QTL_BSA-Sorghum

Release 1.0

Michael Hall

May 26, 2022

Contents

We will performing a comprehensive analysis and processing of two Variant Files from two different species with particular traits of interest. Semi-dwarfism in Sorghum grass and Rice Cold Tolerance.

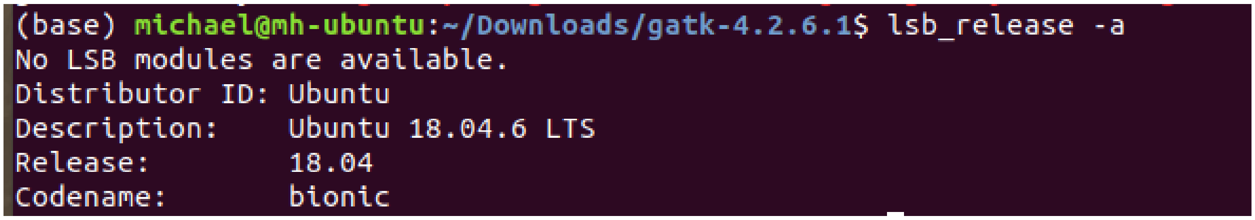
1 QTL_BSA_Crop_Varieties

Author Michael Hall

Date 4/13/2022

Before we begin I like to reveal what my machine specifications are just in case there might be a compatibility issue:

What open source opeating system are you running? Ubuntu 18.04, Code name Bionic, it must be a Tuesday



```
(base) michael@mh-ubuntu:~/Downloads/gatk-4.2.6.1$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS
Release:        18.04
Codename:       bionic
```

1.1 QTLSorghum

QTLseqr is an R package for QTL mapping using NGS Bulk Segregant Analysis.

QTLseqr is still under development and is offered with out any guarantee.

For more detailed instructions please read the vignettehere

For updates read the NEWS.md

1.2 Installation

You can install QTLseqr from github with:

```r
# install devtools first to download packages from github
install.packages("devtools")

# use devtools to install QTLseqr
devtools::install_github("PBGLMichaelHall/QTLseqr")
```

Package Dependencies

Note: Apart from regular package dependencies, there are some Bioconductor tools that we use as well, as such you will be prompted to install support for Bioconductor, if you haven't already. QTLseqr makes use of C++ to make some tasks significantly faster (like counting SNPs). Because of this, in order to install QTLseqr from github you will be required to install some compiling tools (Rtools and Xcode, for Windows and Mac, respectively).

1.3 Citation

If you use QTLseqr in published research, please cite:

> Mansfeld B.N. and Grumet R, QTLseqr: An R package for bulk segregant analysis with next-generation sequencing *The Plant Genome* doi:10.3835/plantgenome2018.01.0006

We also recommend citing the paper for the corresponding method you work with.

QTL-seq method:

> Takagi, H., Abe, A., Yoshida, K., Kosugi, S., Natsume, S., Mitsuoka, C., Uemura, A., Utsushi, H., Tamiru, M., Takuno, S., Innan, H., Cano, L. M., Kamoun, S. and Terauchi, R. (2013), QTL-seq: rapid mapping of quantitative trait loci in rice by whole genome resequencing of DNA from two bulked populations. *Plant J*, 74: 174–183. doi:10.1111/tpj.12105

G prime method:

> Magwene PM, Willis JH, Kelly JK (2011) The Statistics of Bulk Segregant Analysis Using Next Generation Sequencing. *PLOS Computational Biology* 7(11): e1002255. doi.org/10.1371/journal.pcbi.1002255

Abstract

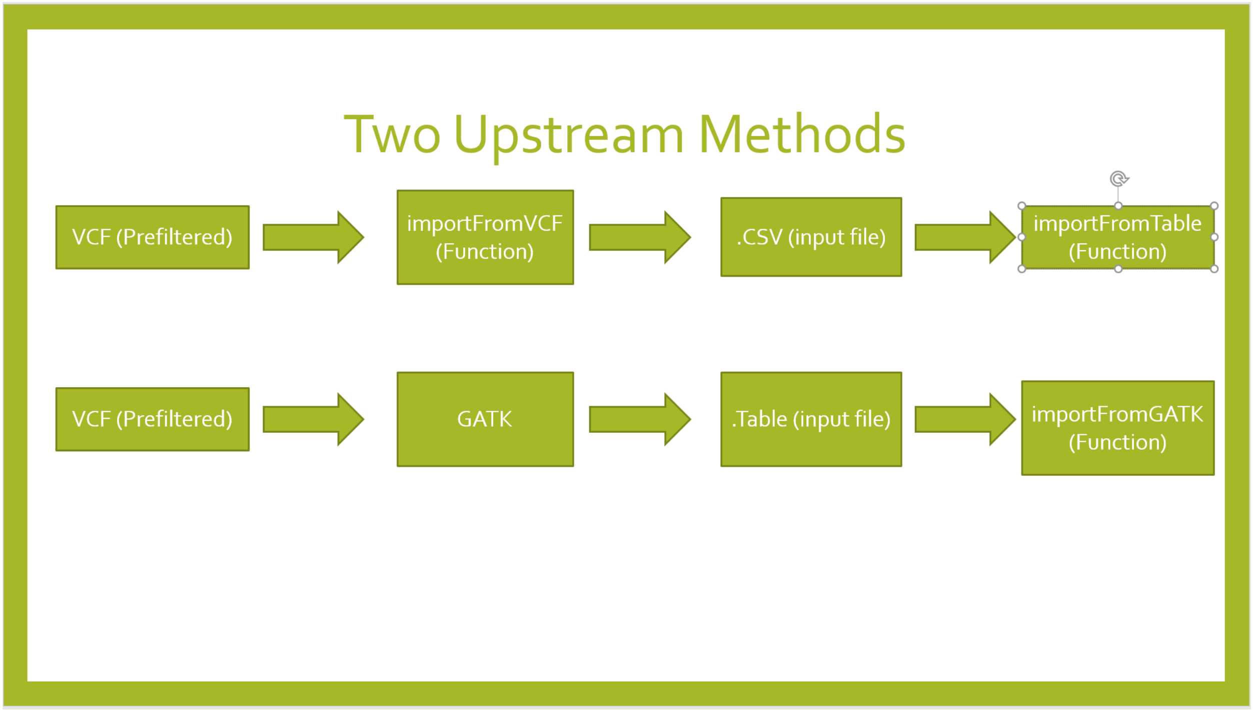
Next Generation Sequencing Bulk Segregant Analysis (NGS-BSA) is efficient in detecting quantitative trait loci (QTL). Despite the popularity of NGS-BSA and the R statistical platform, no R packages are currently available for NGS-BSA. We present QTLseqr, an R package for NGS-BSA that identifies QTL using two statistical approaches: QTL-seq and G'. These approaches use a simulation method and a tricube smoothed G statistic, respectively, to identify and assess statistical significance of QTL. QTLseqr, can import and filter SNP data, calculate SNP distributions, relative allele frequencies, G' values, and log10(p-values), enabling identification and plotting of QTL.

1.4 Examples:

Load/install libraries

```
devtools::install_github(“PBGLMichaelHall/QTLseqr”,force = TRUE)
install.packages(“vcfR”)
install.packages(“tidyr”)
install.packages(“ggplot2”)

library(QTLseqr)
library(vcfR)
library(tidyr)
library(ggplot2)
```

```
**Methods**
```



Two Upstream Methods

VCF (Prefiltered) → importFromVCF (Function) → .CSV (input file) → importFromTable (Function)

VCF (Prefiltered) → GATK → .Table (input file) → importFromGATK (Function)

Set the Working Directory

```
setwd("/home/michael/Desktop/QTLseqr/extdata")
```

1.5 Pre-Filtering Rules

```
Vcf file must only contain bialleleic variants. (filter upstream, e.g., with bcftools␣
↪view -v snps -m2 -M2), also the QTLseqR functions will only take    SNPS, ie, length␣
↪of REF and ALT== 1
```

1.6 Importing Data

importFromVCF

```
df <- importFromVCF(file = "freebayes_D2.filtered.vcf", highBulk = "D2_F2_tt", lowBulk =␣
↪   "D2_F2_TT", filname = "Hall")
```

importFromGATK

An offical Github GATK Genomic Analysis Toolkit repository can be found here to download https://github.com/broadinstitute/gatk

However, we want to clone the repository and make a build:

```
git clone https://github.com/broadinstitute/gatk



**Navigate to find gradlew and type the command:**
```

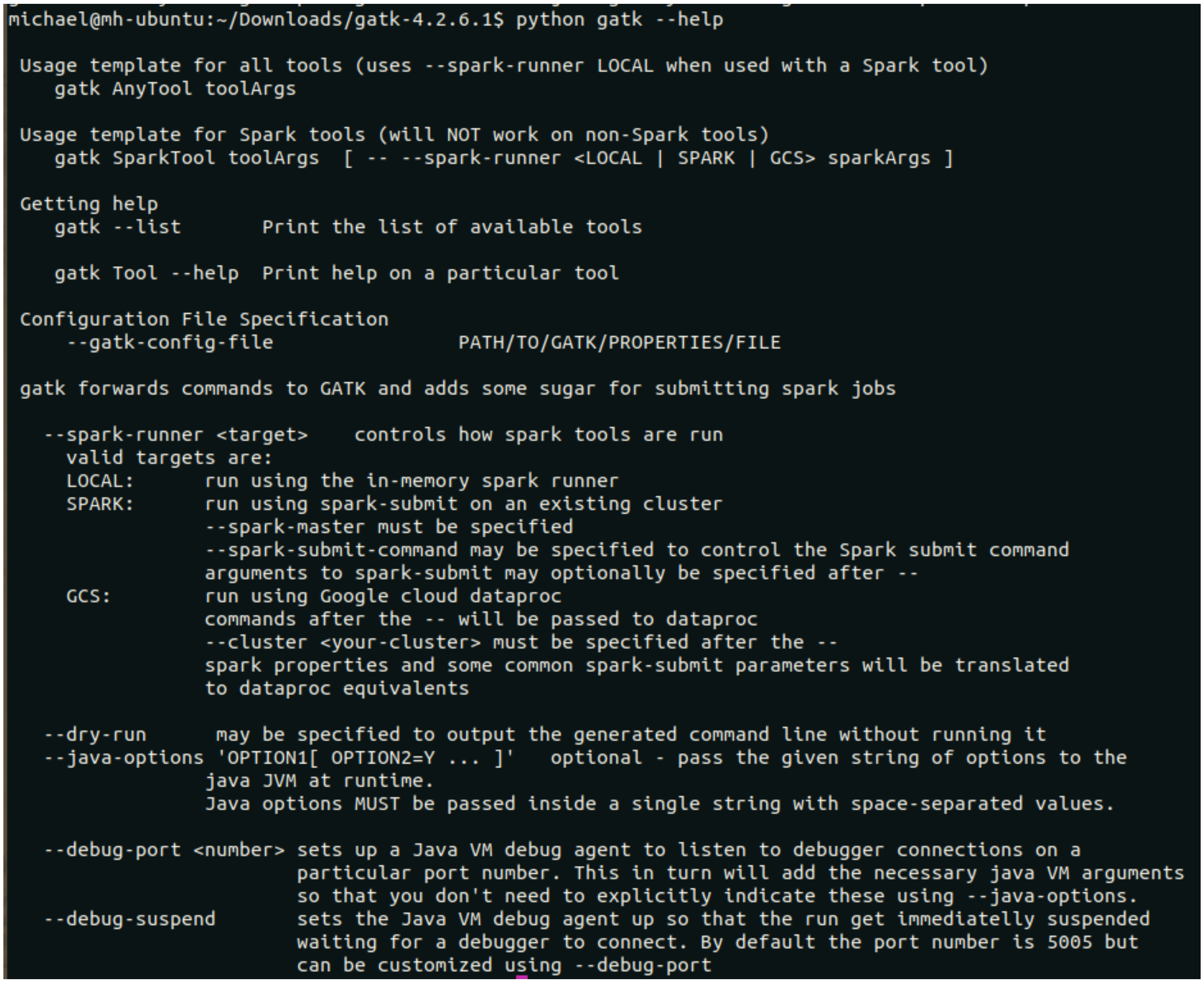
```
gradlew bundle



**To verify it is working invoke python interpreter:**
```

```
python gatk --help
```

```
python gatk --list
```

```
*Base Calling:*
*Copy Number Variant Discovery:*
*Coverage Analysis:*
*Diagnostics and Quality Control:*
*Example Tools:*
*Genotyping Arrays Manipulation:*
*Intervals Manipulation:*
*Metagenomics:*
```

(continues on next page)

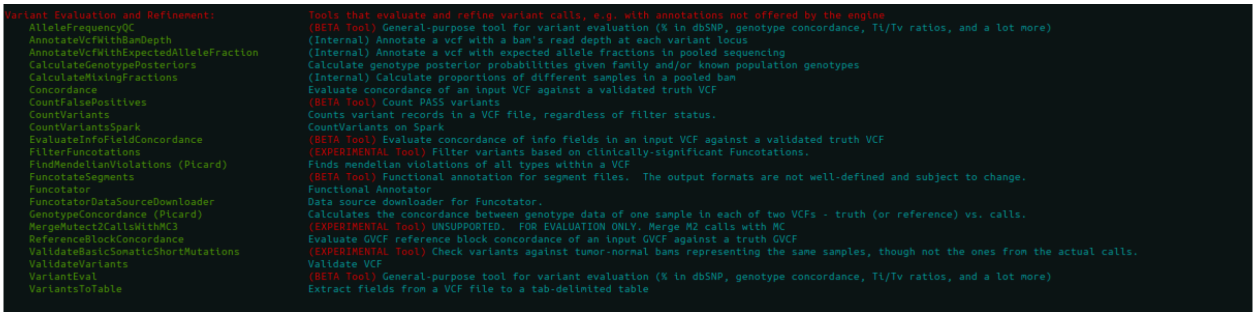


```
michael@mh-ubuntu:~/Downloads/gatk-4.2.6.1$ python gatk --help

 Usage template for all tools (uses --spark-runner LOCAL when used with a Spark tool)
    gatk AnyTool toolArgs

 Usage template for Spark tools (will NOT work on non-Spark tools)
    gatk SparkTool toolArgs  [ -- --spark-runner <LOCAL | SPARK | GCS> sparkArgs ]

 Getting help
    gatk --list       Print the list of available tools

    gatk Tool --help  Print help on a particular tool

 Configuration File Specification
     --gatk-config-file                PATH/TO/GATK/PROPERTIES/FILE

 gatk forwards commands to GATK and adds some sugar for submitting spark jobs

   --spark-runner <target>    controls how spark tools are run
     valid targets are:
     LOCAL:      run using the in-memory spark runner
     SPARK:      run using spark-submit on an existing cluster
                 --spark-master must be specified
                 --spark-submit-command may be specified to control the Spark submit command
                 arguments to spark-submit may optionally be specified after --
     GCS:        run using Google cloud dataproc
                 commands after the -- will be passed to dataproc
                 --cluster <your-cluster> must be specified after the --
                 spark properties and some common spark-submit parameters will be translated
                 to dataproc equivalents

   --dry-run      may be specified to output the generated command line without running it
   --java-options 'OPTION1[ OPTION2=Y ... ]'   optional - pass the given string of options to the
                 java JVM at runtime.
                 Java options MUST be passed inside a single string with space-separated values.

   --debug-port <number> sets up a Java VM debug agent to listen to debugger connections on a
                         particular port number. This in turn will add the necessary java VM arguments
                         so that you don't need to explicitly indicate these using --java-options.
   --debug-suspend       sets the Java VM debug agent up so that the run get immediatelly suspended
                         waiting for a debugger to connect. By default the port number is 5005 but
                         can be customized using --debug-port
```

(continued from previous page)

```
*Methalation-Specific Tools:*
*Other:*
*Read Data Manipulation:*
*Reference:*
*Short Variant Discovery:*
*Structural Variant Discovery:*
*Variant Evaluation and Refinement:*
*Variant Filtering:*
*Variant Manipulation:*
```

# We are most concerned with **Variant Evaluation and Refinement**



```
Variant Evaluation and Refinement:               Tools that evaluate and refine variant calls, e.g. with annotations not offered by the engine
    AlleleFrequencyQC                            (BETA Tool) General-purpose tool for variant evaluation (% in dbSNP, genotype concordance, Ti/Tv ratios, and a lot more)
    AnnotateVcfWithBamDepth                      (Internal) Annotate a vcf with a bam's read depth at each variant locus
    AnnotateVcfWithExpectedAlleleFraction        (Internal) Annotate a vcf with expected allele fractions in pooled sequencing
    CalculateGenotypePosteriors                  Calculate genotype posterior probabilities given family and/or known population genotypes
    CalculateMixingFractions                     (Internal) Calculate proportions of different samples in a pooled bam
    Concordance                                  Evaluate concordance of an input VCF against a validated truth VCF
    CountFalsePositives                          (BETA Tool) Count PASS variants
    CountVariants                                Counts variant records in a VCF file, regardless of filter status.
    CountVariantsSpark                           CountVariants on Spark
    EvaluateInfoFieldConcordance                 (BETA Tool) Evaluate concordance of info fields in an input VCF against a validated truth VCF
    FilterFuncotations                           (EXPERIMENTAL Tool) Filter variants based on clinically-significant Funcotations.
    FindMendelianViolations (Picard)             Finds mendelian violations of all types within a VCF
    FuncotateSegments                            (BETA Tool) Functional annotation for segment files.  The output formats are not well-defined and subject to change.
    Funcotator                                   Functional Annotator
    FuncotatorDataSourceDownloader               Data source downloader for Funcotator.
    GenotypeConcordance (Picard)                 Calculates the concordance between genotype data of one sample in each of two VCFs - truth (or reference) vs. calls.
    MergeMutect2CallsWithMC3                     (EXPERIMENTAL Tool) UNSUPPORTED.  FOR EVALUATION ONLY. Merge M2 calls with MC
    ReferenceBlockConcordance                    Evaluate GVCF reference block concordance of an input GVCF against a truth GVCF
    ValidateBasicSomaticShortMutations           (EXPERIMENTAL Tool) Check variants against tumor-normal bams representing the same samples, though not the ones from the actual calls.
    ValidateVariants                             Validate VCF
    VariantEval                                  (BETA Tool) General-purpose tool for variant evaluation (% in dbSNP, genotype concordance, Ti/Tv ratios, and a lot more)
    VariantsToTable                              Extract fields from a VCF file to a tab-delimited table
```

Fig. 1: **To produce the input file Hall.table, run the following command:**

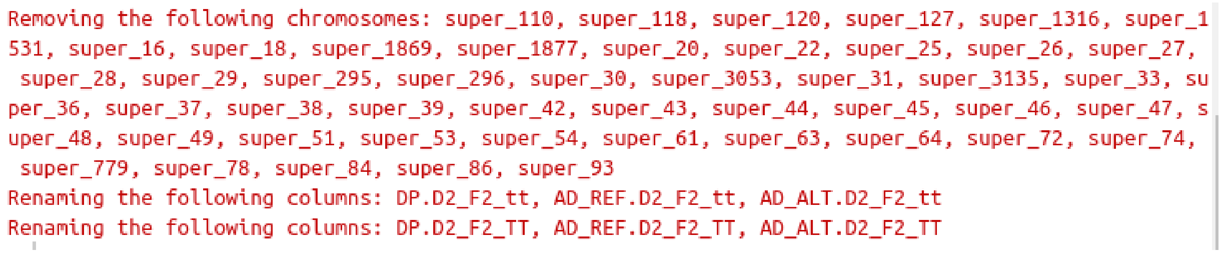
```
python gatk VariantsToTable --variant freebayes_D2.filtered.vcf --fields CHROM --fields POS --fields REF --fields ALT --genotyp-fields AD --genotype-fields DP --genotype-fields GQ --genotype-fields PL --output Hall.table
```

1.7 Input Fields ImportFromVCF

```
**Define High bulk and Low bulk sample names as an input object and define parser generated file name. The file name is generated from ImportFromVCF function.**

HighBulk <- "D2_F2_tt"
LowBulk <- "D2_F2_TT"
file <- "Hall.csv"

**Choose and define which chromosomes/contigs will be included in the analysis. The chromosome/contg names are reverse compatible with VCF names.**

Chroms <- c("Chr01","Chr02","Chr03","Chr04","Chr05","Chr06","Chr07","Chr08","Chr09","Chr10")
```

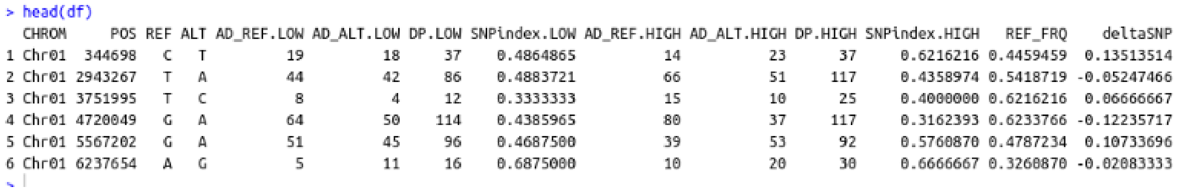
importFromTable

```
df <-
  importFromTable(
    file = file,
    highBulk = HighBulk,
    lowBulk = LowBulk,
    chromList = Chroms
  )
```



```
Removing the following chromosomes: super_110, super_118, super_120, super_127, super_1316, super_1
531, super_16, super_18, super_1869, super_1877, super_20, super_22, super_25, super_26, super_27,
 super_28, super_29, super_295, super_296, super_30, super_3053, super_31, super_3135, super_33, su
per_36, super_37, super_38, super_39, super_42, super_43, super_44, super_45, super_46, super_47, s
uper_48, super_49, super_51, super_53, super_54, super_61, super_63, super_64, super_72, super_74,
 super_779, super_78, super_84, super_86, super_93
Renaming the following columns: DP.D2_F2_tt, AD_REF.D2_F2_tt, AD_ALT.D2_F2_tt
Renaming the following columns: DP.D2_F2_TT, AD_REF.D2_F2_TT, AD_ALT.D2_F2_TT
```

Inspect Header



```
> head(df)
  CHROM     POS REF ALT AD_REF.LOW AD_ALT.LOW DP.LOW SNPindex.LOW AD_REF.HIGH AD_ALT.HIGH DP.HIGH SNPindex.HIGH   REF_FRQ    deltaSNP
1 Chr01  344698   C   T         19         18     37    0.4864865          14          23      37     0.6216216 0.4459459  0.13513514
2 Chr01 2943267   T   A         44         42     86    0.4883721          66          51     117     0.4358974 0.5418719 -0.05247466
3 Chr01 3751995   T   C          8          4     12    0.3333333          15          10      25     0.4000000 0.6216216  0.06666667
4 Chr01 4720049   G   A         64         50    114    0.4385965          80          37     117     0.3162393 0.6233766 -0.12235717
5 Chr01 5567202   G   A         51         45     96    0.4687500          39          53      92     0.5760870 0.4787234  0.10733696
6 Chr01 6237654   A   G          5         11     16    0.6875000          10          20      30     0.6666667 0.3260870 -0.02083333
>
```

1.8 Input Fields ImportFromGATK

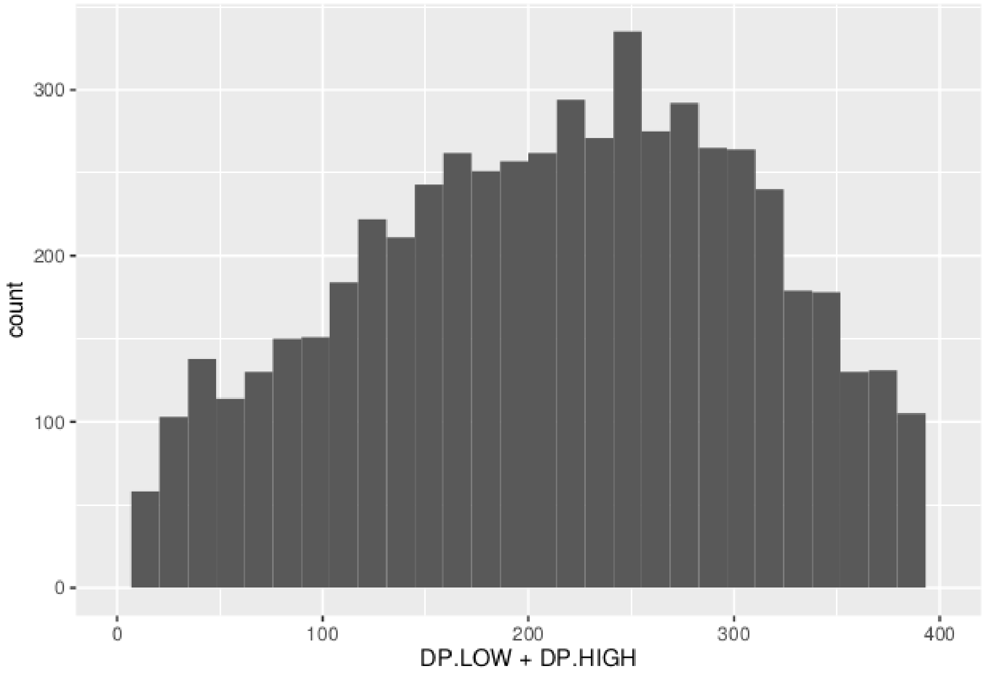
```
**Define Objects High bulk, Low bulk and file given there proper names.**

HighBulk <- "D2_F2_tt"
LowBulk <- "D2_F2_TT"
file <- "Hall.table"

**Choose which chromosomes/contigs will be included in the analysis.**

Chroms <- c("Chr01","Chr02","Chr03","Chr04","Chr05","Chr06","Chr07","Chr08","Chr09",
↪"Chr10")
```

importFromTable

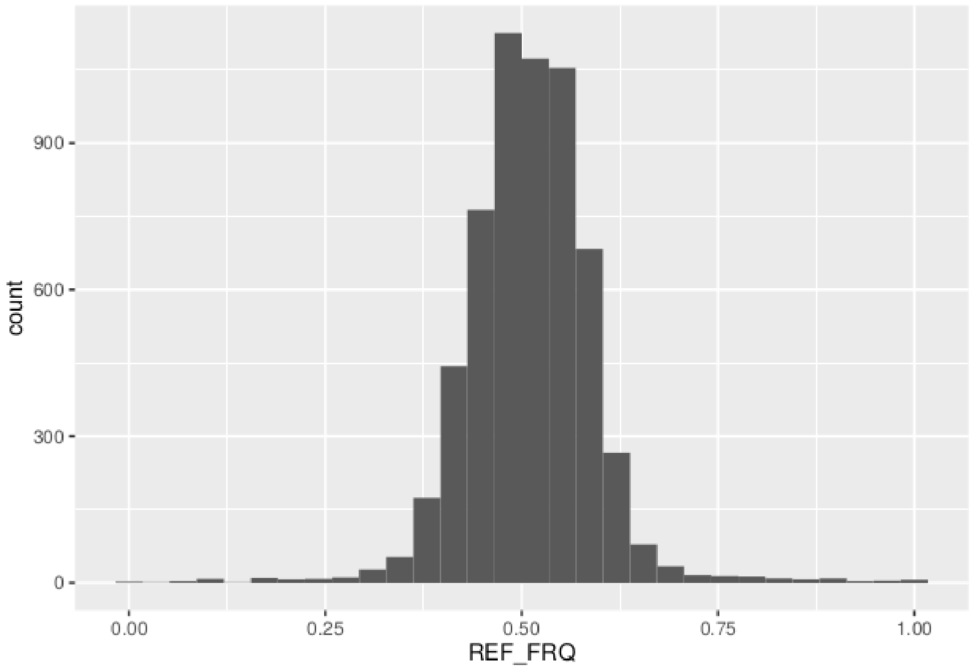
```
df <-
  importFromGATK(
    file = file,
    highBulk = HighBulk,
    lowBulk = LowBulk,
    chromList = Chroms
  )
```

Histograms

```
**Make histograms associated with filtering arguments. Such as Minimum Depth, Maximum␣
↪Depth, Reference Allele Frequency, Minimum Sample Depth, and Genotype Quality.

ggplot(data =df) + geom_histogram(aes(x = DP.LOW + DP.HIGH)) + xlim(0,400)


ggsave(filename = "Depth_Histogram.png",plot=last_plot())
```



```
ggplot(data = df) + geom_histogram(aes(x = REF_FRQ))
ggsave(filename = "Ref_Freq_Histogram.png",plot = last_plot())
```



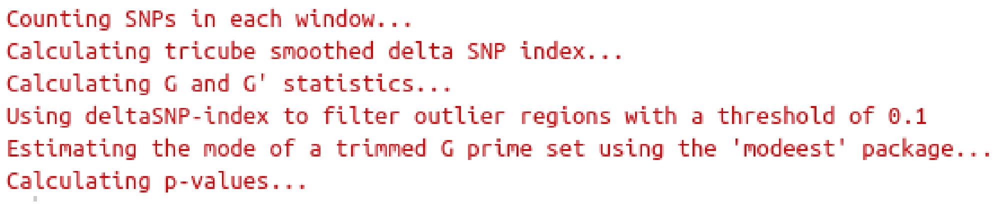
1.9 filterSNPs

```
**Filter SNPs:**
df_filt <- filterSNPs( SNPset = df,
refAlleleFreq = 0.20, minTotalDepth = 100, maxTotalDepth = 400,
minSampleDepth = 40,
minGQ = 0 )
```

```
Filtering by reference allele frequency: 0.2 <= REF_FRQ <= 0.8
...Filtered 75 SNPs
Filtering by total sample read depth: Total DP >= 100
...Filtered 733 SNPs
Filtering by total sample read depth: Total DP <= 400
...Filtered 175 SNPs
Filtering by per sample read depth: DP >= 40
...Filtered 22 SNPs
GQ columns not found. Skipping...
Original SNP number: 5906, Filtered: 1005, Remaining: 4901
```

1.10 runGprimeAnalysis_MH

```
**Run G' analysis:**

df_filt<-runGprimeAnalysis_MH(
  SNPset = df_filt,
  windowSize = 5000000,
  outlierFilter = "deltaSNP",
  filterThreshold = 0.1)
```



```
Counting SNPs in each window...
Calculating tricube smoothed delta SNP index...
Calculating G and G' statistics...
Using deltaSNP-index to filter outlier regions with a threshold of 0.1
Estimating the mode of a trimmed G prime set using the 'modeest' package...
Calculating p-values...
```

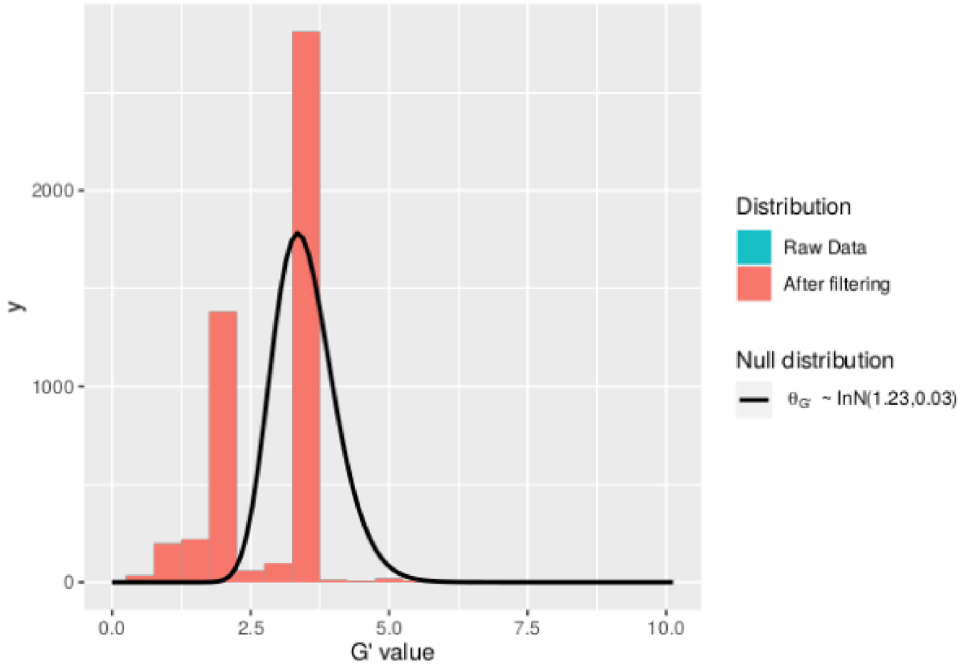
1.11 plotGprimeDist_MH

```
**The plot reveals a skewed G Prime statistic with a really small variance. Perhaps it␣
↪is due to relatively High Coverage with respect to Bulk Sample Sizes and not a lot of␣
↪variants called.**

**In addition, Hampels outlier filter in the second argument can also be changed to
↪"deltaSNP".**

plotGprimeDist(SNPset = df_filt, outlierFilter = "Hampel",filterThreshold = 0.1,␣
↪binwidth = 0.5)
```

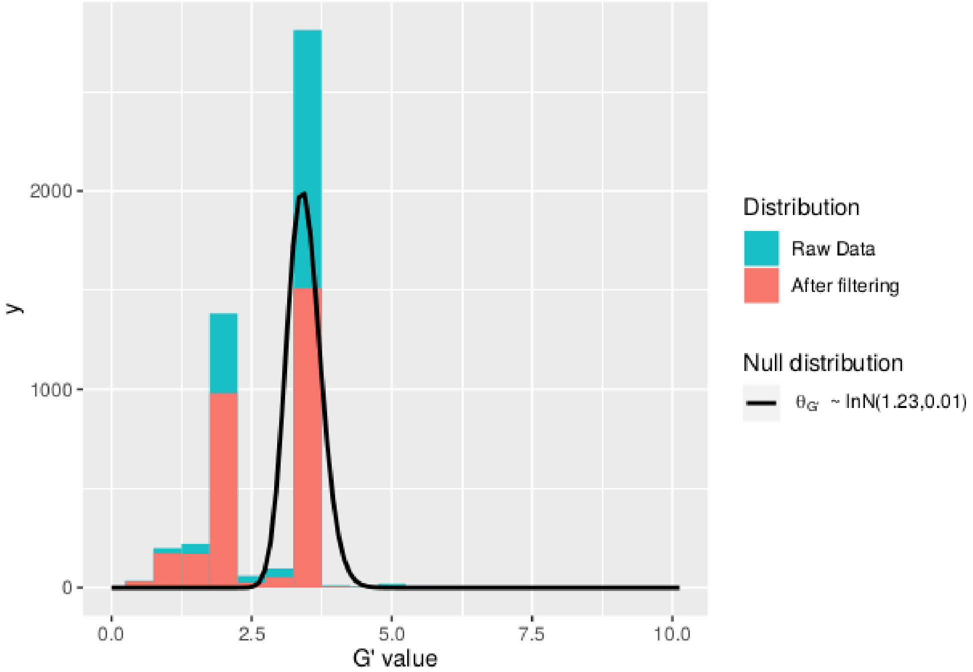
```
**We can see raw data before and after our filtering step**

plotGprimeDist_MH(SNPset = df_filt, outlierFilter = "deltaSNP",filterThreshold = 0.1,
↪binwidth=0.5)
```

1.12 runQTLseqAnalysis_MH

```
**Run QTLseq analysis:**


df_filt2 <- runQTLseqAnalysis_MH(
  SNPset = df_filt,
  windowSize = 5000000,
  popStruc = "F2",
  bulkSize = c(45, 38),
  replications = 10000,
  intervals = c(95, 99)
)
```



y
2000
1000
0
0.0
2.5
5.0
7.5
10.0
G' value
Distribution
🟦 Raw Data
🟥 After filtering
Null distribution
— $\theta_{G'}$ ~ lnN(1.23,0.03)

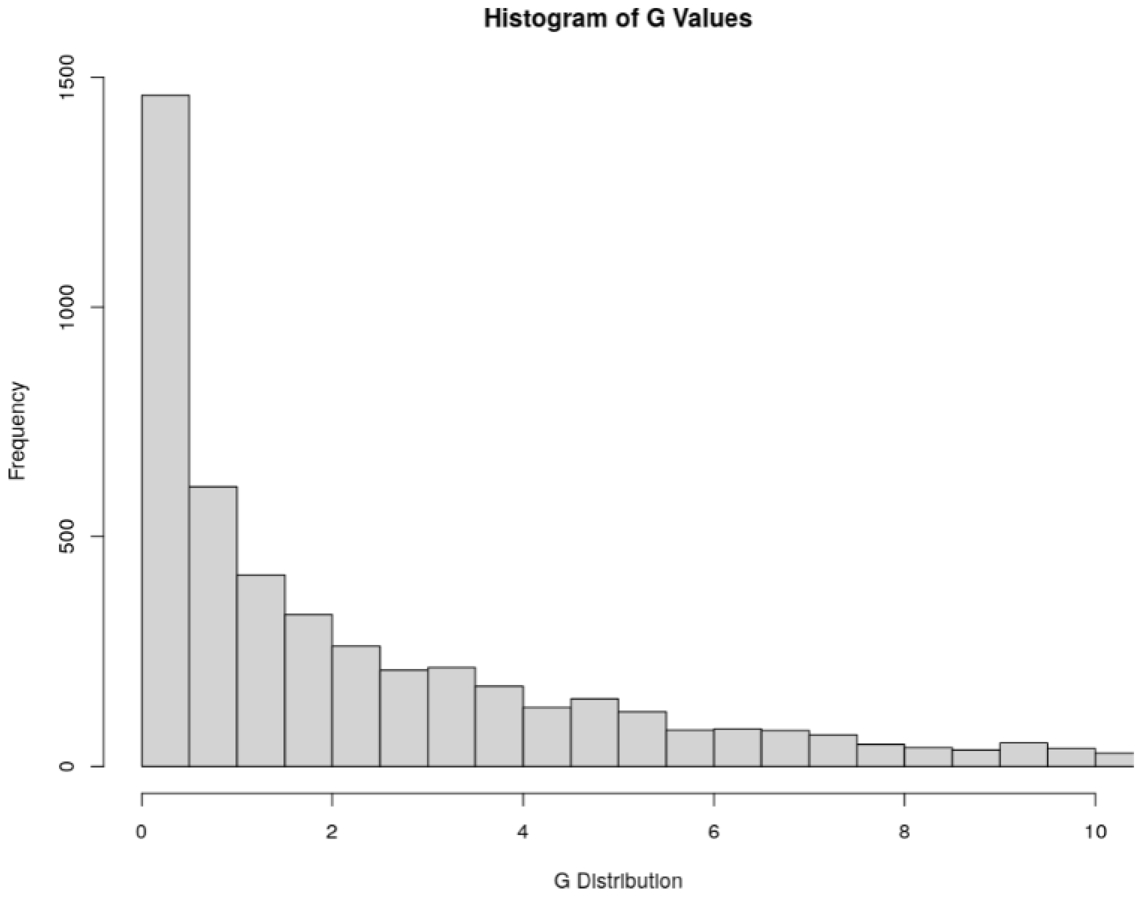


y
2000
1000
0
0.0
2.5
5.0
7.5
10.0
G' value
Distribution
🟦 Raw Data
🟥 After filtering
Null distribution
— $\theta_{G'}$ ~ lnN(1.23,0.01)

```
Counting SNPs in each window...
Calculating tricube smoothed delta SNP index...
Returning the following two sided confidence intervals: 95, 99
Variable 'depth' not defined, using min and max depth from data: 40-198
Assuming bulks selected from F2 population, with 45 and 38 individuals per bulk.
Simulating 10000 SNPs with reads at each depth: 40-198
Keeping SNPs with >= 0.3 SNP-index in both simulated bulks
Joining, by = "tricubeDP"
```

Plot G Statistic Distribution as a Histogram

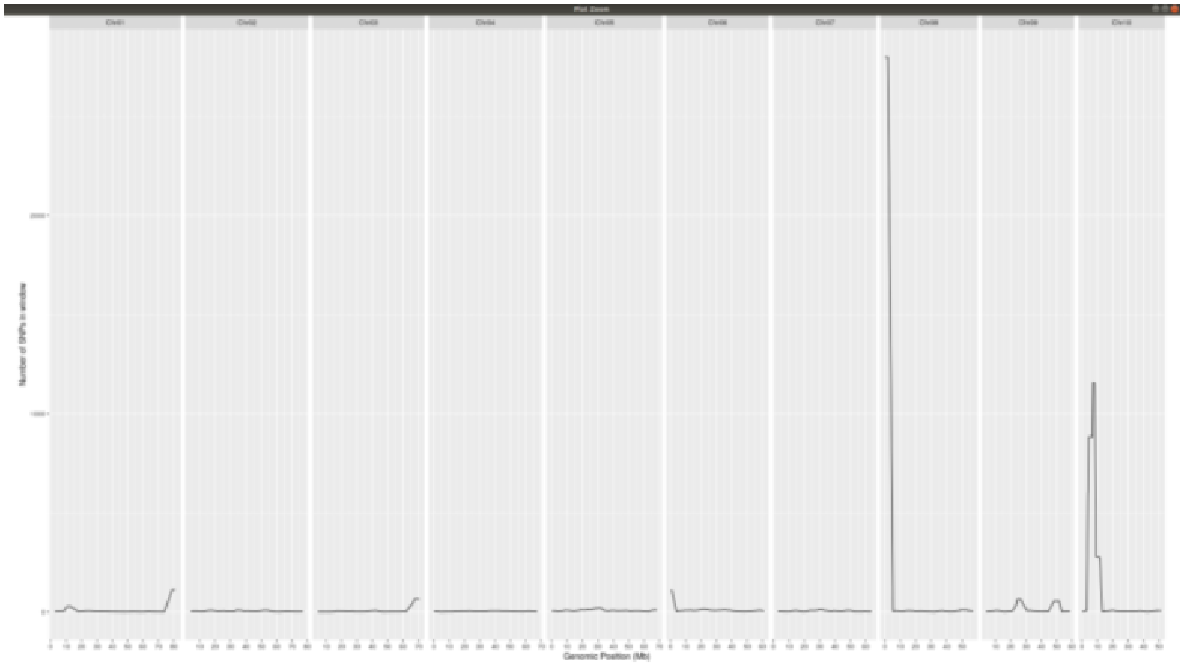
```
hist(df_filt2$G,breaks = 950,xlim = c(0,10),xlab = "G Distribution",main = "Histogram of␣
↪G Values")
```



1.13 plotQTLStats

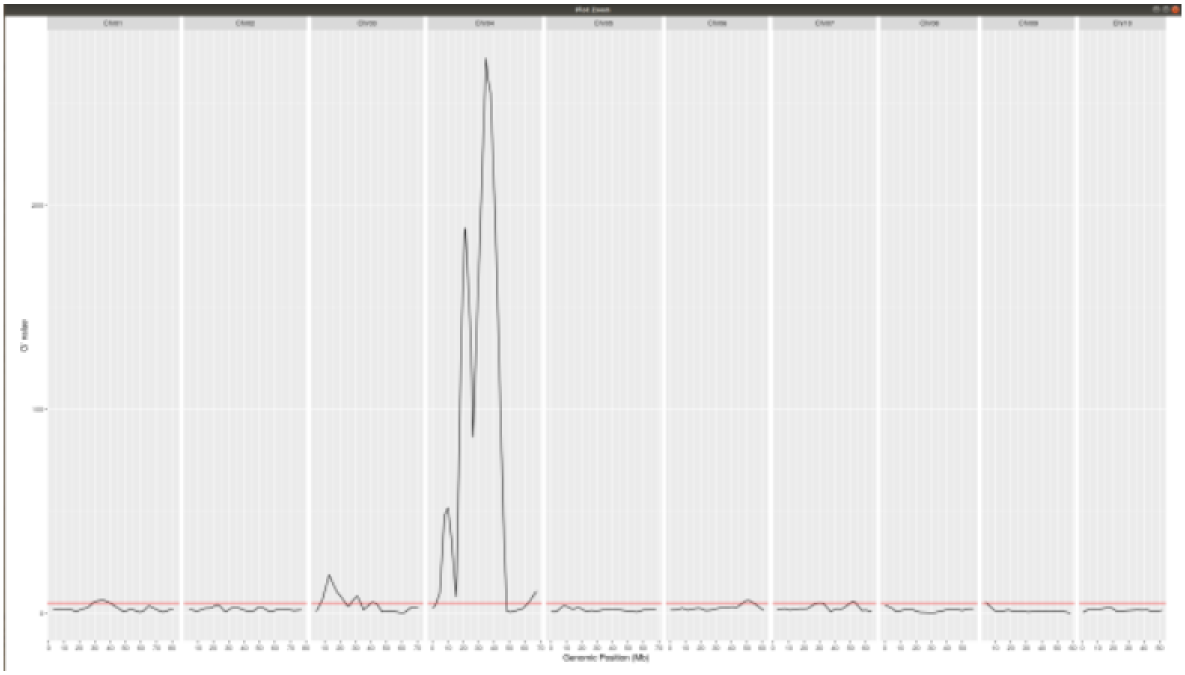
nSNPs

```
**Plot Snps as a function of chromosome and position values**


plotQTLStats(SNPset = df_filt2, var = "nSNPs")
ggsave(filename = "nSNPs.png",plot = last_plot())
```



Gprime

```
**Using QTLStats funciton plot Gprime Statistic with False Discovery Rate Threhshold as␣
↪a third argument boolean operator as TRUE. The q value is used as FDR threshold null␣
↪value is 0.05%.**


plotQTLStats(SNPset = df_filt, var = "Gprime", plotThreshold = TRUE, q = 0.01)
ggsave(filename = "GPrime.png",plot = last_plot())
```



deltaSNP

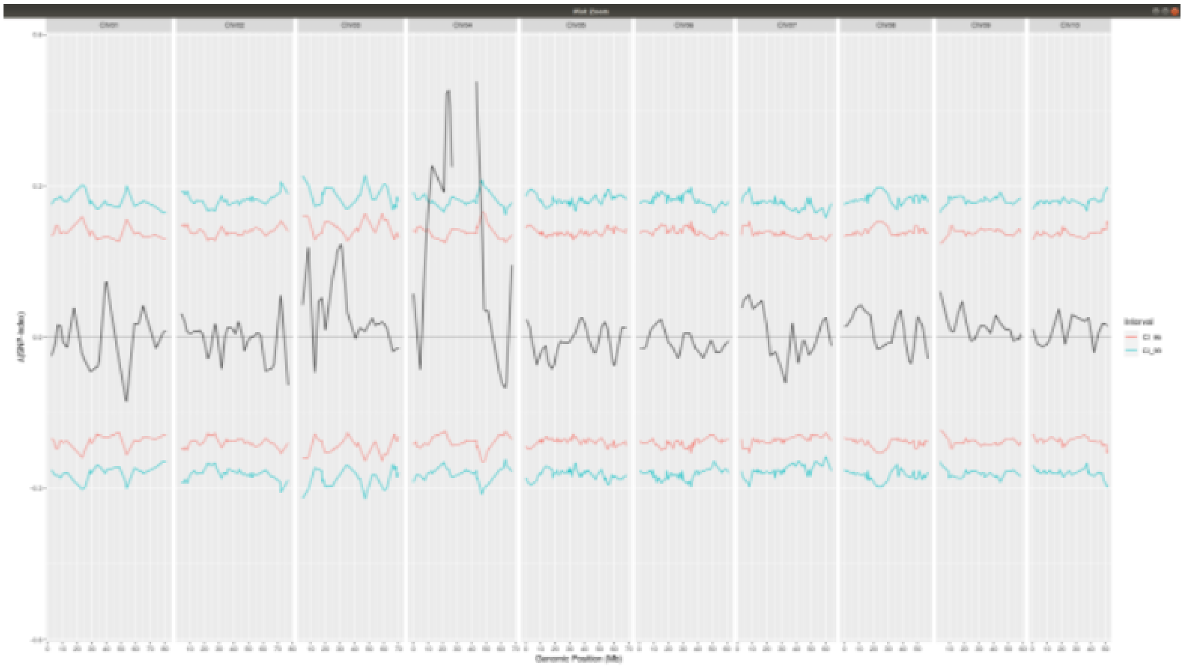
```
**Again using plotQTLStats change second argument varaible to deltaSNP and plot.**

plotQTLStats(SNPset = df_filt2, var = "deltaSNP", plotIntervals  = TRUE)
ggsave(filename = "DeltaSNPInterval.png",plot = last_plot())
```

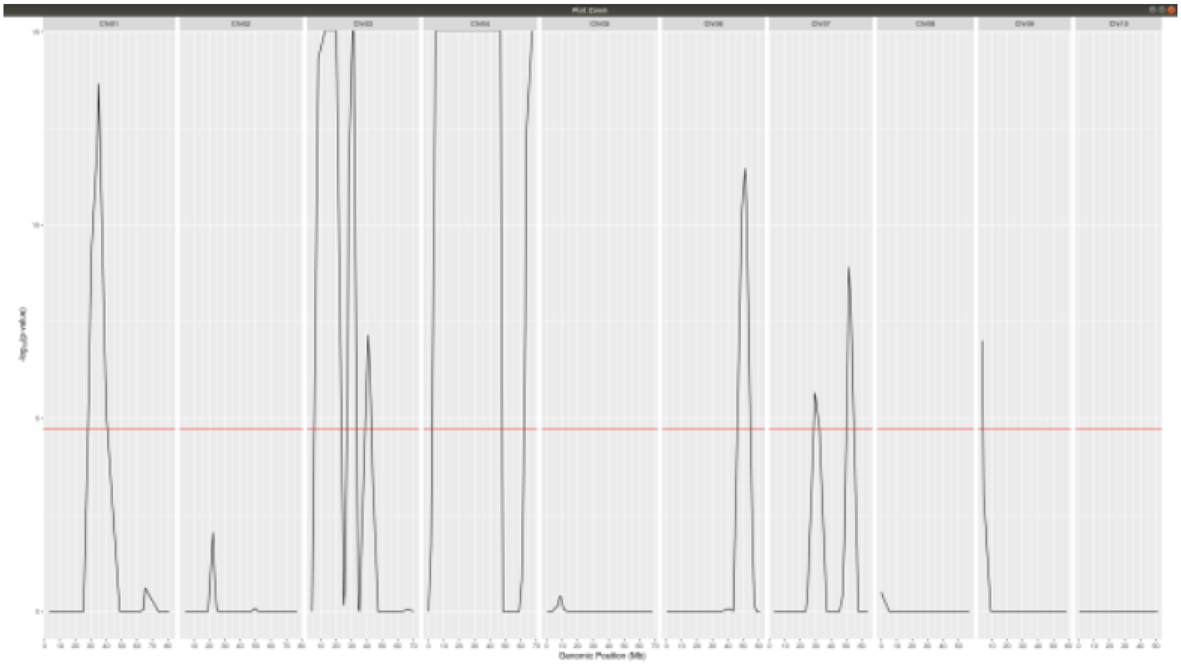
negLog10Pval

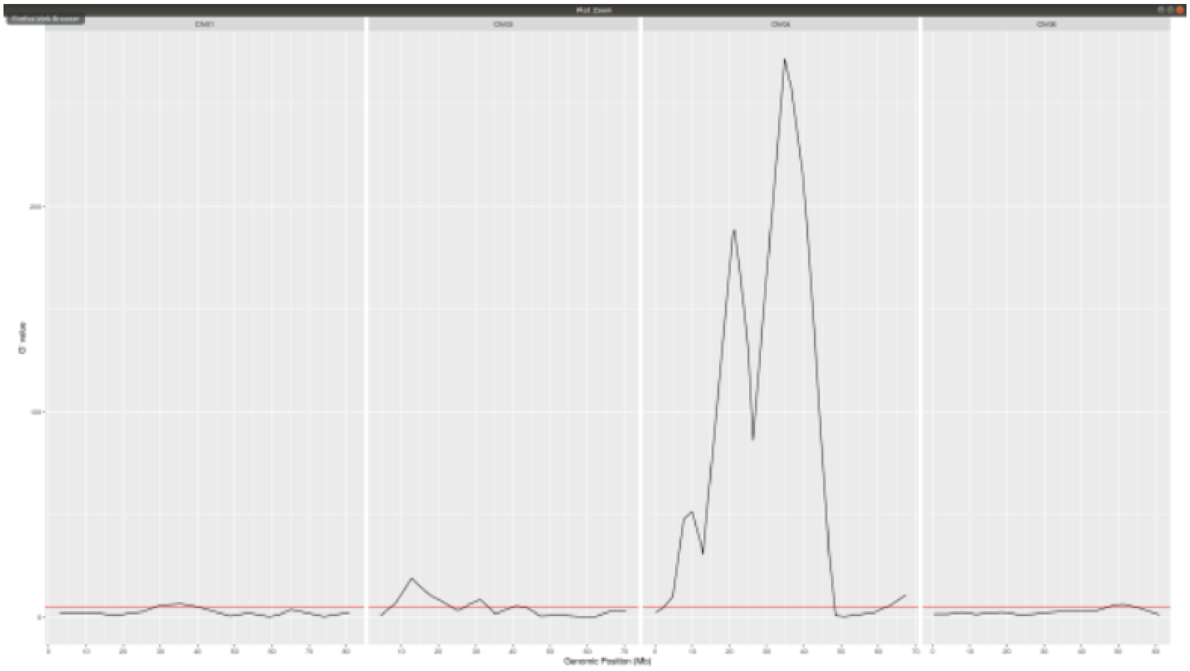
```
**Finally with plotQTLStats plot negLog10Pval.**

plotQTLStats(SNPset = df_filt2, var = "negLog10Pval",plotThreshold = TRUE,q=0.01)
ggsave(filename = "negLog10Pval.png",plot = last_plot())
```

Gprime Subset

```
**Add subset argument to focus on particular chromosomes one, three, four, and six.**
**The reason is due to signficant QTL regions**


plotQTLStats(SNPset = df_filt2, var = "Gprime",plotThreshold = TRUE,q=0.01,subset = c(
↪"Chr01","Chr03","Chr04","Chr06"))
```


Plot Zoom
Chr01
Chr02
Chr03
Chr04
Chr05
Chr06
Chr07
Chr08
Chr09
Chr10
0.4
0.2
0.0
-0.2
-0.4
Δ(SNP-index)
Genomic Position (Mb)
Interval
CI_95
CI_99

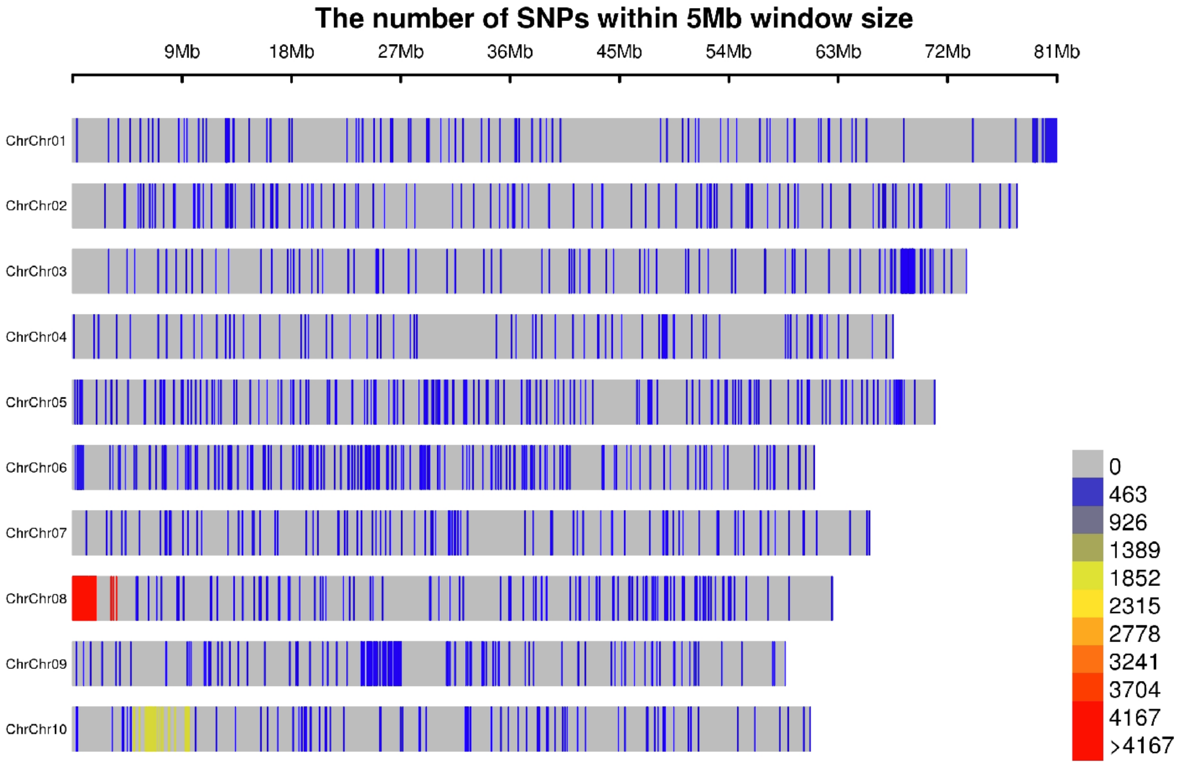

Plot Zoom
Chr01
Chr02
Chr03
Chr04
Chr05
Chr06
Chr07
Chr08
Chr09
Chr10
15
10
5
0
$-\log_{10}(p\text{-value})$
Genomic Position (Mb)



1.14 rMVP Package

SNP Densities

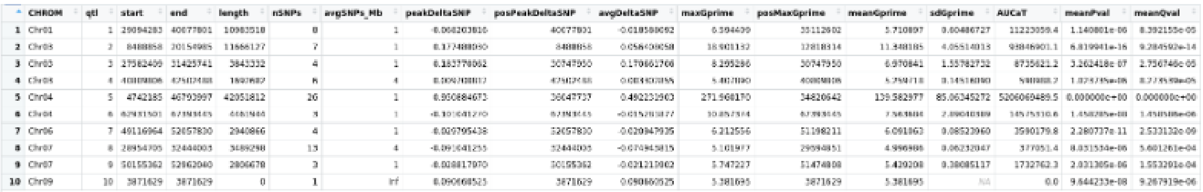
```
install.packages("rMVP")
library(rMVP)
sample<-"Semi_Dwarfism_in_Sorghum"
pathtosample <- "/home/michael/Desktop/QTLseqr/extdata/subset_freebayes_D2.filtered.vcf.
↪gz"
out<- paste0("mvp.",sample,".vcf")
memo<-paste0(sample)
dffile<-paste0("mvp.",sample,".vcf.geno.map")

message("Making MVP data S1")
MVP.Data(fileVCF=pathtosample,
      #filePhe="Phenotype.txt",
      fileKin=FALSE,
      filePC=FALSE,
      out=out)

message("Reading MVP Data S1")
df <- read.table(file = dffile, header=TRUE)
message("Making SNP Density Plots")
MVP.Report.Density(df[,c(1:3)], bin.size = 5000000, col = c("blue", "yellow", "red"),␣
↪memo = memo, file.type = "jpg", dpi=300)
```


The number of SNPs within 5Mb window size
9Mb
18Mb
27Mb
36Mb
45Mb
54Mb
63Mb
72Mb
81Mb
ChrChr01
ChrChr02
ChrChr03
ChrChr04
ChrChr05
ChrChr06
ChrChr07
ChrChr08
ChrChr09
ChrChr10
0
463
926
1389
1852
2315
2778
3241
3704
4167
>4167

1.15 Export summary CSV

```
QTLTable(SNPset = df_filt, alpha = 0.01, export = TRUE, fileName = "my_BSA_QTL.csv")
```

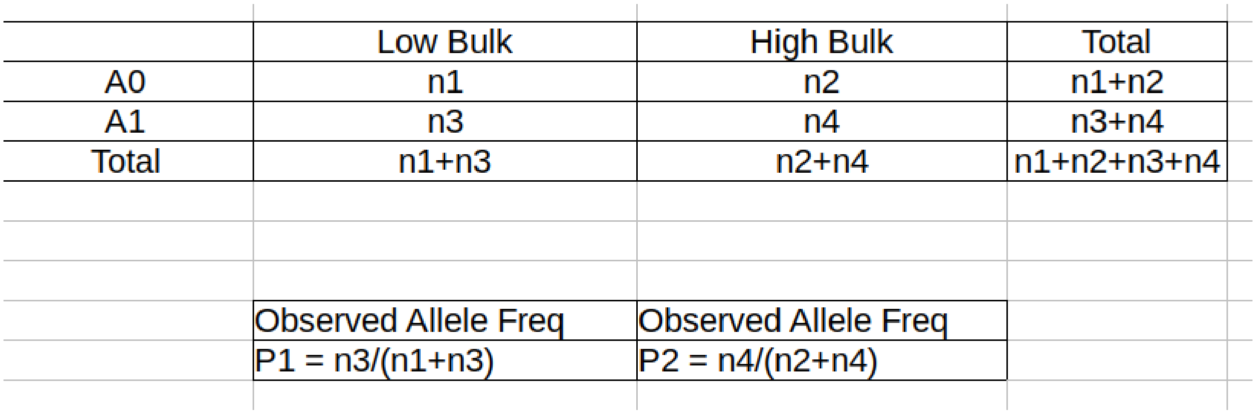
Preview the Summary QTL



	CHROM	qtl	start	end	length	nSNPs	avgSNPs_Mb	peakDeltaSNP	posPeakDeltaSNP	avgDeltaSNP	maxGprime	posMaxGprime	meanGprime	sdGprime	AUCaT	meanPval	meanQval
1	Chr01	1	29094283	40077801	10983518	8	1	-0.068203816	40077801	-0.018588092	6.394439	33112602	5.710897	0.60486727	11223059.4	1.140801e-06	8.392155e-05
2	Chr03	2	8488858	20154985	11666127	7	1	0.177488090	8488858	0.058408058	18.901132	12818314	11.348185	4.05514013	93846901.1	6.819941e-16	9.284592e-14
3	Chr03	3	27582409	31425741	3843332	4	1	0.183778062	30747950	0.170661766	8.295286	30747950	6.970841	1.55782732	8735621.2	3.262418e-07	2.756746e-05
4	Chr03	4	40089806	42502488	1897687	6	4	0.109278817	42502488	0.008403855	5.402890	40089806	5.294718	0.14516080	5808808.7	1.027479e-06	8.773598e-05
5	Chr04	5	4742185	46793997	42051812	26	1	0.950884673	36647737	0.492231903	271.968170	34820642	139.582977	85.06345272	5206069489.5	0.000000e+00	0.000000e+00
6	Chr04	6	62931501	67398445	4461944	3	1	-0.101641270	67398445	-0.015203877	10.857314	67398445	7.563888	2.89040189	14575110.6	1.458285e-08	1.458586e-06
7	Chr06	7	49116964	52057830	2940866	4	1	-0.029795438	52057830	-0.020947935	6.212556	51198211	6.091063	0.08523960	3590179.8	2.280737e-11	2.533132e-09
8	Chr07	8	28954705	32444003	3489298	13	4	-0.091641255	32444003	-0.074945815	5.101977	29594851	4.996986	0.06232047	377051.4	8.031534e-06	5.601261e-04
9	Chr07	9	50155362	52962040	2806678	3	1	-0.028817970	50155362	-0.021212962	5.747227	51474808	5.420208	0.38085117	1732762.3	2.031305e-06	1.553201e-04
10	Chr09	10	3871629	3871629	0	1	Inf	0.090668525	3871629	0.090668525	5.381695	3871629	5.381695	NA	0.0	9.844233e-08	9.267919e-06

1.16 Theory

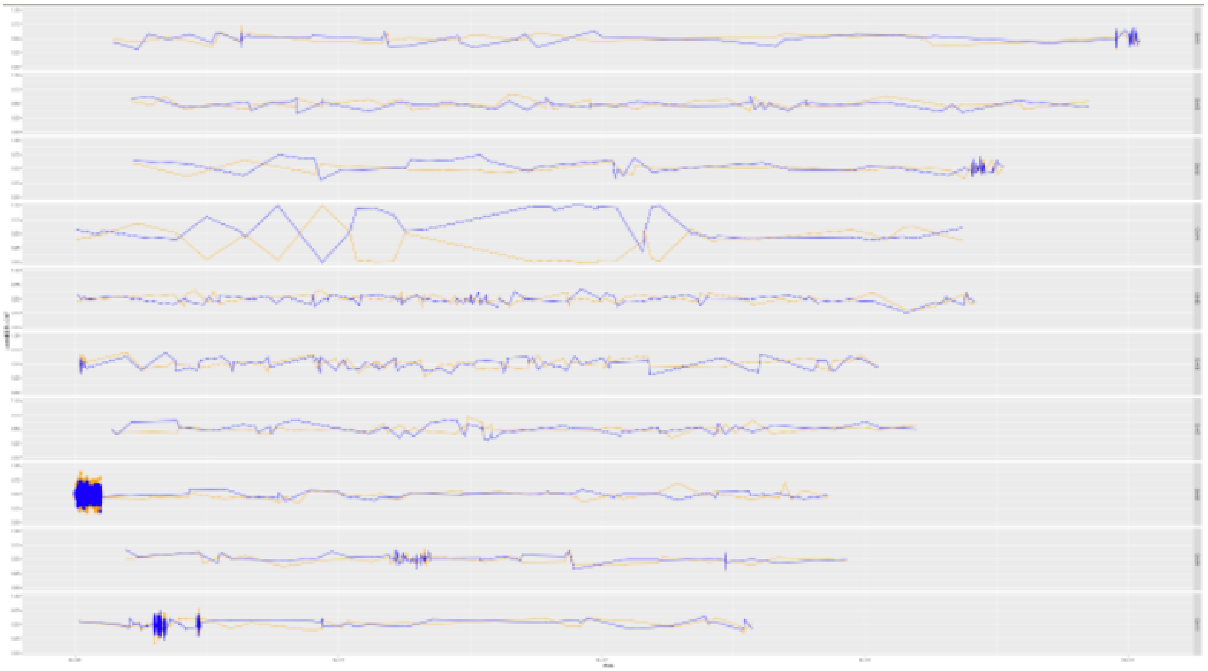
Contigency Table



	Low Bulk	High Bulk	Total
A0	n1	n2	n1+n2
A1	n3	n4	n3+n4
Total	n1+n3	n2+n4	n1+n2+n3+n4
	Observed Allele Freq	Observed Allele Freq	
	P1 = n3/(n1+n3)	P2 = n4/(n2+n4)	

Obs_Allel_Freq

```
**Use the function to plot allele frequencies per chromosome.**
**Second argument size specifes size of scalar factor on nSNPs and if you have a␣
↪relatively small SNP set .001 is a good startin point otherwise set to 1**


Obs_Allele_Freq(SNPSet = df_filt, size = .001)
```



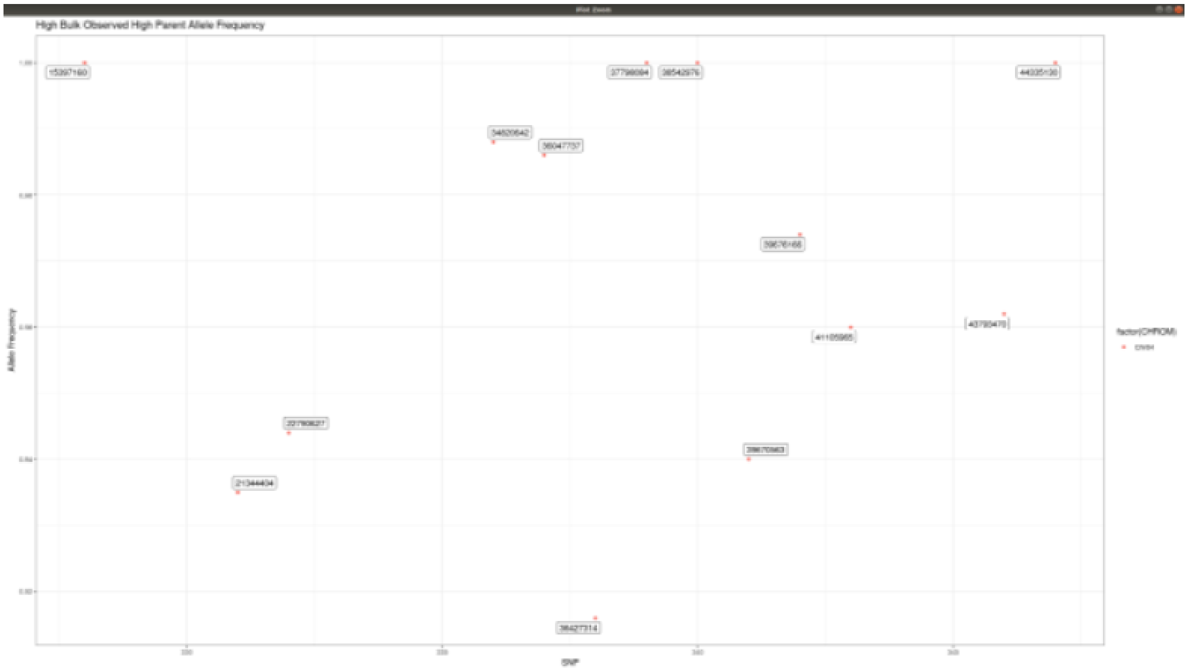
Obs_Allele_Freq2

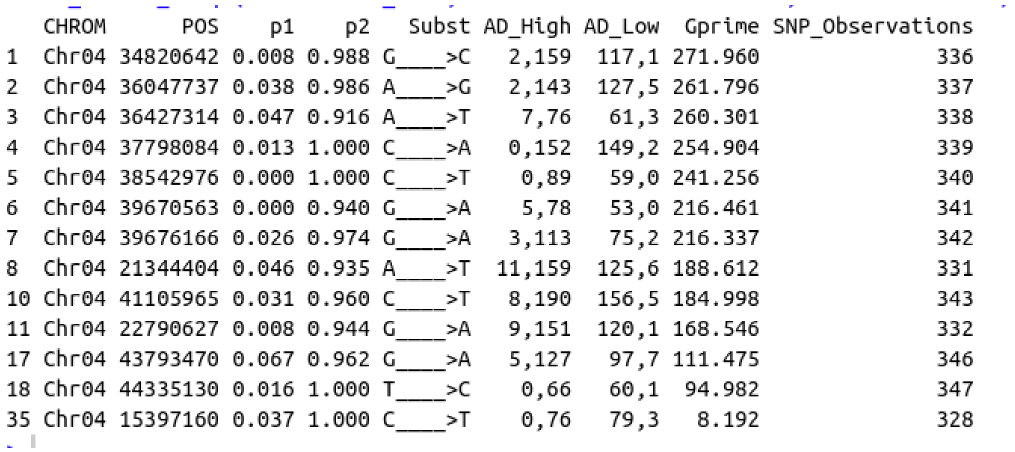
```
**Use the function to investigate chromosomal region of interest**

Obs_Allele_Freq2(SNPSet = df_filt, ChromosomeValue = "Chr04", threshold = .90)
```

Total Coverage and Expected Allelic Frequencies

```
E(n1) = E(n2) = E(n3) = E(n4) = C/2


**Read in the csv file from High bulk tt**

tt<-read.table(file = "D2_F2_tt.csv",header = TRUE,sep = ",")

**Calculate average Coverage per SNP site**

mean(tt$DP)

**Find REalized frequencies**

p1_STAR <- sum(tt$AD_ALT.) / sum(tt$DP)

**Read in the csv file from Low Bulk TT.**

TT<-read.table(file ="D2_F2_TT.csv",header = TRUE,sep=",")

**Calculate average Coverage per SNP sit**
```

(continues on next page)





```
   CHROM      POS    p1    p2  Subst AD_High AD_Low  Gprime SNP_Observations
1  Chr04 34820642 0.008 0.988 G____>C   2,159  117,1 271.960              336
2  Chr04 36047737 0.038 0.986 A____>G   2,143  127,5 261.796              337
3  Chr04 36427314 0.047 0.916 A____>T    7,76   61,3 260.301              338
4  Chr04 37798084 0.013 1.000 C____>A   0,152  149,2 254.904              339
5  Chr04 38542976 0.000 1.000 C____>T    0,89   59,0 241.256              340
6  Chr04 39670563 0.000 0.940 G____>A    5,78   53,0 216.461              341
7  Chr04 39676166 0.026 0.974 G____>A   3,113   75,2 216.337              342
8  Chr04 21344404 0.046 0.935 A____>T  11,159  125,6 188.612              331
10 Chr04 41105965 0.031 0.960 C____>T   8,190  156,5 184.998              343
11 Chr04 22790627 0.008 0.944 G____>A   9,151  120,1 168.546              332
17 Chr04 43793470 0.067 0.962 G____>A   5,127   97,7 111.475              346
18 Chr04 44335130 0.016 1.000 T____>C    0,66   60,1  94.982              347
35 Chr04 15397160 0.037 1.000 C____>T    0,76   79,3   8.192              328
```

(continued from previous page)

```
mean(TT$DP)

**Find Realized frequencies**

p2_STAR <- sum(TT$AD_ALT.) / sum(TT$DP)

**Take the average of the Averages**

C <-(mean(tt$DP)+mean(TT$DP))/2

C<-round(C,0)
**Find Coverage Value**
C
110

E(n1) = E(n2) = E(n3) = E(n4) = C/2 = 55

p2 >> p1 QTL is present
```

1.17 Theory and Analytical Framework of Sampling from BSA

Binomial Sampling

High Bulk

par(mfrow=c(1,1)) **Define Ranges of Success** success <- 0:90

The Difference between realized and Expected Frequencies

ns : Sample Size taken from Low Bulk

2(ns)p1_star ~ Binomial(2(ns),p1)

p1 Expected Frequencies

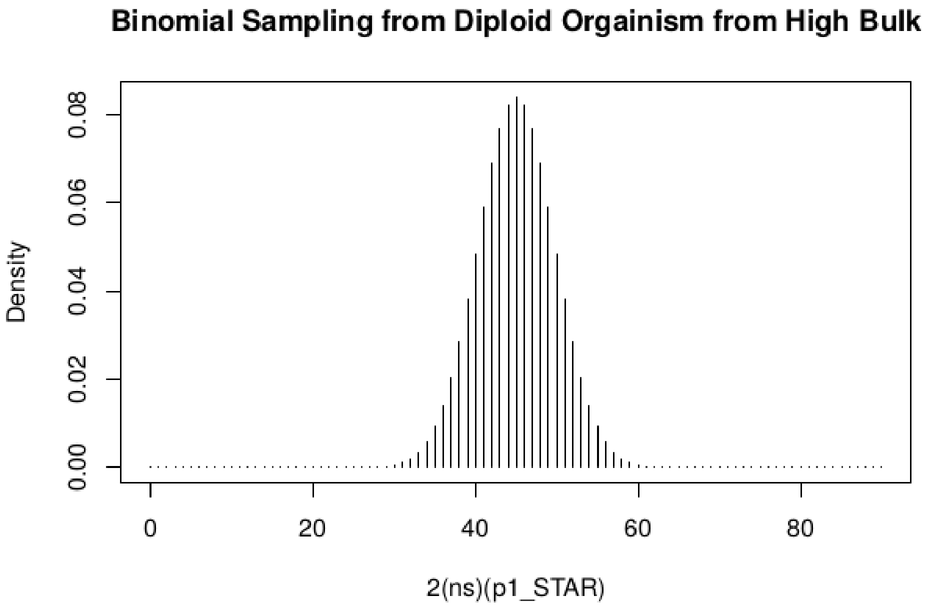
Expected Frequencies:

E(n1) = E(n2) = E(n3) = E(n4) = C/2 = 110

We prefer for accuracy and a powerful G Prime Test to have ns >> C >> 1

However, it is not true in this case.

plot(success, dbinom(success, size = 90, prob = .50), type = “h”,main=”Binomial Sampling from Diploid Orgainism from High Bulk”,xlab=”2(ns)(p1_STAR)”,ylab=”Density”)



Low Bulk

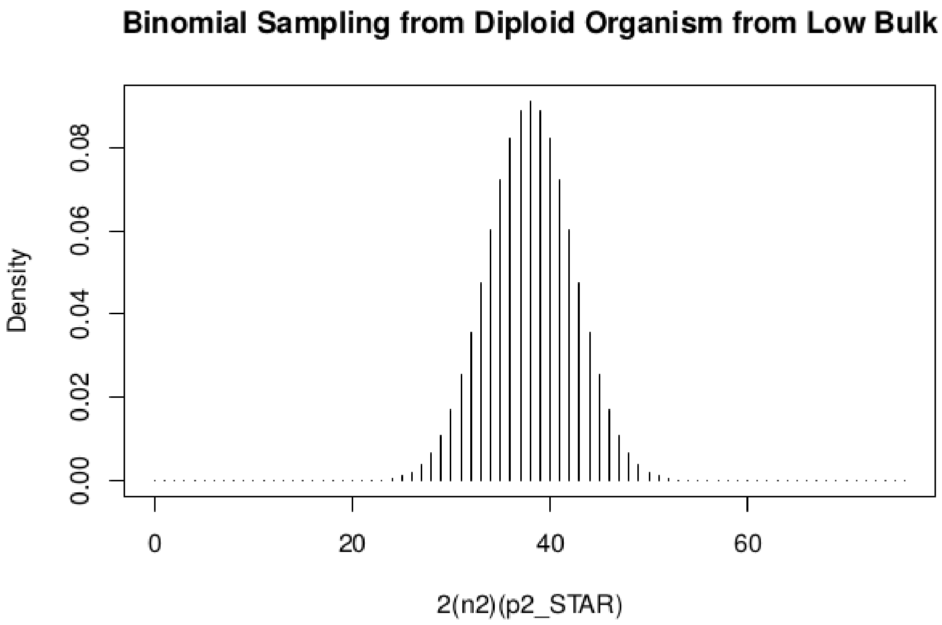
```
**ns : Sample Size from High Bulk**
**2(ns)p2_star ~ Binomial(2(ns),p2)**
**p2 Expected Frequencies**
success <- 0:76
plot(success, dbinom(success, size = 76, prob = 0.5), type = "h",main="Binomial Sampling␣
↪from Diploid Organism from Low Bulk",xlab="2(n2)(p2_STAR)",ylab="Density")
```

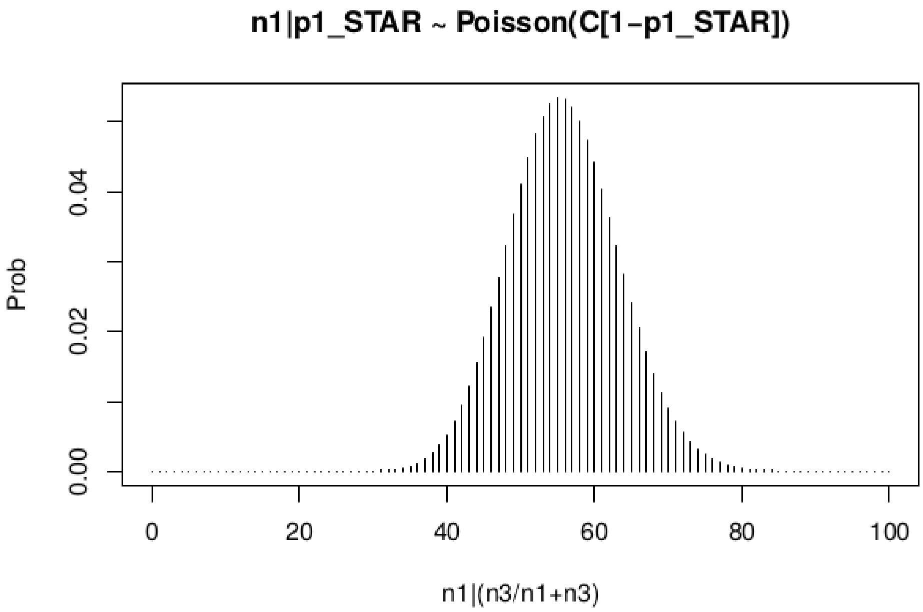
Conditional Distribution of n1 given realized average frequency

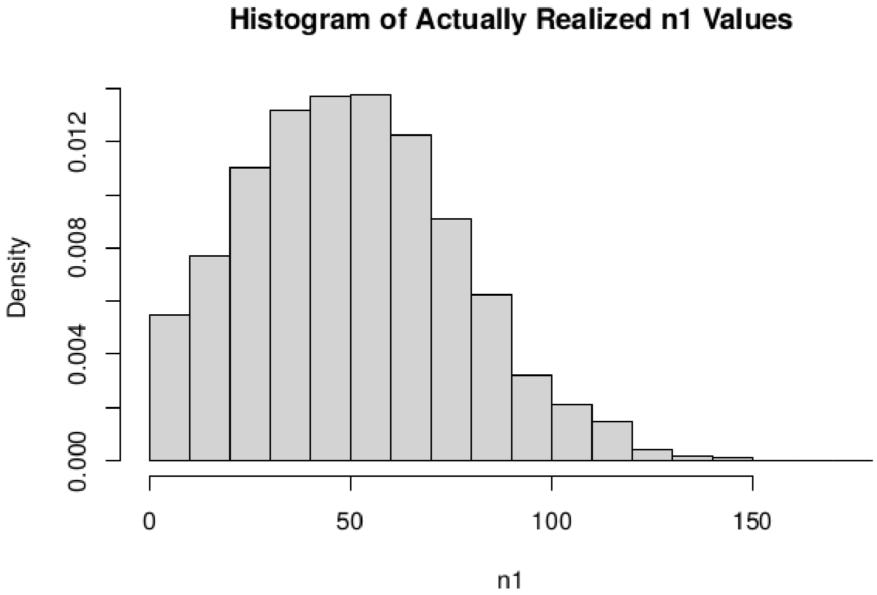
```
par(mfrow=c(1,1))
#Define Ranges of Success (Allele Frequencies High and Low)
success <- 0:100
#n1|p1_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*(1-p1_STAR)), type = 'h',main="n1|p1_STAR ~␣
↪Poisson(C[1-p1_STAR])",xlab="n1|(n3/n1+n3)",ylab="Prob")
```

Observed n1

```
hist(TT$AD_REF., probability = TRUE,main="Histogram of Actually Realized n1 Values",xlab=
↪"n1")
```


Binomial Sampling from Diploid Organism from Low Bulk


n1|p1_STAR ~ Poisson(C[1−p1_STAR])



Conditional Distribution of n2 given realized average frequency

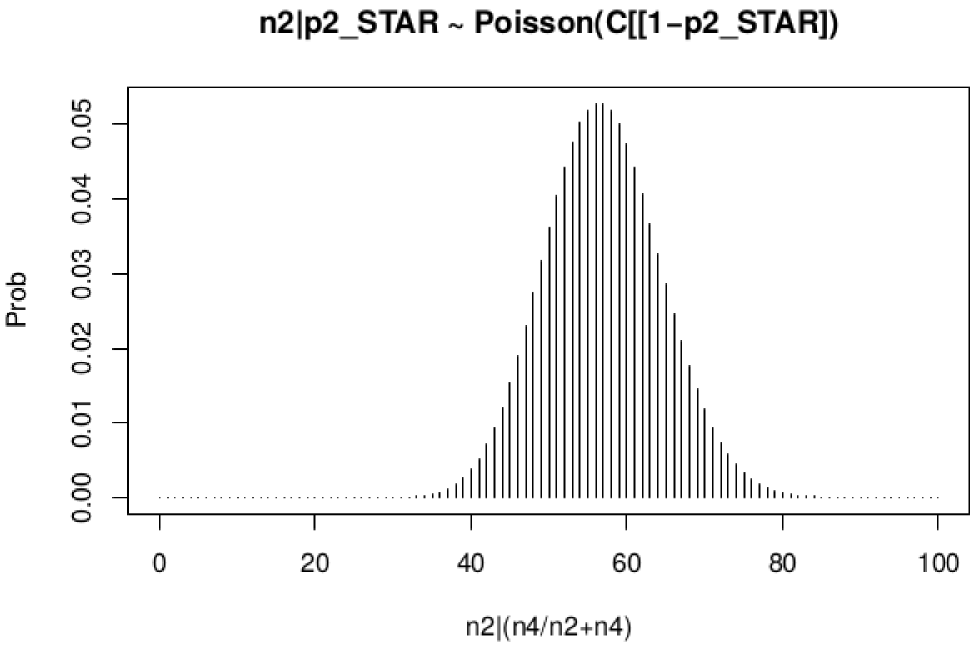
```
#n2|p2_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*(1-p2_STAR)), type='h', main="n2|p2_STAR ~␣
↪Poisson(C[[1-p2_STAR])",xlab="n2|(n4/n2+n4)",ylab="Prob")
```

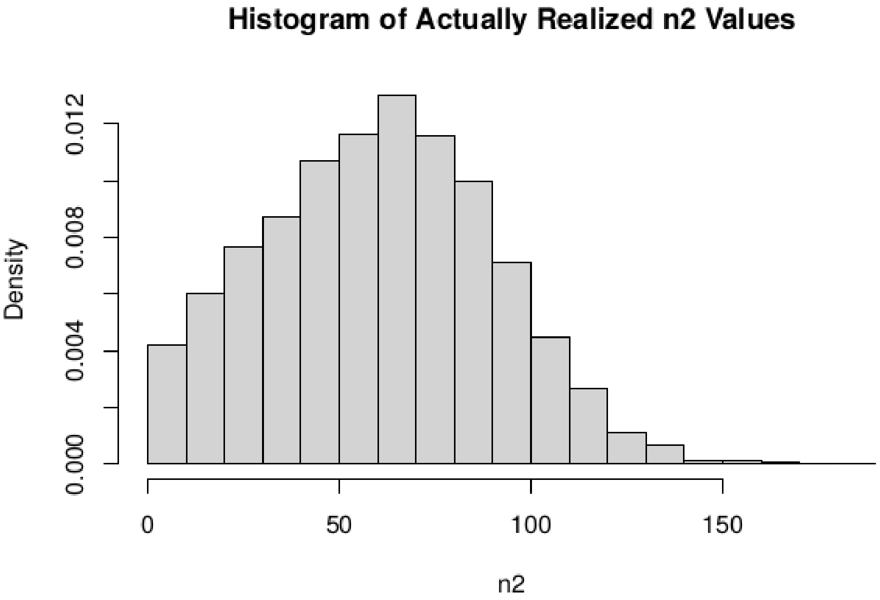
Observed n2

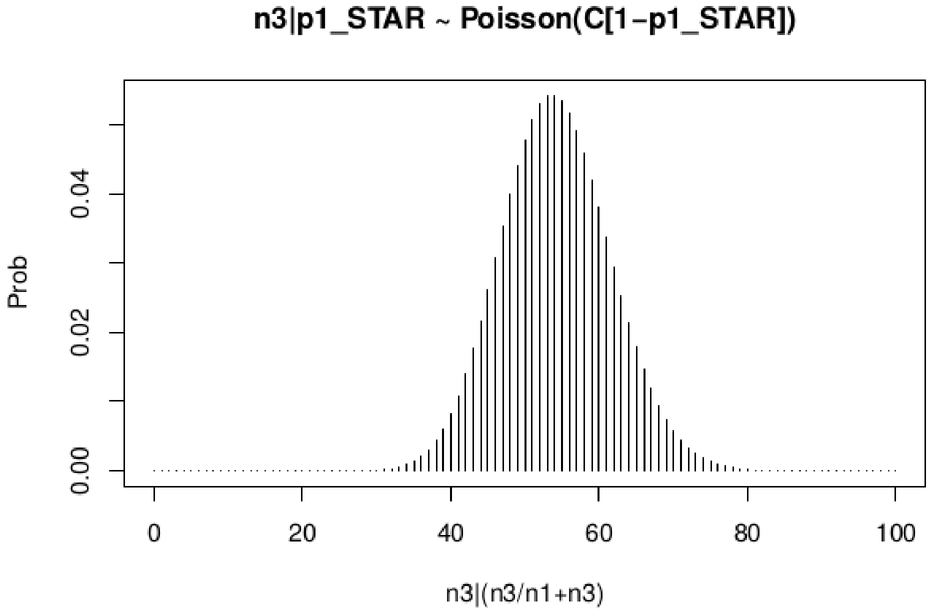
```
hist(tt$AD_REF., probability = TRUE, main = "Histogram of Actually Realized n2 Values",
↪xlab="n2")
```

Conditional Distribution of n3 given realized average frequency

```
#n3|p1_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*p1_STAR),type='h',main="n3|p1_STAR ~ Poisson(C[1-
↪p1_STAR])",xlab="n3|(n3/n1+n3)",ylab="Prob")
```


n2|p2_STAR ~ Poisson(C[[1−p2_STAR])
Prob
0.00
0.01
0.02
0.03
0.04
0.05
0
20
40
60
80
100
n2|(n4/n2+n4)


Histogram of Actually Realized n2 Values
Density
0.000
0.004
0.008
0.012
0
50
100
150
n2



Observed n3

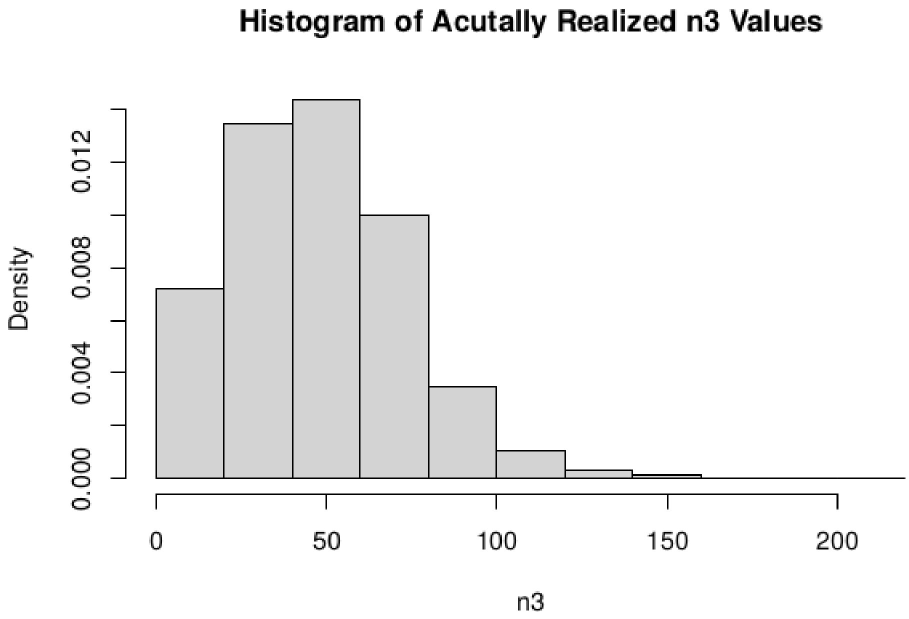
```r
hist(TT$AD_ALT., probability = TRUE, main="Histogram of Acutally Realized n3 Values", xlab="n3")
```

Conditional Distribution of n4 given realized average frequency

```r
#n4|p2_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*p2_STAR), type = 'h',main="n4|p2_STAR ~ Poisson(C[1-p2_STAR])",xlab="n4|n4/(n2+n4)",ylab="Prob")
```

Observed n4

```r
hist(tt$AD_ALT., probability = TRUE, main="Histogram of Acutally Realized n4 Values", xlab="n4")
```



An interdependentaly observed relationship between G and Gprime

2 QTL_Rice_Cold_Tolerance

Author Michael Hall

Date 4/13/2022

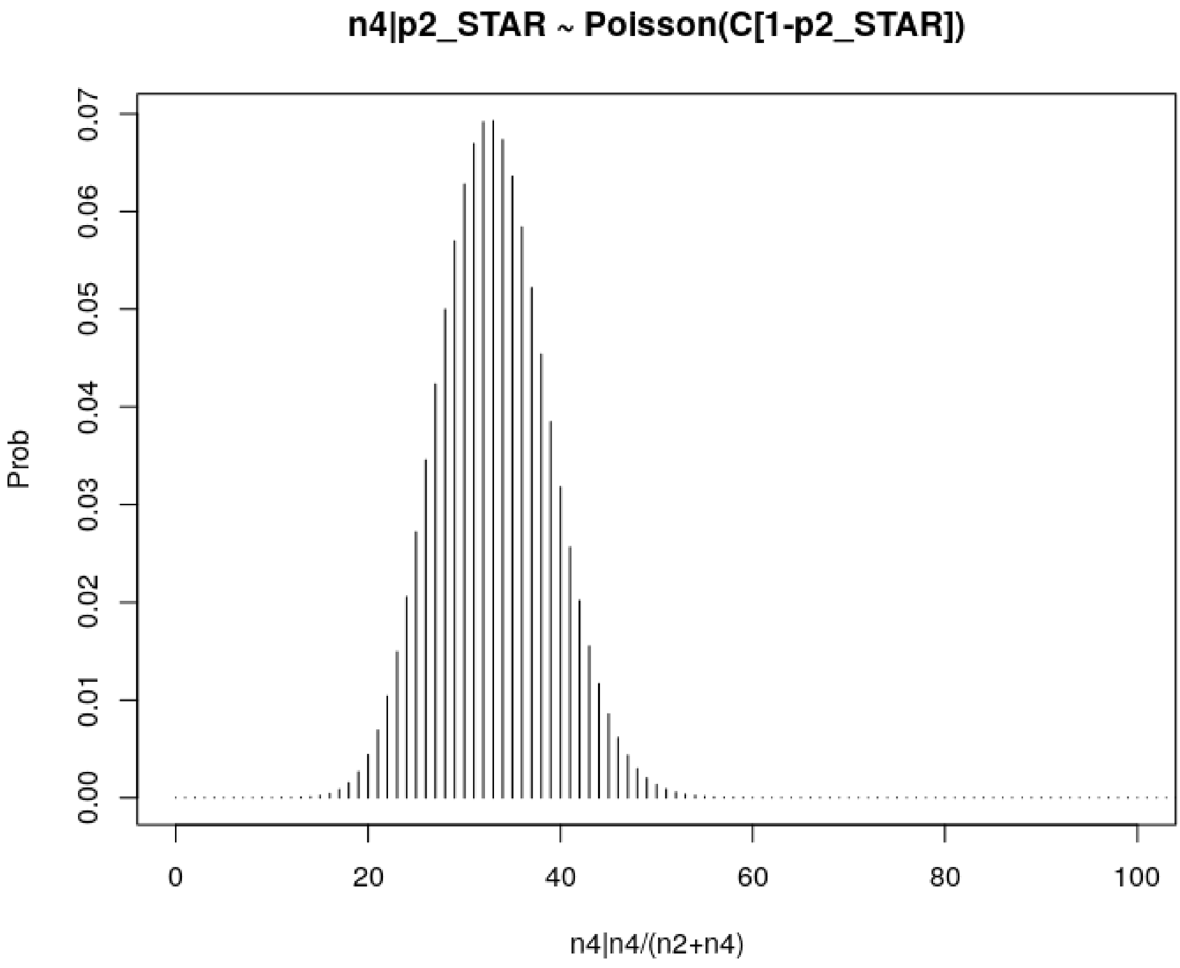
Before we begin I like to reveal what my machine specifications are just in case there might be a compatibility issue:

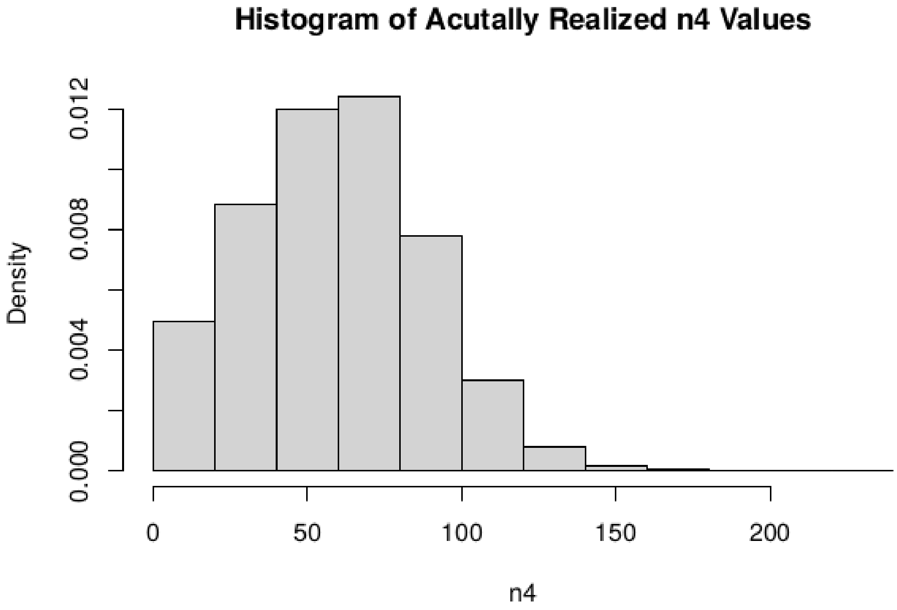
What open source opeating system are you running? Ubuntu 18.04, Code name Bionic, it must be a Tuesday

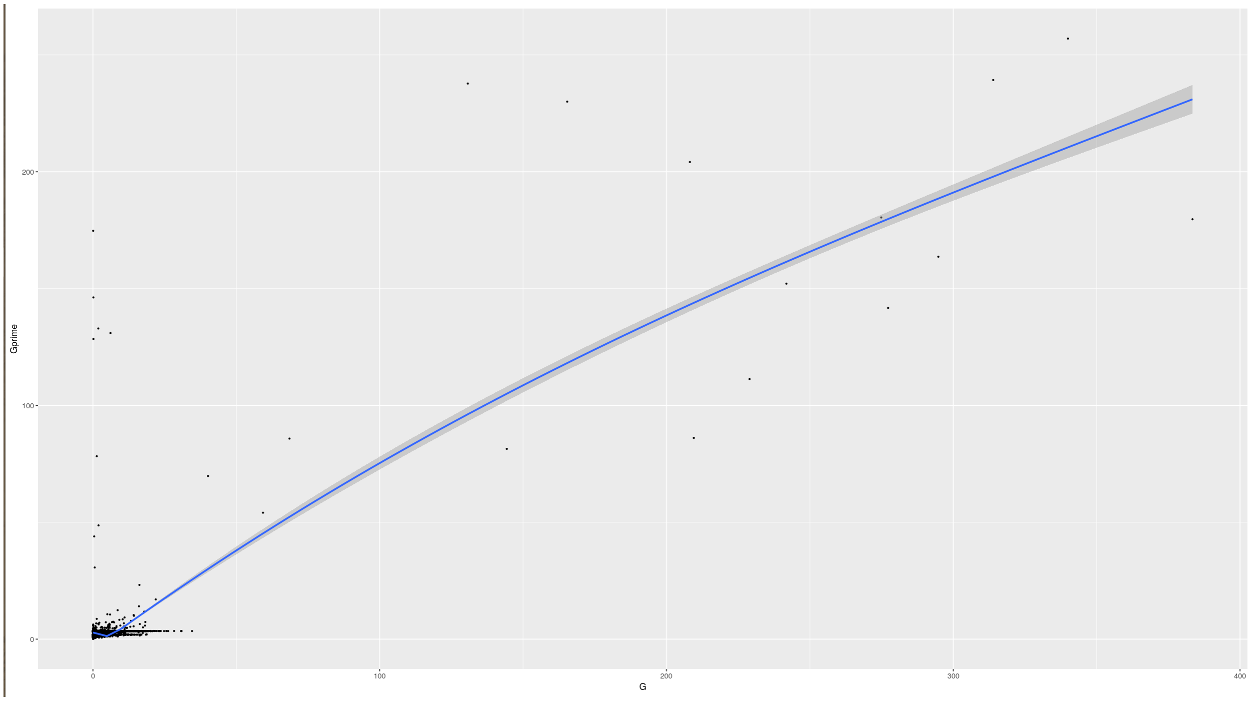
2.1 QTL-Rice-Cold-Tolerance

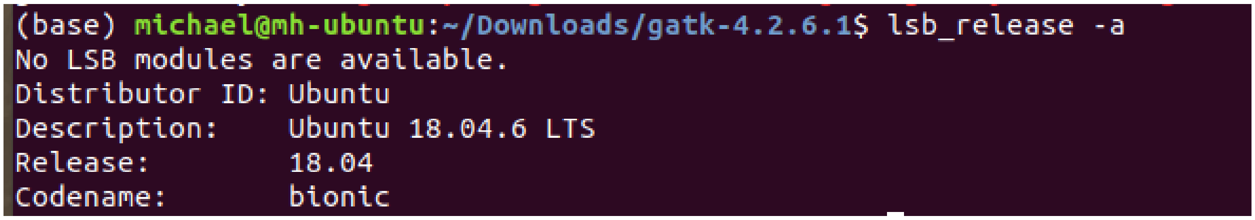
QTLseqr is an R package for QTL mapping using NGS Bulk Segregant Analysis.

QTLseqr is still under development and is offered with out any guarantee.


n4|p2_STAR ~ Poisson(C[1-p2_STAR])
Prob
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0
20
40
60
80
100
n4|n4/(n2+n4)







```
(base) michael@mh-ubuntu:~/Downloads/gatk-4.2.6.1$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS
Release:        18.04
Codename:       bionic
```

For more detailed instructions please read the vignettehere

For updates read the NEWS.md

2.2 Installation

You can install QTLseqr from github with:

```
# install devtools first to download packages from github
install.packages("devtools")

# use devtools to install QTLseqr
devtools::install_github("PBGLMichaelHall/QTLseqr")
```

Package Dependencies

Note: Apart from regular package dependencies, there are some Bioconductor tools that we use as well, as such you will be prompted to install support for Bioconductor, if you haven't already. QTLseqr makes use of C++ to make some tasks significantly faster (like counting SNPs). Because of this, in order to install QTLseqr from github you will be required to install some compiling tools (Rtools and Xcode, for Windows and Mac, respectively).

2.3 Citation

If you use QTLseqr in published research, please cite:

> Mansfeld B.N. and Grumet R, QTLseqr: An R package for bulk segregant analysis with next-generation sequencing *The Plant Genome* doi:10.3835/plantgenome2018.01.0006

We also recommend citing the paper for the corresponding method you work with.

QTL-seq method:

> Takagi, H., Abe, A., Yoshida, K., Kosugi, S., Natsume, S., Mitsuoka, C., Uemura, A., Utsushi, H., Tamiru, M., Takuno, S., Innan, H., Cano, L. M., Kamoun, S. and Terauchi, R. (2013), QTL-seq: rapid mapping of quantitative trait loci in rice by whole genome resequencing of DNA from two bulked populations. *Plant J*, 74: 174–183. doi:10.1111/tpj.12105

G prime method:

> Magwene PM, Willis JH, Kelly JK (2011) The Statistics of Bulk Segregant Analysis Using Next Generation Sequencing. *PLOS Computational Biology* 7(11): e1002255. doi.org/10.1371/journal.pcbi.1002255

Abstract

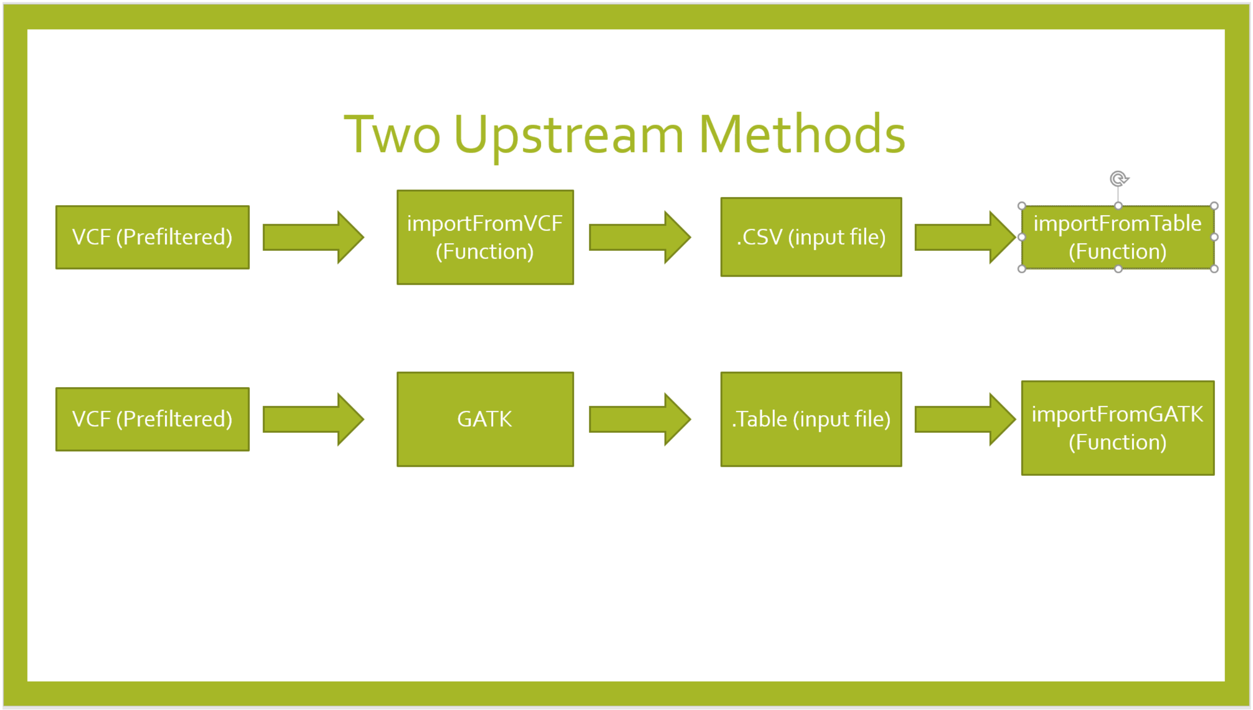
Next Generation Sequencing Bulk Segregant Analysis (NGS-BSA) is efficient in detecting quantitative trait loci (QTL). Despite the popularity of NGS-BSA and the R statistical platform, no R packages are currently available for NGS-BSA. We present QTLseqr, an R package for NGS-BSA that identifies QTL using two statistical approaches: QTL-seq and G'. These approaches use a simulation method and a tricube smoothed G statistic, respectively, to identify and assess statistical significance of QTL. QTLseqr, can import and filter SNP data, calculate SNP distributions, relative allele frequencies, G' values, and log10(p-values), enabling identification and plotting of QTL.

2.4 Examples:

Load/install libraries

```
install.packages(“vcfR”)
install.packages(“tidyr”)
install.packages(“ggplot2”)
devtools::install_github(“PBGLMichaelHall/QTLseqr”,force = TRUE)
library(QTLseqr)
library(vcfR)
library(tidyr)
library(ggplot2)
library(dplyr)
```

```
**Methods**
```



Two Upstream Methods

VCF (Prefiltered) → importFromVCF (Function) → .CSV (input file) → importFromTable (Function)

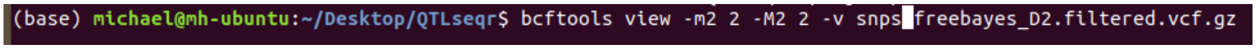
VCF (Prefiltered) → GATK → .Table (input file) → importFromGATK (Function)

Set the Working Directory

```
setwd("/home/michael/Desktop/RiceCold2")
```

2.5 Pre-Filtering Rules

```
Vcf file can contain bialleleic variants before parsing, however, out of a principal␣
↪investigators preference, the user can (filter upstream, e.g., with bcftools view -m2 -
↪M2), also the QTLseqR functions will only call SNPS, so filter out **INDELS** with the␣
↪following command line.
```



```
(base) michael@mh-ubuntu:~/Desktop/QTLseqr$ bcftools view -m2 2 -M2 2 -v snps freebayes_D2.filtered.vcf.gz
```

2.6 The Lonely Parser

Calling my Parser **QTLParser_1_MH** This method requires 4 arguments, a **vcf**, **highBulk**, **lowBulk**, and **filename**. Proceeding this Call you must invoke **importFromTable** before Filtering.

```
df <- QTLParser_1_MH(vcf = "wGQ-Filt-freebayes~bwa~IRGSP-1.0~both-segregant_bulks~
↪filtered-default.vcf", highBulk = "ET-pool-385", lowBulk = "ES-pool-430", filename =
↪"Hall.csv")
```

2.7 Import Data

Method 1 (Biased due to parser configuration)

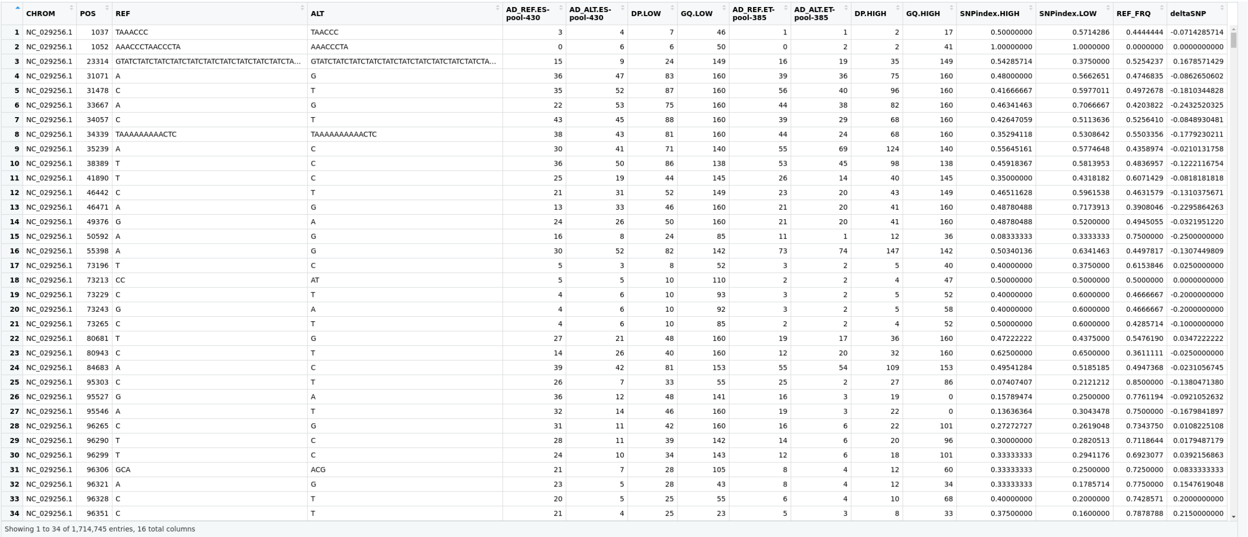
Calling **importFromTable** on Hall.csv file This method requires 5 inputs to 5 arguments, **file**, **highBulk**, **lowBulk**, **chromList** and **sep**.

importFromTable

```
 Chroms <- c("NC_029256.1","NC_029257.1","NC_029258.1","NC_029259.1","NC_029260.1","NC_
↪029261.1","NC_029262.1","NC_029263.1","NC_029264.1","NC_029265.1","NC_029266.1","NC_
↪029267.1")

df <- importFromTable(file = "Hall.csv", highBulk = "ET-pool-385", lowBulk = "ES-pool-430
↪", chromList = Chroms, sep = ",")

**Method 2 (Most convienent)**

Calling **importFromVCF**
This method requires 5 arguments, a vcf **file**, **highBulk**, **lowBulk**,␣
↪**chromList**, **filename**, and **filter.**
**The filtering argument is a Boolean accepting only TRUE or FALSE. If TRUE then it␣
↪filters out all SNPs that did not "PASS" in that INFO field.**
**If it is FALSE then there is no filter applied at all.**
```

importFromVCF

```
Chroms <- c("NC_029256.1","NC_029257.1","NC_029258.1","NC_029259.1","NC_029260.1","NC_
↪029261.1","NC_029262.1","NC_029263.1","NC_029264.1","NC_029265.1","NC_029266.1","NC_
↪029267.1")

df <- importFromVCF(file = "wGQ-Filt-freebayes~bwa~IRGSP-1.0~both-segregant_bulks~
↪filtered-default.vcf",highBulk = "ET-pool-385",lowBulk = "ES-pool-430",chromList =␣
↪Chroms,filename = "Hall",filter = FALSE)
```



	CHROM	POS	REF	ALT	AD_REF.ES-pool-430	AD_ALT.ES-pool-430	DP.LOW	GQ.LOW	AD_REF.ET-pool-385	AD_ALT.ET-pool-385	DP.HIGH	GQ.HIGH	SNPindex.HIGH	SNPindex.LOW	REF_FRQ	deltaSNP
1	NC_029256.1	1037	TAAACCC	TAACCC	3	4	7	46	1	1	2	17	0.50000000	0.5714286	0.4444444	-0.0714285714
2	NC_029256.1	1052	AAACCCTAACCCTA	AAACCCTA	0	6	6	50	0	2	2	41	1.00000000	1.0000000	0.0000000	0.0000000000
3	NC_029256.1	23314	GTATCTATCTATCTATCTATCTATCTATCTATCTATCTATCTA...	GTATCTATCTATCTATCTATCTATCTATCTATCTATCTATCTA...	15	9	24	149	16	19	35	149	0.54285714	0.3750000	0.5254237	0.1678571429
4	NC_029256.1	31071	A	G	36	47	83	160	39	36	75	160	0.48000000	0.5662651	0.4746835	-0.0862650602
5	NC_029256.1	31478	C	T	35	52	87	160	56	40	96	160	0.41666667	0.5977011	0.4972678	-0.1810344828
6	NC_029256.1	33667	A	G	22	53	75	160	44	38	82	160	0.46341463	0.7066667	0.4203822	-0.2432520325
7	NC_029256.1	34057	C	T	43	45	88	160	39	29	68	160	0.42647059	0.5113636	0.5256410	-0.0848930481
8	NC_029256.1	34339	TAAAAAAAAACTC	TAAAAAAAAAACTC	38	43	81	160	44	24	68	160	0.35294118	0.5308642	0.5503356	-0.1779230211
9	NC_029256.1	35239	A	C	30	41	71	140	55	69	124	140	0.55645161	0.5774648	0.4358974	-0.0210131758
10	NC_029256.1	38389	T	C	36	50	86	138	53	45	98	138	0.45918367	0.5813953	0.4836957	-0.1222116754
11	NC_029256.1	41890	T	C	25	19	44	145	26	14	40	145	0.35000000	0.4318182	0.6071429	-0.0818181818
12	NC_029256.1	46442	C	T	21	31	52	149	23	20	43	149	0.46511628	0.5961538	0.4631579	-0.1310375671
13	NC_029256.1	46471	A	G	13	33	46	160	21	20	41	160	0.48780488	0.7173913	0.3908046	-0.2295864263
14	NC_029256.1	49376	G	A	24	26	50	160	21	20	41	160	0.48780488	0.5200000	0.4945055	-0.0321951220
15	NC_029256.1	50592	A	G	16	8	24	85	11	1	12	36	0.08333333	0.3333333	0.7500000	-0.2500000000
16	NC_029256.1	55398	A	G	30	52	82	142	73	74	147	142	0.50340136	0.6341463	0.4497817	-0.1307449809
17	NC_029256.1	73196	T	C	5	3	8	52	3	2	5	40	0.40000000	0.3750000	0.6153846	0.0250000000
18	NC_029256.1	73213	CC	AT	5	5	10	110	2	2	4	47	0.50000000	0.5000000	0.5000000	0.0000000000
19	NC_029256.1	73229	C	T	4	6	10	93	3	2	5	52	0.40000000	0.6000000	0.4666667	-0.2000000000
20	NC_029256.1	73243	G	A	4	6	10	92	3	2	5	58	0.40000000	0.6000000	0.4666667	-0.2000000000
21	NC_029256.1	73265	C	T	4	6	10	85	2	2	4	52	0.50000000	0.6000000	0.4285714	-0.1000000000
22	NC_029256.1	80681	T	G	27	21	48	160	19	17	36	160	0.47222222	0.4375000	0.5476190	0.0347222222
23	NC_029256.1	80943	C	T	14	26	40	160	12	20	32	160	0.62500000	0.6500000	0.3611111	-0.0250000000
24	NC_029256.1	84683	A	C	39	42	81	153	55	54	109	153	0.49541284	0.5185185	0.4947368	-0.0231056745
25	NC_029256.1	95303	C	T	26	7	33	55	25	2	27	86	0.07407407	0.2121212	0.8500000	-0.1380471380
26	NC_029256.1	95527	G	A	36	12	48	141	16	3	19	0	0.15789474	0.2500000	0.7761194	-0.0921052632
27	NC_029256.1	95546	A	T	32	14	46	160	19	3	22	0	0.13636364	0.3043478	0.7500000	-0.1679841897
28	NC_029256.1	96265	C	G	31	11	42	160	16	6	22	101	0.27272727	0.2619048	0.7343750	0.0108225108
29	NC_029256.1	96290	T	C	28	11	39	142	14	6	20	96	0.30000000	0.2820513	0.7118644	0.0179487179
30	NC_029256.1	96299	T	C	24	10	34	143	12	6	18	101	0.33333333	0.2941176	0.6923077	0.0392156863
31	NC_029256.1	96306	GCA	ACG	21	7	28	105	8	4	12	60	0.33333333	0.2500000	0.7250000	0.0833333333
32	NC_029256.1	96321	A	G	23	5	28	43	8	4	12	34	0.33333333	0.1785714	0.7750000	0.1547619048
33	NC_029256.1	96328	C	T	20	5	25	55	6	4	10	68	0.40000000	0.2000000	0.7428571	0.2000000000
34	NC_029256.1	96351	C	T	21	4	25	23	5	3	8	33	0.37500000	0.1600000	0.7878788	0.2150000000

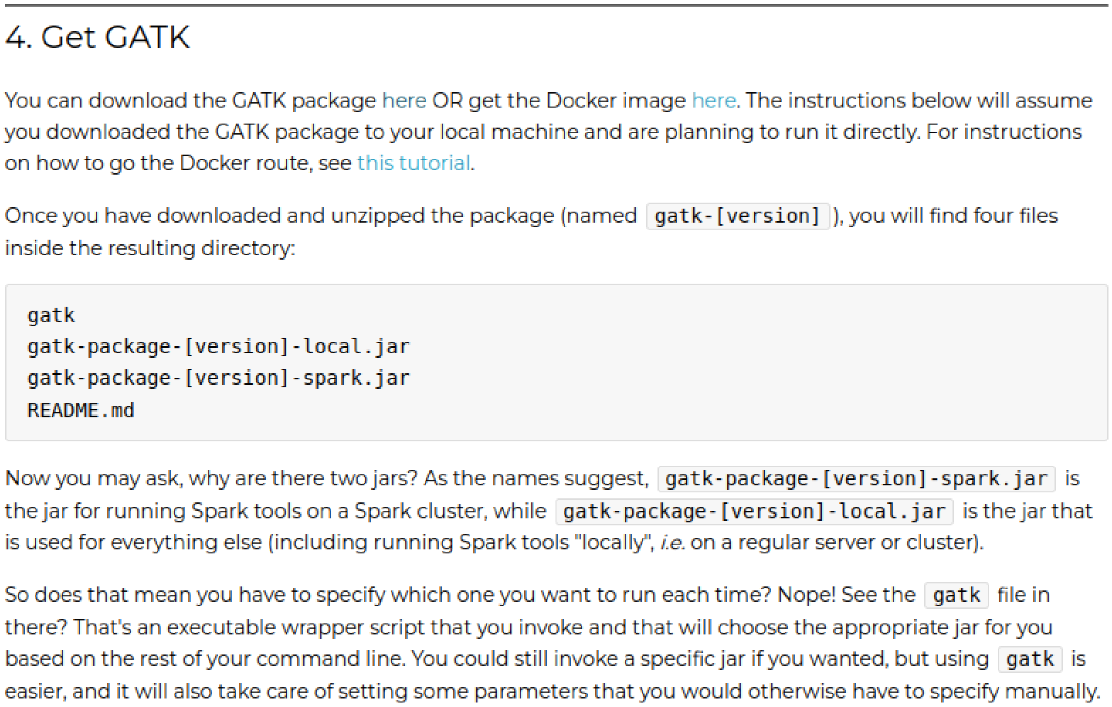
Showing 1 to 34 of 1,714,745 entries, 16 total columns

GATK

Method 3 (Best in my opinion)

Calling **importFromGATK** This method requires 4 arguments, **a vcf file**, **highBulk**, **lowBulk**, and **chromlist**. If you do not have the software on your machine, first visit this website. https://gatk.broadinstitute.org/hc/en-us/articles/360036194592-Getting-started-with-GATK4 Go to section 4 and click the first from left to right **here** hyperlink

```
 Chroms <- c("NC_029256.1","NC_029257.1","NC_029258.1","NC_029259.1","NC_029260.1","NC_
↪029261.1","NC_029262.1","NC_029263.1","NC_029264.1","NC_029265.1","NC_029266.1","NC_
↪029267.1")

df <- importFromGATK(file = "Hall.table", highBulk = "ET-pool-385", lowBulk = "ES-pool-
↪430", chromlist = Chroms)

**Method 1 is the most biased and therefore cuts out more SNPs than Methods 2 & 3 which␣
↪produce nearly identical SNP sets.**
```



4. Get GATK

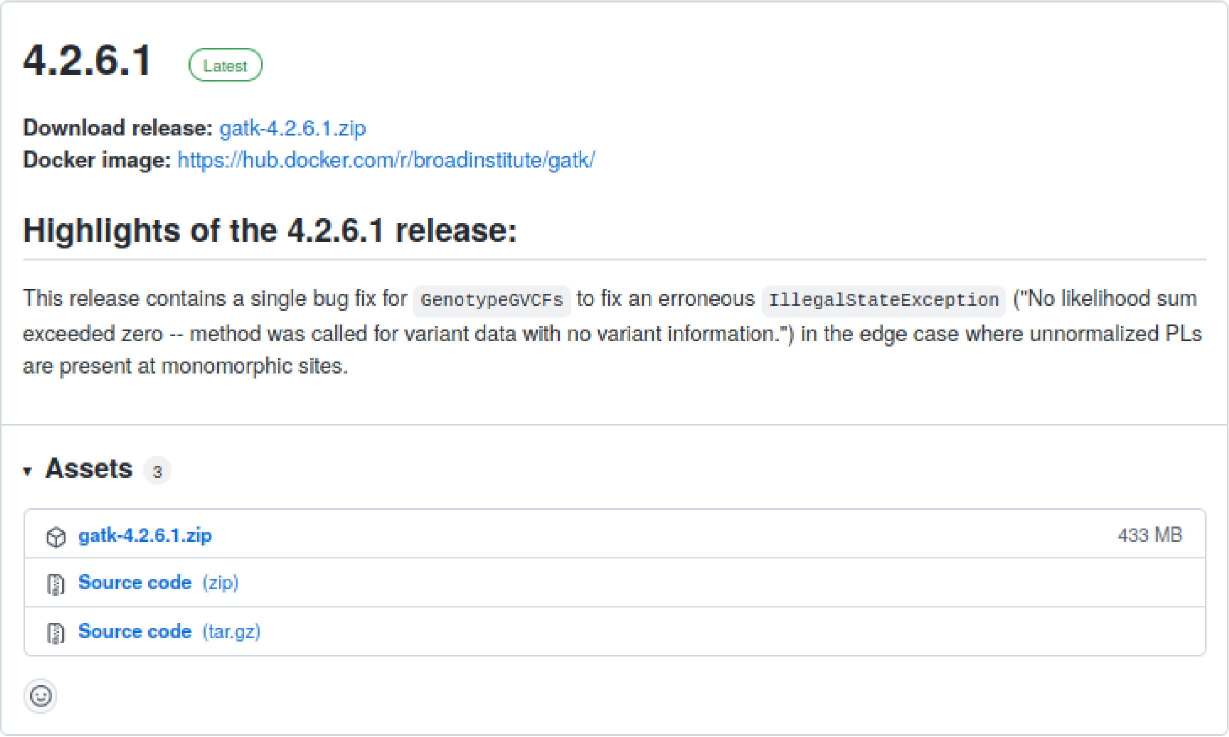
You can download the GATK package here OR get the Docker image here. The instructions below will assume you downloaded the GATK package to your local machine and are planning to run it directly. For instructions on how to go the Docker route, see this tutorial.

Once you have downloaded and unzipped the package (named `gatk-[version]`), you will find four files inside the resulting directory:

```
gatk
gatk-package-[version]-local.jar
gatk-package-[version]-spark.jar
README.md
```

Now you may ask, why are there two jars? As the names suggest, `gatk-package-[version]-spark.jar` is the jar for running Spark tools on a Spark cluster, while `gatk-package-[version]-local.jar` is the jar that is used for everything else (including running Spark tools "locally", *i.e.* on a regular server or cluster).

So does that mean you have to specify which one you want to run each time? Nope! See the `gatk` file in there? That's an executable wrapper script that you invoke and that will choose the appropriate jar for you based on the rest of your command line. You could still invoke a specific jar if you wanted, but using `gatk` is easier, and it will also take care of setting some parameters that you would otherwise have to specify manually.



4.2.6.1

Download release: gatk-4.2.6.1.zip
Docker image: https://hub.docker.com/r/broadinstitute/gatk/

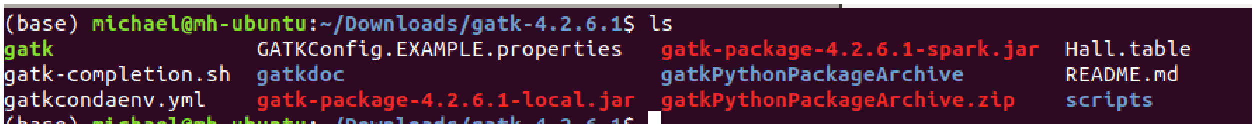
Highlights of the 4.2.6.1 release:

This release contains a single bug fix for `GenotypeGVCFs` to fix an erroneous `IllegalStateException` ("No likelihood sum exceeded zero -- method was called for variant data with no variant information.") in the edge case where unnormalized PLs are present at monomorphic sites.

▾ **Assets** 3

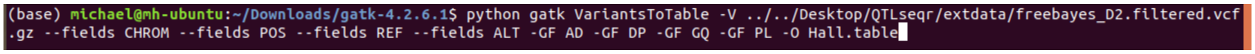
gatk-4.2.6.1.zip	433 MB
Source code (zip)	
Source code (tar.gz)	

Fig. 2: **Navigate to the folder containg gatk executable python script**



```
(base) michael@mh-ubuntu:~/Downloads/gatk-4.2.6.1$ ls
gatk                 GATKConfig.EXAMPLE.properties    gatk-package-4.2.6.1-spark.jar  Hall.table
gatk-completion.sh   gatkdoc                          gatkPythonPackageArchive        README.md
gatkcondaenv.yml     gatk-package-4.2.6.1-local.jar   gatkPythonPackageArchive.zip    scripts
(base) michael@mh-ubuntu:~/Downloads/gatk-4.2.6.1$
```

Fig. 3: Call **VariantsToTable** sub executable program with all appropriate flags



```
(base) michael@mh-ubuntu:~/Downloads/gatk-4.2.6.1$ python gatk VariantsToTable -V ../../Desktop/QTLseqr/extdata/freebayes_D2.filtered.vcf
.gz --fields CHROM --fields POS --fields REF --fields ALT -GF AD -GF DP -GF GQ -GF PL -O Hall.table
```

Fig. 4: This should produce a file called **Hall.table**

2.8 Input Fields

```
#Set High bulk and Low bulk sample names and parser generated file name
#The file name is generated from the QTLParser_1_MH function in line 119

HighBulk <- "ET-pool-385"
LowBulk <- "ES-pool-430"
file <- "Hall.csv"

#Choose which chromosomes/contigs will be included in the analysis,

Chroms <- c("NC_029256.1","NC_029257.1","NC_029258.1","NC_029259.1","NC_029260.1","NC_
↪029261.1","NC_029262.1","NC_029263.1","NC_029264.1","NC_029265.1","NC_029266.1","NC_
↪029267.1")
```

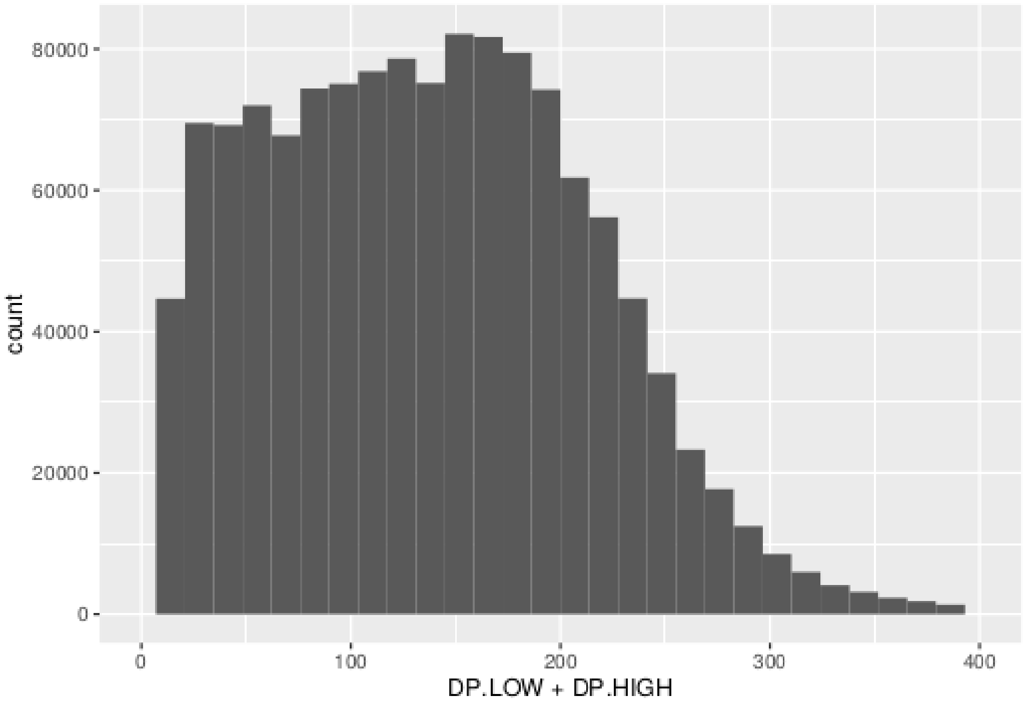
2.9 importFromTable

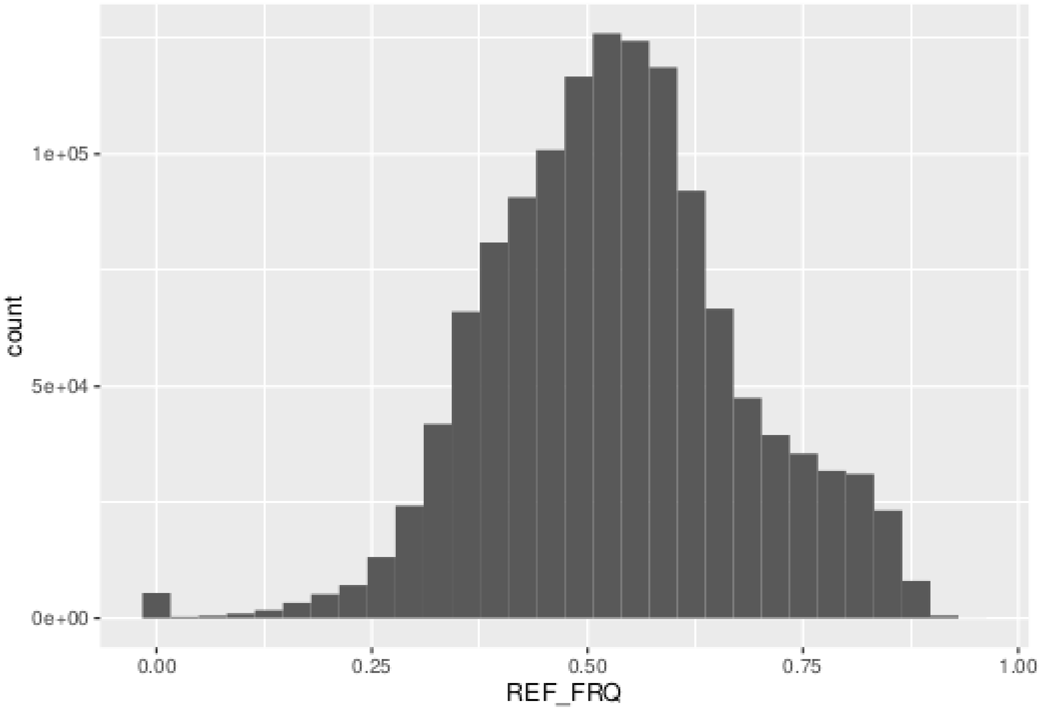
```
df <-
  importFromTable(
    file = file,
    highBulk = HighBulk,
    lowBulk = LowBulk,
    chromList = Chroms
  )
```

Histograms

```
#plot histograms associated with filtering arguments such as mamximum and minumum Total␣
↪Depths and reference Allele Frequency to determine cut off        values
ggplot(data =df) + geom_histogram(aes(x = DP.LOW + DP.HIGH)) + xlim(0,400)
ggsave(filename = "Depth_Histogram.png",plot=last_plot())
```

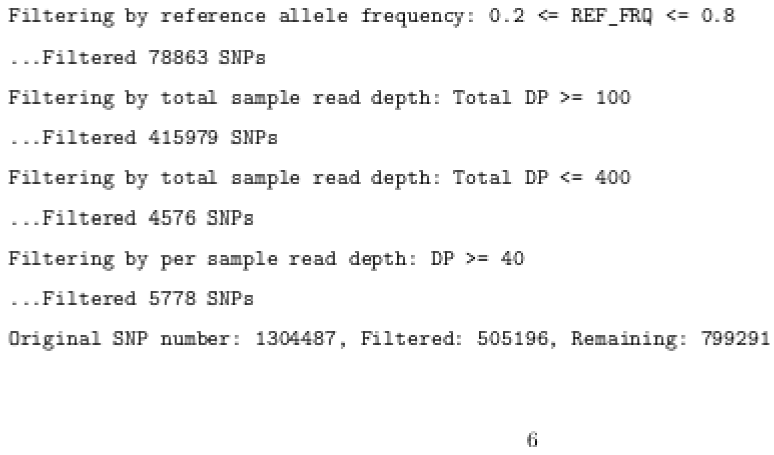
```
ggplot(data = df) + geom_histogram(aes(x = REF_FRQ))
ggsave(filename = "Ref_Freq_Histogram.png",plot = last_plot())
```


80000
60000
40000
20000
0
count
0
100
200
300
400
DP.LOW + DP.HIGH


1e+05
5e+04
0e+00
count
0.00
0.25
0.50
0.75
1.00
REF_FRQ

2.10 filterSNPs

```
#Filter SNPs based on some criteria
df_filt <- filterSNPs( SNPset = df,
refAlleleFreq = 0.20, minTotalDepth = 100, maxTotalDepth = 400,
minSampleDepth = 40,
# minGQ = 0 )
```

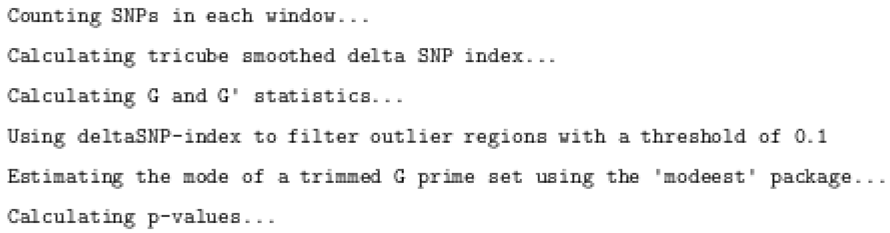


```
Filtering by reference allele frequency: 0.2 <= REF_FRQ <= 0.8

...Filtered 78863 SNPs

Filtering by total sample read depth: Total DP >= 100

...Filtered 415979 SNPs

Filtering by total sample read depth: Total DP <= 400

...Filtered 4576 SNPs

Filtering by per sample read depth: DP >= 40

...Filtered 5778 SNPs

Original SNP number: 1304487, Filtered: 505196, Remaining: 799291
```

6

2.11 runGprimeAnalysis_MH

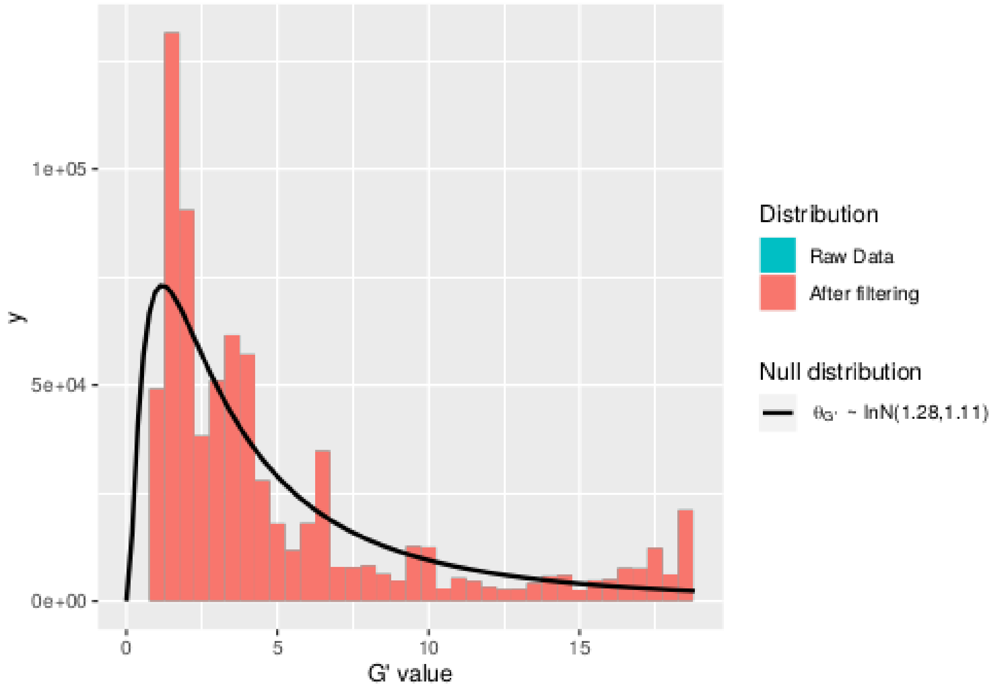
```
#Run G' analysis
df_filt<-runGprimeAnalysis(
  SNPset = df_filt,
  windowSize = 1e6,
  outlierFilter = "deltaSNP",
  filterThreshold = 0.1)
```



```
Counting SNPs in each window...

Calculating tricube smoothed delta SNP index...

Calculating G and G' statistics...

Using deltaSNP-index to filter outlier regions with a threshold of 0.1

Estimating the mode of a trimmed G prime set using the 'modeest' package...

Calculating p-values...
```

2.12 plotGprimeDist_MH

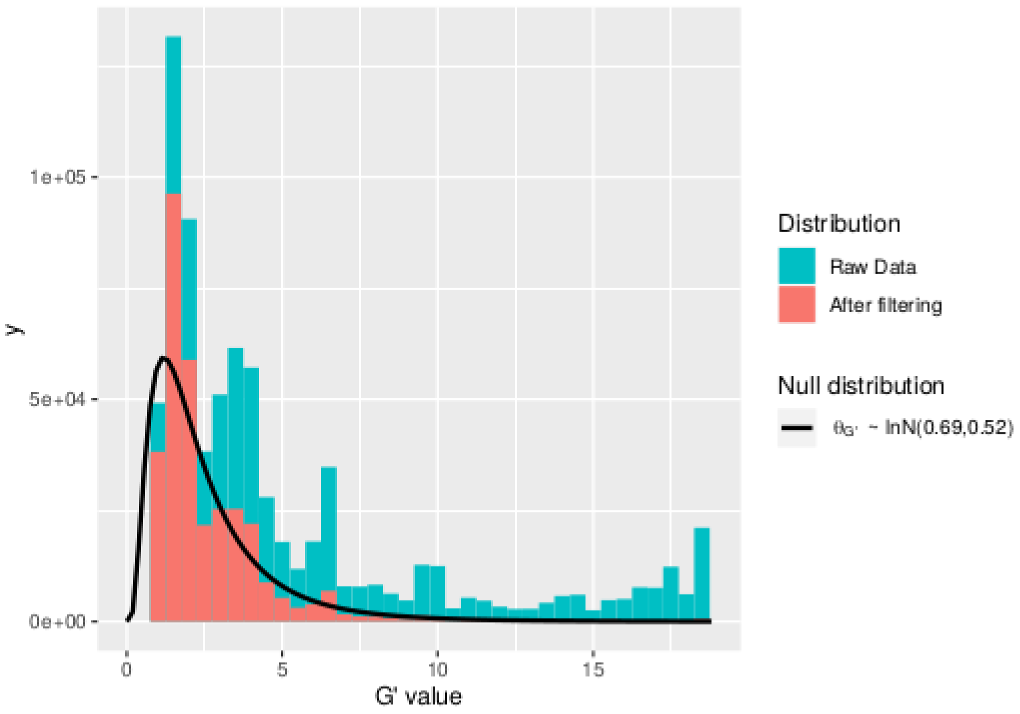
```
#The plot reveals a skewed G Prime statistic with a really small variance. Perhaps it is␣
↪due to the small number of variants called.
#In addition, Hampels outlier filter in the second argument, can also be changed to
↪"deltaSNP"

plotGprimeDist(SNPset = df_filt, outlierFilter = "Hampel")
```



```
#We can see raw data before and after our filtering step

plotGprimeDist(SNPset = df_filt, outlierFilter = "deltaSNP",filterThreshold = 0.1)
```

2.13 runQTLseqAnalysis_MH

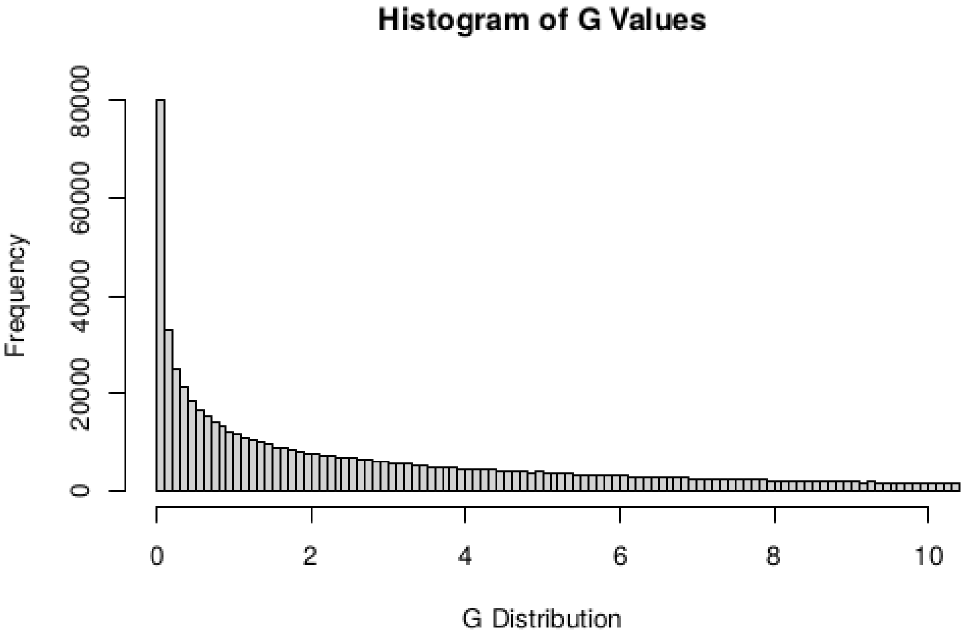
```
#Run QTLseq analysis
df_filt2 <- runQTLseqAnalysis(
  SNPset = df_filt,
  windowSize = 1e6,
  popStruc = "F2",
  bulkSize = c(430, 385),
  replications = 10000,
  intervals = c(95, 99)
)
```



```
Counting SNPs in each window...

Calculating tricube smoothed delta SNP index...

Returning the following two sided confidence intervals: 95, 99

Variable 'depth' not defined, using min and max depth from data: 40-197

Assuming bulks selected from F2 population, with 430 and 385 individuals per bulk.

Simulating 10000 SNPs with reads at each depth: 40-197

Keeping SNPs with >= 0.3 SNP-index in both simulated bulks

Joining, by = "tricubeDP"
```

Plot G Statistic Distribution as a Histogram

```
hist(df_filt2$G,breaks = 950,xlim = c(0,10),xlab = "G Distribution",main = "Histogram of␣
↪G Values")
```



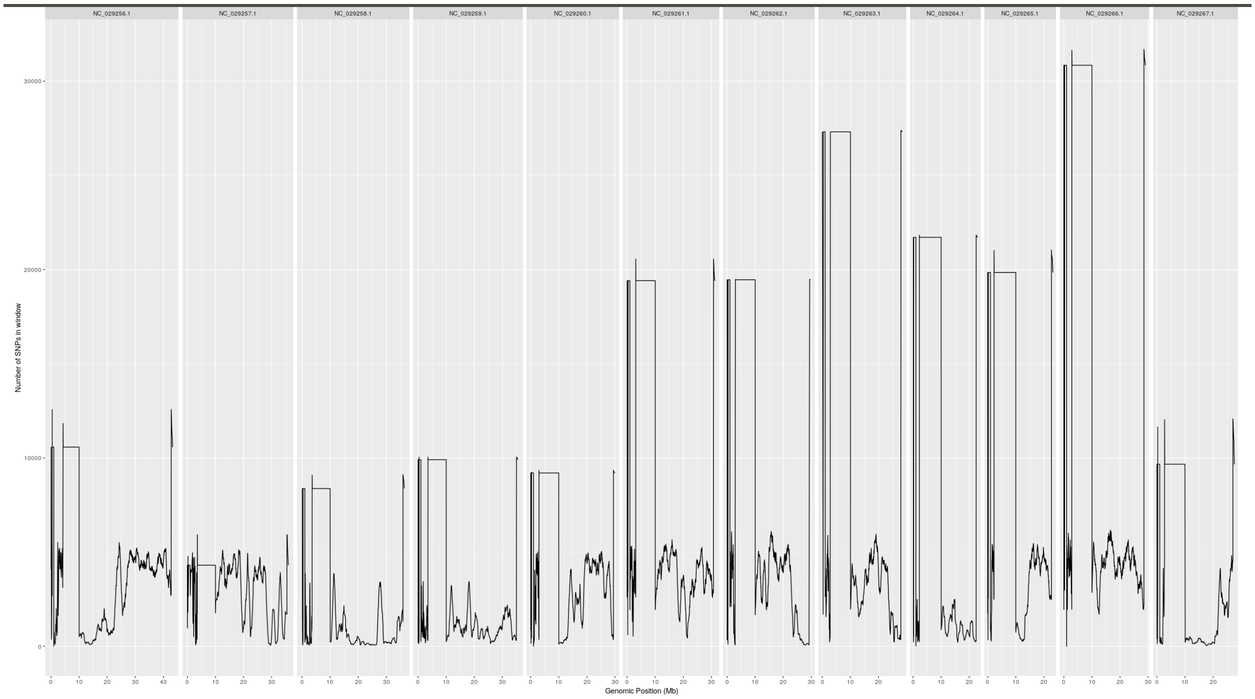
2.14 plotQTLStats

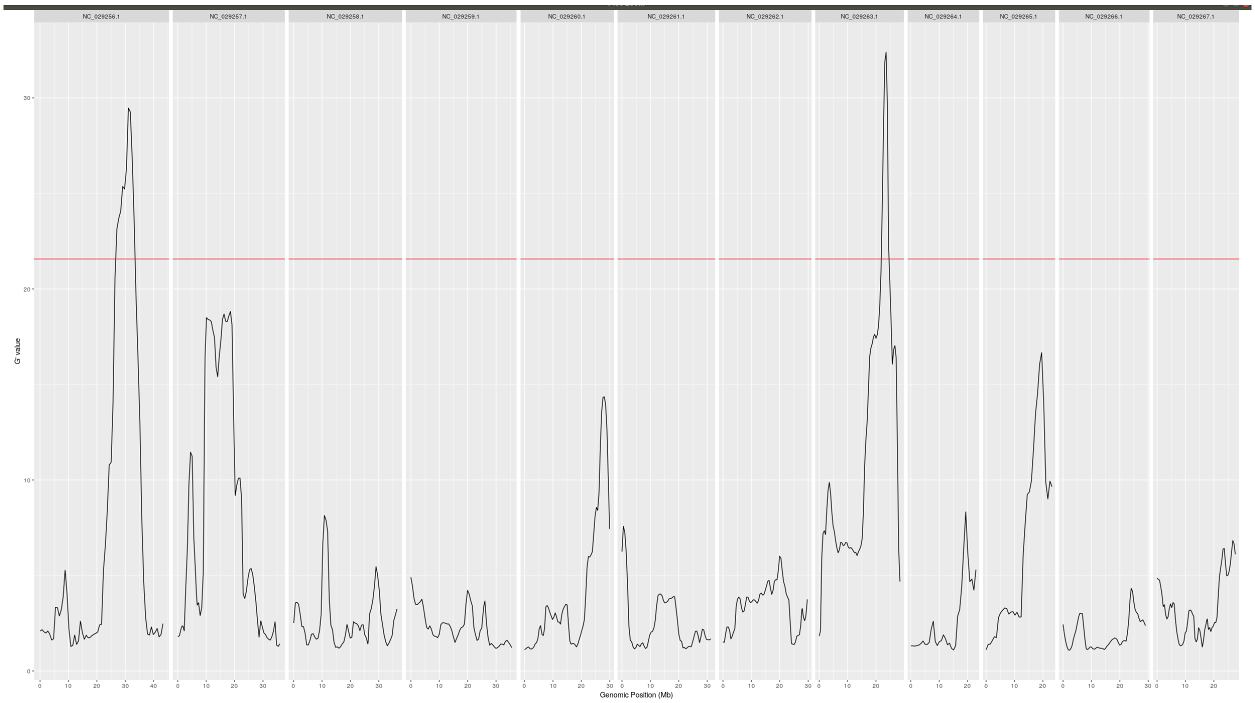
nSNPs

```
#Plot Snps as a function of chromosome and position values

plotQTLStats(SNPset = df_filt2, var = "nSNPs")
ggsave(filename = "nSNPs.png",plot = last_plot())
```

Gprime

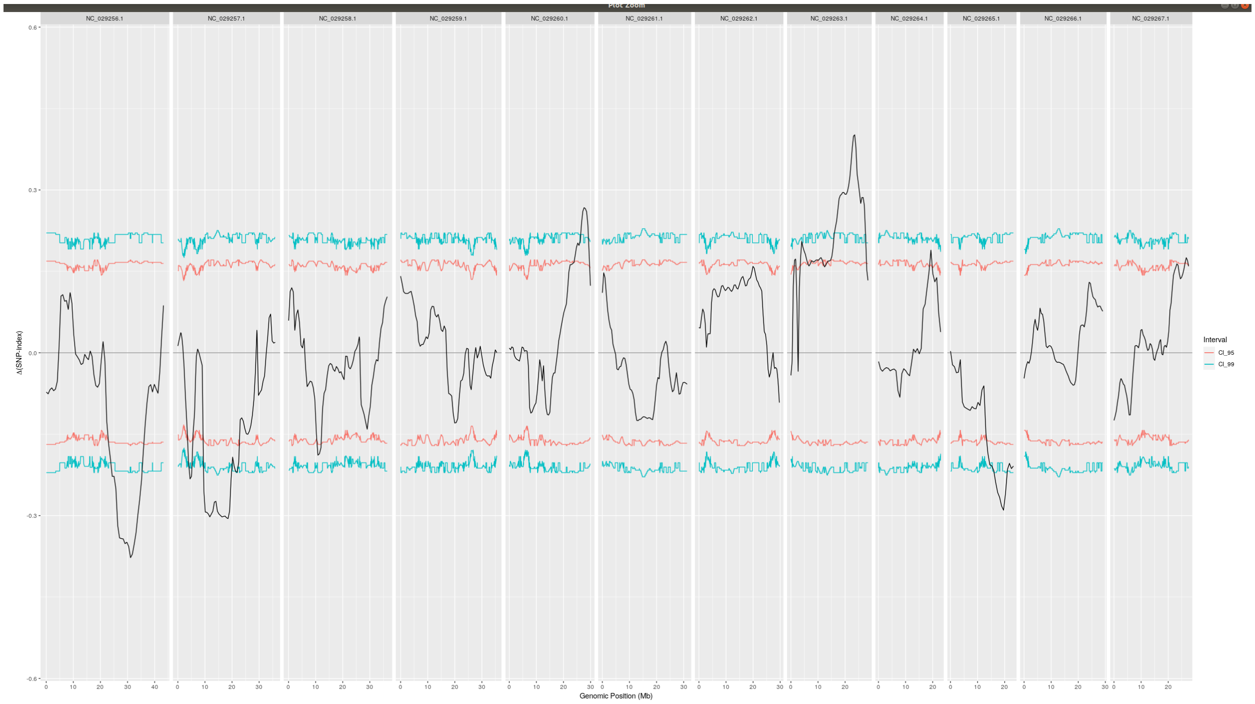
```
#Using QTLStats funciton plot Gprime Statistic with False Discovery Rate Threhshold as a␣
↪third argument boolean operator as TRUE. The q value is used as FDR threshold null␣
↪value is 0.05%.

plotQTLStats(SNPset = df_filt, var = "Gprime", plotThreshold = TRUE, q = 0.01)
ggsave(filename = "GPrime.png",plot = last_plot())
```


NC_029256.1
NC_029257.1
NC_029258.1
NC_029259.1
NC_029260.1
NC_029261.1
NC_029262.1
NC_029263.1
NC_029264.1
NC_029265.1
NC_029266.1
NC_029267.1
Number of SNPs in window
30000
20000
10000
0
0 10 20 30 40
Genomic Position (Mb)


NC_029256.1
NC_029257.1
NC_029258.1
NC_029259.1
NC_029260.1
NC_029261.1
NC_029262.1
NC_029263.1
NC_029264.1
NC_029265.1
NC_029266.1
NC_029267.1
G' value
30
20
10
0
0 10 20 30 40
Genomic Position (Mb)

deltaSNP

```
#Again using plotQTLStats change second argument varaible to deltaSNP and plot.

plotQTLStats(SNPset = df_filt2, var = "deltaSNP", plotIntervals  = TRUE)
ggsave(filename = "DeltaSNPInterval.png",plot = last_plot())
```

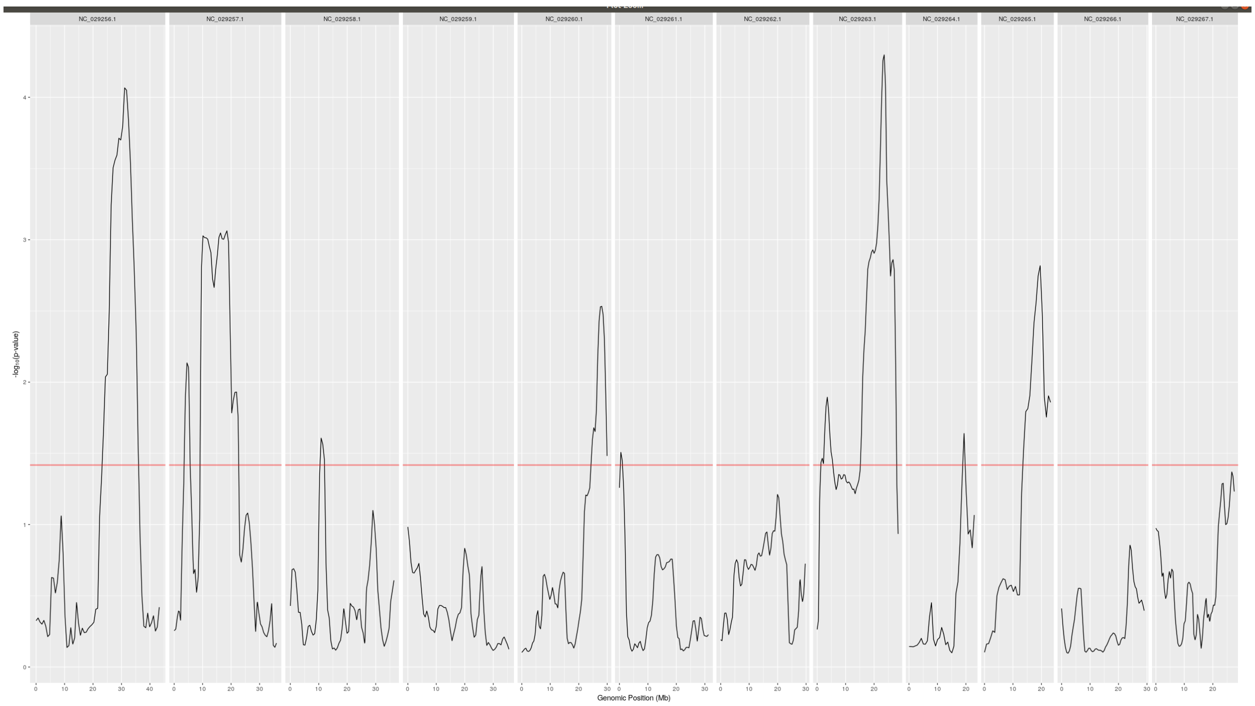


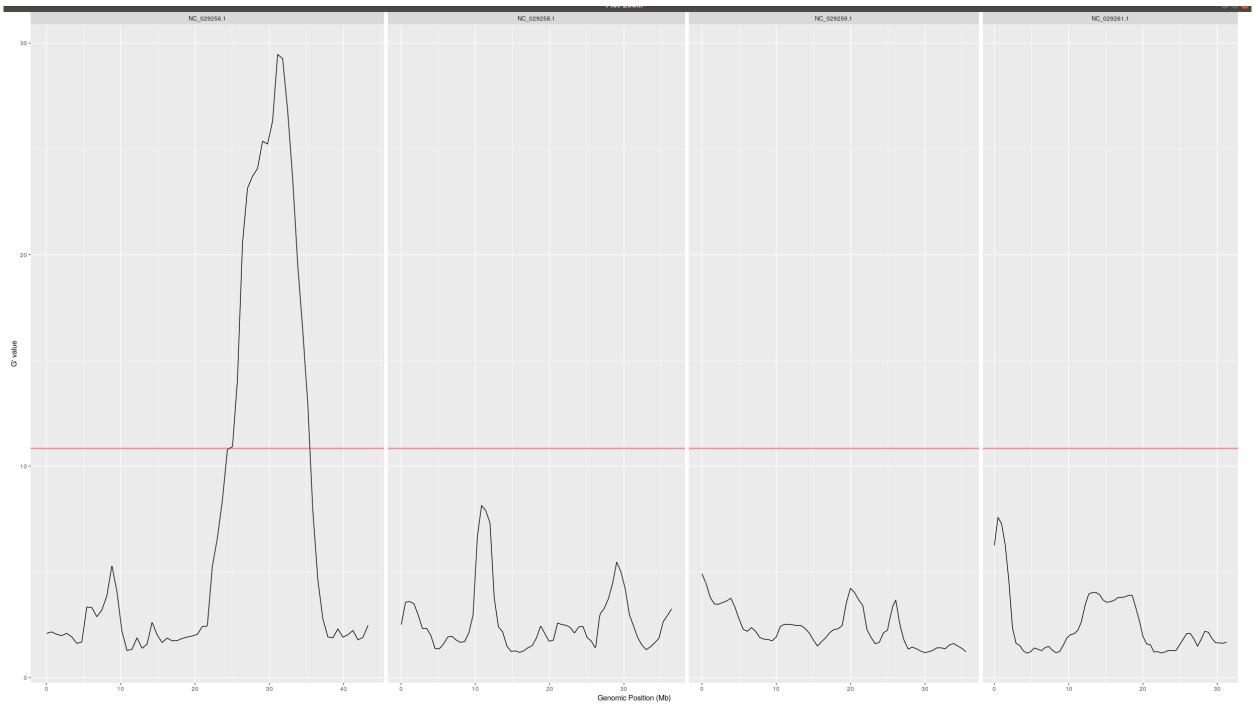
negLog10Pval

```
#Finally with plotQTLStats plot negLog10Pval

plotQTLStats(SNPset = df_filt, var = "negLog10Pval",plotThreshold = TRUE,q=0.15)
ggsave(filename = "negLog10Pval.png",plot = last_plot())
```

Gprime Subset

```
#Add subset argument to focus on particular chromosomes one, three, four, and six.
#The reason is due to signficant QTL regions
plotQTLStats(SNPset = df_filt, var = "Gprime",plotThreshold = TRUE,q=0.05,subset = c("NC_
↪029256.1","NC_029258.1","NC_029259.1","NC_029261.1"))
```


Plot Zoom
NC_029256.1
NC_029257.1
NC_029258.1
NC_029259.1
NC_029260.1
NC_029261.1
NC_029262.1
NC_029263.1
NC_029264.1
NC_029265.1
NC_029266.1
NC_029267.1
$-\log_{10}$(p-value)
0
1
2
3
4
0
10
20
30
40
Genomic Position (Mb)


Plot Zoom
NC_029256.1
NC_029258.1
NC_029259.1
NC_029261.1
G' value
0
10
20
30
40
Genomic Position (Mb)

2.15 rMVP Package

SNP Densities

```
#install.packages("rMVP")
library(rMVP)
sample<-"Semi_Dwarfism_in_Sorghum"
pathtosample <- "/home/michael/Desktop/QTLseqr/extdata/subset_freebayes_D2.filtered.vcf.
↪gz"
out<- paste0("mvp.",sample,".vcf")
memo<-paste0(sample)
dffile<-paste0("mvp.",sample,".vcf.geno.map")

message("Making MVP data S1")
MVP.Data(fileVCF=pathtosample,
      #filePhe="Phenotype.txt",
      fileKin=FALSE,
      filePC=FALSE,
      out=out)

message("Reading MVP Data S1")
df <- read.table(file = dffile, header=TRUE)
message("Making SNP Density Plots")
MVP.Report.Density(df[,c(1:3)], bin.size = 1000000, col = c("blue", "yellow", "red"),␣
↪memo = memo, file.type = "jpg", dpi=300)
```

2.16 Export summary CSV

```
QTLTable(SNPset = df_filt, alpha = 0.01, export = TRUE, fileName = "my_BSA_QTL.csv")
```

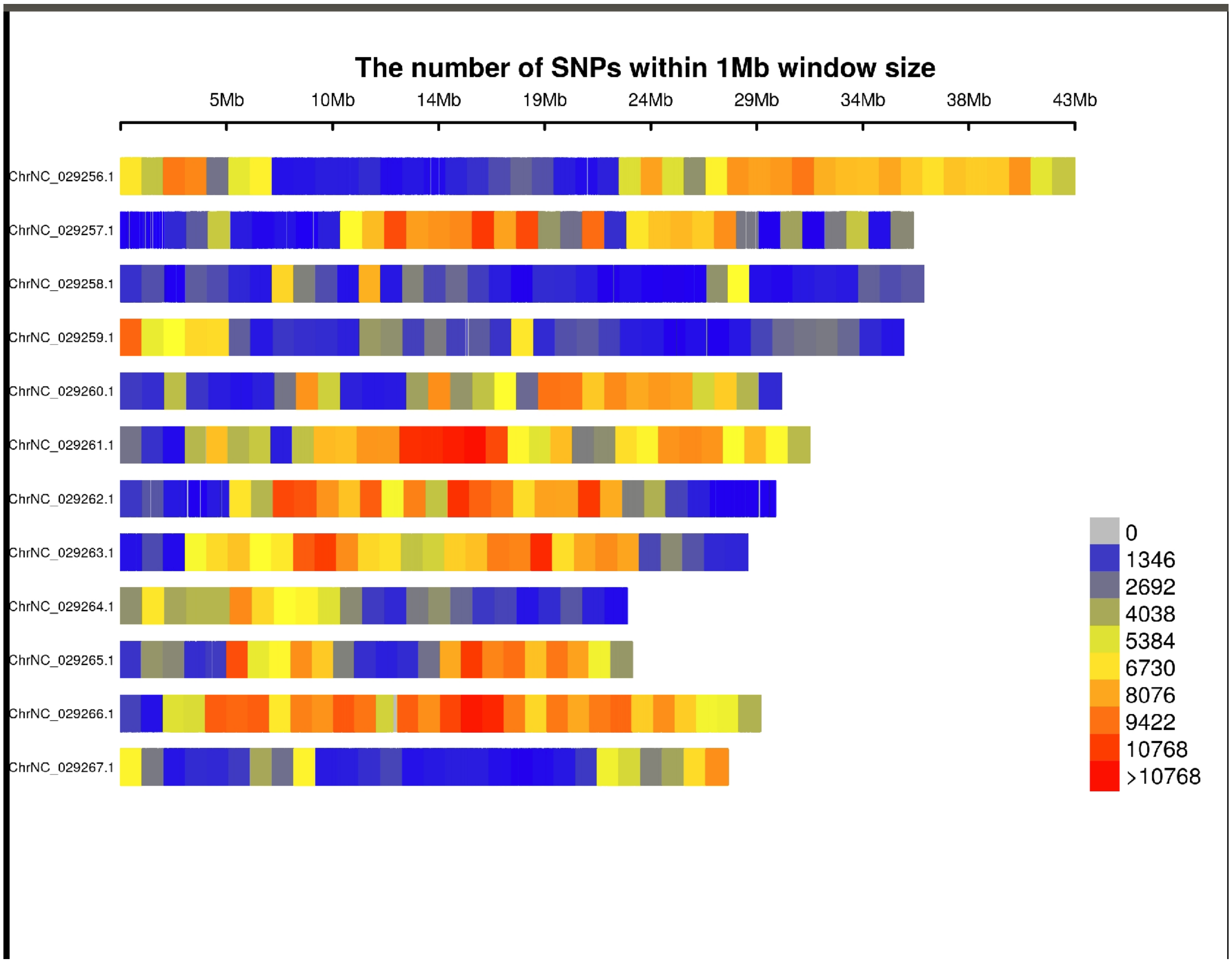
Preview the Summary QTL

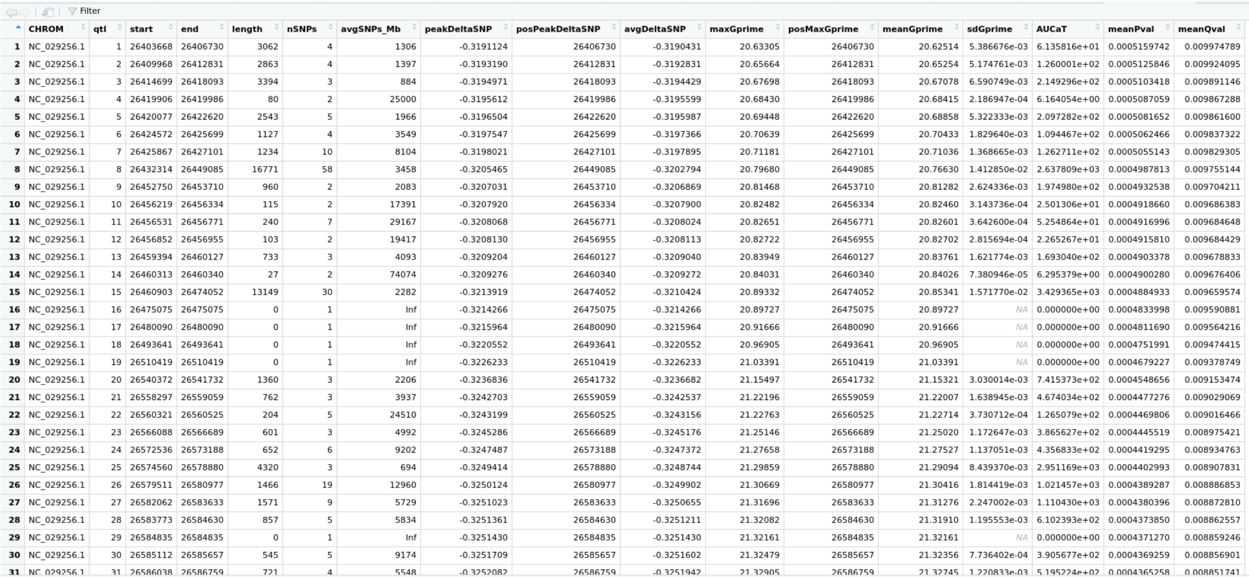
2.17 Theory

Contigency Table

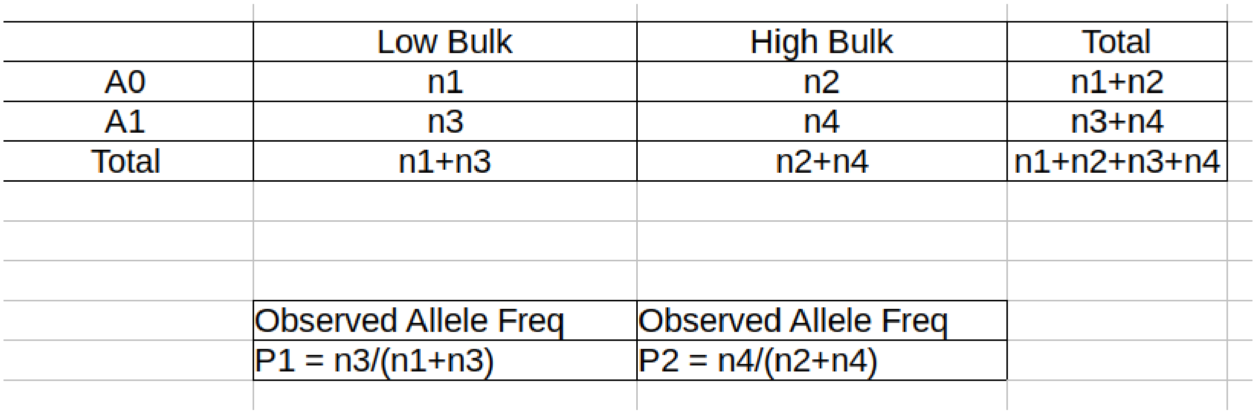
Obs_Allel_Freq

```
#Use the function to plot allele frequencies per chromosome
#Second argument size specifes size of scalar factor on nSNPs and if you have a␣
↪relatively small SNP set .001 is a good startin point otherwise set to 1
Obs_Allele_Freq(SNPSet = df_filt, size = 1)
```





	CHROM	qtl	start	end	length	nSNPs	avgSNPs_Mb	peakDeltaSNP	posPeakDeltaSNP	avgDeltaSNP	maxGprime	posMaxGprime	meanGprime	sdGprime	AUCaT	meanPval	meanQval
1	NC_029256.1	1	26403668	26406730	3062	4	1306	-0.3191124	26406730	-0.3190431	20.63305	26406730	20.62514	5.386676e-03	6.135816e+01	0.0005159742	0.009974789
2	NC_029256.1	2	26409968	26412831	2863	4	1397	-0.3193190	26412831	-0.3192831	20.65664	26412831	20.65254	5.174761e-03	1.260001e+02	0.0005125846	0.009924095
3	NC_029256.1	3	26414699	26418093	3394	3	884	-0.3194971	26418093	-0.3194429	20.67698	26418093	20.67078	6.590749e-03	2.149296e+02	0.0005103418	0.009891146
4	NC_029256.1	4	26419906	26419986	80	2	25000	-0.3195612	26419986	-0.3195599	20.68430	26419986	20.68415	2.186947e-04	6.164054e+00	0.0005087059	0.009867288
5	NC_029256.1	5	26420077	26422620	2543	5	1966	-0.3196504	26422620	-0.3195987	20.69448	26422620	20.68858	5.322333e-03	2.097282e+02	0.0005081652	0.009861600
6	NC_029256.1	6	26424572	26425699	1127	4	3549	-0.3197547	26425699	-0.3197366	20.70639	26425699	20.70433	1.829640e-03	1.094467e+02	0.0005062466	0.009837322
7	NC_029256.1	7	26425867	26427101	1234	10	8104	-0.3198021	26427101	-0.3197895	20.71181	26427101	20.71036	1.368665e-03	1.262711e+02	0.0005055143	0.009829305
8	NC_029256.1	8	26432314	26449085	16771	58	3458	-0.3205465	26449085	-0.3202794	20.79680	26449085	20.76630	1.412850e-02	2.637809e+03	0.0004987813	0.009755144
9	NC_029256.1	9	26452750	26453710	960	2	2083	-0.3207031	26453710	-0.3206869	20.81468	26453710	20.81282	2.624336e-03	1.974980e+02	0.0004932538	0.009704211
10	NC_029256.1	10	26456219	26456334	115	2	17391	-0.3207920	26456334	-0.3207900	20.82482	26456334	20.82460	3.143736e-04	2.501306e+01	0.0004918660	0.009686383
11	NC_029256.1	11	26456531	26456771	240	7	29167	-0.3208068	26456771	-0.3208024	20.82651	26456771	20.82601	3.642600e-04	5.254864e+01	0.0004916996	0.009684648
12	NC_029256.1	12	26456852	26456955	103	2	19417	-0.3208130	26456955	-0.3208113	20.82722	26456955	20.82702	2.815694e-04	2.265267e+01	0.0004915810	0.009684429
13	NC_029256.1	13	26459394	26460127	733	3	4093	-0.3209204	26460127	-0.3209040	20.83949	26460127	20.83761	1.621774e-03	1.693040e+02	0.0004903378	0.009678833
14	NC_029256.1	14	26460313	26460340	27	2	74074	-0.3209276	26460340	-0.3209272	20.84031	26460340	20.84026	7.380946e-05	6.295379e+00	0.0004900280	0.009676406
15	NC_029256.1	15	26460903	26474052	13149	30	2282	-0.3213919	26474052	-0.3210424	20.89332	26474052	20.85341	1.571770e-02	3.429365e+03	0.0004884933	0.009659574
16	NC_029256.1	16	26475075	26475075	0	1	Inf	-0.3214266	26475075	-0.3214266	20.89727	26475075	20.89727	NA	0.000000e+00	0.0004833998	0.009590881
17	NC_029256.1	17	26480090	26480090	0	1	Inf	-0.3215964	26480090	-0.3215964	20.91666	26480090	20.91666	NA	0.000000e+00	0.0004811690	0.009564216
18	NC_029256.1	18	26493641	26493641	0	1	Inf	-0.3220552	26493641	-0.3220552	20.96905	26493641	20.96905	NA	0.000000e+00	0.0004751991	0.009474415
19	NC_029256.1	19	26510419	26510419	0	1	Inf	-0.3226233	26510419	-0.3226233	21.03391	26510419	21.03391	NA	0.000000e+00	0.0004679227	0.009378749
20	NC_029256.1	20	26540372	26541732	1360	3	2206	-0.3236836	26541732	-0.3236682	21.15497	26541732	21.15321	3.030014e-03	7.415373e+02	0.0004548656	0.009153474
21	NC_029256.1	21	26558297	26559059	762	3	3937	-0.3242703	26559059	-0.3242537	21.22196	26559059	21.22007	1.638945e-03	4.674034e+02	0.0004477276	0.009029069
22	NC_029256.1	22	26560321	26560525	204	5	24510	-0.3243199	26560525	-0.3243156	21.22763	26560525	21.22714	3.730712e-04	1.265079e+02	0.0004469806	0.009016466
23	NC_029256.1	23	26566088	26566689	601	3	4992	-0.3245286	26566689	-0.3245176	21.25146	26566689	21.25020	1.172647e-03	3.865627e+02	0.0004445519	0.008975421
24	NC_029256.1	24	26572536	26573188	652	6	9202	-0.3247487	26573188	-0.3247372	21.27658	26573188	21.27527	1.137051e-03	4.356833e+02	0.0004419295	0.008934763
25	NC_029256.1	25	26574560	26578880	4320	3	694	-0.3249414	26578880	-0.3248744	21.29859	26578880	21.29094	8.439370e-03	2.951169e+03	0.0004402993	0.008907831
26	NC_029256.1	26	26579511	26580977	1466	19	12960	-0.3250124	26580977	-0.3249902	21.30669	26580977	21.30416	1.814419e-03	1.021457e+03	0.0004389287	0.008886853
27	NC_029256.1	27	26582062	26583633	1571	9	5729	-0.3251023	26583633	-0.3250655	21.31696	26583633	21.31276	2.247002e-03	1.110430e+03	0.0004380396	0.008872810
28	NC_029256.1	28	26583773	26584630	857	5	5834	-0.3251361	26584630	-0.3251211	21.32082	26584630	21.31910	1.195553e-03	6.102393e+02	0.0004373850	0.008862557
29	NC_029256.1	29	26584835	26584835	0	1	Inf	-0.3251430	26584835	-0.3251430	21.32161	26584835	21.32161	NA	0.000000e+00	0.0004371270	0.008859246
30	NC_029256.1	30	26585112	26585657	545	5	9174	-0.3251709	26585657	-0.3251602	21.32479	26585657	21.32356	7.736402e-04	3.905677e+02	0.0004369259	0.008856901
31	NC_029256.1	31	26586038	26586759	721	4	5548	-0.3252082	26586759	-0.3251942	21.32905	26586759	21.32745	1.220833e-03	5.195224e+02	0.0004365258	0.008851741

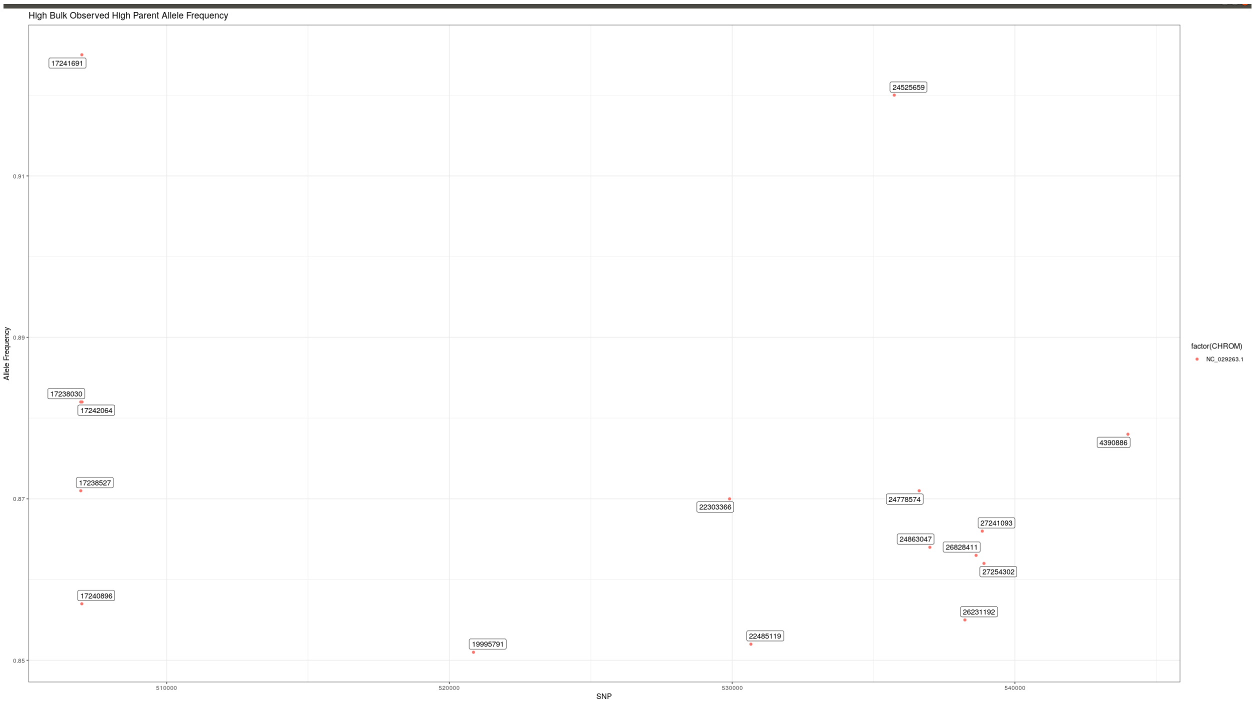


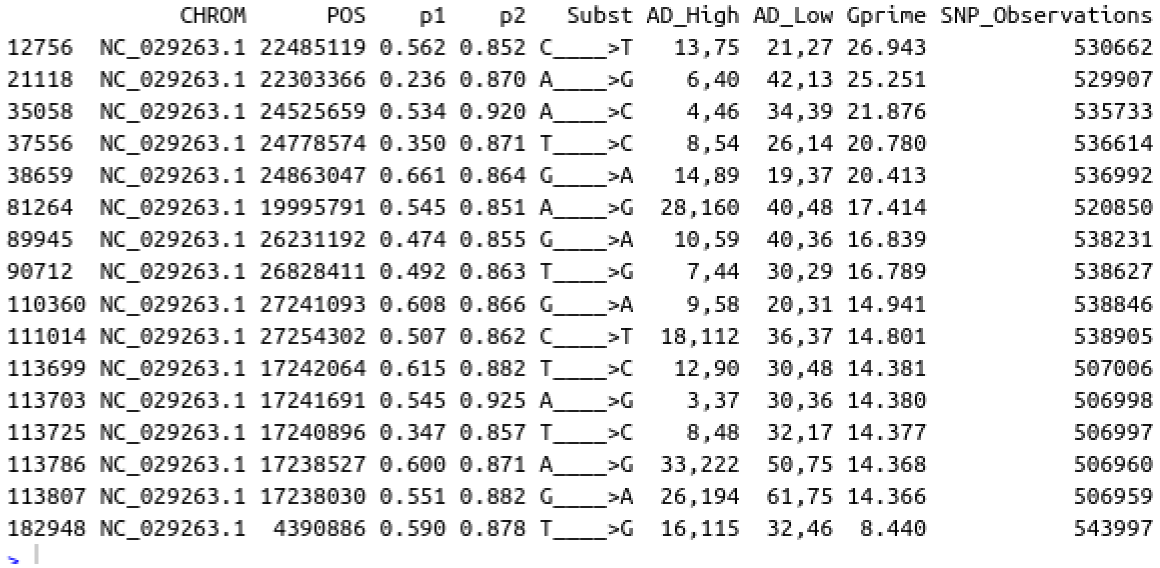
	Low Bulk	High Bulk	Total
A0	n1	n2	n1+n2
A1	n3	n4	n3+n4
Total	n1+n3	n2+n4	n1+n2+n3+n4

Observed Allele Freq	Observed Allele Freq
P1 = n3/(n1+n3)	P2 = n4/(n2+n4)

Obs_Allele_Freq2

```
#Use the function to plot allele frequencies per chromosome
#Second argument size specifes size of scalar factor on nSNPs and if you have a relatively small SNP set .001 is a good startin point otherwise set to 1
##Use the function to investigate chromosomal region of interest
Obs_Allele_Freq2(SNPSet = df_filt, ChromosomeValue = "NC_029263.1", threshold = .85)
```





```
             CHROM      POS    p1    p2   Subst AD_High AD_Low Gprime SNP_Observations
12756  NC_029263.1 22485119 0.562 0.852 C____>T   13,75  21,27 26.943           530662
21118  NC_029263.1 22303366 0.236 0.870 A____>G    6,40  42,13 25.251           529907
35058  NC_029263.1 24525659 0.534 0.920 A____>C    4,46  34,39 21.876           535733
37556  NC_029263.1 24778574 0.350 0.871 T____>C    8,54  26,14 20.780           536614
38659  NC_029263.1 24863047 0.661 0.864 G____>A   14,89  19,37 20.413           536992
81264  NC_029263.1 19995791 0.545 0.851 A____>G  28,160  40,48 17.414           520850
89945  NC_029263.1 26231192 0.474 0.855 G____>A   10,59  40,36 16.839           538231
90712  NC_029263.1 26828411 0.492 0.863 T____>G    7,44  30,29 16.789           538627
110360 NC_029263.1 27241093 0.608 0.866 G____>A    9,58  20,31 14.941           538846
111014 NC_029263.1 27254302 0.507 0.862 C____>T  18,112  36,37 14.801           538905
113699 NC_029263.1 17242064 0.615 0.882 T____>C   12,90  30,48 14.381           507006
113703 NC_029263.1 17241691 0.545 0.925 A____>G    3,37  30,36 14.380           506998
113725 NC_029263.1 17240896 0.347 0.857 T____>C    8,48  32,17 14.377           506997
113786 NC_029263.1 17238527 0.600 0.871 A____>G  33,222  50,75 14.368           506960
113807 NC_029263.1 17238030 0.551 0.882 G____>A  26,194  61,75 14.366           506959
182948 NC_029263.1  4390886 0.590 0.878 T____>G  16,115  32,46  8.440           543997
> 
```

Total Coverage and Expected Allelic Frequencies

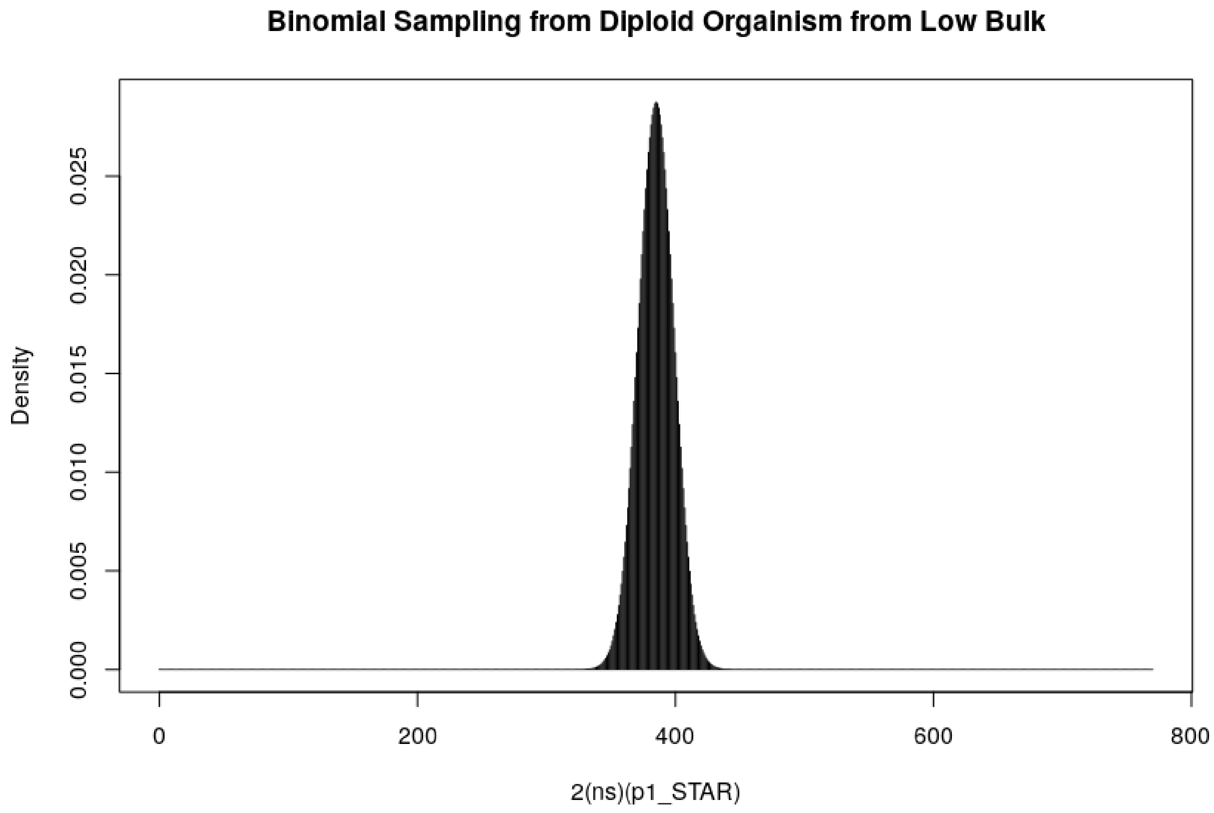
```
#Assuming average sequencing coverage (C) expected values for n1,n2,n3,n4
E(n1) = E(n2) = E(n3) = E(n4) = C/2 = 35

# Read in the csv file from High bulk tt
tt<-read.table(file = "ET-pool-385.csv",header = TRUE,sep = ",")
# Calculate average Coverage per SNP site
mean(tt$DP)
# Find REalized frequencies
p1_STAR <- sum(tt$AD_ALT.) / sum(tt$DP)

# Read in the csv file from Low Bulk TT
TT<-read.table(file ="ES-pool-430.csv",header = TRUE,sep=",")
# Calculate average Coverage per SNP sit
mean(TT$DP)
# Find Realized frequencies
p2_STAR <- sum(TT$AD_ALT.) / sum(TT$DP)
# Take the average of the Averages
C <-(mean(tt$DP)+mean(TT$DP))/2
C<-round(C,0)
#Average Coverage
70
C/2 = 35

p2 >> p1 QTL is present
However, ns >> C >> 1 is TRUE
```

2.18 Chromosomal Sampling Theory and an Analytical Framework with respect to Bulk Segregant Analysis

Binomial Sampling

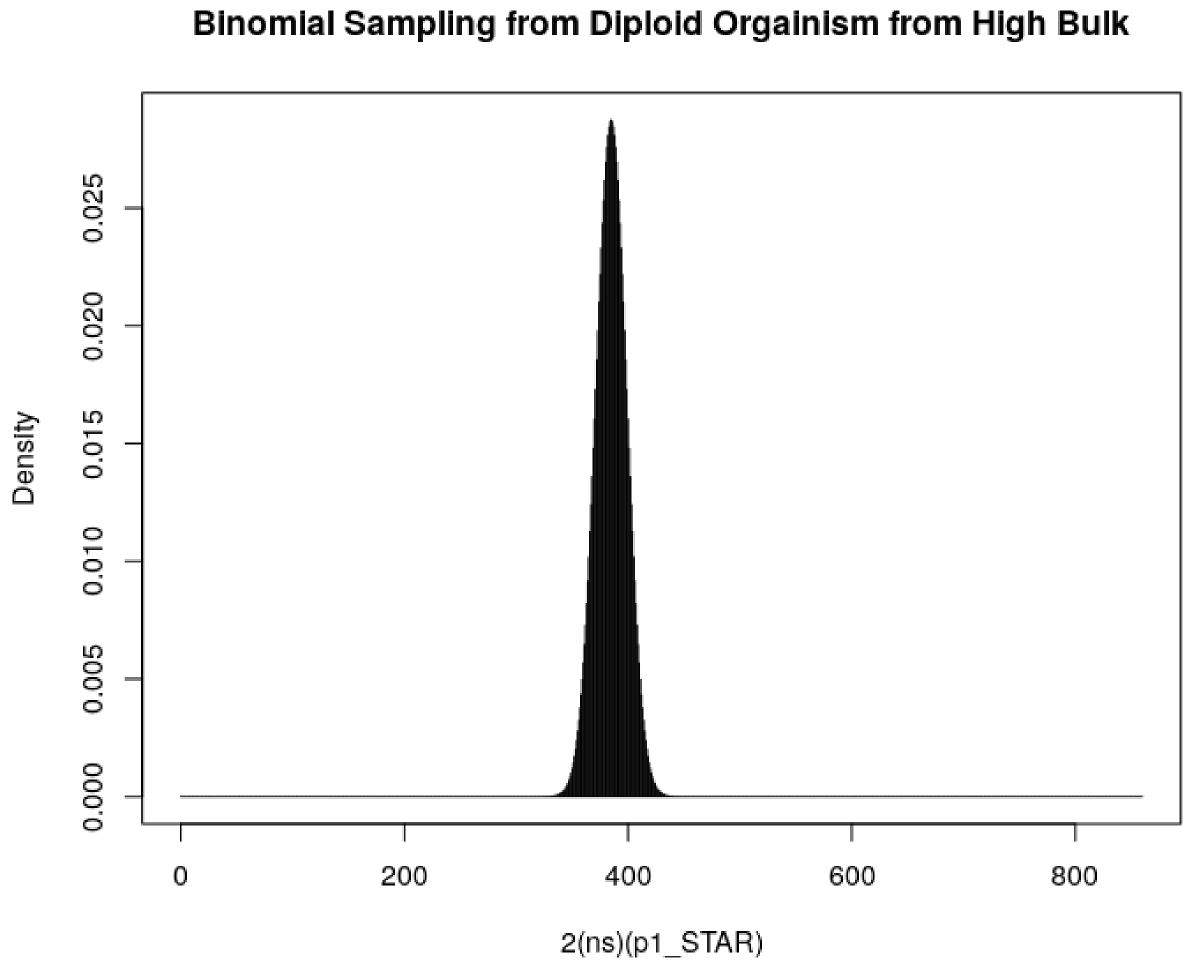
Low Bulk

```r
setwd("/home/michael/Desktop/QTLseqr/extdata")
# Theory and Analytical Framework of Sampling from BSA
par(mfrow=c(1,1))
# Define Ranges of Success
# Sample Size from High Bulk sn = 385
success <- 0:770
# The Difference between realized and Expected Frequencies
# ns : Sample Size taken from Low Bulk
# 2(ns)p1_star ~ Binomial(2(ns),p1)
# p1 Expected Frequencies
# Expected Frequencies:
# E(n1) = E(n2) = E(n3) = E(n4) = C/2 = 110
# We prefer for accuracy to have ns >> C >> 1
plot(success, dbinom(success, size = 770, prob = .50), type = "h",main="Binomial␣
↪Sampling from Diploid Orgainism from Low Bulk",xlab="2(ns)(    p1_STAR)",ylab="Density")
```



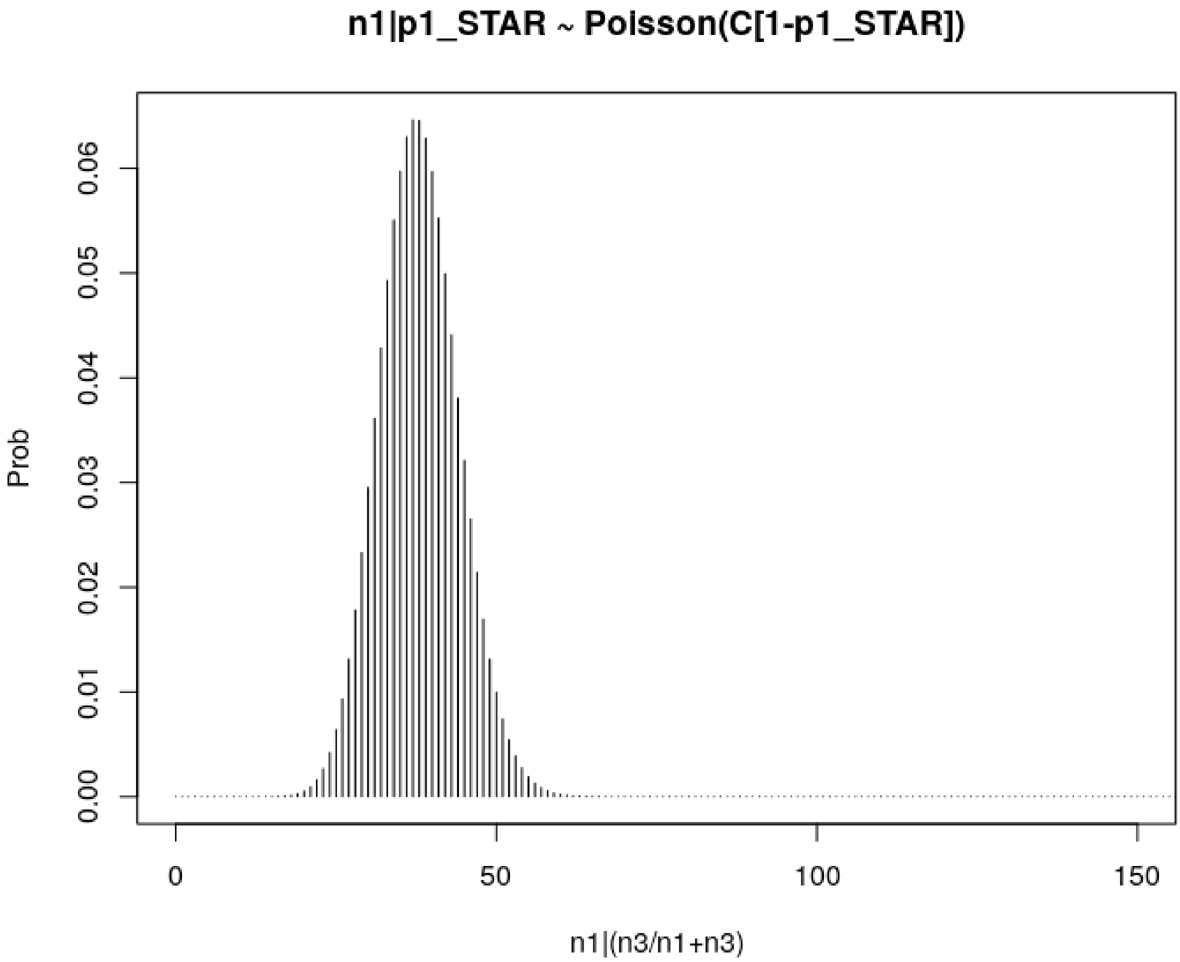
High Bulk

```
# ns : Sample Size from High Bulk
# 2(ns)p2_star ~ Binomial(2(ns),p2)
# p2 Expected Frequencies
success <- 0:860
plot(success, dbinom(success, size = 860, prob = 0.5), type = "h",main="Binomial␣
↪Sampling from Diploid Organism from High Bulk",xlab="2(n2)(p2_STAR)",ylab="Density")
```



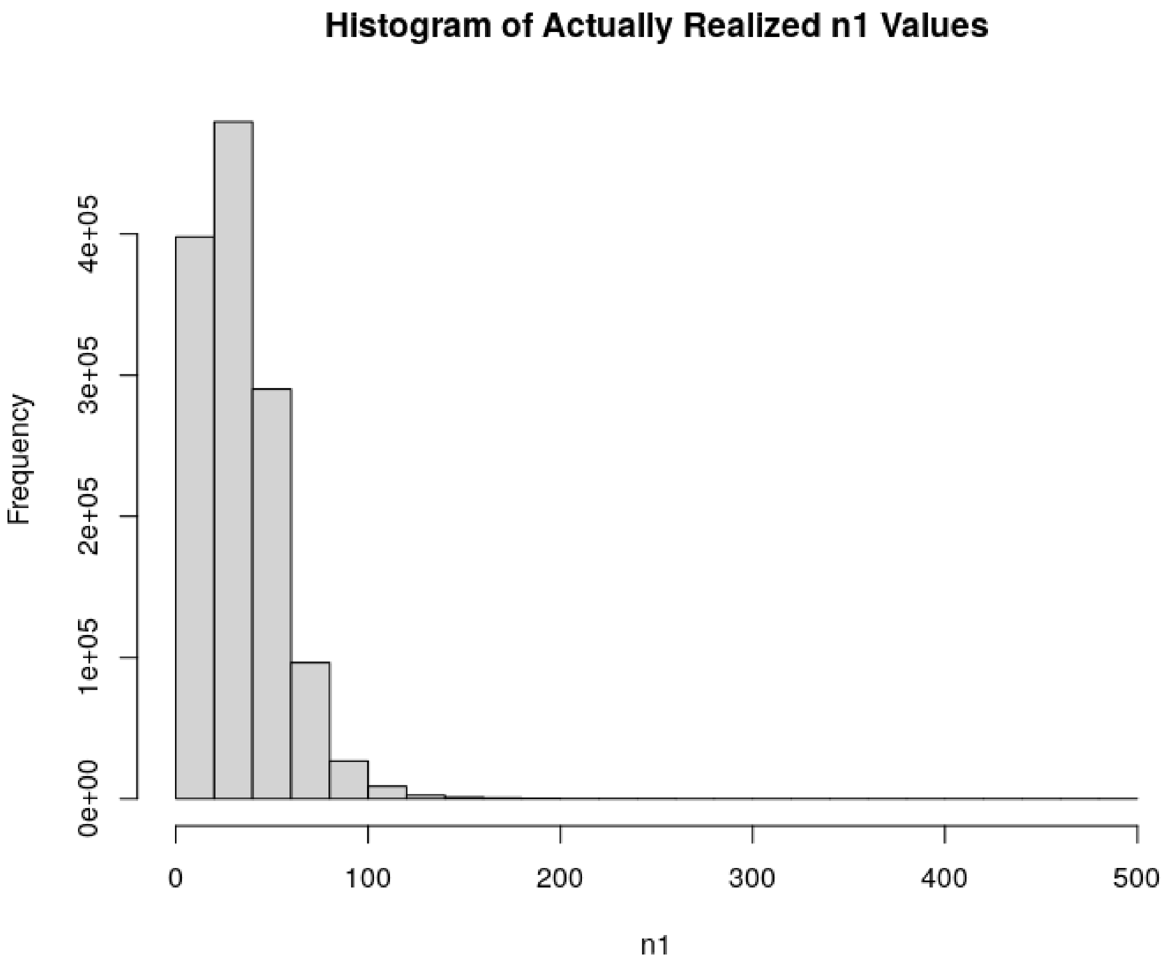
Conditional Distribution of n1 given realized average frequency

```
par(mfrow=c(1,1))
#Define Ranges of Success (Allele Frequencies High and Low)
success <- 0:100
#n1|p1_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*(1-p1_STAR)), type = 'h',main="n1|p1_STAR ~␣
↪Poisson(C[1-p1_STAR])",xlab="n1|(n3/n1+n3)",ylab="Prob")
```



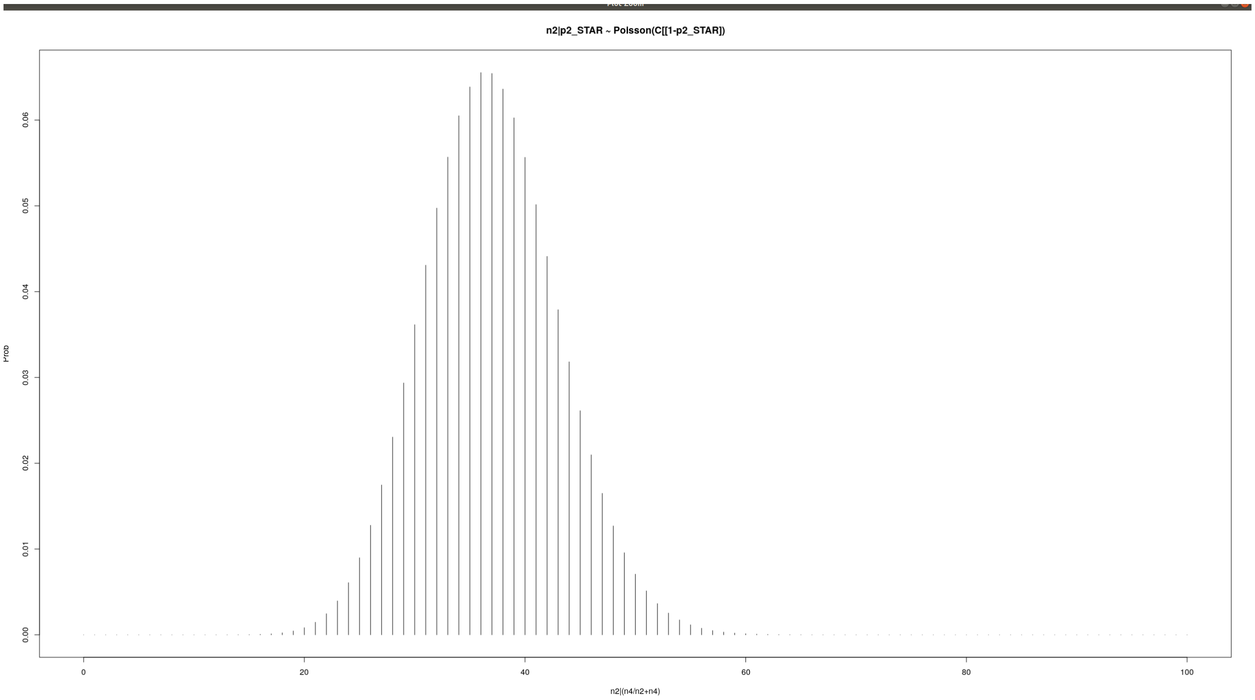
Observed n1

```
# Filter outliers
TT <- TT %>% filter(AD_REF. <= 500)

hist(TT$AD_REF., probability = FALSE,main="Histogram of Actually Realized n1 Values",
↪xlab="n1",breaks = "Sturges")
```



Conditional Distribution of n2 given realized average frequency

```
#n2|p2_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*(1-p2_STAR)), type='h', main="n2|p2_STAR ~␣
↪Poisson(C[[1-p2_STAR])",xlab="n2|(n4/n2+n4)",ylab="Prob")
```



Observed n2

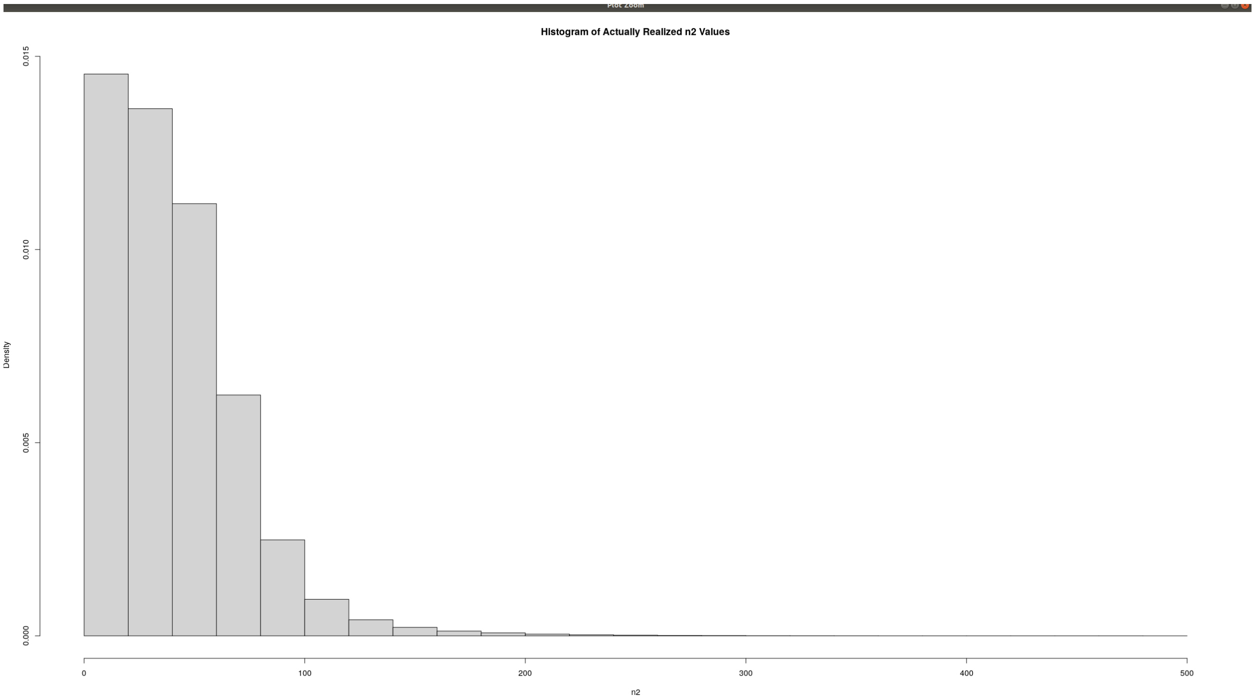
```
tt <- tt %>% filter(AD_REF. <= 500)
hist(tt$AD_REF., probability = TRUE, main = "Histogram of Actually Realized n2 Values",
↪xlab="n2")
```

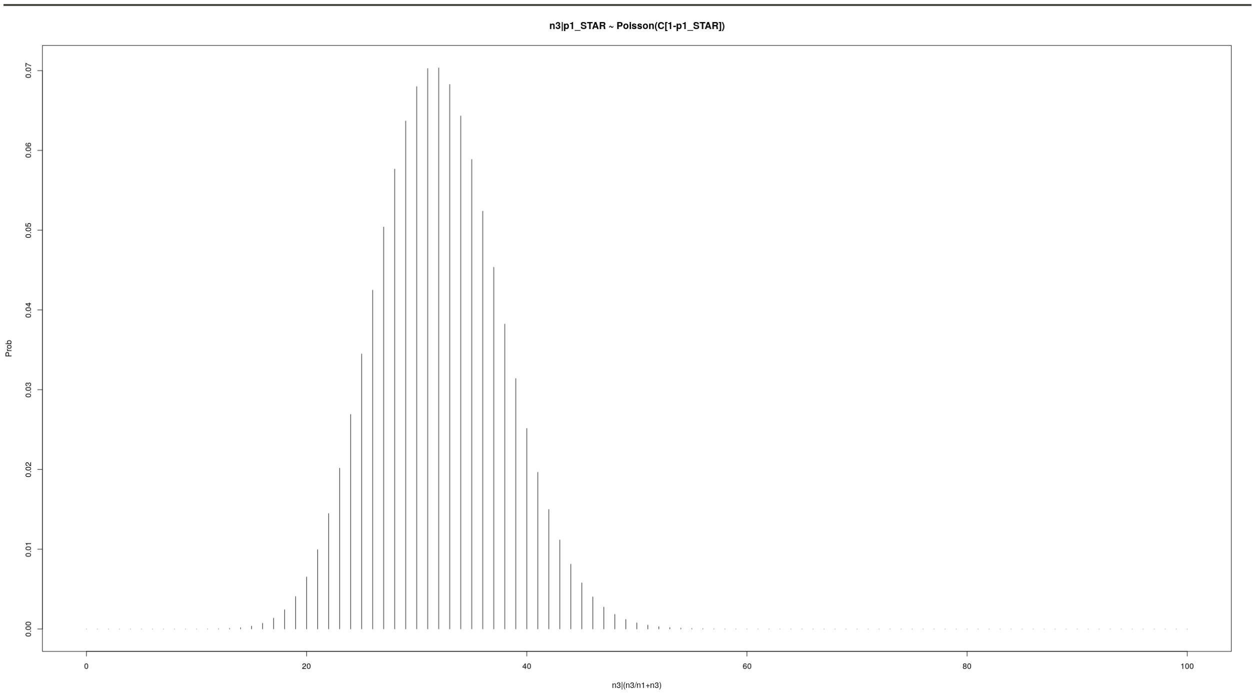
Conditional Distribution of n3 given realized average frequency

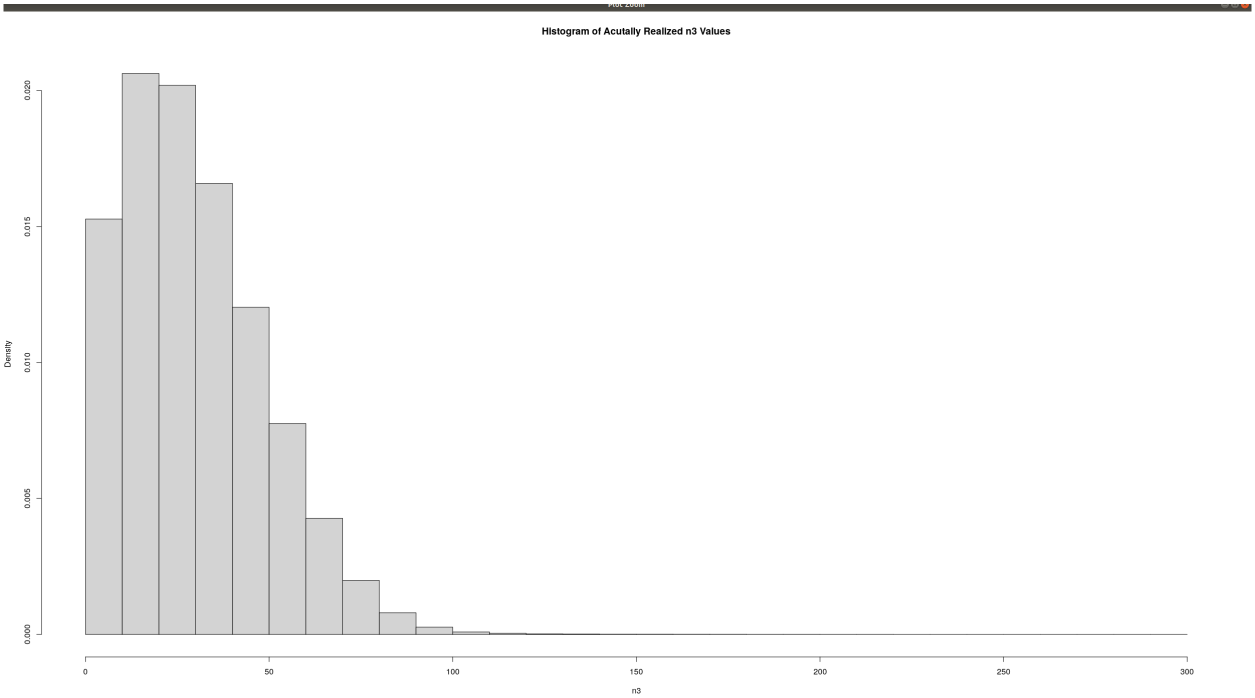
```
#n3|p1_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*p1_STAR),type='h',main="n3|p1_STAR ~ Poisson(C[1-
↪p1_STAR])",xlab="n3|(n3/n1+n3)",ylab="Prob")
```

Observed n3

```
TT <- TT %>% filter(AD_ALT. <= 300)
hist(TT$AD_ALT., probability = TRUE, main="Histogram of Acutally Realized n3 Values",
↪xlab="n3")
```


Plot Zoom
Histogram of Actually Realized n2 Values
Density
0.015
0.010
0.005
0.000
0
100
200
300
400
500
n2


n3|p1_STAR ~ Poisson(C[1-p1_STAR])
Prob
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
n3|(n3/n1+n3)



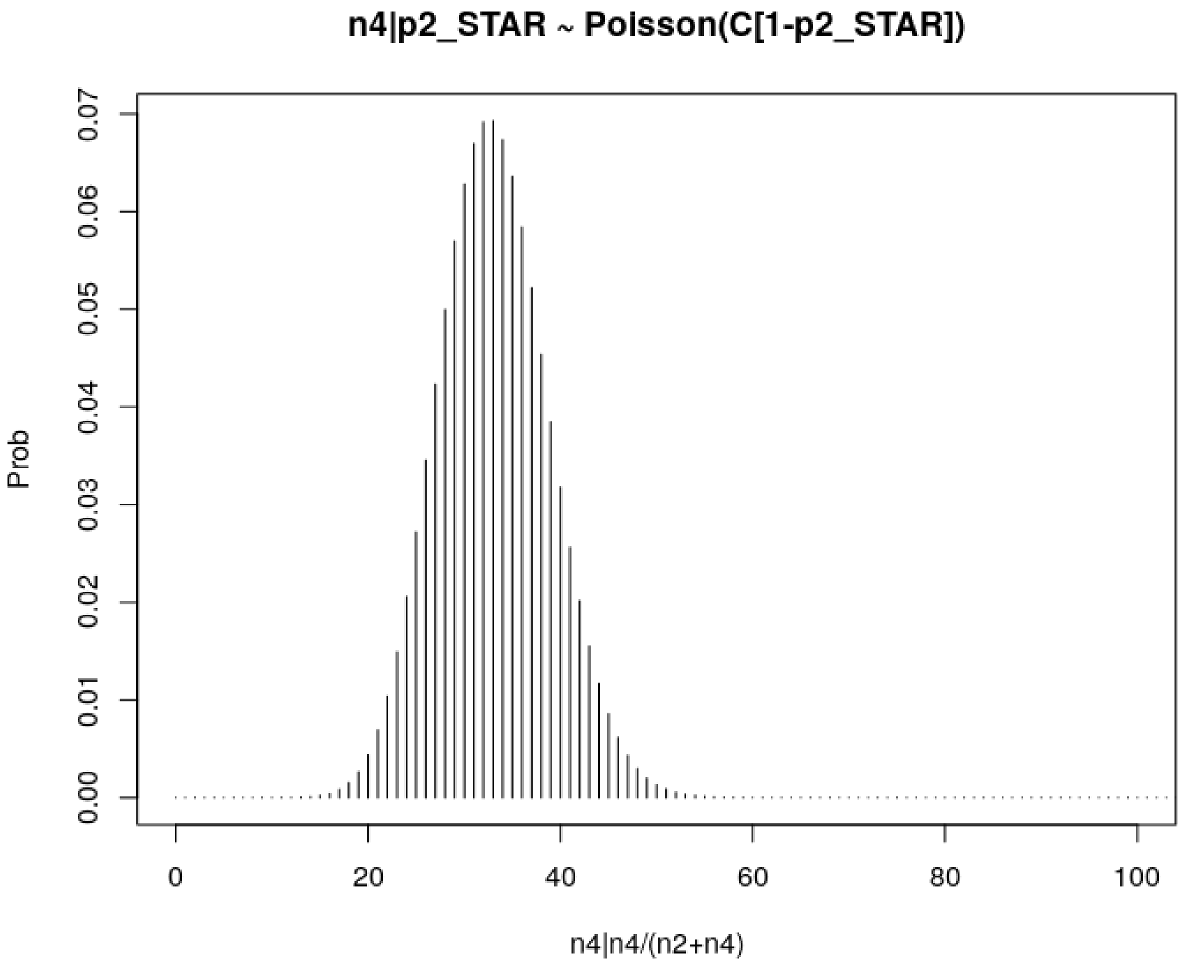
Conditional Distribution of n4 given realized average frequency

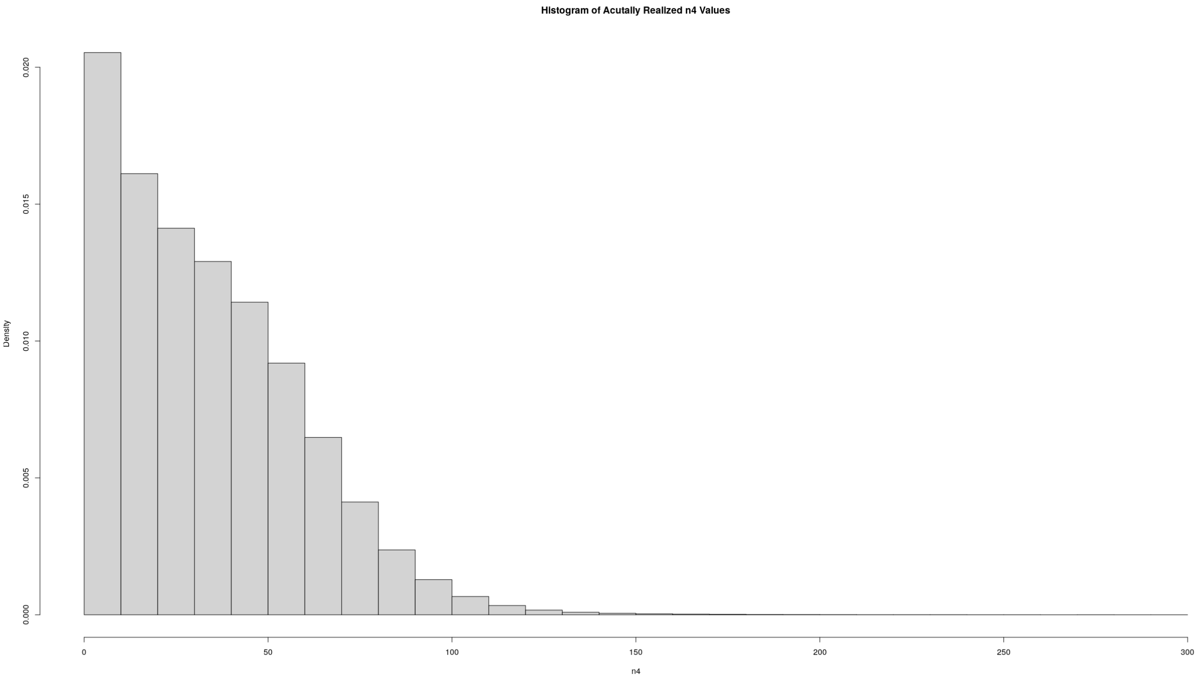
```
#n4|p2_star ~ Poisson(lambda)
plot(success, dpois(success, lambda = C*p2_STAR), type = 'h',main="n4|p2_STAR ~␣
↪Poisson(C[1-p2_STAR])",xlab="n4|n4/(n2+n4)",ylab="Prob")
```

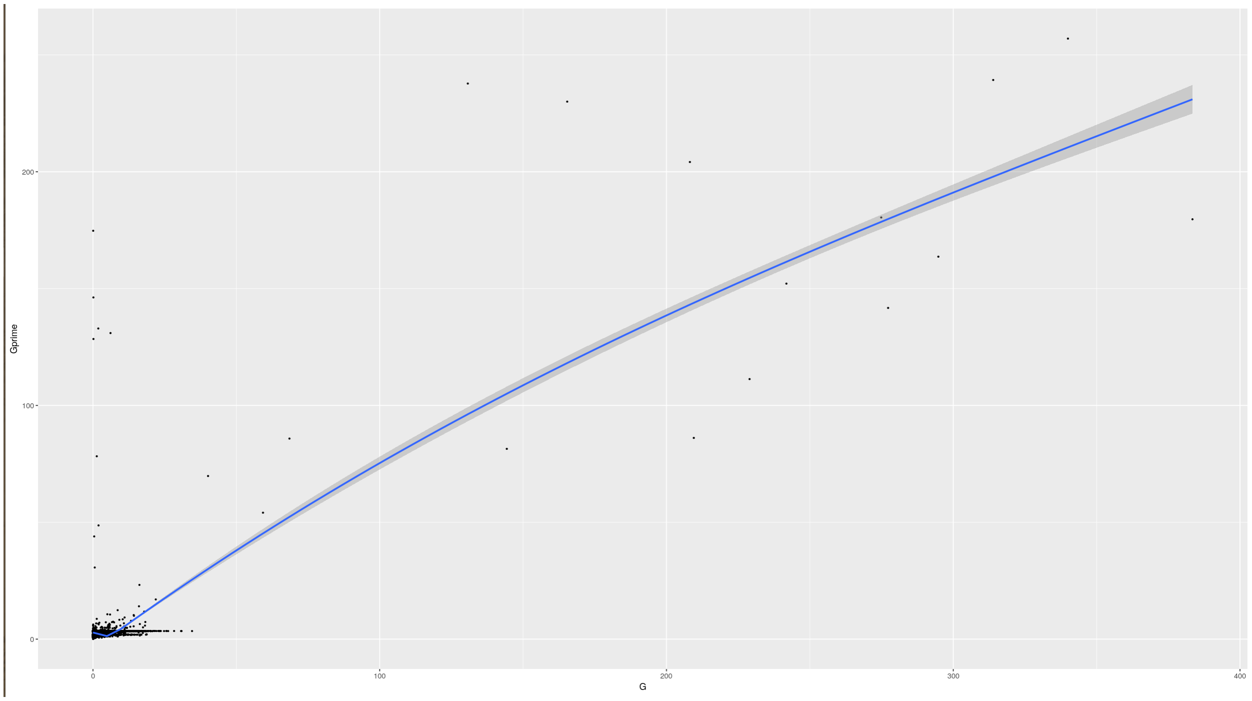
Observed n4

```
hist(tt$AD_ALT., probability = TRUE, main="Histogram of Acutally Realized n4 Values",
↪xlab="n4")
```

An interdependentaly observed relationship between G and Gprime


n4|p2_STAR ~ Poisson(C[1-p2_STAR])
Prob
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0
20
40
60
80
100
n4|n4/(n2+n4)


Histogram of Acutally Realized n4 Values
Density
0.000
0.005
0.010
0.015
0.020
0
50
100
150
200
250
300
n4


Gprime
0
100
200
0
100
200
300
400
G