gritsr3 Version-0.1.1 2016-06-28 22:31:06 PDT

Contents

1 Introduction

For questions or corrections, please email gailgongster@gmail.com

`gritsr` is an acronym for "Genome Resampling to Infer Test Significance R"

The main function of this package `gritsr` calculates the p-values of a family of statistics that test the association between a specified set of M genetic markers, called target markers, and a binary

trait. These statistics include three "basic" statistics–the squared burden statistic, the linear kernel (SKAT) statistic, and a new case-based kernel statistic–as well as an ensemble of four statistics that optimize linear combinations of the three basic statistics. All seven statistics are summarized in Table 1 of the manuscript.

The test statistics use observations on a number N of people, or subjects, who are sampled based on their unaffected/affected status of a binary trait such as prostate cancer, $y_n = 1(n\text{-th subject is affected})$. Also the calculations require a user-specified trait probability `e_y[n]` (for example the predicted value of y_n from a logistic regression on nongenetic covariates and principle components of ancestry), and a row vector of genotypes (either the dosage or carriage of the minor allele at a marker) $g_n = (g_{n1}, \cdots, g_{nM})$ at the M target markers. Define the genotype matrix of target markers, or more briefly, "genotype matrix" G to be the $N \times M$ matrix whose (n, m)-th component is the genotype of the n-th subject at the m-th target marker)

The test statistics take into account covariances of the elements in G. For pairwise covariances among the M markers (summarized by Γ in the manuscript) our calculations use the empirical covariance matrix of G. To account for pairwise covariances among the subjects, we require the user to provide a Ψ matrix consisting of interpersonal genotype correlation coefficients (described in the manuscript by the same notation).

Obtaining the p-values of the optimized statistics further requires resampling from sets of markers located throughout the human genome. We require M sampling sets, one for each target marker. The sampling set for the m-th target marker requires markers whose minor allele frequencies match closely with that of its target marker. Also the sampling sets should be independent, i.e. a marker chosen from one sampling set should be statistically independent from a marker chosen from a second set. The gritsr3 package can create these sampling sets given a user-provided data set containing genotypes of the N subjects at a large number of markers (in addition to the target markers) throughout the human genome. (In the analysis of the prostate data in the manuscript, we obtained `126702` markers, but we would have preferred even more.)

The creation of the sampling sets uses hotspots from "A Fine-Scale Map of Recombination Rates and Hotspots Across the Human Genome", Myers, Simon;Bottolo, Leonardo;Freeman, Colin;McVean, Gil;Donnelly, Peter Science; Oct 14, 2005; 310, 5746; ProQuest Research Library, pg. 321.

TOP

2 Installing `gritsr3`

If you have not yet installed R, download the latest version (at least 3.2.4) for your operating system at:

http://www.r-project.org

Run the R application. To install the `gritsr3` package, enter the following lines of code to the R prompt:

```
install.packages("devtools")
library(devtools)
install_github("gailg/gritsr3")
```

```
if("gritsr3" %in% rownames(installed.packages())){
  print("gritsr3 installed successfully--you are good to go!")
} else {
  print("something went wrong--ask for help")
}
```

If your installation was successful, you should see the message

```
[1] "gritsr3 installed successfully--you are good to go!"
```

TOP

3 Data required

The function `gritsr` requires quite large data, and typically, you would organize your data in Plink.(http://pngu.mgh.harvard.edu/~purcell/plink/). We assume that you can bring your data into R and create the following data sets.

3.1 `bim`

A `data.frame` containing (at least) the three columns `chromosome`, `snp`, `bp` and L rows corresponding to a very large number of L markers. These markers include the target markers and those used to build the sampling sets. The l-th row of `bim` summarizes the `l`-th marker and corresponds to the `l`-th column of `genotype`. `bim` could be the `.bim` file from plink.

3.2 `genotype`

A matrix with N rows and L columns, whose (n, l)-th element records either the number (0, 1 or 2) of minor alleles of marker l found in the n-th subject or an indicator (0 or 1) for the n-th person carrying at least one minor allele at marker l. The l-th column for `genotype` corresponds to the `l`-th row of `bim`. `genotype` could be the `.bed` file from plink after massaging genotype information into either dosage or carriage. (`genotype` is a matrix containing target markers AND sampling set markers; the "genotype matrix" G is a matrix containing just the target markers.)

3.3 `target_markers`

A vector that is a subset of the column `bim$snp`. This vector names the target markers.

3.4 `fam`

A `data.frame` containing N rows and at least the two columns y and `e_y` where $y_n = 1(n\text{-th person is affected})$ and `e_y[n]` is a trait probability, the predicted value of y_n based on nongenetic covariates and possibly principle components of ancestry. We performed a logistic

regression of y on age, membership into family or case/control data, and their interaction, and used `e_y` to be the fitted values of the logistic regression.

3.5 `Psi` = Ψ

A matrix with N rows and N columns. Ψ_{n_1,n_2} is the correlation for the genotype at one marker between the n_1-th and n_2-th subjects. Ψ can estimated from known family pedigree structures and/or from the subjects' genotypes at markers independent of those in the target set. The diagonal elements are all unity and two people known to be full sibs and who have parents known to be completely unrelated have correlation 1/2

3.6 `hotspot`

The `gritsr3` package provides this data set for your convenience. This file contains (at least) the columns `chromosome` and `center`. `center` describes the location of hotspots and `chromosome` describes the number of the chromosome (`1:22`).

TOP

4 An example data set

The package `gritsr3` provides an example data set which I will use to illustrate how to use the package. You can access this example data with the following commands.

```
library(gritsr3)
bim = alternative_example$bim
genotype = alternative_example$genotype
fam = alternative_example$fam
target_markers = alternative_example$target_markers[3:5]
Psi = alternative_example$Psi
target_markers = target_markers
```

4.1 `bim` has 24898 rows and includes the columns `chromosome`, `snp`, and `bp`

```
str(bim)
```

```
## 'data.frame':    24898 obs. of  6 variables:
##  $ chromosome: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ snp       : chr  "exm2216283" "exm55" "exm94" "exm158" ...
##  $ morgans   : num  100 0 0 0 0 0 0 0 0 0 ...
##  $ bp        : int  564766 865694 874501 878744 879381 880160 880502 881918 889387 90042
##  $ allele_1  : chr  "C" "T" "T" "C" ...
```

```
##  $ allele_2  : chr  "T" "C" "C" "G" ...
```

4.2 genotype has 200 rows and 24898 columns

I chose this smaller number of columns to keep the example small.

```
str(genotype)
```

```
##  int [1:200, 1:24898] 0 0 0 0 0 0 0 1 0 0 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:200] "1" "2" "3" "4" ...
##   ..$ : chr [1:24898] "exm2216283" "exm55" "exm94" "exm158" ...
```

The column names of `genotype` must match `bim$snp`:

```
all(colnames(genotype) == bim$snp)
```

```
## [1] TRUE
```

4.3 target_markers is a character vector containing 3

```
str(target_markers)
```

```
##  chr [1:3] "exm1061853" "exm1061861" "exm1061863"
```

The elements in `target_markers` must be included in `bim$snp`

```
all(target_markers %in% bim$snp)
```

```
## [1] TRUE
```

4.4 fam has 200 rows and includes the columns y and e_y

```
str(fam)
```

```
## 'data.frame':    200 obs. of  3 variables:
##  $ id : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ y  : int  1 1 1 1 1 1 1 1 1 0 ...
##  $ e_y: num  0.847 0.865 0.812 0.789 0.82 ...
```

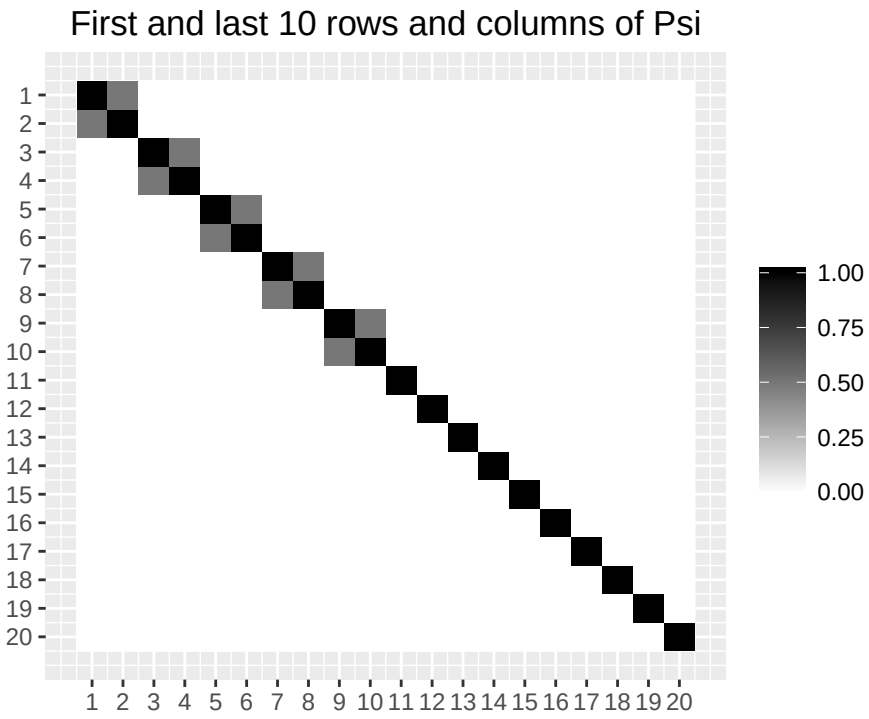
4.5 Psi is a 200 $\times$ 200 square matrix

```
str(Psi)
```

```
##  num [1:200, 1:200] 1 0.5 0 0 0 0 0 0 0 0 ...
```

The following figure reflects the fact that the first 100 people are made up of 50 sib pairs, and the last 100 people‘ are independent.

```
NNN = nrow(fam)
first_ten = 1:10
last_ten = NNN - (9:0)
matrix_image_fn(Psi[c(first_ten, last_ten), c(first_ten, last_ten)],
                main = "First and last 10 rows and columns of Psi")
```



4.6 `hotspot` is available as soon as you enter `gritsr3`:

```
str(hotspot)
```

```
## 'data.frame':    26177 obs. of  4 variables:
##  $ chromosome: chr  "1" "1" "1" "1" ...
##  $ center    : int  969634 1605634 1865633 1903633 2081633 2196633 2287633 2537633 25906
##  $ start     : int  610634 1602634 1830633 1900633 2074633 2188633 2263633 2534633 25806
##  $ end       : num  1105634 1607634 1892633 1905633 2091633 ...
```

5 Check that your genotype matrix has full rank

```
library(Matrix)
genotype_target_markers = genotype[, target_markers]
list(target_markers = target_markers,
     rank = as.numeric(rankMatrix( genotype_target_markers)))
```

```
## $target_markers
## [1] "exm1061853" "exm1061861" "exm1061863"
##
## $rank
## [1] 3
```

6 gritsr_sampling_set

Once you have your data in R, use the function `gritsr_sampling_set` to create your sampling sets.

6.1 A description of how `gritsr_sampling_set` builds the sampling sets

Each column in `genotype` corresponds to a row in `bim`, and both correspond to a marker. After removing the target markers, we want to form from the remaining markers M sampling sets, one for each of the target markers. Recall that the two conditions we require of the sampling sets are (1) the markers in the sampling set for a target marker has minor allele frequencies that match closely with that of the target marker and (2) a marker choosen from one sampling set is statistically independent of a marker chosen from a second sampling set.

We assume that hotspots from Myers et.al. cut the genome into independent segments, and so markers residing within two consecutive hotspots are independent of markers residing within another two consecutive hotspots. Defining a segment to be a set of markers that all lie within two consecutive hotspots, we obtain a (large) set of independent segments. If we assign each segment to at most one target marker (or one target marker's sampling set), we insure that the M sampling sets are independent, satisfying condition (2).

Our goal then becomes to assign segments to sampling sets so that each segment can contribute as many matching markers as possible to the sampling set it is assigned to, and also so that the minimum size across all sampling sets is maximized.

A segment can contribute to a target marker if it contains at least one marker that matches the target marker. If a segment can contribute to only one target marker, it is an easy decision to assign that segment to that target marker. However, if a segment can contribute to multiple target markers, then we need to decide which of the target markers we should assigned it to.

It turns out that there are many independent segment, and it is convenient to make the decision randomly according to a probability function $p = (p_1, \cdots, p_M)$ which assigns the segment to target marker m with probability p_m.

Let c_m count the number of markers in the segment that match target marker m. We tried different ways to build assignment probability functions p_m which depends on c_m. We may use $p_m = c_m$ (normalized) if we are using the counts for weights or $p_m = 1(c_m > 0)$ if not. We may or may not divide by C_m, the "popularity" of sampling set m, defined to be the sum of the c_m's over all the segments. We may or may not further adjust by dividing by the "expected" number of markers in each sampling set gotten by summing $p_m * c_m$ over all the segments (each segment giving a possibly different c_m).

When using a given probability function p, we often find that a few target snps have "deficient" sampling sets (their sizes being quite small compared to the sizes of other target snps). The probability p can be adjusted slightly to give these deficient target markers some additional weight: If a segment has no markers that match the deficient markers, keep p as before; otherwise zero out all p_m except those corresponding to the deficient ones.

Because sampling sets gotten by assigning a segment to a target marker m with probability p_m can vary from try to try, we can try several times and choose the one that give us the maximum of the minimum sampling set size.

6.2 The arguments of `gritsr_sampling_set`

The following is the function header of `gritsr_sampling_set` showing which arguments are needed and in which order.

```
gritsr_sampling_set(
  bim, genotype, target_markers, hotspot,
  epsilon_on_log_scale = 0.02,
  use_count_as_weight = FALSE,
  use_popularity = FALSE,
  use_expected = FALSE,
  markers_needing_help = NULL,
  N_try = 100
)
```

`bim`, `genotype`, `target_markers`, and `hotspot` are data sets already discussed in An example data set.

6.2.1 `epsilon_on_log_scale`

A positive small real number used to parametrize the matching

When creating the m-th sampling set for target marker with minor allele frequency p_m, only those markers whose minor allele frequencies falling within the interval $p_m \times \left[1-\texttt{epsilon\_on\_log\_scale}, 1+\texttt{epsilon\_on\_log\_scale}\right]$ can be included in the sampling set.

6.2.2 `use_count_as_weight`, `use_popularity`, `use_expected`

Logical arguments to tell `gritsr_sampling_set` how to build the assignment probabilities.

If `use_count_as_weight == TRUE`, then we begin by defining $p_m = c_m$, where c_m counts the number of markers in the segment that match target marker m. Otherwise we begin with $p_m = 1(c_m > 0)$.

If `use_popularity == TRUE`, then we also divide p_m by C_m where C_m is the "popularity" of sampling set m, equal to the sum of the c_m's over all the segments.

If `use_expected == TRUE`, then we also divide p_m by E_m equal to the sum of $p_m * c_m$ over all the segments (each segment giving a possibly different c_m).

6.2.3 markers_needing_help

Either NULL or an integer vector that is a subset of $1, \cdots, M$.

If markers_needing_help is a integer vector numbering a subset of $1, \cdots, M$, then for each segment, if it has no markers that match ones named in markers_needing_help, keep p as before; otherwise zero out all p_m except those giving positive weights to markers_needing_help.

6.2.4 N_try

A positive integer equal to the number of times we use the resulting probability p to assign segments to sampling sets. An assemblage of sampling sets which maximizes the minimum size is chosen.

6.3 Example calls to gritsr_sampling_set

```
set.seed(2)
sampling_set = gritsr_sampling_set(
  bim, genotype, target_markers, hotspot,
  epsilon_on_log_scale = 0.02,
  use_count_as_weight = FALSE,
  use_popularity = FALSE,
  use_expected = FALSE,
  markers_needing_help = NULL,
  N_try = 100
)
print(sampling_set)
```

```
## $inputs
## use_count_as_weight      use_popularity        use_expected
##               FALSE               FALSE               FALSE
##
## $markers_needing_help
## NULL
##
## $sampling_set_report
##      min p_target    max set_size
## 1 0.0175   0.0175 0.0175      497
## 2 0.0800   0.0800 0.0800      412
## 3 0.0025   0.0025 0.0025      423
##
## $minimum_sampling_set_size
## [1] 412
```

```
sampling_set = gritsr_sampling_set(
  bim, genotype, target_markers, hotspot,
  epsilon_on_log_scale = 0.02,
```

```
  use_count_as_weight = FALSE,
  use_popularity = TRUE,
  use_expected = TRUE,
  markers_needing_help = c(2, 3),
  N_try = 100
)
print(sampling_set)
```

```
## $inputs
## use_count_as_weight      use_popularity        use_expected
##               FALSE                TRUE                TRUE
##
## $markers_needing_help
## [1] 2 3
##
## $sampling_set_report
##      min p_target    max set_size
## 1 0.0175   0.0175 0.0175      438
## 2 0.0800   0.0800 0.0800      440
## 3 0.0025   0.0025 0.0025      443
##
## $minimum_sampling_set_size
## [1] 438
```

TOP

7 gritsr

7.1 A description of gritsr

`gritsr` calculates the p-values of the following seven statistics: the squared burden Q_B, the SKAT Q_S, the case based Q_C, and optimal linear combinations Q_{BS}, Q_{BC}, Q_{SC}, Q_{BSC}. The p-value of any linear combination of the basic statistics Q_B, Q_S, and Q_C can be calculated using the `davies` function from the `CompQuadForm` package, but the theory no longer applies when a statistic is an optimal one in a universe of linear combinations. We obtain the null distribution of these optimal statistics by resorting to a simulation.

Recall our notation: G is the $N \times M$ genotype matrix, y is the N-vector indicator for disease, and `e_y` is an N-vector of trait probabilities. Any linear combination of the basic statistics can be written as a quadratic form $Q(\alpha) = Z^T A(\alpha) Z$ where Z is a linear function of the two vectors WGy and WGp, W is a diagonal matrix that weights the target markers, $A(\alpha)$ is a square matrix that does not depend on G, and α are the coeffients of the linear combination. Z is a multivariate normal vector when N is large, with mean μ and covariance matrix Σ which depend on the first two moments of G. For a fixed value of α, the `davies` function can calculate the p-value for $Q(\alpha)$. Each optimal statistic Q_{BS}, Q_{BC}, Q_{SC}, Q_{BSC} is a monotonoic (decreasing) function of the smallest p-value of $Q(\alpha)$ over a set of values of α.

To run a simulation to get the null distribution of an optimal statistic Q_{optimal}, we need to obtain simulated observations of the genotype matrix $\left\{\tilde{G}^{(1)}, \cdots, \tilde{G}^{(S)}\right\}$. On the s-th simulated genotype matrix $\tilde{G}_s$, we calculate $Q(\alpha)$ over the set of values of α and obtain $Q^{(s)}_{\text{optimal}}$. The p-value of Q_{optimal} is the proportion of $\left\{Q^{(1)}_{\text{optimal}}, \cdots, Q^{(S)}_{\text{optimal}}\right\}$ that exceed Q_{optimal}.

The question becomes how to generate each $\tilde{G}^{(s)}$ that has the same distribution as the observed genotype matrix G? We propose *Genome Resampling*. The idea is to observe, for each person in the sample, not only the target markers but many additional markers (along the genome), and then form $\tilde{G}^{(s)}$ by sampling from these additional markers, each marker contributing a column to $\tilde{G}^{(s)}$. The attractiveness of this approach is that the rows of the resulting $\tilde{G}^{(s)}$ have the same covariance as the rows of the target genotype matrix (because. Simulations involving different family designs and different degrees of correlations of the target markers show that it is not necessary to match column covariances of $\tilde{G}^{(s)}$ to those of the oberved genotype matrix for this approach to still give good test sizes. Provided the sampled markers are independent and their minor allele frequencies match, using genome resampling gives good test sizes. We believe it is possible to have some nonzero correlations between the sampled markers, but since there can be such a wide variety of possible correlations if we allowed them, we chose to limit our simulation studies and therefore our prescription to independent sampling sets.

7.2 The arguments of `gritsr`

The following is the function header of `gritsr` showing which arguments are needed and in which order.

`gritsr(fam, Psi, sampling_set, N_sim_reps)`

`fam` and `Psi` are data sets already discussed in An example data set.

`sampling_set` is the result of `gritsr_sampling_set` and you may follow the example from gritsr_sampling_set

7.2.1 'N_sim_reps

An positive integer which specifies the number of replications in the something. TOP

7.3 Example calls to `gritsr`

These take some time to run.

```
start = Sys.time()
gritsr(fam, Psi, sampling_set, N_sim_reps = 1000)
```

```
##              B           S           C    BS    SC    BC   BSC
## 1 0.0072327253 0.027497087 0.010899414 0.006 0.018 0.003 0.002
```

```
elapsed_time = Sys.time() - start
paste("1000 sim reps required",
      round(elapsed_time, 1), attributes(elapsed_time)$units)
```

```
## [1] "1000 sim reps required 7.6 mins"
```

```
start = Sys.time()
gritsr(fam, Psi, sampling_set, N_sim_reps = 2000)
```

```
##              B           S           C    BS    SC    BC    BSC
## 1 0.0072327253 0.027497087 0.010899414 0.008 0.016 0.004 0.0045
```

```
elapsed_time = Sys.time() - start
paste("2000 sim reps required",
      round(elapsed_time, 1), attributes(elapsed_time)$units)
```

```
## [1] "2000 sim reps required 15.3 mins"
```

8 All the code in one place

```
library(gritsr3)

# Preparing the data
bim = alternative_example$bim
genotype = alternative_example$genotype
fam = alternative_example$fam
target_markers = alternative_example$target_markers[3:5]
Psi = alternative_example$Psi
target_markers = target_markers

NNN = nrow(fam)
first_ten = 1:10
last_ten = NNN - (9:0)
matrix_image_fn(Psi[c(first_ten, last_ten), c(first_ten, last_ten)],
                main = "First and last 10 rows and columns of Psi")

# Checking the rank of the genotype_target_markers matrix
library(Matrix)
genotype_target_markers = genotype[, target_markers]
list(target_markers = target_markers,
     rank = as.numeric(rankMatrix( genotype_target_markers)))

# Creating the sampling_set
set.seed(2)
sampling_set = gritsr_sampling_set(
  bim, genotype, target_markers, hotspot,
```

```
  epsilon_on_log_scale = 0.02,
  use_count_as_weight = FALSE,
  use_popularity = FALSE,
  use_expected = FALSE,
  markers_needing_help = NULL,
  N_try = 100
)
print(sampling_set)

sampling_set = gritsr_sampling_set(
  bim, genotype, target_markers, hotspot,
  epsilon_on_log_scale = 0.02,
  use_count_as_weight = FALSE,
  use_popularity = TRUE,
  use_expected = TRUE,
  markers_needing_help = c(2, 3),
  N_try = 100
)
print(sampling_set)

# Calling gritsr with N_sim_reps = 1000 and 2000
start = Sys.time()
gritsr(fam, Psi, sampling_set, N_sim_reps = 1000)
elapsed_time = Sys.time() - start
paste("1000 sim reps required",
      round(elapsed_time, 1), attributes(elapsed_time)$units)
start = Sys.time()
gritsr(fam, Psi, sampling_set, N_sim_reps = 2000)
elapsed_time = Sys.time() - start
paste("2000 sim reps required",
      round(elapsed_time, 1), attributes(elapsed_time)$units)
```

‘‘‘