Association Mapping: GWAS and Sequencing Data

Tiffany Timbers

August 20, 2015

Introduction to GWAS

A genome-wide association study (GWAS) is a method which tests whether genetic variants are associated with a trait (*e.g.,* phenotype or disease outcome). The method compares the DNA sequences of two groups: individuals with the phenotype or disease outcome (cases) and individuals without the phenotype or disease outcome (controls). Individuals DNA is read via SNP arrays or sequencing and if one variant appears significantly more frequently in the pool of individuals with the disease then the variant is said to be "associated" with the disease.

New considerations when performing GWAS on sequencing data

GWAS was developed for data from SNP microarrays and has begun to be subsequently adapted for next-generation sequencing data. Thus there are several considerations when doing GWAS with sequencing data, including: * how to deal with rare variants (these didn't exist in SNP array experiments)? * focus only on the exome? or look at whole-genome? * how to deal with synonomous mutations?

Advantages when performing GWAS on sequencing data

- potentially increased power due to rare variants potentially being more sever than common variants
- sequencing gives a complete picture of genetic variation (*e.g.,* SNPs, Copy Number Variants (CNVs) and insertions-deletions)
- can predict severity of variants from sequencing data and use this to weight analysis

Outline of project workflow for GWAS of sequencing data using the Sequence Kernel Association Test (SKAT)

1. Create a single `.vcf` file containing all the variants (as rows) and all the individuals (as columns)
2. Choose to perform association analysis via either gene sets or windows (*e.g.* SNP/variant set is defined to be those variants which fall within a chosen region size).
3. Create a SNP set ID (`.SSID`) file which indicates which variants belong to which set (*e.g.,* gene)
4. (optional) Create a custom weights file to assign weights to each variant

5. Create binary `plink` files from the .vcf file
6. Append the phenotype/disease outcome to the `plink .fam` file
7. (optional) Create co-variate file
8. Apply association test (*e.g.,* `SKAT`)
9. Apply mutliple testing correction to determine which genes/regions are significantly associated with the phenotype/disease outcome

Let's do it!

Dependencies

Our program dependencies are: the `Bash Shell`, `Perl`, `bgzip`, `tabix`, `R` and `R` packages `SKAT`, `dplyr`, `stringr` and `fdrtool`, as well as the program `plink`. Our data dependencies are: one `.vcf` file for each individual and a single phenotype/disease outcome file with 2 columns (the first being the individual ID and the second being the phenotype/disease outcome).

Create a merged `.vcf` file containing all the variants and all the individuals

In the `data` directory we have a collection of 480 `.vcf` files, each of which represent the whole-genome sequences of individual isogenic nematode, *Caenorhabditis elegans*, strains from the Million Mutation Project (http://genome.sfu.ca/mmp/about.html) (Thompson *et. al., Genome Research*, 2013) called against the VC2010 reference strain (a derivative of N2). Given that these are isogenic strains, we will treat each strain as an individual for this analysis.

```
# Bash Shell
ls data/*.vcf | head -5
```

```
## data/VC10118.vcf
## data/VC10129.vcf
## data/VC10130.vcf
## data/VC20007.vcf
## data/VC20009.vcf
```

Next, we use the VCFtools program `vcf-merge` to create a single merged `.vcf` file containing all the variants and all the individuals.

```
# Bash Shell
## CAUTION: LONG-RUN TIME PROCESS

# pre-process .vcf files so they are compatible with vcf-merge
for file in data/*.vcf; do bgzip -c $file > $file.gz; tabix -p vcf $file.gz; done
ls data/*.vcf.gz | head -5

# merge them together (note - if you run this twice you will get an error message
because it will try to merge MMP_vcf-merge.vcf.gz)
ulimit -n 8000 # this is because vcf-merge will open many files simultaneously
vcf-merge data/*.vcf.gz | bgzip -c > data/MMP_vcf-merge.vcf.gz
ls data/MMP_vcf-merge.vcf.gz
```

Then we uncompress `MMP.vcf.gz` so we can use it to create a snp set ID (`SSID`) file, as well as binary `plink` files.

```
# Bash Shell

gunzip data/MMP_vcf-merge.vcf.gz
grep -v -E '##' data/MMP_vcf-merge.vcf | cut -f 10-15 | head -5
```

```
## VC10118  VC10129 VC10130 VC20007 VC20009 VC20017
## .    .   .   .   .   .
## .    .   .   .   .   .
## .    .   .   .   .   .
## .    .   .   .   .   .
```

When we merged the files, we see that for each individual we either have “1/1” or “.” for a genotype. vcf-merge assumed that if there was no variant info for an individual when it merged the file that the data was missing. In the case of this data this is not true. For this dataset, each individual .vcf files simply did not contain all variants for this whole dataset because each file would be ~ 860,000 lines long and filled with mostly “0/0”. But we need those “0/0” for our analysis and so we need to replace the “.” with “0/0”. We can use the gsub function in awk to do this.

```
# Bash Shell

# replace  "." with "0/0""
awk '{gsub("\t[.]","\t0/0",$0); print;}' data/MMP_vcf-merge.vcf > data/MMP.vcf
grep -v -E '##' data/MMP.vcf | cut -f 10-15 | head -5
```

```
## VC10118  VC10129 VC10130 VC20007 VC20009 VC20017
## 0/0  0/0 0/0 0/0 0/0 0/0
## 0/0  0/0 0/0 0/0 0/0 0/0
## 0/0  0/0 0/0 0/0 0/0 0/0
## 0/0  0/0 0/0 0/0 0/0 0/0
```

Create a SNP set ID (`.SSID`) file which indicates which variants belong to which set (*e.g.,* gene)

For this analysis, we will define our SNP/variant sets by genes, thus we will omit all other DNA outside of exons and introns from our analysis. We also want to omit silent mutations in coding regions To do this we need to create a snp set ID (`.SSID`) file which indicates which variants belong to which set (*e.g.,* gene). `SSID` files contain 2 columns, the first being the gene name and the second being the variant name. Note - you can define these SNP sets however you want for your own analysis. For example, you could define them by biochemical pathways as opposed to genes to potentially increase power.

First we extract all the lines from the `.vcf` file with gene names using a series of `grep` commands. Note - these commands depend on how you defined your gene/sequence names in the `INFO` column. In the case of the `.vcf` files we are using here, gene/sequence names are found either after `SN=` or `CODING=` .

```
# Bash Shell

grep -E -v '#' data/MMP.vcf | grep -E 'SN=[A-Z0-9.]{4,}|CODING=' | grep -E -v 'S
N=I;' | grep -E -v 'GRANT=0|GRANT=NA' | cut -f 1-8 > data/gene_variants.txt
head -5 data/gene_variants.txt
```

```
## I    9752    gk362559    C    T    173.00  PASS    AAC=R->Q;AC=2;ACGT
N=0,0,0,7,0;AN=2;CMQ=60;DP20=7;GERP=3.35;GF=coding_exon;GRANT=43;PHASTCONS=0.998;P
HYLOP=2.060;PN=Y74C9A.3;SF=258;SN=Y74C9A.3
## I    9785    gk100026    G    A    71.50   PASS    AAC=A->V;AC=2;ACGT
N=3,0,0,0,0;AN=2;CMQ=60;DP20=3;GERP=-6.24;GF=coding_exon;GRANT=64;PHASTCONS=0.62
2;PHYLOP=0.101;PN=Y74C9A.3;SF=151;SN=Y74C9A.3
## I    10102   gk343340    T    C    222.00  PASS    AAC=Y->C;AC=2;ACGTN=0,1
0,0,0,0;AN=2;CMQ=60;DP20=10;GERP=NA;GF=coding_exon;GRANT=194;PHASTCONS=0.957;PHYLO
P=0.734;PN=Y74C9A.3;SF=212;SN=Y74C9A.3
## I    13569   gk960507    T    <DEL>   0/0 PASS    AC=2;AN=2;CODING=NA;END=1381
1;NCOV=0.00;NCRNA=NA;NQDEV=-2.47;SF=320;SPAN=7;SVLEN=-242;SVTYPE=DEL
## I    16500   gk913332    C    T    170.00  PASS    AAC=A->V;AC=2;ACGTN=0,0,0,1
1,0;AN=2;CMQ=60;DP20=11;GERP=3.35;GF=coding_exon;GRANT=64;PHASTCONS=0.996;PHYLO
P=2.060;PN=nlp-40;SF=467;SN=Y74C9A.2
```

Next we need to extract the gene and the sequence names from `data/gene_variants.txt` and save them in a file called `MMP.SSID` . We will do this in R for simplicity, but it could be done faster with another language (*e.g.,* Perl)

```
# R

## load data/gene_variants.txt into R
gene_variants <- read.table(file = "data/gene_variants.txt", stringsAsFactors = FA
LSE)
head(gene_variants)
```

```
##   V1    V2       V3 V4    V5     V6   V7
## 1  I  9752 gk362559  C     T 173.00 PASS
## 2  I  9785 gk100026  G     A  71.50 PASS
## 3  I 10102 gk343340  T     C 222.00 PASS
## 4  I 13569 gk960507  T <DEL>    0/0 PASS
## 5  I 16500 gk913332  C     T 170.00 PASS
## 6  I 19071 gk541423  C     T 222.00 PASS
##
V8
## 1  AAC=R->Q;AC=2;ACGTN=0,0,0,7,0;AN=2;CMQ=60;DP20=7;GERP=3.35;GF=coding_exon;GR
ANT=43;PHASTCONS=0.998;PHYLOP=2.060;PN=Y74C9A.3;SF=258;SN=Y74C9A.3
## 2 AAC=A->V;AC=2;ACGTN=3,0,0,0,0;AN=2;CMQ=60;DP20=3;GERP=-6.24;GF=coding_exon;GR
ANT=64;PHASTCONS=0.622;PHYLOP=0.101;PN=Y74C9A.3;SF=151;SN=Y74C9A.3
## 3 AAC=Y->C;AC=2;ACGTN=0,10,0,0,0;AN=2;CMQ=60;DP20=10;GERP=NA;GF=coding_exon;GRA
NT=194;PHASTCONS=0.957;PHYLOP=0.734;PN=Y74C9A.3;SF=212;SN=Y74C9A.3
## 4                                                AC=2;AN=2;CODING=NA;END=1381
1;NCOV=0.00;NCRNA=NA;NQDEV=-2.47;SF=320;SPAN=7;SVLEN=-242;SVTYPE=DEL
## 5  AAC=A->V;AC=2;ACGTN=0,0,0,11,0;AN=2;CMQ=60;DP20=11;GERP=3.35;GF=coding_exo
n;GRANT=64;PHASTCONS=0.996;PHYLOP=2.060;PN=nlp-40;SF=467;SN=Y74C9A.2
## 6        AAC=G->E;AC=2;ACGTN=0,0,0,15,0;AN=2;CMQ=60;DP20=15;GERP=NA;GF=coding_e
xon;GRANT=98;PHASTCONS=NA;PHYLOP=NA;PN=Y74C9A.4;SF=376;SN=Y74C9A.4
```

```
# load stringr library to perform regular expressions
library(stringr)

# use str_extract() to capture gene/sequence name from INFO column (V8 in the gen
e_variants data frame)
pattern  <- "SN=[a-zA-Z0-9.]{1,}|CODING=[a-zA-Z0-9.]{1,}"
gene  <- str_extract(gene_variants$V8, pattern)
gene  <- sub("SN=|CODING=", "", gene)
head(gene)
```

```
## [1] "Y74C9A.3" "Y74C9A.3" "Y74C9A.3" "NA"       "Y74C9A.2" "Y74C9A.4"
```

```
# make a new data frame containing 2 columns, gene and variants (gene_variants$V3)
SSID <- data.frame(gene, gene_variants$V3)
head(SSID)
```

```
##       gene gene_variants.V3
## 1 Y74C9A.3         gk362559
## 2 Y74C9A.3         gk100026
## 3 Y74C9A.3         gk343340
## 4       NA         gk960507
## 5 Y74C9A.2         gk913332
## 6 Y74C9A.4         gk541423
```

```
# remove NA's (sometimes CODING=NA and we want to remove those)
SSID <- subset(SSID, gene!="NA")

# save the SSID file
write.table(x = SSID, file = "data/MMP.SSID", row.names = FALSE, col.names = FALS
E, append = FALSE, quote = FALSE)
```

Create binary `plink` files from the .vcf file

`plink` (Purcell *et al., American Journal of Human Genetics*, 2007) is one of the most widely used programs to perform traditional GWAS (*e.g.,* using SNP array data) and thus most newly developed tools accept `plink` files as input. In addition to association analysis tools, `plink` now also contains tools to convert `.vcf` files to `plink` files.

```
# Bash Shell

plink --vcf data/MMP.vcf --allow-extra-chr --no-fid --no-parents --no-sex --no-phe
no --out data/MMP
ls data/MMP*
```

Append the phenotype/disease outcome to the `plink .fam` file

The `.fam` file does not yet contain the phenotype/disease outcome data. We need to manually add this to the `.fam` by removing column 6 from the `.fam` file and joining it by strain/individual with the file containing our phenotype/disease outcome data. Because these files are smaller (2-6 column and hundreds of to tens of thousands of rows) we can load these files into memory and will take advantage of the powerful `dplyr` package in `R` to do this.

```
# R

# load dplyr library (we will use this to join the files)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# read in phenotypes file
pheno_file <- tbl_df(read.table(file = "data/phenotype_dichotomous.csv", header =
TRUE, stringsAsFactors = FALSE))
head(pheno_file)
```

```
## Source: local data frame [6 x 2]
##
##    strain phenotype
## 1 VC20138         1
## 2 VC20214         1
## 3 VC20267         1
## 4 VC20337         1
## 5 VC20448         1
## 6 VC20520         1
```

```
# read in fam file
fam_file <- tbl_df(read.table(file = "data/MMP.fam", stringsAsFactors = FALSE))
head(fam_file)
```

```
## Source: local data frame [6 x 6]
##
##        V1      V2 V3 V4 V5 V6
## 1 VC10118 VC10118  0  0  0 -9
## 2 VC10129 VC10129  0  0  0 -9
## 3 VC10130 VC10130  0  0  0 -9
## 4 VC20007 VC20007  0  0  0 -9
## 5 VC20009 VC20009  0  0  0 -9
## 6 VC20017 VC20017  0  0  0 -9
```

```
# name columns and drop column V6 (we will replace this with the phenotype column)
colnames(fam_file)[1] <- "strain"
fam_file$V6 <- NULL
head(fam_file)
```

```
## Source: local data frame [6 x 5]
##
##    strain      V2 V3 V4 V5
## 1 VC10118 VC10118  0  0  0
## 2 VC10129 VC10129  0  0  0
## 3 VC10130 VC10130  0  0  0
## 4 VC20007 VC20007  0  0  0
## 5 VC20009 VC20009  0  0  0
## 6 VC20017 VC20017  0  0  0
```

```
#  use dplyr to join columns 1-5 of fam_file with columns 1-2 of pheno_file, join by column 1
fam_w_phenos <- left_join(x = fam_file, y = pheno_file)
```

```
## Joining by: "strain"
```

```
head(fam_w_phenos)
```

```
## Source: local data frame [6 x 6]
##
##    strain      V2 V3 V4 V5 phenotype
## 1 VC10118 VC10118  0  0  0         0
## 2 VC10129 VC10129  0  0  0         0
## 3 VC10130 VC10130  0  0  0         0
## 4 VC20007 VC20007  0  0  0         0
## 5 VC20009 VC20009  0  0  0         1
## 6 VC20017 VC20017  0  0  0         0
```

```
write.table(x = fam_w_phenos, file = "data/MMP.fam", row.names = FALSE, col.names = FALSE, quote = FALSE, append = FALSE)
```

```
# Bash Shell

head -5 data/MMP.fam
```

```
## VC10118 VC10118 0 0 0 0
## VC10129 VC10129 0 0 0 0
## VC10130 VC10130 0 0 0 0
## VC20007 VC20007 0 0 0 0
## VC20009 VC20009 0 0 0 1
```

Apply association test (*e.g.,* `SKAT`)

Now that we have our `SSID` file and our binary `plink` files we can do the association analysis. We will work in R to do this and use the SKAT package (Wu *et al., American Journal of Human Genetics*, 2011).

```
# R
## CAUTION: LONG-RUN TIME PROCESS

# load SKAT library (to do association analysis)
library(SKAT)

# Generate a SNP set data file (`.SSD`) and an `.info` file from binary plink form
ated data files using user specified SNP sets.
Generate_SSD_SetID('data/MMP.bed', 'data/MMP.bim', 'data/MMP.fam', 'data/MMP.SSI
D', 'data/MMP.SSD', 'MMP.info')
```

```
##
## Check duplicated SNPs in each SNP set
## No duplicate
## Warning: SSD file has more SNP sets then SetID file.It happens when SNPs in set
s are not contiguous!
## SKAT generates a temporary SetID file with contiguous SNP sets. However the ord
er of SNP sets will be different from the order of SNP sets in the original SetID
file!
##
##
## Check duplicated SNPs in each SNP set
## No duplicate
## 476 Samples, 15652 Sets, 46563 Total SNPs
## [1] "SSD and Info files are created!"
```

```
# read in fam file
fam_file <- read.table(file = "data/MMP.fam")

# open .SSD and .info files we just created
SSD.info <- Open_SSD('data/MMP.SSD', 'MMP.info')
```

```
## 476 Samples, 15652 Sets, 46563 Total SNPs
## Open the SSD file
```

```
# create null model based on phenotypes (and co-variates if you have any)
set.seed(100)
Null_Model <- SKAT_Null_Model(fam_file$V6 ~ 1, out_type="D")
```

```
## Sample size (non-missing y and X) = 476, which is < 2000. The small sample adju
stment is applied!
```

```
# perform SKAT on all sets of variants
#All_SKAT_Data  <- SKAT.SSD.All(SSD.info, Null_Model)
All_SKAT_Data  <- SKAT.SSD.All(SSD.INFO = SSD.info, obj = Null_Model)
```

```
## Warning: The missing genotype rate is 0.001050. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.002101. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.001050. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.000525. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.000525. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.000420. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.001050. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.002101. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.002101. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.000420. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.000420. Imputation is applied.
```

SKAT.SSD.All returns a dataframe that contains `SetID`, p-values (`P.value`), the number of markers in the SNP sets (`N.Marker.All`), and the number of markers to test for an association after excluding non-polymorphic or high missing rates markers (`N.Marker.Test`).

```
# R

# Checkout SKAT results
head(All_SKAT_Data$results)
```

```
##    SetID   P.value N.Marker.All N.Marker.Test
## 1 2L52.1 0.4864633            1             1
## 2 4R79.1 0.4933038            1             1
## 3 4R79.2 0.6488933            3             3
## 4 6R55.2 0.5947937            2             2
## 5 AC3.10 0.4880504            1             1
## 6  AC3.2 0.4889657            1             1
```

The results are sorted via SetID, and not p-value. Thus, to find which genes are most highly associated with our phenotype/disease outcome we must sort the results by p-value.

```
# R

# sort SKAT results by p-value
head(All_SKAT_Data$results[order(All_SKAT_Data$results$P.value),])
```

```
##            SetID      P.value N.Marker.All N.Marker.Test
## 3147    C50F4.11 0.0002159104            6             6
## 14737 Y97E10AR.5 0.0006961214            4             4
## 4686     F18E2.5 0.0007662204            4             4
## 12434  Y17G7B.15 0.0008058830            8             8
## 9824       R74.1 0.0008609688            8             8
## 3781     F01D4.9 0.0011937994            7             7
```

```
#save SKAT results
write.table(x = All_SKAT_Data$results, file = "data/SKAT_all-pvals.results", row.n
ames = FALSE, col.names = TRUE, quote = FALSE, append = FALSE)
```

Apply mutliple testing correction

We have just performed association tests on > 1000 genes… We need to correct for multiple testing. There are many ways to do this, including Bonferroni correction, False discovery rate and resampling. We will choose a “medium” stringency multiple correction adjustment, the false discovery rate.

```
# R

# load fdrtool library
library(fdrtool)

# calculate q-values
qvals <- fdrtool(All_SKAT_Data$results$P.value, statistic = "pvalue", cutoff.metho
d="fndr", plot = FALSE)
```

```
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
```

```
All_SKAT_Data$results$Q.value <- qvals$qval

# sort SKAT results by q-value
All_SKAT_Data$results <-All_SKAT_Data$results[order(All_SKAT_Data$results$P.valu
e),]
All_SKAT_Data$results[1:20,]
```

```
##            SetID      P.value N.Marker.All N.Marker.Test   Q.value
## 3147    C50F4.11 0.0002159104            6             6 0.0257245
## 14737 Y97E10AR.5 0.0006961214            4             4 0.0257245
## 4686     F18E2.5 0.0007662204            4             4 0.0257245
## 12434  Y17G7B.15 0.0008058830            8             8 0.0257245
## 9824       R74.1 0.0008609688            8             8 0.0257245
## 3781     F01D4.9 0.0011937994            7             7 0.0257245
## 12152 Y105C5A.15 0.0014396498            5             5 0.0257245
## 7741     F59G1.2 0.0015636097            5             5 0.0257245
## 6297     F43D9.2 0.0016161015            5             5 0.0257245
## 7340    F56D12.4 0.0016746634           10            10 0.0257245
## 4724     F19B2.8 0.0016795828            2             2 0.0257245
## 7479     F57F5.2 0.0017129592            2             2 0.0257245
## 11854    W04D2.1 0.0017620209            5             5 0.0257245
## 505      C01G5.5 0.0017661726            2             2 0.0257245
## 12124  Y102A5C.1 0.0017741545            2             2 0.0257245
## 325      B0432.5 0.0017900697            2             2 0.0257245
## 5274    F28A12.2 0.0017985007            2             2 0.0257245
## 10988    T20B3.4 0.0018419621            2             2 0.0257245
## 797     C05D11.5 0.0018578262            2             2 0.0257245
## 720      C04F5.1 0.0018597892            2             2 0.0257245
```

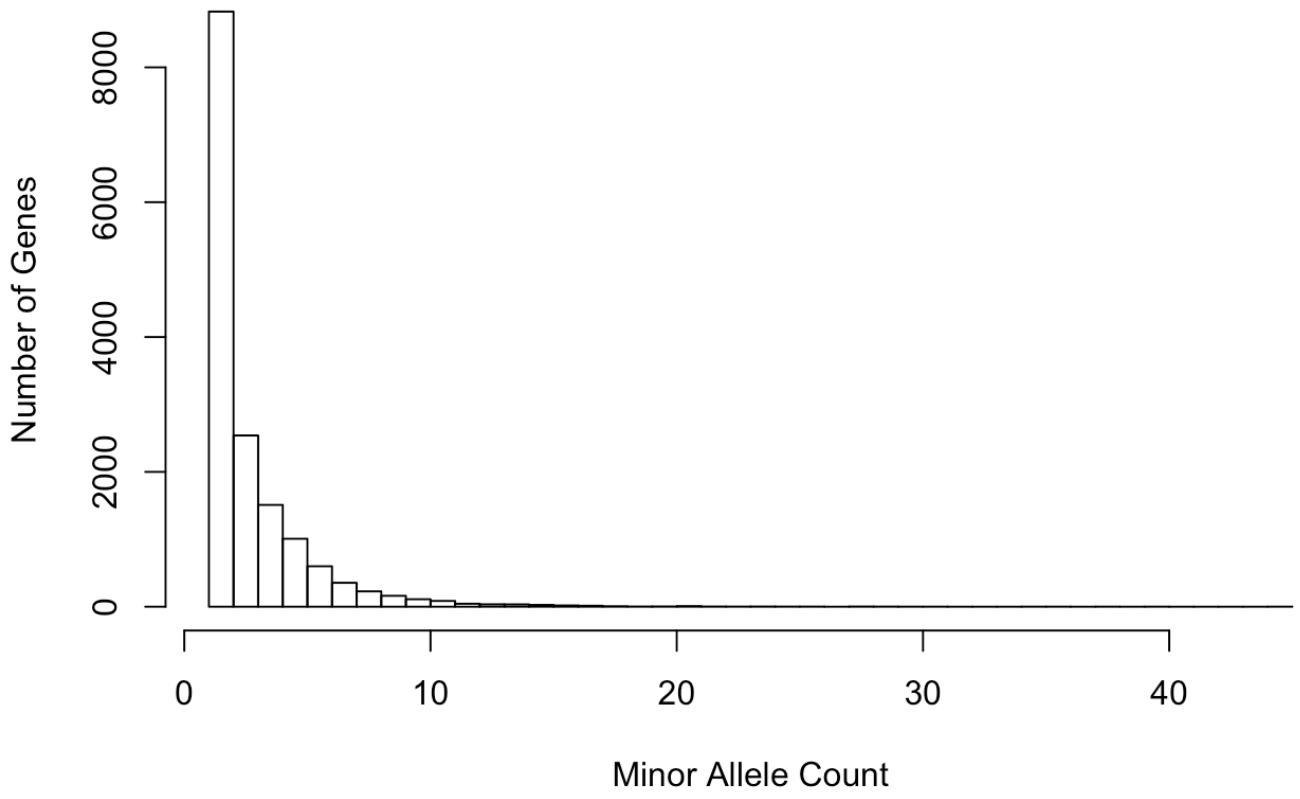
```
#save SKAT results
write.table(x = All_SKAT_Data$results, file = "data/SKAT_all-qvals.results", row.n
ames = FALSE, col.names = TRUE, quote = FALSE, append = FALSE)
```

We find that if we choose a 30% false-discovery rate cut-off that we find that a lot of genes are significantly associated with the phenotype. This doesn't make much sense? This is due to a skewed p-value distrubution from these samples which arises from the distribution of minor allele counts in the Million Mutation project strains which exponentially decreases from 1 to N.

```
# R

# load plyr library so we can count how many variants are in each snp set
library(plyr)
```

```
## ------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dp
lyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------
##
## Attaching package: 'plyr'
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
# load snp set ID (.SSID) file
SSID <- read.table(file = "data/MMP.SSID", stringsAsFactors = FALSE)

# count how many variants are in each snp set
MACs <- count(SSID[,1])

# plot this as a histogram
MAC_hist <- hist(x <- (MACs[MACs$freq < 50,2]), breaks = 50, xlab = "Minor Allele
Count", ylab = "Number of Genes", main = "Allele distribution")
```



Thus, to use false-discovery rate we need to decide on a minimum minor allele count for SNP sets we want to include our analysis. This is important beyond correctiong for multiple comparisons, for example, how meaningful is a gene that only has a single variant in the population being looked at? How much confidence do we have that it would be associated (or not associated) with the individuals phenotype/disease outcome?
What is the minimum number of variants in SNP set to include it in the analysis? Unfortuneatly, this is a question that as of yet has no solid answer and researchers must make their own decision on the minimum minor allele count for a SNP set must be.

From the histogram of the minor allele count per gene for this dataset, a minimum minor allele count of 7 seems to be reasonable. Thus, we will reduce our SNP set ID (`.SSID`) file to only those genes which meet that criteria.

```
# R

# load .SSID file and give meaningful column names
SSID <- read.table(file = "data/MMP.SSID", header=FALSE)
colnames(SSID) <- c("gene", "variant")
head(SSID)
```

```
##       gene  variant
## 1 Y74C9A.3 gk362559
## 2 Y74C9A.3 gk100026
## 3 Y74C9A.3 gk343340
## 4 Y74C9A.2 gk913332
## 5 Y74C9A.4 gk541423
## 6 Y74C9A.4 gk561603
```

```
# make a list of the variants to keep (i.e. plyr::count the number of occurrences of each gene)
SSID_count <- count(SSID[,1])
head(SSID_count)
```

```
##        x freq
## 1 2L52.1    1
## 2 4R79.1    1
## 3 4R79.2    3
## 4 6R55.2    2
## 5 AC3.10    1
## 6  AC3.2    1
```

```
# make SSID file with only those variants (i.e. reduce the SSID list to only those with >= 6 variants)
genes_to_include <- data.frame(SSID_count[SSID_count$freq >= 5, 1])

# give the data frame meaningful column names
colnames(genes_to_include) <- ("gene")
head(genes_to_include)
```

```
##       gene
## 1   AH10.1
## 2    AH6.1
## 3  B0019.1
## 4  B0019.2
## 5 B0024.14
## 6  B0024.6
```

```
# reduce SSID to contain only those genes from the list genes_to_include via dplyr::semi_join
reduced_SSID <- semi_join(SSID, genes_to_include)
```

```
## Joining by: "gene"
```

```
head(reduced_SSID)
```

```
##     gene  variant
## 1 AH10.1 gk253070
## 2 AH10.1 gk518421
## 3 AH10.1 gk738126
## 4 AH10.1 gk348203
## 5 AH10.1 gk733854
## 6 AH10.1 gk253074
```

```
# save SSID file
write.table(x = reduced_SSID, file = "data/MMP-reduced.SSID", row.names = FALSE, c
ol.names = FALSE, quote = FALSE)
```

Now can run SKAT with this reduced SNP set ID (`.SSID`) file:

```
# R
## CAUTION: LONG-RUN TIME PROCESS

# load SKAT library (to do association analysis)
library(SKAT)

# Generate a SNP set data file (`.SSD`) and an `.info` file from binary plink form
ated data files using user specified SNP sets.
Generate_SSD_SetID('data/MMP.bed', 'data/MMP.bim', 'data/MMP.fam', 'data/MMP-reduc
ed.SSID', 'data/MMP-reduced.SSD', 'MMP-reduced.info')
```

```
##
## Check duplicated SNPs in each SNP set
## No duplicate
## 476 Samples, 2775 Sets, 20266 Total SNPs
## [1] "SSD and Info files are created!"
```

```
# open .SSD and .info files we just created
SSD.info_reduced <- Open_SSD('data/MMP-reduced.SSD', 'MMP-reduced.info')
```

```
## Close the opened SSD file: /Users/tiffanytimbers/Documents/SKAT_NGS-2015/data/M
MP.SSD
## 476 Samples, 2775 Sets, 20266 Total SNPs
## Open the SSD file
```

```
# null model doesn't take in info from .SSID, so we don't need to run that again
# perform SKAT on all sets of variants
All_SKAT_Data_reduced  <- SKAT.SSD.All(SSD.INFO = SSD.info_reduced, obj = Null_Mod
el)
```

```
## Warning: The missing genotype rate is 0.000420. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.000420. Imputation is applied.
```

```
## Warning: The missing genotype rate is 0.000420. Imputation is applied.
```

```
# sort SKAT results by p-value
head(All_SKAT_Data_reduced$results[order(All_SKAT_Data_reduced$results$P.value),])
```

```
##           SetID      P.value N.Marker.All N.Marker.Test
## 518    C50F4.11 0.0002159104            6             6
## 2175  Y17G7B.15 0.0008058830            8             8
## 1717      R74.1 0.0008609688            8             8
## 627     F01D4.9 0.0011937994            7             7
## 2121 Y105C5A.15 0.0014396498            5             5
## 1336    F59G1.2 0.0015636097            5             5
```

```
# calculate q-values
qvals <- fdrtool(All_SKAT_Data_reduced$results$P.value, statistic = "pvalue", cuto
ff.method="fndr", plot = FALSE)
```

```
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
```

```
All_SKAT_Data_reduced$results$Q.value <- qvals$qval

# sort SKAT results by q-value
All_SKAT_Data_reduced$results <- All_SKAT_Data_reduced$results[order(All_SKAT_Dat
a_reduced$results$P.value),]
All_SKAT_Data_reduced$results[1:20,]
```

```
##           SetID      P.value N.Marker.All N.Marker.Test   Q.value
## 518    C50F4.11 0.0002159104            6             6 0.1976838
## 2175  Y17G7B.15 0.0008058830            8             8 0.1976838
## 1717      R74.1 0.0008609688            8             8 0.1976838
## 627     F01D4.9 0.0011937994            7             7 0.1976838
## 2121 Y105C5A.15 0.0014396498            5             5 0.1976838
## 1336    F59G1.2 0.0015636097            5             5 0.1976838
## 1076    F43D9.2 0.0016161015            5             5 0.1976838
## 1273   F56D12.4 0.0016746634           10            10 0.1976838
## 2069    W04D2.1 0.0017620209            5             5 0.1976838
## 12      B0205.4 0.0025454732            6             6 0.2030985
## 1350   H06I04.5 0.0026216957            6             6 0.2034629
## 1171    F52D2.6 0.0027301847            6             6 0.2039485
## 1081    F44B9.6 0.0027864153            6             6 0.2041861
## 799     F19H8.1 0.0028842725            8             8 0.2045788
## 1529    K12H4.8 0.0033944150           11            11 0.2062765
## 1198    F53H1.1 0.0034791584            6             6 0.2065125
## 595     D2085.5 0.0049403727           10            10 0.2093502
## 2249  Y39B6A.20 0.0051514868            7             7 0.2096312
## 255     C18D4.2 0.0052446565           14            14 0.2097483
## 132    C05D12.2 0.0052490933            7             7 0.2097538
```

```
#save SKAT results
write.table(x = All_SKAT_Data_reduced$results, file = "data/SKAT_all_reduced.resul
ts", row.names = FALSE, col.names = TRUE, quote = FALSE, append = FALSE)
```

Now we can see that only a handful of genes are associated with the phenotype - a much more reasonable number. So what's next? The gold standard in human studies would to be to perform an independent data collection and analysis on another sample of the population and if any genes show up as signficant in both sets of analyses then we would have a fair anount confidence that their disruption may cause the phenotpye/disease outcome.

In genetically manipuable organisms you could instead experimentally confirm that mutations in these genes are causative for the phenotpye/disease outcome via Crispr/Casp, genetic rescue, RNAi and/or mapping experiments.

References:

1. Kumar P, Henikoff S, Ng PC. 2009. Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. Nat Protoc 4: 1073-1081.

2. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC (2007) PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

3. Ramensky V, Bork P, Sunyaev S. 2002. Human non-synonymous SNPs: server and survey. Nucleic Acids Res 30: 3894-3900.

4. Thompson O, Edgley M, Strasbourger P, Flibotte S, Ewing B, Adair R, Au V, Chaudry I, Fernando L, Hutter H et al. 2013. The Million Mutation Project: A new approach to genetics in Caenorhabditis elegans.

Genome Res doi:gr.157651.113 (doi:gr.157651.113) [pii] 10.1101/gr.157651.113.
5. Wu MC, Lee S, Cai T, Li Y, Boehnke M, Lin X. 2011. Rare-variant association testing for sequencing data with the sequence kernel association test. American journal of human genetics 89: 82-93.