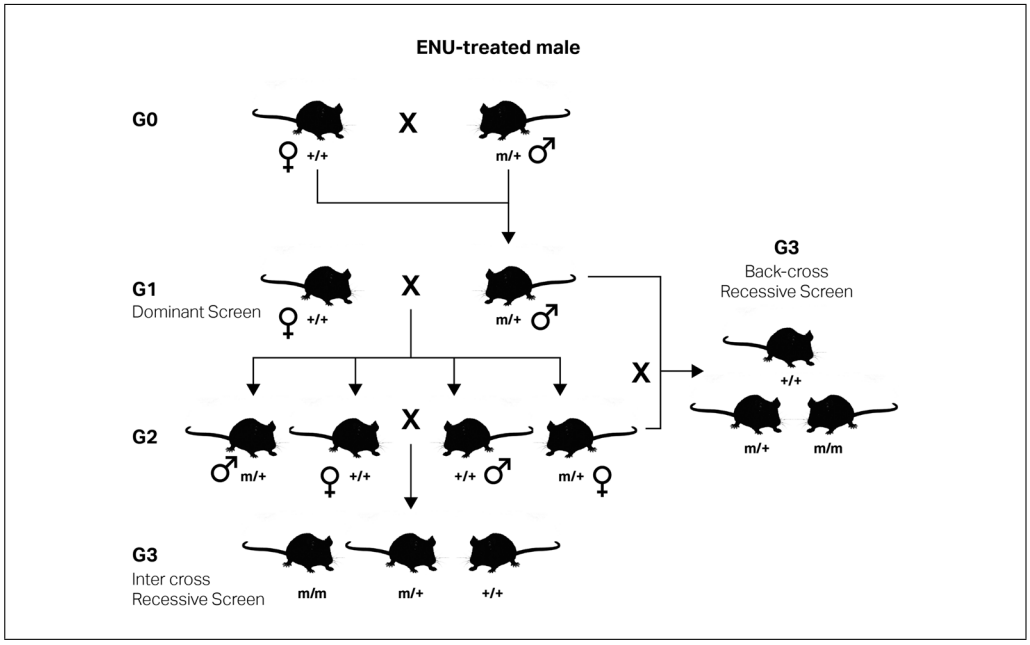


Figure 2 ENU-mutagenized males are mated to wild-type females. Each G1 offspring carries a unique set of mutations and therefore will exhibit different phenotypes. To segregate the phenotypes with a mutation or mutations, the G1 male mice are mated to wild-type females. The G2 offspring are then either intercrossed to each other or mated back to the original G1. The G3 progeny can be phenotyped for both recessive and dominant mutations. Typically, wild-type females of a different genetic background are used in the breeding scheme to facilitate the genetic mapping of any phenodeviant G3 offspring.

short reads of 75- to 100-base-pair (bp) length. Though sequencing quality can decrease with longer reads, especially toward the 3' end, the increase in read length improves mapping. Proper quality control (QC) can access the accuracy of longer (100-bp) reads. If needed, read trimming can be used to improve mapping. We recommend trimming reads after accessing quality of the FASTQC report generated in Basic Protocol 1, and use of TrimGalore for that purpose (Krueger, 2019; Martin, 2011). For a qualitative review of adapter trimming methods, see Fabbro, Scalabrin, Morgante, and Giorgi (2013). If other sequencing protocols are followed, the choice of aligner and pre-processing QC steps will need to be adjusted. If other sequencing protocols are followed, starting from Basic Protocol 1, step 17, the rest of the Basic Protocol 1 and subsequent protocols can be followed without adjustment provided that a SAM file of aligned reads can be supplied. NGS can produce either single-end or paired-end reads (Sengupta et al., 2011) from DNA fragments produced from a library of templates from the mouse genome. This article assumes that readers will be using paired-end reads, though Basic Protocol 1 may be modified to use single-end reads and the subsequent protocols are not dependent upon single- or paired-end design. While single-end reads sequence each DNA fragment once, paired-end reads sequence each twice, once in each direction starting at each end of the fragment. This produces twice the number of reads for the same amount of sequenced fragments. Paired-end sequencing is more costly than single-end; however, it offers several advantages by indicating positions of reads relative to each other. This facilitates more accurate identification of insertions and deletions (indels) than single-end reads, and increases reliability of SNPs called, especially in repetitive regions. For a reliable assessment of sequencing technology, see Park and Kim (2016); also see Shendure, Porreca, and Church (2008) for relevant reviews.

COMPUTATIONAL RESOURCE REQUIREMENTS

NGS data sets are large; thus, a large amount of computational resources are required to efficiently handle such data. A single sequencing experiment can produce several gigabytes of compressed reads, and extensive memory (RAM) is needed to process such volumes of data. Modern computers with at least 8 GB of RAM and 500 GB of disk space are sufficient for small analyses, provided that a modern 64-bit processor and multiple cores are included. This tutorial assumes a Unix-like environment, either Linux or a recent version of Mac OS. Windows users may be served by using a virtual machine to emulate a Linux distribution, as many software tools used in this article are not supported in Windows environments. Having multiple physical or virtual cores will allow parallel processing at certain steps, increasing processing speed dramatically. While such a configuration on a personal workstation may suffice, in practice, servers or cloud computing environments are utilized. An introduction to using remote servers is beyond the scope of this article, which has been written to be easily implemented either locally on a workstation, or remotely on a server directly, without need to use a grid engine to submit work to distributed computing resources.

ALIGNMENT OF SEQUENCING READS AND QUALITY CONTROL

BASIC PROTOCOL 1

This protocol describes how to align sequences to the current mouse reference genome. Starting with some housekeeping setup to make this and following protocols accessible in any Unix-like computing environment, the current reference genome is acquired and processed with SAMtools (Li et al., 2009). The FASTQC tool is used to access the quality of the NGS reads themselves before initiating alignment (Andews, 2015). Sequencing reads are then aligned to the genome using the Burrows-Wheeler Aligner (BWA; Li & Durbin, 2009). Picard Tools is used to mark duplicate reads in the file, and SAMtools is again used for additional postprocessing (Picard Toolkit, 2018). The created files are used in Basic Protocol 2 to detect variants.

Materials

Hardware

Workstation (Mac OS X, Linux, or any Unix-like system) with 500 GB of RAM, a 64-bit processor with at least two real or virtual cores, and 500 GB free disk space either on the workstation or in an external drive. An alternative is a remote server meeting at least these specifications. Administrator privileges are required for initial software setup, but do not have to belong to the end user. See Computational Resource Requirements for more information.

Software

A Java Runtime Environment (JRE) can be verified by visiting *https://www.java.com/en/download/installed.jsp?detect=jre*.
A Docker platform installation (Merkel, 2014)
FASTQC version 0.11.18 (Andews, 2015)
SAMtools version 1.9 (Li et al., 2009)
GATK version 4.0.11.0 (McKenna et al., 2010)
BWA version 0.7.17 (Li & Durbin, 2009)

Unix tools

The following bash commands are used, which should be installed on any modern Linux or Mac OS X distribution:
wget, tar, sed, cat

Datasets

Raw sequencing reads in FASTQ format from a sequencer. In absence of user-generated data, the Sequence Read Archive (*https://www.ncbi.nlm.nih.gov/sra*) may be investigated for test data (Leinonen, Sugawara, Shumway, & International Nucleotide Sequence Database Collaboration, 2011).

Setup software and file paths

In this and the following protocols, a working directory must be used. Software can likewise be stored in any accessible directory. Setting up one-time aliases for these directories enables seamless integration with this protocol. See below to set up aliases. If the software directory is `/NGS/Software/MouseMutations`, create a variable "`mySoftware`" to store this location. If the working directory is `/NGS/working_projects/MouseMutations`, create a variable "`myProjects`" to store this location.

1. Set up aliases for your directories by typing:

```
mySoftware=/NGS/Software/MouseMutations
myProject=/NGS/working_projects/MouseMutations
```

   Once the data are obtained in FASTQ format, the initial step is to align the sequence reads to the genome. Before starting this endeavor, it is useful to access the quality of the sequences themselves. FASTQC is a community-adopted program to access sequence quality. FASTQ files are stored in the "`myProject`" directory and labeled as:

```
sample_1.fastq
sample_2.fastq
```

Perform quality control

2. Each FASTQ file is from one of the "paired ends" of a single experiment. If experimental protocols use single-end reads, each sequencing run will produce one `sample.fastq` file instead of two. The `_1` and `_2` labels denote files, which are paired ends of the same experiment. This protocol may be modified to use single-end reads by typing "`sample.fastq`" instead of both "`sample_1.fastq`" and "`sample_2.fastq`" in steps 2 and 16 of this protocol. Run the following command to execute FASTQC on both of the FASTQ files. This will output reports in FASTQ format (html files and zip files). Adding "`--threads 8`" allows the program to use eight computational threads to speed up processing. Change this to match the settings on the workstation in use. To access how many threads are available on a Linux environment, type:

```
nproc
```

   Or on a Mac OS X environment type:

```
sysctl hw.logicalcpu
```

Output will inform the code below, as well as any further decisions on the number of threads or processes available for use. FASTQC reports will be generated and may be interpreted by viewing documentation in "Internet Resources."

```
$mySoftware/FastQC/fastqc *fastq --threads 8
```

Obtain and index the mouse genome

To conveniently obtain the mm10 genome from mouse strain C57BL/6J, Illumina creates builds by Ensembl, UCSC, and the NCBI. The Mouse Genomes Project contains draft assembled genomes for 16 laboratory- and wild-derived strains, which may be used in place of the standard mm10 genome as biological questions dictate. These may be accessed at *https://www.sanger.ac.uk/science/data/mouse-genomes-project* (Adams, Doran, Lilue, & Keane, 2015).

3. Download the mm10 UCSC genome:

```
wget ftp://igenome:G3nom3s4u@ussd-ftp.illumina.com/
Mus_musculus/UCSC/mm10/Mus_musculus_UCSC_mm10.tar.gz
```

4. Uncompress the files:

```
tar -zxvf Mus_musculus_UCSC_mm10.tar.gz
```

5. Change to the `Chromosomes` directory:

```
cd Mus_musculus/UCSC/mm10/Sequence/Chromosomes/
```

6. Edit a header in the `.fa` files downloaded to remove "`chr`":

```
for f in *.fa; do sed -i 's/chr//g' $f; done;
```

7. Combine individual chromosomes with the "`cat`" command into one output "`ref_mm10.fasta" file`:

```
cat chr1.fa chr2.fa chr3.fa chr4.fa chr5.fa chr6.fa
chr7.fa chr8.fa chr9.fa chr10.fa chr11.fa chr12.fa
chr13.fa chr14.fa chr15.fa chr16.fa chr17.fa chr18.fa
chr19.fa chrM.fa chrX.fa chrY.fa > ref_mm10.fasta
```

8. Move the `.fasta` file into a new "`musRefs`" directory and move back to the project directory:

```
mkdir -m 777 $myProject/musRefs_10/
mv ref_mm10.fasta $myProject/musRefs_10/
cd $myProject
```

9. Index the mm10 genome for use with BWA:

```
$mySoftware/bwa/bwa index -a bwtsw musRefs_10/
ref_mm10.fasta
```

10. Enable permissions to edit the files in musRefs_10:

```
cd musRefs_10; chmod 777 *
```

11. Index the mm10 genome for use with SAMtools:

```
$mySoftware/samtools/bin/samtools faidx ref_mm10.fasta
```

12. Index the mm10 genome for use with the Genome Analysis Toolkit (GATK). If not previously downloaded, pull the following GATK image by typing:

```
docker pull broadinstitute/gatk:4.0.11.0
```

13. Then, mount the GATK image and re-create the `myProject` variable:

```
docker run -t -i -v $PWD:/gatk/MouseMutations
broadinstitute/gatk:4.0.11.0 /bin/bash
myProject=/gatk/MouseMutations
```

14. Use GATK to create the index:

```
$mySoftware/gatk-4.0.11.0/gatk CreateSequence
Dictionary --REFERENCE=$myProject/ref_mm10.fasta
--OUTPUT=$myProject/ref_mm10.dict
```

15. Move back out of the Docker container and into the `$myProject` directory:

```
exit; cd $myProject
```

Align experimental reads to the mouse genome and format results

16. With the mm10 reference genome FASTA file and the two mouse FASTQ files given below, run BWA to align the genome to mm10. In the command below, the BWA program is installed in the `$mySoftware/bwa/bwa` directory. "`mem`" is called to align 70 bp to 1 Mbp-length reads with "`-t`" of 30 threads. Adjust the number of threads to fit the workstation. The FASTQ file has been indexed as above and is stored in `musRefs_10/ref_m10.fasta`. As stated above, paired-end sequencing has produced two FASTQ files, one for each pair. If single-end sequencing has been used, simply give one FASTQ file. BWA assumes that the two FASTQ files given are mates, with one FASTQ file being interpreted as "`reads.fastq`" and the other as the corresponding "`mates.fastq`" pair. See the BWA documentation for

more information. Run the line below. The end result will be an aligned "SAM" file in the `$myProject` directory.

```
$mySoftware/bwa/bwa mem -t 30 musRefs_10/ref_mm10.fasta
sample_1.fastq sample_2.fastq > sample.sam
```

17. Use GATK's included version of Picard Tools (Picard Toolkit, 2018) to add read group names to the SAM file created above. Then mount the Docker image to use GATK, including a map to the local `$myProject` directory which you are currently in. Once there, re-create the `$myProject` variable in the Docker container:

```
docker run -t -i -v $PWD:/gatk/MouseMutations
broadinstitute/gatk:4.0.11.0 /bin/bash
myProject=/gatk/MouseMutations
```

18. Now, use Picard Tools to allow GATK's HaplotypeCaller to work correctly. The values passed to number a read group (`--RGID`), a read library (`--RGLB`), and platform (`--RGPL`), a platform unit or run barcode (`--RGPU`), and a sample name (`--RGSM`) can be either arbitrary or have assigned meaning.

```
gatk AddOrReplaceReadGroups --INPUT=$myProject/
sample.sam --OUTPUT==$myProject/sample.named.sam
--RGID=1 --RGLB=lib1 --RGPL=illumina --RGPU=unit1
--RGSM=1
```

Sort the aligned reads

19. Next, sort the SAM file by leftmost coordinates to allow Picard Tools to correctly find duplicates, and compress the SAM file into its binary analog, a BAM file. This is done using Picard Tools bundled with GATK and executing the `SortSam` command. The output of this command is a sorted BAM file. Do this in one step as shown below.

```
gatk SortSam --INPUT==$myProject/sample.named.sam --
OUTPUT==$myProject/sample.sorted.bam --SORT_ORDER=
coordinate
```

Deduplicate aligned, sorted reads

20. To mark potential PCR artifacts such as duplicate reads, run the MarkDuplicates Picard Tools function in GATK. The input "`INPUT`" flag is assigned to the sorted BAM above. The output of the new file "`OUTPUT`" ends in `sorted.rmDub.bam`. The MarkDuplicates command requires a metrics file to be included to output statistics; call this `Input.tmp_file`. Ask MarkDuplicates to remove duplicates found via the `REMOVE_DUPLICATES=true` option, and to assume the file is sorted (`ASSUME_SORTED`). Lastly, set the VALIDATION_STRINGENCY to silent to allow processing of the entire file without the program failing if errors are encountered. Use a "`tmp/`" directory as your temporary "TMP_DIR" to store files, which will be deleted after the completion of the protocol.

```
gatk MarkDuplicates --INPUT==$myProject/sample.
sorted.bam --OUTPUT==$myProject/sample.sorted.
rmDup.bam --METRICS_FILE=Input.tmp_file
--REMOVE_DUPLICATES=true --ASSUME_SORTED=true
--VALIDATION_STRINGENCY=SILENT --TMP_DIR=tmp/
```

21. Exit the Docker container by typing:

```
exit
```

    to return to the `$myProject` directory in the native shell (set above as `/NGS/working_projects/MouseMutations`).

Index the deduplicated reads

22. Lastly, index the deduplicated BAM file with SAMtools to run GATK options in Basic Protocol 2. This will create an index file in the `$myProject` directory.

```
$mySoftware/samtools/bin/samtools index
$myProject/sample.sorted.rmDup.bam
```

    Now the sorted, deduplicated, indexed BAM file is ready to be analyzed with the GATK.

BASIC PROTOCOL 2

MUTATION DETECTION AND VARIANT GENOTYPING

This protocol focuses on using the GATK for variant discovery and genotyping. While different tools can be used for this protocol, GATK is in current development and well documented. Of note, previous versions of GATK included custom-built multiprocessing ability. New versions (>4.0.8) use Spark (Zaharia, Chowdhury, Franklin, Shenker, & Stoica, 2010) to enable multiprocessing. However, this is under development and should not be used for production work until moved out of "Beta" release, as results via Spark currently do not match those shown below in this protocol. Input for this protocol is the SAM file from Basic Protocol 1. First, the Haplotype Caller is used to build a de novo assembly of haplotypes in regions of higher-than-expected variation. This facilitates calling indels and SNPs that may be close together or in high-complexity regions. Next, the re-assembled haplotypes are used as input for calling genotypes with the Genotype Caller to produce a Variant Call Format (VCF) file. Lastly, this VCF file is annotated based on specific QC hard filters and is then ready to be compared to known mutations in Basic Protocol 3.

Materials

Hardware

Workstation (Mac OS X, Linux, or any Unix-like system) with 500 GB of RAM, a 64-bit processor with at least two real or virtual cores, and 500 GB free disk space either on the workstation or in an external drive. An alternative is a remote server meeting at least these specifications. Administrator privileges are required for initial software setup, but do not have to belong to the end user. See Computational Resource Requirements for more information.

Software

A JRE can be verified by visiting *https://www.java.com/en/download/installed.jsp?detect=jre*
GATK version 4.0.11.0
Docker

Datasets

The sorted, deduplicated, indexed BAM file from Basic Protocol 1:

```
$myProject/sample.sorted.rmDup.bam
```

The indexed mouse reference genome from Basic Protocol 1 along with its index and dictionary:

```
$myProject/musRefs_10/ref_mm10.fasta
$myProject/musRefs_10/ref_mm10.dict
$myProject/musRefs_10/ref_mm10.fasta.fai
```

1. First, use the Haplotype Caller in GATK to call local haplotype assemblies for your sorted, deduplicated BAM file. First, load the Docker image and mount the `$myProject` directory as in Basic Protocol 1:

```
docker run -t -i -v $PWD:/gatk/MouseMutations
broadinstitute/gatk:4.0.11.0 /bin/bash
myProject=/gatk/MouseMutations
```

2. Use the built in GATK script to call the `GenomeAnalysisTK.jar` file (the GATK program) within the GATK Docker image. `--java-options "-Xmx4g"` indicate that you are giving 4 GB of memory to the JRE. GATK is using the HaplotypeCaller program, and the "`-R`" flag indicates the directory of your reference FASTA file for the mouse genome. The "`-I`" flag points to your sorted deduplicated BAM file. The `-O` flag indicates where the output will be stored (a VCF file). Lastly, the "`-ERC`" GVCF flag indicates that the algorithm will output an intermediate genomic VCF. The resulting output file will be `$myProject/sample.sorted.rmDup.g.vcf`, a genomic VCF fie. While superficially the same as a VCF file that will be will encountered later, this also encodes information from the reference genome (`-R`) and indicates both variant site records and blocks of nonvariant sites. Type:

```
gatk --java-options "-Xmx4G" HaplotypeCaller -R
$myProject/musRefs_10/ref_mm10.fasta -I
$myProject/sample.sorted.rmDup.bam -O
$myProject/sample.sorted.rmDup.g.vcf -ERC GVCF
```

Once this file is created by the HaplotypeCaller, it can be genotyped with the GenotypeGVCFs function. Simply pass the new "`g.vcf`" file to this function as below. Here, call GATK as above, but specify the GenotypeGVCFs function. The reference FASTA is the same as above, and the input is your "`g.vcf`" from the Haplotype Caller. The output is "`toFilter-gatk.vcf`," which will be used for variant filtering.

3. Type:

```
gatk GenotypeGVCFs -R $myProject/musRefs_10/ref_
mm10.fasta -V $myProject/sample.sorted.rmDup.g.
vcf -O $myProject/toFilter-gatk.vcf
```

4. Various filters can be applied. It is recommended to filter by read depth and sequencing quality. While in humans GATK's VQSR (variant recalibration based on 1000 Genomes and other training data) can be used, in mouse no such models exist due to lack of training data. Thus, hard filters may be applied. The expected result of this step is to encode the seventh field, FILTER, of `toFilter-gatk.vcf` with either the string "`PASS`" or another code indicating that the SNP should be filtered out early in Basic Protocol 3. The filter below will filter based on a quality score of 30 and a read depth for each SNP of at least 10×. For variants that do not pass this simple filtration, instead of "`PASS`," the "`--filter-name`" of "`myDepthQualityFilter`" will be input. Output will be a VCF file with these filter options added.

```
gatk VariantFiltration -R $myProject/musRefs_10/
ref_mm10.fasta -V $myProject/toFilter-gatk.vcf -
O $myProject/filtered-gatk.vcf --filter-expression
"QUAL > 30.0 && DP == 10" --filter-name "myDepth
QualityFilter"
```

5. Finally, exit the Docker container to return to the rest of the protocol. Continue with Basic Protocol 3 unless Alternate Protocol or Support Protocol 2 apply. Alternate Protocol addresses experiments with multiple biological replicates, a common situation. While this protocol can be repeated for each replicate, Alternate Protocol automates the process. Support Protocol 2 provides for additional "hard filtering" steps, which may be needed if the quality of the experimental data is suspect, and provides a template for adding additional filtering steps.

```
exit
```

BASIC PROTOCOL 3

NOVEL MUTATION DISCOVERY

After detecting mutations with the GATK pipeline described above, we wish to compare our findings to known external results, and therefore annotate only novel findings. If annotating all mutations is desirable (and one does not want to annotate only novel mutations), this protocol may be skipped, and the VCF file from Basic Protocol 2 can be used as input to Basic Protocol 4. This protocol will produce a file with unknown variants. We will then annotate unknown variants in Basic Protocol 4.

Materials

Hardware

Workstation (Mac OS X, Linux, or any Unix-like system) with 500 GB of RAM, a 64-bit processor with at least two real or virtual cores, and 500 GB free disk space either on the workstation or in an external drive. An alternative is a remote server meeting at least these specifications. Administrator privileges are required for initial software setup, but do not have to belong to the end user. See Computational Resource Requirements for more information.

Unix tools

The following bash commands are used, which should be installed on any modern Linux or Mac OS X distribution:

wget, gunzip, awk, cut, comm

Datasets

The filtered GATK processed VCF file from Basic Protocol 2 (or the alternate or support protocols, as mentioned above):

```
$myProject/filtered-gatk.vcf
```

A reference panel of known SNPs downloaded from the Sanger Institute (see below) will be acquired during this protocol. Sanger provides current data updated via the URL *ftp://ftp-mouse.sanger.ac.uk/current_snps*.

1. Make sure to be in the myProject working directory:

```
cd $myProject
```

2. Download the mouse known mutation file with wget:

```
wget ftp://ftp-mouse.sanger.ac.uk/current_snps/
mgp.v5.merged.snps_all.dbSNP142.vcf.gz
```

3. Unzip the mouse mutation file. This will output one file, `mgp.v5.merged.smps_all.dbSNP142.vcf`:

```
gunzip mgp.v5.merged.snps_all.dbSNP142.vcf.gz
```

Note that in practice every mouse colony is different and may contain its own specific mutations due to drift. Laboratories combine the file above with known mutations from their own colony to filter out variants they have previously characterized. It is suggested that current researchers do the same. Going forward, assume that the unzipped file above contains combined mutations from one's own lab and the known SNPs database.

4. Next, remove the words "`chr`" from the "`$myProject/filtered-gatk.vcf`" file in column 1 of the VCF file, and remove all headers, using the "grep" tool. Then, filter each field for only variants that passed filters from the GATK protocol using "awk." To clean up output for comparing this merged dpSNP file to the filtered GATK file, "cut" the fields 1, 2, 4, and 5, then "sort" them into a temporary file called "a." Field 3, which holds SNP rs IDs, will be withheld and re-annotated in Basic Protocol 4:

```
grep -v '^#' mgp.v5.merged.snps_all.dbSNP142.vcf |
awk '$7 == "PASS"' | cut -f1,2,4,5 | sort -u > a
```

5. Next, repeat step four (above) with the `gatk.vcf` file, creating a temporary file called "b":

```
grep -v '^#' filtered-gatk.vcf | awk '$7 == "PASS"' |
cut -f1,2,4,5 | sort -u > b
```

Table 1 The Structure of Mutant.vcf Passed to VEP for Annotation[a]

Field position	Field name	Field meaning	Example
1	CHROM	Chromosome number	2
2	POS	Genomic coordinate	112876894
3	ID	SNP ID	rs246375767
4	REF	Reference allele	C
5	ALT	Alternate allele	T

[a]Output from the "`mutant.vcf`" file explained. Each field can be used by VEP and other resources to characterize known and novel variants. The ID field is marked as "." if the SNP ID is unknown. Indels, CNVs, and other mutation types are characterized by the same file format and can also be input in to variant prediction protocols (see Basic Protocol 4).

6. Then, compare the files with the "comm" utility. The comm flag "`-13`" excludes lines unique to file "`a`" with flag 1 (the dpSNP reference) and in common to both (flag 3), keeping only field 2 of standard comm output, the lines unique to only "`b`," and the formatted `filtered-gatk.vcf` file holding unique mutations called `novel_snps`:

```
comm -13 a b > novel_snps
```

7. Lastly, `novel_snps` must be formatted to restore the lost column "3" cut out in steps 4 and 5, using a "`.`" to indicate yet-unknown SNPs. These will be annotated with Ensembl-sourced rsIDs in Basic Protocol 4. This line produces `mutant_vcf`, which is the input for Basic Protocol 4:

```
awk -F' ' '{$2 = $2 FS ".”; print $1," ",$2," ",$3,
" ",$4}' novel_snps > mutant.vcf
```

The resulting `mutant.vcf` file contains potentially novel SNVs. The five fields referenced above are explained in Table 1.

The only input needed for Basic Protocol 4 is the resulting file, `mutant.vcf`. This protocol involves downloading reference databases specific to your strain of mouse automatically. Characterize by Basic Protocol 4.

BASIC PROTOCOL 4

VARIANT INTERPRETATION

To characterize genomic variants identified as unique to our experiment above, we employ the Variant Effect Predictor (VEP) from Ensembl (McLaren et al., 2016). VEP is available in many formats, but to maintain compatibility it is becoming best practice to use a Docker instance. This ensures that VEP and all its dependencies are managed in a self-contained ecosystem and do not interfere with any other programs you may have installed on your system.

Materials

Hardware

Workstation (Mac OS X, Linux, or any Unix-like system) with 500 GB of RAM, a 64-bit processor with at least two real or virtual cores, and 500 GB free disk space either on the workstation or in an external drive. An alternative is a remote server meeting at least these specifications. Administrator privileges are required

for initial software setup, but do not have to belong to the end user. See Computational Resource Requirements for more information.

Software

Docker version 18.06 or greater
The Ensembl Variant Predictor Docker container (version 95.1)

Datasets

The `mutant.vcf` file produced in Basic Protocol 3:

```
$myProject/mutant.vcf
```

Download VEP

1. Clone VEP Docker image and run VEP. Cloning the Docker image will take time initially and does not need to be done repeatedly. Run:

   ```
   docker pull ensemblorg/ensembl-vep:release_95.1
   ```

   Run the following to call up the help message (optional), which gives parameters useful for diagnosing problems and expanding annotation options:

   ```
   docker run -t -i ensemblorg/ensembl-
   vep:release_95.1./vep --help
   ```

2. After VEP is installed, create a directory (folder) in which to put the `mutant.vcf` file:

   ```
   mkdir vep_data
   ```

3. Then, change the permissions to make sure that you can read, write, and execute programs in the directory.

   ```
   chmod a+rw vep_data
   ```

4. Now, move into that directory and copy the "`mutant.vcf`" file there:

   ```
   cd vep_data
   mv $myProject/mutant.vcf.
   ```

5. To make the following steps easier and in line with the official VEP tutorial, create a temporary variable in the bash shell which will locate the `VEP_data` folder. Using `$PWD` (below) ensures that this code snippet will work wherever one is in the workstation directories.

   ```
   myAnnotation=$PWD
   ```

Install VEP application programming interface (API)

The Docker image contains everything needed to run VEP. To install VEP within the Docker container, run the perl `INSTALL.pl` script contained in the Docker image. To do

this, simply enter the command under step 6. The "`t`" flag indicates that a pesudo-TTY is enabled so Docker can pass the image signals, the "`i`" enables the user to add information to the Docker image via standard input, and the "`v`" flag binds the "`$myAnnotation`" directory, where the "`mutant.vcf`" file is stored, to the "`/opt/vep/.vep`" hidden directory in the VEP Docker image. Doing this allows local files to be stored and accessed via Docker, and allows files generated in the ongoing analysis to be locally accessible after closing the VEP Docker image down.

6. Then type:

```
docker run -t -i -v $myAnnotation:/opt/vep/.vep
ensemblorg/ensembl-vep:release_95.1 perl INSTALL.pl
```

7. Next, a prompt will appear asking to install the API. If this is the first time using VEP, type "`y`." If not, type "`n`" unless upgrading to version 95.1 of the API:

```
Do you want to continue installing the API (y/n)? y
```

Install the mouse genome annotation and FASTA files

8. If installing the API (or upgrading to a new version), type "`y`" to proceed with overwriting the current cache directory. You will also be asked if you wish to cache any files. To cache the mouse genome variant information, type "`y`" at the prompt as depicted below:

```
Do you want to install any cache files (y/n)? y
```

   The mouse VEP annotation is listed in version 95.1 of VEP as 145:

```
145: mus_musculus_vep_95_GRCm38.tar.gz
```

9. Type `145` at the prompt, and it will install the mouse VEP v94 annotation files.

10. Having the files stored locally on the workstation or server will ensure a quick analysis time. Likewise, installing the human annotations may be useful for any comparisons of mouse SNP consequences to human data. Downloading and processing VEP cache data will take some time, but dramatically speed up future analyses, and need only be done once. VEP will then ask if it should install FASTA files. This is not required, but is suggested if this protocol is to be used frequently. Type "`y`" at the prompt to install FASTA files:

```
Do you want to install any FASTA files (y/n)? y
```

    There are several mouse strains sequenced, with option 82 being the default mm10 reference genome on strain C57BL/6J, listed as mus_musculus. Other mouse genomes are available from *ftp://ftp.ensembl.org/pub/release-95/fasta/*.

```
82: mus_musculus
```

11. Type `82` at the prompt to download Mus_musculus FASTA files at the question mark "`?`".

```
? 82
```

Install third-party plugins

The VEP installer will then ask if any plugins should be installed. Again, these are optional, but type "`y`" to install, then browse the list as required. They are useful if using offline mode, which can help troubleshooting when acquiring data via secure connections.

```
y
```

12. Type `0` at the next prompt next to the question mark "`?`". This will install all available plugins for mouse data, which are useful for giving various computational predictions of variant effects. If some installations fail, do not panic; they are still optional.

```
? 0
```

Annotate variants

Finally, VEP is fully ready to use. The Docker image will exit, presenting the working directory "`myAnnotation`." You will recall that the `mutant.vcf` file was previously moved to the `$myAnnotation` directory.

13. Run the Docker image and load the bash prompt by typing:

```
docker run -t -i -v $myAnnotation:/opt/vep/.vep
ensemblorg/ensembl-vep:release_95.1 /bin/bash
```

14. Running this command brings up a new bash prompt. To locate the `$myAnnotation` directory which we mapped above to `/opt/vep/.vep`, run the following command:

```
vep@29f222ab2c50:~/src/ensembl-vep$ ls /opt/vep/.vep/
```

The mutant file, plugins, and the mouse reference cache will each be seen and be ready for analysis.

15. Next, to annotate the "`mutant.vcf`" file, run the code below. This runs VEP with the "`mutant.vcf`" as input, with the cache of data and running all options. See Table 2 for explanation of flags and output:

```
./vep -i /opt/vep/.vep/mutant.vcf --cache
--everything --species mus_musculus --output_file
/opt/vep/.vep/mutant_VEP.txt --stats_file /opt/
vep/.vep/mutant_VEP.html -offline
```

Table 2 Output Options Passed to VEP[a]

Option name	Use
`--cache`	Uses cache FASTA mouse data
`--everything`	Runs all available analyses and plugins
`--species`	Species mouse species
`--output_file`	Creates a text output file in `/opt/vep/.vep/mutant_VEP.txt`
`--stats_file`	Creates a web page with summary statistics in `/opt/vep/.vep/mutant_VEP.html`
`--offline`	Uses our cache and plugins without online access

[a]Output options for VEP with explanations. Note that after `--species`, one types a Latin or English name. If `--offline` is set, then the species name must be in Latin, as in `--species mus_musculus`.

An error message will indicate indicating "`INFO: disabling PolyPhen`." This is normal, as the polyphen option is only relevant for human. These options produce several files, which are explained in Table 2.

A section of output from VEP annotation is depicted below in text format (the standard output, in this case being "`mutant_VEP.txt`"). An additional output file is an interactive website file, "`mutant_VEP.html`." This file can be loaded in any web browser. Both of these files will be in the "`$myAnnotation`" directory. The `mutant_VEP.txt` file will contain several lines of headers, each of which explains a field in the tab delimited file. The first 13 fields describe the variant, its position in the genome, and any coding consequences to the allele and gene in which it inheres. The last column, number 14, describes many potential sources of data which are explained in the headers of the file.

As is evident on inspection of the headers in `mutant_VEP.txt`, there are many header flags (rows starting with ##), as are common with VCF output files. Scrolling through these headers reveals the purpose of comments in the "Extra" column, where annotations are separated by "`;`" semicolons. Otherwise, fields are tab separated.

After giving the `--everything` flag above, VEP annotates SNPs with many tools downloaded. VEP annotations with the `--everything` flag are described in Table 3.

It is worth noting that using the "`--everything`" flag is the most convenient way to retrieve annotations for mouse. However, the default VEP output is largely meant for human data, and flags such as "`-af`" called by "`--everything`" will include the allelic frequency in ethnic populations reported in the 1000 Genomes Project. These are not relevant to mouse work, but it is advised to keep them in the workflow in case there is a need to annotate human SNPs.

A cleaner way to access variants on a global scale is to look at the `mutant_VEP.html` file. It contains summary statistics on mutation consequences, coding consequences, variants by chromosome, and position in the coding protein. Figure 1 shows a screenshot of that output. Note that the output generated in this protocol is from a sample of the `mutant.vcf` file of parts of the "X" and "Y" chromosome. We see that most variants are either intergenic or intronic.

Table 3 VEP Annotates SNVs with Several Resources[a]

Option name	Use
`--sift b`	AA substitution via homology
`--polyphen b`	AA substitution effect on structure
`--ccds`	Adds CCDS transcript ID
`--uniprot`	Adds Uniprot ID
`--hgvs`	Adds HGVS ID
`--symbol`	Adds gene symbol
`--numbers`	Adds exon/intron numbers
`--domains`	Adds overlapping domains
`--regulatory`	Overlapping regulatory regions
`--canonical`	Canonical transcript
`--protein`	Ensembl protein ID
`--biotype`	Biotype of transcript
`--uniprot`	Uniprot ID
`--tsl`	Transcript support level
`--appris`	Isoform annotation
`--gene_phenotype`	Phenotype of overlapped gene
`--pubmed`	Pubmed IDs
`--variant_class`	Sequence ontology variant class

[a]All options that VEP will call if given the `--everything` flag. Note that PolyPhen is only available for humans, and SIFT can also be species-specific but will work for mouse.

16. Scroll through the downloaded web page, which can open in any modern web browser. While data based on the SNVs and variants in real mice will be different, the analysis format will be similar. To work with the text file itself, the best approach is to use the command line. Opening the output of VEP may not be feasible in a modern word processor, especially in a program such as MS Word, due to memory constraints. We recommend using the text editor "nano," installed on all Mac and Linux systems. To peruse the text file, one may search for a gene or SNP of interest using the "grep" command. To use grep to search for the gene Vipr2, type:

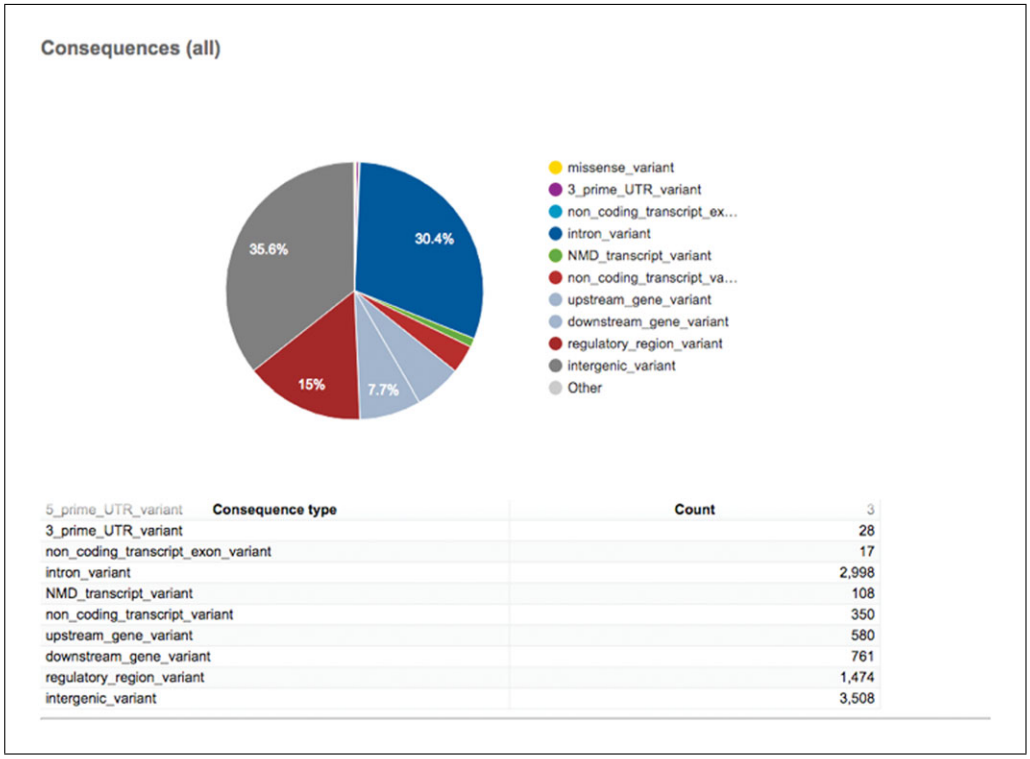
```
grep Vipr2 mutant_VEP.txt
```

If any variants are annotated to that gene, they will appear on the screen. VEP also includes a filter function. To explore this capability, type the following within the VEP Docker to display the help message:

```
./filter_vep --help
```

17. Lastly, exit the Docker image. Type:

```
exit
```



5_prime_UTR_variant	Consequence type	Count	3
3_prime_UTR_variant			28
non_coding_transcript_exon_variant			17
intron_variant			2,998
NMD_transcript_variant			108
non_coding_transcript_variant			350
upstream_gene_variant			580
downstream_gene_variant			761
regulatory_region_variant			1,474
intergenic_variant			3,508

Figure 3 A screenshot of the html web page generated by VEP displayed in a web browser. Note that tables displayed above are interactive, and you may scroll to find more information. The summaries generated are useful for overall characterization of mutations.

This ends Basic Protocol 4. Expected output relevant for further analysis are two files produced by VEP described above (Fig. 3). The html file provides a graphical overview of mutations and effects, while the annotated VCF file can be integrated into many analysis tools and has been analyzed for many consequences, as described in Table 3.

SUPPORT PROTOCOL 1

INSTALLATION OF NEEDED SOFTWARE

This protocol is written to aid in the installation of software needed for the analysis detailed in this article. Full instructions and help can be obtained from the websites referred to throughout the article and in the references provided.

Materials

Hardware

Workstation (Mac OS X, Linux, or any Unix-like system) with 500 GB of RAM, a 64-bit processor with at least two real or virtual cores, and 500 GB free disk space either on the workstation or an external drive. An alternative is a remote server meeting at least these specifications. Administrator privileges are required for initial software setup, but do not have to belong to the end user. See Computational Resource Requirements for more information.

Software

A JRE can be verified by visiting *https://www.java.com/en/download/installed.jsp?detect=jre*

Docker

Install Docker

1. To install Docker on a Linux system (assuming Ubuntu or Debian is used), type:

```
$ sudo apt-get update; apt-get install docker-ce;
```

   This will update the cache of available programs and install the community edition of Docker. Type the above command in the shell/terminal of choice, and a current version of Docker will be installed. This command first updates your cache and then installs docker-ce. If you have older versions of Docker installed, it is advisable to delete them first. Further guidance can be found at:

   https://docs.docker.com/install/linux/docker-ce/ubuntu/.

   On a system running Mac OS X, Docker can be installed by visiting *https://hub.docker.com/editions/community/docker-ce-desktop-mac* and following instructions.

Install FASTQC

2. To install FASTQC, first change to the "`$mySoftware`" directory by running:

```
cd $mySoftware
```

3. Then, use wget to download the compressed zip file:

```
wget https://www.bioinformatics.babraham.ac.uk/
projects/fastqc/fastqc_v0.11.8.zip
```

4. Lastly, unzip the downloaded zip file via the "unzip" utility pre-installed with Linux or Mac OS X:

```
unzip fastqc_v0.11.8.zip
```

Install BWA

5. To install the BWA software, pre-compiled binaries are available for many systems. An alternative approach is to use the "git" utility. Assuming a Linux Ubuntu environment, install git with:

```
sudo apt-get update; sudo apt-get install git
```

6. Clone the git directory into "`$mySoftware`":

```
git clone https://github.com/lh3/bwa.git
```

7. Change into the new "`bwa`" directory:

```
cd bwa
```

8. Make the file that will install BWA:

```
make
```

Install SAMtools

9. Change back into the "$mySoftware" directory.

```
cd $mySoftware
```

10. To install SAMtools, first download the current version into your $mySoftware directory. This will install version 1.9, current as of this writing:

```
wget https://github.com/samtools/samtools/releases/
download/1.9/samtools-1.9.tar.bz2
```

11. Next, extract the file with the "tar" command:

```
tar -xvjf samtools-1.9.tar.bz2
```

12. Change into the samtools directory:

```
cd samtools-1.9/
```

13. Then, change a configuration file to point to $mySoftware/samtools:

```
./configure --prefix=$mySoftware/samtools
```

14. Finally, build the SAMtools installation:

```
make; make install
```

Pull GATK and VEP images

15. Pull the GATK version 4.0.11.0 Docker image:

```
docker pull broadinstitute/gatk:4.0.11.0
```

16. Pull VEP v 95.1:

```
docker pull ensemblorg/ensembl-vep:release_95.1
```

ALTERNATE PROTOCOL

MUTATION DETECTION WITH MULTIPLE BIOLOGICAL REPLICATES

In practice, mutation detection is often performed on many mice to capture natural biological variation, to account for the variable penetrance of mutations, and to ensure statistical power to find variations of interest. Below, Alternate Protocol duplicates the

steps in Basic Protocol 2, but for multiple files. Of note, the entire process in Basic Protocol 1 (above) must be run for each file.

Materials

Hardware

Workstation (Mac OS X, Linux, or any Unix-like system) with 500 GB of RAM, a 64-bit processor with at least two real or virtual cores, and 500 GB free disk space either on the workstation or in an external drive. An alternative is a remote server meeting at least these specifications. Administrator privileges are required for initial software setup, but do not have to belong to the end user. See Computational Resource Requirements for more information.

Software

A JRE can be verified by visiting *https://www.java.com/en/download/installed.jsp?detect=jre*
GATK version 4.0.11.0
Docker

Datasets

For each input pair of FASTQ files from paired-end experiments, there should be a resulting deduplicated, sorted, indexed SAM file in the associated `$myProject` directory. When performed on one sample, the resulting sample files, and index files. The directory `musRefs_10` should contain everything generated in Basic Protocol 1:

```
$myProject/musRefs_10/ref_mm10.fasta
$myProject/musRefs_10/ref_mm10.dict
$myProject/musRefs_10/ref_mm10.fasta.fai
```

To proceed, assume that multiple samples have been processed generating each of the files below, with names "`sample1`" and "`sample2`" indicating the indexed BAM files for each of two replicates:

```
$myProject/sample1.sorted.rmDup.bam
$myProject/sample1.sorted.rmDup.bam.bai
$myProject/sample2.sorted.rmDup.bam
$myProject/sample2.sorted.rmDup.bam.bai
```

NOTE: See Basic Protocol 2 for a full explanation of the options passed to GATK below.

1. Within the `$myProject` directory, enter the GATK Docker container as in Basic Protocol 1, changing into the `$myProject` directory to batch process files:

```
docker run -t -i -v $PWD:/gatk/MouseMutations broad-
institute/gatk:4.0.11.0 /bin/bash
myProject=/gatk/MouseMutations
cd $myProject
```

2. In the code below, a variable "`f`" is created to hold the names of each sample BAM file, and an additional variable is created to hold a new name for each output file "`newF`." In a "for loop," call the GATK Haplotype Caller for each of the sample files above, generating a "`g.vcf`" file for each sample:

```
for f in *.sorted.rmDup.bam; do newF=${f/bam/g.vcf};
/gatk/gatk --java-options "-Xmx4G" HaplotypeCaller -R
$myProject/musRefs_10/ref_mm10.fasta -I $myProject/$f
-O $myProject/$newF -ERC GVCF; done;
```

This produces several `.g.vcf` files. These "`.g.vcf`" files produced by GATK need to be consolidated with the CombineGCVFs function, which will produce one `.g.vcf` for combined genotype calling. To enable this, first create a file of arguments to read into the CombineGCVFs function. The function takes a list of each variant on the command line, and this can be automated by creating a file listing these commands.

3. Below, search for each `.g.vcf` file and store it in the "`i`" variable, then paste each variable into a "`myArguments.txt`" file:

```
for i in *.g.vcf; echo "--variant $i" >> myArguments.
txt; done;
```

Next, use the list of annotations to combine the "`.g.vcf`" files with the Combine GVCFs function. Note that unlike the above analogous step in Basic Protocol 2, no "`-V`" flag is passed containing the "`.g.vcf`" file; instead, these files are listed in the "`myArguments.tt`" file.

4. Type the following to produce one `toFilter-gatk.vcf` file:

```
/gatk/gatk GenotypeGVCFs -R $myProject/musRefs_10/
ref_mm10.fasta -O $myProject/toFilter-gatk.vcf
--arguments_file myArguments.txt
```

This final step duplicates the last step of Basic Protocol 2, above (refer to Basic Protocol 2, step 3).

5. Type:

```
/gatk/gatk VariantFiltration -R $myProject/musRefs
_10/ref_mm10.fasta -V $myProject/toFilter-gatk.vcf
-O $myProject/filtered-gatk.vcf --filter-expression
"QUAL > 30.0 && DP == 10" --filter-name "myDepth
QualityFilter"
```

6. Lastly, exit the Docker container:

```
exit
```

SUPPORT PROTOCOL 2

VARIANT GENOTYPING WITH ADDITIONAL HARD FILTERING

As a substitute for the last step of Basic Protocol 2 and the identical step in Alternate Protocol, many additional filtering options may be passed to GATK's VariantFiltration function. Above, filtering was performed to filter on Phred score (base calling quality) and read depth. This support protocol demonstrates how to edit the command above to include other recommended filtering steps.

Materials

Hardware

Workstation (Mac OS X, Linux, or any Unix-like system) with 500 GB of RAM, a 64-bit processor with at least two real or virtual cores, and 500 GB free disk space either on the workstation or in an external drive. An alternative is a remote server meeting at least these specifications. Administrator privileges are required for initial software setup, but do not have to belong to the end user. See Computational Resource Requirements for more information.

Software

A JRE can be verified by visiting *https://www.java.com/en/download/installed.jsp?detect=jre*
GATK version 4.0.11.0
Docker

Datasets

The reference genome and associated files should be included as above in Basic Protocol 2:

```
$myProject/musRefs_10/ref_mm10.fasta
$myProject/musRefs_10/ref_mm10.dict
$myProject/musRefs_10/ref_mm10.fasta.fai
```

Input to the filtering step is the VCF file generated above:

```
$myProject/toFilter-gatk.vcf
```

1. Start the GATK Docker container as described above in Basic Protocol 1:

```
docker run -t -i -v $PWD:/gatk/MouseMutations
broadinstitute/gatk:4.0.11.0 /bin/bash
myProject=/gatk/MouseMutations
```

2. Apply the code below. The output file, "`filtered-gatk.vcf`," will be identical to that at the end of Basic Protocol 2, except different variants will have a "`PASS`" flag, and the new filter name is "`myComplexFilter`." Type:

```
/gatk/gatk VariantFiltration -R $myProject/
musRefs_10/ref_mm10.fasta -V $myProject/toFilter-
gatk.vcf -O $myProject/filtered-gatk.vcf --filter-
expression "QD < 2.0 || FS > 60.0 || MQ < 40.0
|| MQRankSum < -12.5 || ReadPosRankSum < -8.0"
--filter-name "myComplexFilter"
```

Each flag above performs a QC step. `QD < 2` divides the variant confidence (QUAL) by unfiltered read depth of all model samples. `MQ < 40.0` accesses mapping quality across all samples. `FS > 60` performs a Fisher's exact test to detect strand bias. `MQRankSum < --13.5` tests differences in mapping qualities between reference and alternate alleles using a z-approximation of a Mann-Whitney U-test. `ReadPosRankSum < --8.0` applies this test to the distance of an alternative allele to the end of a read to screen false positives in heterozygous alleles.

The "||" symbol is a logical OR, indicating that a `myComplexFilter` flag will be entered for a variant if any of the above conditions are met. Additional information can be found via GATK documentation: *https://software.broadinstitute.org/gatk/documentation/article?id=11069*.

3. Exit the GATK Docker container:

```
exit
```

COMMENTARY

Background Information

Mutation detection begins with the alignment of NGS reads to the mouse reference genome (see Fig. 1). Although there are over three billion bases in the mouse genome, reads of length ≥25 are typically sufficient to cover the entire mouse genome. Short single-end reads are at a disadvantage, as they may contain a run of bases that are homologous to various regions of the genome, and as such can be placed at various regions of the genome, producing erroneous mapped regions that may have a downstream erroneous effect. The more popular paired-end reads contain a central spacer of approximately 125 bases and are mapped to positions where both reads are placed together. This placement reduces the problem of numerous mapping events due to homologous regions and increases the number of unique mapping events even when there are mismatches or gaps (Chen et al., 2012). The number of reads that map to a given genomic location is termed read depth or coverage. Sequence variants in the aligned genome are deviations from the reference sequence; these range from small differences including SNPs and indels to the larger and complicated variations including inversions and translocations. In a sequence alignment, heterozygous variations manifest as positions where approximately half of the reads match the reference and the other reads differ from the reference, although the theory rarely matches reality.

There has been a lot of debate on how much read depth or coverage is required to successfully call a small sequence variant such as a SNP or small indel. A sequencing run performed by one of the large sequencing companies such as Illumina (Goodwin, McPherson, & McCombie, 2016) generates reads randomly, but they are not distributed equally across the genome. Instead, some regions of the genome are better covered by sequence reads than others (Sims, Sudbery, Ilott, Heger, & Ponting, 2014). For instance, repetitive DNA regions are poorly covered because they can generate multiple, very similar short sequence reads, and the mapping of these regions may puzzle the assembly program, resulting in no or ambiguous mapping and erroneous alignments. Thus, the more frequently a base is sequenced, the more reliable that base is. Sequence variant detection with <20× read depth is commonplace (Potter et al., 2016), as is multiplexing samples with a shallow read depth (Bull et al., 2013). However, for detecting homozygous SNPs, it is advisable to have a read depth of 15×, and heterozygous SNPs should have a read depth of 33× (Bentley et al., 2008). Alongside the read depth are other parameters that are important to mutation detection, including base-mapping quality and the mapping quality of the surrounding sequence. Many SNP callers have built-in scoring systems to distinguish a genuine SNP from an error. For example, when GATK encounters a region of high variation, it reassembles mapped reads in that region, and then uses a hidden Markov model to determine the likelihood of haplotypes (and then alleles) for each variant site. By default, filters are included based on mapping quality and read quality, specifically according to the Phred score of each called base and read depth. Filtering by read depth ensures that a sufficient number of reads were aligned to each base. Phred scores come precalculated by sequencing machines based on a standard reference recovery rate for the specific chemistry of any given sequencer. These scores are encoded in FASTQ files directly, and allow filtering by the log likelihood of miscalling any given base. Furthermore, filtration programs (see Support Protocol 2) filter for strand bias, mapping quality, and sequencing artifacts, including combination scores such as QD (quality by depth), which incorporates a Phred-scaled probability that samples are homozygous for a reference allele and read depth.

When calling variants, two critical choices may affect results: the choice of the transcript set to use (Ensembl, RefSeq, UCSC) and the choice of variant calling software (SAMtools or GATK, for example). As many bioinformatics tools accept Ensembl transcript IDs as input for further analysis, and McCarthy et al. (2014) found Ensembl to outperform Refseq when discovering loss of function variants, we recommend using Ensembl transcripts and have done so in this article. GATK's realignment of variant-rich regions produces a higher positive predictive value than SAMtools; thus, it is recommended for variant detection in mouse, especially when multiple variants may be expected (Pirooznia et al., 2014). There are many variant annotators freely available to annotate SNPs. Among the most popular are VEP and Annovar (Wang, Li, & Hakonarson, 2010). Discrepancies in definitions of sequence variants partially explain the difference in annotation recovery between VEP and Annovar (McCarthy et al., 2014). VEP currently annotates sequences using, among other tools, the Sequence Ontology, which allows the specific variant types used by VEP to be consistent across experiments and databases (Eilbeck et al., 2005). Lastly, VEP was chosen because of the Docker implementation. Docker enables software to be managed easily and updated within a standard environment; this will facilitate expanding this article as newer variant detection algorithms become available in the future.

It is important to note that this protocol is optimized for detecting germline mutations. When detecting and annotating somatic (cancer) mutations, different tools should be used to take into account extreme aneuploidy and incorporate a "panel of normals." Such a panel is taken from normal (noncancerous) tissue sequenced with the same technology as the mutant data. It seeks to capture technical artifacts and thus improve the accuracy of variant calling.

Troubleshooting

Troubleshooting is summarized in Table 4. For additional problems, each tool installed (BWA, FASTQC, SAMTools, GATK, VEP) has documentation and can be run with "`--help`" flags.

The Java Virtual Machine must be assigned memory to handle large datasets encountered in mouse genome experiments. If, when executing any GATK, Picard Tools, or FASTQC programs an error occurs, which references an out of memory error, for example:

```
Exception in thread "main"
java.lang.OutOfMemoryError:
Java heap space
```

then an additional argument must be made to each call to the program in question. Instead of typing:

```
/gatk/gatk HaplotypeCaller
... {other options}
```

type:

```
/gatk/gatk --java-options
"-Xmx4G" HaplotypeCaller
... {other options}
```

The "`-Xmx4G`" option allocates 4 GB of RAM to the Java Virtual Machine. This option can be increased to fit the workstation memory limits and the size of the datasets being analyzed.

GATK is called with a wrapper script. To fix this issue with any other program directly called by the Java program itself, use the following syntax:

```
java -jar program.jar
[program arguments]
```

Understanding Results

The anticipated results are ultimately reports from VEP, as described in Basic Protocol 4, yielding both a text file of raw results and an html file of pre-made summary statistics. SNVs and structural variations may be identified. Within the text file of annotations, variations will be annotated with known rsIDs, Ensembl gene IDs for any intergenic variations found, and possible consequences of each variation. This may later be mined for more information relating to variants themselves, or be combined with previous knowledge to inform systems biology studies.

Time Considerations

Each protocol varies in time of execution. Once Support Protocol 1 (installation of software) is completed, Basic Protocol 1 is the largest consumer of time. Indexing the mouse genome takes several hours on one processor. By multithreading in a server environment, the alignment steps in Basic Protocol 1 can be reduced from days (when many replicates are involved) to hours. Basic Protocol 2 and Support Protocol 2 likewise take several hours; with multithreading via Spark, GATK

Table 4 Troubleshooting Guide for Mutation Detection in Mouse[a]

Problem	Possible cause	Solution
Basic Protocol 4 produces multiple results	VEP was run on a multisample VCF	Use GATK to combine each VCF before running Basic Protocol 4, or run as separate processes
No variants flagged as "`PASS`" in GATK	Low coverage	Pool samples of low coverage together if appropriate via the HaplotypeCaller and GenotypeGVCF
Command not found	Typos	Re-check typing in command prompt
Java heap space error	Low Java memory	Run Java command with "`Xmx4G`" flag
Java heap space error	Low Docker container memory	Run Docker with:`docker run -m=4g {imageID}` where 4g is 4 GB; increase this as needed
Files not found	`$myProject/ $mySoftware` not initialized	Re-initialize bash variables as in Basic Protocol 1
Error response from daemon; bad response from Docker engine	Docker engine not started	Start Docker. Mac OS: run open `--background -a Docker`. Linux: `sudo systemctl start docker`

[a]For help with specific problems not covered, check the documentation for each program. Always make sure to check for typographical errors, and be sure that each command is executed in the proper working directory (folder). Lastly, make sure that any bash variables such as "`myProject`" are initialized.

filtering steps can be completed in 2 h for small samples. As software is constantly improving, speeds are expected to greatly improve with increased multithreading support. Time of execution of Basic Protocols 3 and 4 largely depend on the number of novel and previously known variants found, but it is expected that this step will be completed in a matter of hours. Interpretation of variants and identifying variants with biological meaning in the context of any given experiment may take time depending on a researchers' domain expertise.

Acknowledgments

Research reported in this publication was supported by the Medical Research Council (MC_U142684171) and National Human Genome Research Institute of the National Institutes of Health (UM1HG006370). The authors also would like to Gareth Banks for help on the breeding scheme.

Conflicts of Interest

The authors declare no conflicts of interest. The funding organizations had no role in the design of this study, data collection, analysis or interpretation, or preparation of the manuscript, and did not approve, disapprove, or delay publication of the work.

Literature Cited

Adams, D. J., Doran, A. G., Lilue, J., & Keane, T. M. (2015). The Mouse Genomes Project: A repository of inbred laboratory mouse strain genomes. *Mammalian Genome: Official Journal of the International Mammalian Genome Society*, *26*(9–10), 403–412. doi: 10.1007/s00335-015-9579-6.

Andews, S. (2015). FastQC: A quality control tool for high throughput sequence data (version 0.11.4). Retrieved from *https://www.bioinformatics.babraham.ac.uk/projects/fastqc/*.

Balling, R. (2001). ENU mutagenesis: Analyzing gene function in mice. *Annual Review of Genomics and Human Genetics*, *2*(1), 463–492. doi: 10.1146/annurev.genom.2.1.463.

Bennett, S. (2004). Solexa Ltd. *Pharmacogenomics*, *5*(4), 433–438. doi: 10.1517/14622416.5.4.433.

Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., ... Smith, A. J. (2008). Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, *456*(7218), 53–59. doi: 10.1038/nature07517.

Bull, K. R., Rimmer, A. J., Siggs, O. M., Miosge, L. A., Roots, C. M., Enders, A., ... Cornall, R. J. (2013). Unlocking the bottleneck in forward genetics using whole-genome sequencing and identity by descent to isolate causative mutations. *PLOS Genetics*, *9*(1), e1003219. doi: 10.1371/journal.pgen.1003219.

Chen, Y., Negre, N., Li, Q., Mieczkowska, J. O., Slattery, M., Liu, T., ... Liu, X. S. (2012). Systematic evaluation of factors influencing ChIP-seq fidelity. *Nature Methods*, *9*(6), 609–614. doi: 10.1038/nmeth.1985.

Concepcion, D., Seburn, K. L., Wen, G., Frankel, W. N., & Hamilton, B. A. (2004). Mutation rate and predicted phenotypic target sizes in ethylnitrosourea-treated mice. *Genetics*, *168*(2), 953–959. doi: 10.1534/genetics.104.029843.

Eilbeck, K., Lewis, S. E., Mungall, C. J., Yandell, M., Stein, L., Durbin, R., & Ashburner, M. (2005). The Sequence Ontology: A tool for the unification of genome annotations. *Genome Biology*, *6*(5), R44. doi: 10.1186/gb-2005-6-5-r44.

Fabbro, C. D., Scalabrin, S., Morgante, M., & Giorgi, F. M. (2013). An extensive evaluation of read trimming effects on illumina NGS data analysis. *PLoS One*, *8*(12), e85024. doi: 10.1371/journal.pone.0085024.

Goodwin, S., McPherson, J. D., & McCombie, W. R. (2016). Coming of age: Ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, *17*(6), 333–351. doi: 10.1038/nrg.2016.49.

Justice, M. J., Noveroske, J. K., Weber, J. S., Zheng, B., & Bradley, A. (1999). Mouse ENU mutagenesis. *Human Molecular Genetics*, *8*(10), 1955–1963. doi: 10.1093/hmg/8.10.1955.

Korlach, J., Gedman, G., Kingan, S. B., Chin, C.-S., Howard, J. T., Audet, J.-N., ... Jarvis, E. D. (2017). De novo PacBio long-read and phased avian genome assemblies correct and add to reference genes generated with intermediate and short reads. *GigaScience*, *6*(10), 1–16. doi: 10.1093/gigascience/gix085.

Krueger, F. (2019). Trim Galore (version 0.6.0). Retrieved from *https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/*.

Leinonen, R., Sugawara, H., & Shumway, M., & International Nucleotide Sequence Database Collaboration. (2011). The sequence read archive. *Nucleic Acids Research*, *39*(Database issue), D19–D21. doi: 10.1093/nar/gkq1019.

Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, *25*(14), 1754–1760. doi: 10.1093/bioinformatics/btp324.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., & Homer, N. ... 1000 Genome Project Data Processing Subgroup. (2009). The sequence alignment/map format and SAMtools. *Bioinformatics*, *25*(16), 2078–2079. doi: 10.1093/bioinformatics/btp352.

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.Journal*, *17*(1), 10–12. doi: 10.14806/ej.17.1.200.

McCarthy, D. J., Humburg, P., Kanapin, A., Rivas, M. A., Gaulton, K., & Cazier, J.-B. ... The WGS500 Consortium. (2014). Choice of transcripts and software has a large effect on variant annotation. *Genome Medicine*, *6*(3), 26. doi: 10.1186/gm543.

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., ... DePristo, M. A. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, *20*(9), 1297–1303. doi: 10.1101/gr.107524.110.

McLaren, W., Gil, L., Hunt, S. E., Riat, H. S., Ritchie, G. R. S., Thormann, A., ... Cunningham, F. (2016). The Ensembl variant effect predictor. *Genome Biology*, *17*(1), 122. doi: 10.1186/s13059-016-0974-4.

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, *2014*(239). Retrieved from *http://dl.acm.org/citation.cfm?id=2600239.2600241*.

Mianné, J., Chessum, L., Kumar, S., Aguilar, C., Codner, G., Hutchison, M., ... Bowl, M. R. (2016). Correction of the auditory phenotype in C57BL/6N mice via CRISPR/Cas9-mediated homology directed repair. *Genome Medicine*, *8*(1), 16.

Mouse Genome Sequencing Consortium. (2002). Initial sequencing and comparative analysis of the mouse genome. *Nature*, *420*(6915), 520–562. doi: 10.1038/nature01262.

O'Brien, T. P., & Frankel, W. N. (2004). Moving forward with chemical mutagenesis in the mouse. *The Journal of Physiology*, *554*(1), 13–21. doi: 10.1113/jphysiol.2003.049494.

Panda, S. K., Wefers, B., Ortiz, O., Floss, T., Schmid, B., Haass, C., ... Kühn, R. (2013). Highly efficient targeted mutagenesis in mice using TALENs. *Genetics*, *195*(3), 703–713. doi: 10.1534/genetics.113.156570.

Park, S. T., & Kim, J. (2016). Trends in next-generation sequencing and a new Era for

whole genome sequencing. *International Neurology Journal*, *20*(Suppl 2), S76–S83. doi: 10.5213/inj.1632742.371.

Picard Toolkit. (2018). Retrieved from *http://broadinstitute.github.io/picard/*.

Pirooznia, M., Kramer, M., Parla, J., Goes, F. S., Potash, J. B., McCombie, W. R., & Zandi, P. P. (2014). Validation and assessment of variant calling pipelines for next-generation sequencing. *Human Genomics*, *8*, 14. doi: 10.1186/1479-7364-8-14.

Potter, P. K., Bowl, M. R., Jeyarajan, P., Wisby, L., Blease, A., Goldsworthy, M. E., . . . Brown, S. D. M. (2016). Novel gene function revealed by mouse mutagenesis screens for models of age-related disease. *Nature Communications*, *7*, 12444. doi: 10.1038/ncomms12444.

Qian, L., Mahaffey, J. P., Alcorn, H. L., & Anderson, K. V. (2011). Tissue-specific roles of Axin2 in the inhibition and activation of Wnt signaling in the mouse embryo. *Proceedings of the National Academy of Sciences of the United States of America*, *108*(21), 8692–8697. doi: 10.1073/pnas.1100328108.

Sengupta, S., Bolin, J. M., Ruotti, V., Nguyen, B. K., Thomson, J. A., Elwell, A. L., & Stewart, R. (2011). Single read and paired end mRNA-Seq Illumina libraries from 10 nanograms total RNA. *Journal of Visualized Experiments*, *56*, e3340. doi: 10.3791/3340.

Shendure, J. A., Porreca, G. J., & Church, G. M. (2008). Overview of DNA sequencing strategies. *Current Protocols in Molecular Biology*, *81*, 7.1.1–7.1.11. doi: 10.1002/0471142727.mb0701s81.

Sims, D., Sudbery, I., Ilott, N. E., Heger, A., & Ponting, C. P. (2014). Sequencing depth and coverage: Key considerations in genomic analyses. *Nature Reviews Genetics*, *15*(2), 121–132. doi: 10.1038/nrg3642.

Wang, K., Li, M., & Hakonarson, H. (2010). ANNOVAR: Functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Research*, *38*(16), e164. doi: 10.1093/nar/gkq603.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, June 22–25, Boston, MA. Retrieved from *http://dl.acm.org/citation.cfm?id=1863103.1863113*.

Key References

Butkiewicz, M., & Bush, W. S. (2016). In silico functional annotation of genomic variation. *Current Protocols in Human Genetics*, *88*(1), 6.15.1–6.15.17. doi: 10.1002/0471142905.hg0615s88.

Buteiwcz and Bush (2016) provide a detailed overview variant annotation, explaining why certain elements may be annotated for biological importance by means of the Sequence Ontology.

Internet Resources

https://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

For information on evaluating the QC status of your reads, the FastQC website has excellent tutorials outside the scope of this protocol. As of this writing date, the following tutorial video is available from the Babraham institute via YouTube: http://www.youtube.com/watch?v=bz93ReOv87Y.

https://www.ensembl.org/info/docs/tools/vep/index.html.

See the link above for access to VEP's web interface, which provides an alternative means of uploading a VCF file and retrieving results. For direct links to sources which VEP uses for annotating mouse variants, including a version number for each resource, see: https://www.ensembl.org/info/genome/variation/species/sources_documentation.html#mus_musculus.

https://software.broadinstitute.org/gatk/documentation/.

The user guide for GATK provides best practices, links to tool documentation, and information about the continual growth of the GATK resource.