Conducting Linear Regression Based GWAS

Introduction

The goal of this exampel is to demonstrate how you can use functions within the Staphopia API to conduct a GWAS. For this demonstration we will be using an already published

Let's Begin

Import Packages

```
library(staphopia)
library(ggplot2)
```

Minimum Allele Frequency

We can set a constant for minimum allele frequency (MAF) at which we'll filter out the rare SNPs.

```
MAF <- 0.05
```

Acquiring the VISA tag

For ease of access Alam et al. 2014 has been stored under the Tag 'visa-gwas-2014'. The first thing we'll need to do is get information (id, comment, etc. . .) about the tag using the `get_tag_by_name` by name function.

```
tag <- get_tag_by_name('visa-gwas-2014')
tag
```

```
## $id
## [1] 10
##
## $tag
## [1] "visa-gwas-2014"
##
## $comment
## [1] "PMID=24787619; hVISA GWAS Study, Tim Read Lab, 75 sequenced samples"
##
## $user
## [1] 5
##
## $status
## [1] 200
```

Acquiring all samples associated with tag 'visa-gwas-2014'

Using the id of the tag (visa-gwas-2014) which we just acquired, we can easily pull down all the samples assocaited with tag with the `get_sampels_by_tag` function.

```
samples <- get_samples_by_tag(tag$id)
head(samples)
```

```
##       db_tag user_id is_paired sample_tag sample_id is_public is_published
## 1 ena_000001       5      TRUE  SRX476958       685      TRUE        FALSE
## 2 ena_000002       5      TRUE  SRX476959       686      TRUE        FALSE
## 3 ena_000003       5      TRUE  SRX476960       687      TRUE        FALSE
## 4 ena_000004       5      TRUE  SRX476961       688      TRUE        FALSE
## 5 ena_000005       5      TRUE  SRX476962       689      TRUE        FALSE
## 6 ena_000006       5      TRUE  SRX476963       690      TRUE        FALSE
```

Now the Phenotype

These data tested the phenotype of 74 *S. aureus* samples against Vancomycin. In this demonstration we will use the minimum inhibitatory concentrations (MIC ug/ml) as determined by the Etest.

First we will use `get_resistance` and specify the *antibiotic* as vancomycin and the *test* as Etest.

```
vancomycin <- get_resistance(antibiotic="vancomycin", test="etest")
vancomycin
```

```
## $id
## [1] 1
##
## $antibiotic
## [1] "Vancomycin"
##
## $test
## [1] "Etest"
##
## $unit
## [1] "MIC (ug/ml)"
##
## $status
## [1] 200
```

Now using the resistance id we just acquired, we can pull the resistance phenotype for each of samples but only pull down those associated with Vancomycin Etest. This is done using `get_resistance_by_sample` and setting the *resistance_id* parameter.

```
phenotype <- get_resistance_by_samples(samples$sample_id,
                                       resistance_id=vancomycin$id)
head(phenotype)
```

```
##    mic phenotype sample_id
## 1:   1      VSSA       685
## 2:   1      VSSA       689
## 3:   1      VSSA       694
## 4:   1      VSSA       695
## 5:   1      VSSA       696
## 6: 1.5      VSSA       686
```

Merge phenotype and samples, then sort by sample_tag

Nothing much going on here. In this step we are simply merging the phenotype data frame into the samples data frame. Then we sort the samples data frame by *sample_tag*.

```
samples <- merge(samples, phenotype)
samples <- samples[order(samples$sample_tag),]
head(samples)
```

```
##   sample_id     db_tag user_id is_paired sample_tag is_public is_published
## 1       685 ena_000001       5      TRUE  SRX476958      TRUE        FALSE
## 2       686 ena_000002       5      TRUE  SRX476959      TRUE        FALSE
## 3       687 ena_000003       5      TRUE  SRX476960      TRUE        FALSE
## 4       688 ena_000004       5      TRUE  SRX476961      TRUE        FALSE
## 5       689 ena_000005       5      TRUE  SRX476962      TRUE        FALSE
## 6       690 ena_000006       5      TRUE  SRX476963      TRUE        FALSE
##   mic phenotype
## 1   1      VSSA
## 2 1.5      VSSA
## 3 1.5      VSSA
## 4 1.5      VSSA
## 5   1      VSSA
## 6 1.5      VSSA
```

Acquiring all SNPs in each sample

We're conducting a GWAS, so we probably need some SNPs. There are two ways to get the SNPs associated with a sample. You can get them by sample with `get_snps(sample_id)` or you can get them by all samples at once using `get_snps_by_samples(sample_ids)`. In our case since we have 74 samples, we'll be using `get_snps_by_samples`.

```
snps <- get_snps_by_samples(samples$sample_id)
head(snps)
```

```
##    snp_id comment_id filters_id sample_id
## 1:    185          1         29       685
## 2:    454          1         29       689
## 3:    475          1         29       689
## 4:    561          1         29       686
## 5:    722          1         29       689
## 6:    799          1         29       689
```

Annotating *snp_id*

As you might notice we have a list of samples and snps, but they're only ids. There's not much biological information behind these ids. So let's use the `get_snps_in_bulk` to pull down all of the SNPs at once. In order to do so we must pass *snps$snp_id*. It will take care of only pulling the unique snp_ids.

```
snp_info <- get_snps_in_bulk(snps$snp_id)
head(snp_info[,1:8])
```

```
## [1] 1 2 3 4 5 6
```

Present/Absent SNP matrix

For our GWAS we'll need to build a simple matrix. In this matrix each column will represent a sample and each row a SNP position. Now for each SNP we need to determine if its present (1) or absent (0) in each sample. No worries! There's a function for that! Using `create_snp_matrix`, we can pass all the info we've acquired *snps*, *samples* and *snp_info*. Leaving us with something like the following.

```
snp_matrix <- create_snp_matrix(snps, samples, snp_info)
head(snp_matrix[,1:6])
```

```
##     SRX476958 SRX476959 SRX476960 SRX476961 SRX476962 SRX476963
## 13          0         0         0         0         0         0
## 62          1         0         0         0         0         0
## 66          0         0         0         0         0         0
## 89          0         0         0         0         0         0
## 93          0         0         0         0         0         0
## 137         0         0         0         0         0         0
```

Conducting a linear regression based GWAS

Now the moment we are all here for, GWAS! We'll be using R's bnuilt in `lm` (linear regression) function. It produces the same output as PLINK, just a bit slower. In order to do so, we can use the `run_gwas` function and pass our *snp_matrix* and the phenotype (*sample$mic*) to be tested.

```
gwas <- run_gwas(snp_matrix, samples$mic)
head(gwas)
```

```
##   position      pval       freq      logp
## 1       13 0.7058609 0.01351351 0.1512809
## 2       62 0.2510824 0.02702703 0.6001837
## 3       66 0.6822817 0.01351351 0.1660362
## 4       89 0.6737525 0.17567568 0.1714996
## 5       93 0.6737525 0.17567568 0.1714996
## 6      137 0.6737525 0.17567568 0.1714996
```

Remove SNPs < MAF

Rare SNPs may produce spurious results, so lets remove those SNPs that are at a frequency less than our minimum allele frequency (*MAF*) which we set earlier.

```
MAF
```

```
## [1] 0.05
```

```
# Pre-filter length
nrow(gwas)
```

```
## [1] 83487
```

```
gwas <- gwas[gwas$freq >= MAF,]

# Post-filter length
nrow(gwas)
```

```
## [1] 26401
```

```
head(gwas)
```

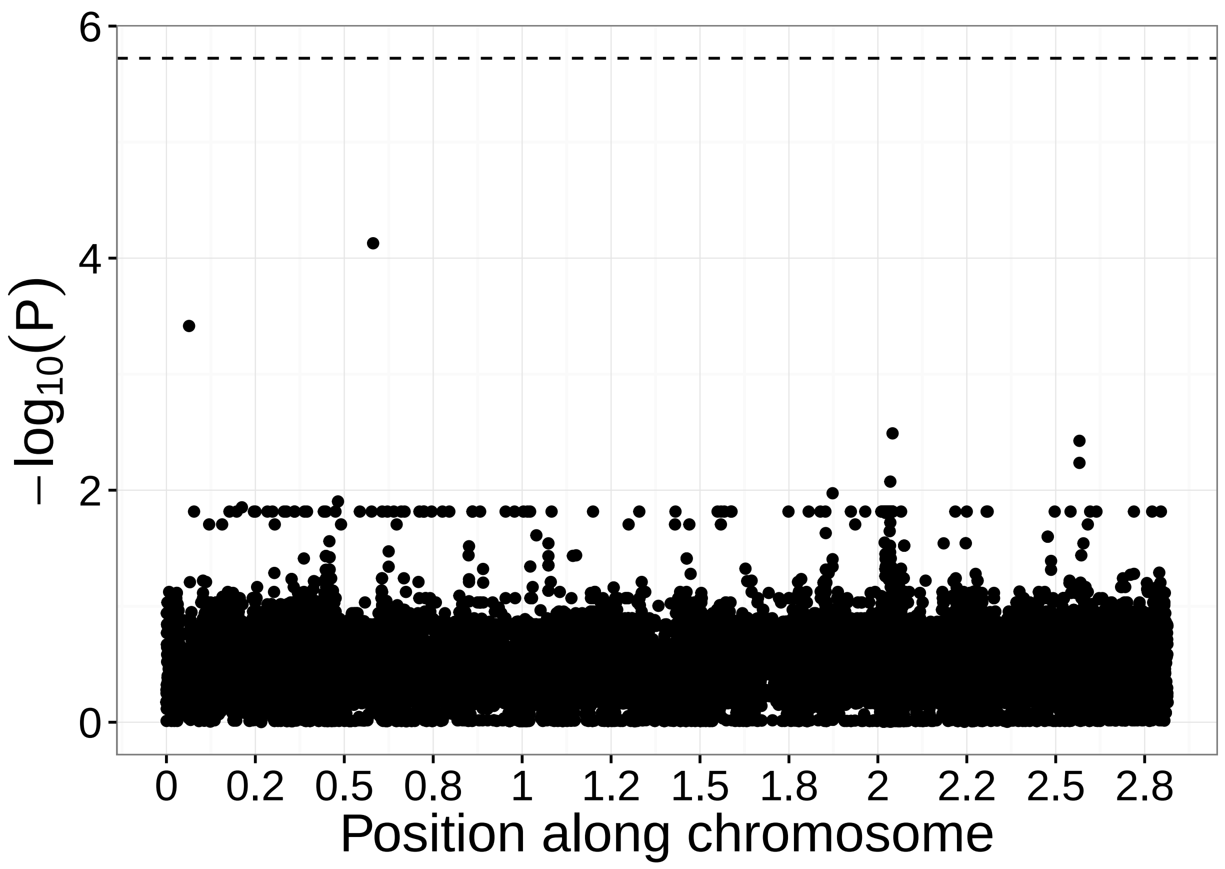
```
##    position      pval       freq      logp
## 4        89 0.6737525 0.17567568 0.1714996
## 5        93 0.6737525 0.17567568 0.1714996
## 6       137 0.6737525 0.17567568 0.1714996
## 7       152 0.6640341 0.05405405 0.1778096
## 10      165 0.5259754 0.20270270 0.2790346
## 18      290 0.5619739 0.06756757 0.2502839
```

Visualize the results

We can use `manhattan_plot` and `qq_plot` to visualize the results of our GWAS. By default each of the plots will use Bonferroni correction to determine significant SNPs.

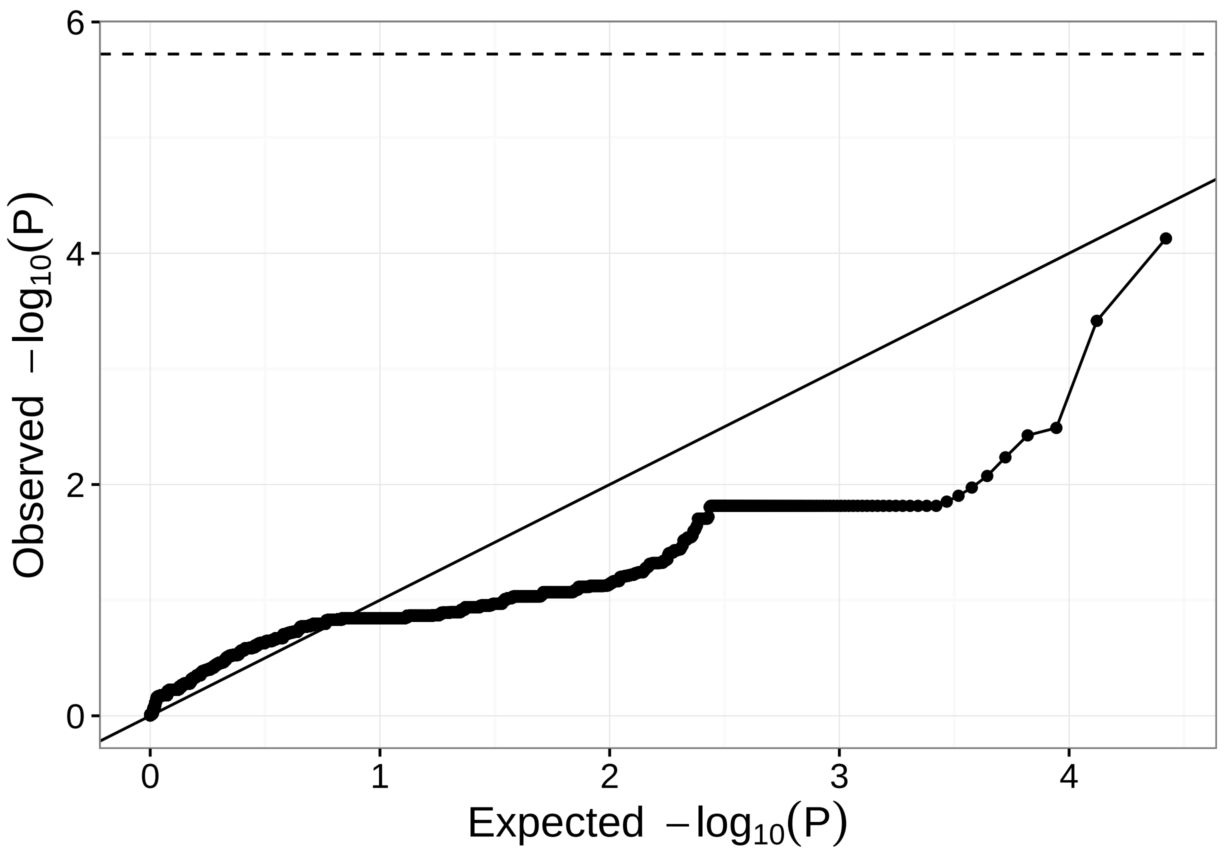
Manhattan Plot

```
manhattan_plot(gwas)
```



QQ Plot

```
qq_plot(gwas[with(gwas, order(logp)), ])
```



The End