RSNPset: An Open R Package for Genome-Wide SNP Set Analysis on the Basis of Efficient Scores

December 16, 2014

Abstract

This package offers the capability to conduct genome-wide inference for case-control studies, including studies with censored phenotypes, using efficient score statistics. This document is intended to supplement the package and function documentation, to more thoroughly address finer statistical points encountered when applying this approach.

1 Introduction

Analysis of high-throughput genomic assay data on the basis of the score is asymptotically equivalent to Wald or likelihood ratio tests, offers higher computational efficiency, greater stability, and more readily lends itself to the use of resampling methods. Though the estimation of unknown nuisance parameters may induce variability on the score, the use of the efficient score accounts for this.

The RSNPset package provides a software implementation of efficient score statistics in genome-wide SNP set analysis of complex traits. This document provides an overview of the package, its methods for statistical inference, some example usage, and an explanation of the statistical assumptions of its arguments and options. By way of example, the SNP sets discussed here use genes as the loci of interest, i.e. they are sets of SNPs relevant to specific genes, but the approach is suitable for other genomic loci, including pathways and bands.

2 Score Calculations

We begin by establishing some notation. Let the number of patients be denoted by n (indexed by i), and the number of SNPs be denoted m (indexed by j). The complete set of genotypes for all patients forms the matrix G, where the genotype for SNP j in patient i is denoted by

G_{ij}. The vector of outcomes for all patients is denoted Y, where the outcome for patient i is denoted by Y_i. (Or, for right-censored endpoints, what is observed is (Y_i, Δ_i), where Y_i is the event time and $\Delta_i \in \{0, 1\}$ is the event indicator).

The marginal null hypothesis for SNP j (that the variant is not associated with the outcome) is denoted by H_j. This hypothesis is tested using an efficient score statistic whose numerator is of the form $\sum_{i=1}^{n} U_{ij}$, where U_{ij} denotes the contribution of patient i to the efficient score~[2] statistic. Each of K SNP sets, J_k, are composed of m_k SNPs, and the null hypothesis for each SNP set is denoted by $\mathbb{H}_k = \cap_j H_j$ where $j \in J_k$. Table 1 shows the specific equations for U_{ij} for different types of outcomes.

Outcome	Score	U_{ij}
$Y_i \in \{0, 1\}$	Binomial	$(G_{ij} - \bar{G}_j)(Y_i - \bar{Y})$
$Y_i \in \mathbb{R}$	Gaussian	$(G_{ij} - \bar{G}_j)(Y_i - \bar{Y})(\frac{1}{n}\sum_{l=1}^{n}(Y_l - \bar{Y})^2)^{-1}$
$Y_i \in (0, \infty), \Delta_i \in \{0, 1\}$	Cox	$\Delta_i(G_{ij} - a_j(i)b(i)^{-1})$

Table 1: Summary of score calculations. Here $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ and $\bar{G}_j = \frac{1}{n}\sum_{i=1}^{n} G_{ij}$. For the Gaussian score, the calculation of the MLE estimator of the variance, $\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2$, is omitted. For the Cox score, $a_j(i) = \sum_{l=1}^{n} \mathbb{I}[Y_l \geq Y_i] G_{lj}$ and $b(i) = \sum_{l=1}^{n} \mathbb{I}[Y_l \geq Y_i]$.

To derive the efficient score for the SNP set, we construct $\boldsymbol{U}_{k,n}$, the $n \times m_k$ matrix whose (i, j) element is U_{ij}. Then $\boldsymbol{U}_{\bullet k,n} = (\sum_{i=1}^{n} U_{i1}, \ldots, \sum_{i=1}^{n} U_{im_k})^T$ is the corresponding vector of the m_k marginal score statistics, and $\boldsymbol{\Sigma}_{k,n} = \boldsymbol{U}_{k,n}^T \boldsymbol{U}_{k,n}$ is the corresponding covariance matrix. The efficient score statistic for the SNP set hypothesis $\mathbb{H}_k$ then is $W_k = \boldsymbol{U}_{\bullet k,n}^T \boldsymbol{\Sigma}_{k,n}^{+} \boldsymbol{U}_{\bullet k,n}$, where $\boldsymbol{\Sigma}_{k,n}^{+}$ is the Penrose-Moore inverse of $\boldsymbol{\Sigma}_{k,n}^{+}$.

Under suitable regularity conditions~[1], W_k converges in distribution to a chi-squared distribution with degrees of freedom $\nu = R_k \in \{1, \ldots, m_k\}$ and centrality parameter δ, denoted by $\chi^2[\nu, \delta]$. As R_k is an unknown parameter, under $\mathbb{H}_k$ we approximate the null distribution of W_k by a $\chi^2[r_k, 0]$ distribution, where r_k is the rank of $\boldsymbol{\Sigma}_{k,n}$. Thus we are able to compute the efficient score and asymptotic p-value for each of the K SNP sets.

2.1 Notes on `rspset()`

Users should be aware that this function does not check to confirm that the elements of a SNP set are present in the matrix G before executing. No results are returned for SNP sets for which no SNPs are present or which include **any** SNPs with **missing** values. If a SNP set contains some column names that are not present in the argument `G`, the function executes without objection and returns a test statistic based on the subset of columns that *are* present. For this reason, the `summary()` function provides counts of the number of SNP sets dropped from the analysis and the number of SNP sets defined to include SNPs that are not present in the data, if any, (in addition to other execution information).

2.2 Options for `rspset()`

The following options imply specific assumptions about the model or data. It is important to confirm that these assumptions are valid when executing the analysis.

`v.permute` - Under the assumption that $\mathbf{\Sigma}_{k,n}$ for SNP set J_k is the same across permutations, setting this value to `FALSE` saves processing time by not re-calculating the variance of the permutation replicates.

`ret.rank` - Under the assumption that the rank of $\mathbf{\Sigma}_{k,n}$ for SNP set J_k is invariant under permutation, setting this option to `FALSE` reduces the size of the returned object by not returning ranks for the permutation replicates. Ranks are returned only for the observed data.

`pinv.check` - The calculation of the efficient score relies on the Penrose-Moore inverse of the variance matrices. Setting this argument to `TRUE` returns several diagnostic measures of the accuracy of this inverse. Departure of these values from zero indicates poor performance of the Penrose-Moore inverse. These diagnostics are returned as an attribute that can be accessed via the `summary()` function.

3 Hypothesis Testing

At minimum, the `rspset.pvalue()` function returns asymptotic p-values and false discovery rate (FDR) adjusted q-values for each SNP set. Setting the `rspset()` argument `B` > 0 allows `rspset.pvalue()` to return permutation p-values (and FDR adjusted permutation q-values) as well. By default, the `rspset.pvalue()` function uses the `qvalue` package to compute the q-values, though the `qfun` argument can be used to provide a custom definition.

3.1 Notes on `rspset.pvalue()`

Several considerations need to be made in testing the association of a SNP set with the outcome. Firstly, meaningful unadjusted permutation p-values require `B` $> K/\alpha$ (on the order of 10^7 permutations for a genome-wide study) in order to account for false positives due to multiple testing. Note that in the event that none of the permutation replicates generates a more extreme result than the observed value, a permutation p-value of $1/$`B` is returned (instead of 0).

Second, by default `rspset()` and `rspset.pvalue()` operate in accordance with the assumption that each $\mathbf{\Sigma}_{k,n}$ is invariant under permutation, in which case the degrees of freedom of the chi-squared distribution for the statistics, W_k, are the same across permutations. If that assumption is valid, then the observed and permutation statistics for a SNP set are directly comparable. However, if the ranks of the $\mathbf{\Sigma}_{k,n}$ differ across permutations, then the

chi-squared distributions differ in degrees of freedom, so the raw statistics are not comparable. In this latter case, the arguments of `rspset()` should be set to return the ranks of the permutation replicates (`ret.rank = TRUE`), and `rspset.pvalue()` should be run with the `pval.transform` argument set to `TRUE`. When `pval.transform = TRUE`, instead of comparing the observed and permutation statistics, the permutation p-values are determined by comparing observed asymptotic p-values to the asymptotic p-values of the permutation replicates.

3.2 Options for `rspset.pvalue()`

`pval.transform` - As mentioned above, when this option is set to `TRUE`, the function uses the ranks of the permutation replicates to get permutation replicate p-values (which are identically distributed, and thus comparable) to determine the empirical permutation p-values of the statistics.

`qfun` - The `qvalue()` function, which is used internally in this function, may fail when used on a small number of replications or SNP sets. This argument is used to define a new q-value function, or to assign arguments for the `qvalue()` function. For example, `qfun = function(x){qvalue(x, robust = TRUE)$qvalue}` can be used to avoid generating errors if the number of SNP sets is small.

4 An Example Analysis

In practice, analyses might include hundreds of patients and thousands of SNP sets, spanning tens of thousands of SNPs. For the purposes of this demonstration, we simulate a more manageable example. Users should also note than an important precursor to genome-wide analyses is quality control of the genotypic data. As our data is simulated and complete, we can omit this step and proceed with our analysis.

4.1 Simulating the Data

First, we generate a cohort of `n` patients and their outcome data, i.e., the traits we wish to analyze: case-control status, LDL level, and survival time/survival status.

```
set.seed(123)

n <- 100
status <- rbinom(n, 1, 0.5)
table(status)


## status
```

```
##  0  1
## 53 47
```

```
LDL <- rnorm(n, mean=115, sd=35)
quantile(LDL)
```

```
##        0%       25%       50%       75%      100%
##  34.17909  91.86624 113.23968 133.43291 191.55665
```

```
time <- rexp(n, 1/10)
event <- rbinom(n, 1, 0.9)
quantile(time)
```

```
##         0%        25%        50%        75%       100%
##  0.2417337  3.6029326  7.6329649 15.4336484 43.6591094
```

```
table(event)
```

```
## event
##  0  1
## 10 90
```

Next we simulate the genomic data. Here we are using allele counts, but the methodology is also applicable to expression levels. For each SNP, we let the probability of having a mutant allele be a random value selected from a uniform distribution across the interval $(0.1, 0.9)$.

```
m <- 500

G <- matrix(as.double(rbinom(n*m, 2, runif(n*m,.1,.9))), n, m)
dim(G)
```

```
## [1] 100 500
```

We must label all of the SNPs so that they can be referenced by the SNP sets. The rows correspond to the genotypes of each patient, and the columns are the allele counts for each SNP.

```
rsIDs <- paste0("rs100",1:m)
colnames(G) <- rsIDs
```

```
G[1:5,1:5]
```

```
##      rs1001 rs1002 rs1003 rs1004 rs1005
## [1,]      2      1      0      2      0
## [2,]      0      0      2      1      2
## [3,]      1      0      1      2      0
## [4,]      2      2      1      2      2
## [5,]      1      0      0      0      1
```

Next we make the SNP sets. For our example we compose `K = 10` sets of between three and fifty SNPs at random, but we imagine them to represent groups of SNPs related to specific genes (or perhaps some other genomic loci, such as bands or pathways).

```
K <- 10
genes <- paste0("XYZ",1:K)
geneSets <- lapply(sample(3:50, size=K, replace=TRUE), sample, x=rsIDs)
names(geneSets) <- genes

unlist(lapply(geneSets, length))
```

```
##  XYZ1  XYZ2  XYZ3  XYZ4  XYZ5  XYZ6  XYZ7  XYZ8  XYZ9 XYZ10
##    45    23     3     9    36    18    19     8    32    14
```

For a systematic approach to generating gene-based SNP sets from real genomic data, see the `snplist`~[3] package.

4.2 Conducting the Analysis

After installing its dependent packages, we load `RSNPset`. The fact that `RSNPset` utilizes the `doRNG` package gives us the ability to set a seed so that our analyses are reproducible.

```
library(RSNPset)
set.seed(456)
```

We look first at our binary phenotype, case-control status. Again, a dramatically greater number of permutations is generally required in order to attain meaningful results. We use just a few here for the purposes of demonstration.

```
ccres <- rsnpset(Y=status, G=G, snp.sets=geneSets, score="binomial", B=10, ret.rank=TRUE
```

```
## Loading required package:  foreach
## Loading required package:  rngtools
## Loading required package:  pkgmaker
## Loading required package:  registry
```

The resulting object, `ccres`, is a list of `B+1` data frames. Each data frame has one row per (non-empty) SNP set and a column, `W`, containing the test statistics. The first data frame contains the statistics for the observed data, while the remainder correspond to the permutation results. Since we set `ret.rank = TRUE`, each data frame also contains a column with the rank of the covariance matrix, $\Sigma_{k,n}$, i.e. the degrees of freedom for our chi-squared test. The first data frame also contains a column, `m`, denoting the number of SNPs in each SNP set.

```
ccres[["Observed"]]
```

```
##                W rank  m
## XYZ1  49.496824   45 45
## XYZ2  14.456745   23 23
## XYZ3   3.948585    3  3
## XYZ4  17.489361    9  9
## XYZ5  34.604769   36 36
## XYZ6   8.831058   18 18
## XYZ7  20.132735   19 19
## XYZ8   3.550083    8  8
## XYZ9  37.233857   32 32
## XYZ10 15.317483   14 14
```

The `summary()` function displays the execution parameters for the returned object.

```
summary(ccres)
```

```
## - Efficient score statistics based on 100 samples.
## - SNP sets range in size from 3 to 45.
## - 0 SNP sets were not included in the analysis
## - 0 SNP sets contained SNPs that were not included in the analysis.
## - 10 permutation replicates were computed.
## - ret.rank = TRUE : The ranks of the permutation variance matrices were returned.
## - v.permute = TRUE : Variance was recomputed for each permutation replicate.
## [1] NA
```

We use `rspset.pvalue()` to check for evidence for rejecting our null hypotheses. Since we have the ranks for the permutation replicates, we set `pval.transform = TRUE` to ensure the results across permutations are comparable.

```
rsnpset.pvalue(ccres, pval.transform=TRUE)
```

```
##                W rank  m          p  pB  PB         Q        QB
## XYZ1  49.496824   45 45 0.29845127 0.2 0.9 0.6443850 0.6666667
## XYZ2  14.456745   23 23 0.91304199 0.9 1.0 0.9635286 1.0000000
## XYZ3   3.948585    3  3 0.26706978 0.3 0.8 0.6443850 0.6666667
## XYZ4  17.489361    9  9 0.04158187 0.1 0.3 0.4158187 0.6666667
## XYZ5  34.604769   36 36 0.53493377 0.7 1.0 0.7641911 1.0000000
## XYZ6   8.831058   18 18 0.96352860 1.0 1.0 0.9635286 1.0000000
## XYZ7  20.132735   19 19 0.38663103 0.4 0.9 0.6443850 0.6666667
## XYZ8   3.550083    8  8 0.89526815 0.8 1.0 0.9635286 1.0000000
## XYZ9  37.233857   32 32 0.24071560 0.3 0.7 0.6443850 0.6666667
## XYZ10 15.317483   14 14 0.35680211 0.4 0.9 0.6443850 0.6666667
```

As explained above, setting `pval.transform = TRUE` means the function compares asymptotic p-values across permutations, instead of raw statistics, in computing the permutation p-values (`pB`). Having computed these p-values, function also returns family-wise error adjusted p-values (`PB`), in addition to the asymptotic (`p`) and FDR adjusted p-values (`Q` and `QB`). As expected, since the data are simulated, we find no significant association between any SNP sets and the outcome. (Recall that the asymptotic p-value for *XYZ4* is not significant under multiple testing).

We move on to the continuous phenotype, LDL level. Here we revert to the default values for the arguments `ret.rank` and `v.permute`. In practice, when used on a much larger set of data, these changes offer the potential for a significant reduction in both the processing time and the size of the returned object.

```
ldlres <- rsnpset(Y=LDL, G=G, snp.sets=geneSets, score="gaussian", B=10)
```

Since the default is `ret.rank = FALSE`, only the first data frame contains the column with the ranks of the covariance matrices.

```
ldlres[["Observed"]]
```

```
##                W rank  m
## XYZ1  30.668778   45 45
## XYZ2  27.955484   23 23
## XYZ3   7.854819    3  3
```

```
## XYZ4   5.302372    9  9
## XYZ5  44.631803   36 36
## XYZ6  16.320559   18 18
## XYZ7  19.061289   19 19
## XYZ8   5.834179    8  8
## XYZ9  33.492914   32 32
## XYZ10  8.165996   14 14
```

```
ldlres[["Permutation.1"]]
```

```
##                W
## XYZ1  43.641053
## XYZ2  24.583065
## XYZ3   9.144861
## XYZ4  21.242080
## XYZ5  31.218468
## XYZ6  21.755842
## XYZ7  17.627167
## XYZ8   6.083224
## XYZ9  25.883371
## XYZ10 13.402500
```

Here the statistics of the permutation replicates are assumed to have the same degrees of freedom as the observed statistics.

```
rsnpset.pvalue(ldlres)
```

```
##                W rank  m         p  pB         Q   QB
## XYZ1  30.668778   45 45 0.9491987 1.0 0.9491987 1.00
## XYZ2  27.955484   23 23 0.2174697 0.1 0.7248990 0.25
## XYZ3   7.854819    3  3 0.0491094 0.1 0.4910940 0.25
## XYZ4   5.302372    9  9 0.8071934 0.8 0.9491987 1.00
## XYZ5  44.631803   36 36 0.1531673 0.1 0.7248990 0.25
## XYZ6  16.320559   18 18 0.5701863 0.4 0.9491987 0.80
## XYZ7  19.061289   19 19 0.4529107 0.6 0.9058215 1.00
## XYZ8   5.834179    8  8 0.6658001 0.8 0.9491987 1.00
## XYZ9  33.492914   32 32 0.3947629 0.1 0.9058215 0.25
## XYZ10  8.165996   14 14 0.8804998 0.9 0.9491987 1.00
```

Again, these permutation p-values (`pBk`) are computed under the assumption of identical covariance matrices, so the observed and permutation statistics are directly comparable.

Lastly, we look at our right-censored (time to event) phenotype.

```
tteres <- rsnpset(Y=time, delta=event, G=G, snp.sets=geneSets, score="cox", B=10, pinv.c
```

By setting `pinv.check = TRUE`, the returned object now includes an attribute giving five diagnostic measures of the Penrose-Moore Inverse computed for each SNP set in all of the permutation replicates and the observed data. These are accessed via the `summary()` function. The (B+1) $\times K \times 5$ data frame can be captured for examination.

```
pinv.diag <- summary(tteres)

## - Efficient score statistics based on 100 samples.
## - SNP sets range in size from 3 to 45.
## - 0 SNP sets were not included in the analysis
## - 0 SNP sets contained SNPs that were not included in the analysis.
## - 10 permutation replicates were computed.
## - ret.rank = FALSE : The ranks of the permutation variance matrices were not returned
## - v.permute = TRUE : Variance was recomputed for each permutation replicate.
## - pinv.tol = 7.8e-08

pinv.diag[["Observed"]]

##               d0           d1           d2           d3           d4
## 1  4.580336e-12 4.572343e-12 1.025352e-14 1.098930e-13 1.095877e-13
## 2  2.316369e-12 2.415845e-12 4.319461e-16 1.621750e-14 1.620579e-14
## 3  4.973799e-14 3.552714e-14 1.387779e-17 6.245005e-17 4.943962e-17
## 4  6.195933e-12 6.181722e-12 1.568190e-15 2.101097e-14 2.118965e-14
## 5  5.414336e-12 5.414336e-12 8.605963e-15 5.448211e-13 5.448558e-13
## 6  3.228617e-11 3.222711e-11 9.499292e-15 8.204375e-14 8.203854e-14
## 7  3.481659e-12 3.510081e-12 2.126771e-15 1.277234e-14 1.281397e-14
## 8  1.126210e-12 1.110667e-12 5.127463e-16 1.070888e-14 1.075095e-14
## 9  2.067679e-12 2.039258e-12 1.314920e-15 4.520516e-14 4.528669e-14
## 10 1.192912e-11 1.192024e-11 4.807786e-15 1.747422e-13 1.749087e-13

unlist(lapply(pinv.diag, max))

##       Observed  Permutation.1  Permutation.2  Permutation.3
##   3.228617e-11   4.040146e-11   1.968870e-11   2.260236e-11
##  Permutation.4  Permutation.5  Permutation.6  Permutation.7
##   2.794565e-11   2.551026e-11   2.748379e-11   2.933120e-11
##  Permutation.8  Permutation.9 Permutation.10
##   2.814460e-11   2.918599e-11   1.972289e-11
```

The diagnostic measures are all close to zero, indicating the Penrose-Moore Inverses are performing well, so we proceed to examining the p-values.

```
ttepvals <- rsnpset.pvalue(tteres)
ttepvals
```

```
##                W rank  m         p  pB         Q        QB
## XYZ1  55.944016   45 45 0.1270435 0.1 0.1631446 0.5767282
## XYZ2  16.004060   23 23 0.8550949 0.8 0.2282156 0.7177063
## XYZ3   3.619096    3  3 0.3056411 0.4 0.1631446 0.5767282
## XYZ4  12.586511    9  9 0.1822259 0.4 0.1631446 0.5767282
## XYZ5  33.905012   36 36 0.5685958 0.5 0.1921430 0.5767282
## XYZ6  22.151629   18 18 0.2253228 0.4 0.1631446 0.5767282
## XYZ7  16.139634   19 19 0.6479420 0.9 0.1921430 0.7266776
## XYZ8   7.782686    8  8 0.4549809 0.2 0.1921430 0.5767282
## XYZ9  35.530468   32 32 0.3054573 0.5 0.1631446 0.5767282
## XYZ10 11.595316   14 14 0.6387660 0.7 0.1921430 0.7064921
```

A `summary()` method is also available for the results of `rspset.pvalue()`, by default returning the top ten most significant SNP sets by asymptotic p-value. The `verbose = TRUE` option gives additional information about the calculations.

```
summary(ttepvals, verbose=TRUE)
```

```
##
## - Permutation p-values (pB) come from comparison of
## test statistics across 10 replications.
## - Q-values based on 10 SNP sets.
##                W rank  m         p  pB         Q        QB
## XYZ1  55.944016   45 45 0.1270435 0.1 0.1631446 0.5767282
## XYZ4  12.586511    9  9 0.1822259 0.4 0.1631446 0.5767282
## XYZ6  22.151629   18 18 0.2253228 0.4 0.1631446 0.5767282
## XYZ9  35.530468   32 32 0.3054573 0.5 0.1631446 0.5767282
## XYZ3   3.619096    3  3 0.3056411 0.4 0.1631446 0.5767282
## XYZ8   7.782686    8  8 0.4549809 0.2 0.1921430 0.5767282
## XYZ5  33.905012   36 36 0.5685958 0.5 0.1921430 0.5767282
## XYZ10 11.595316   14 14 0.6387660 0.7 0.1921430 0.7064921
## XYZ7  16.139634   19 19 0.6479420 0.9 0.1921430 0.7266776
## XYZ2  16.004060   23 23 0.8550949 0.8 0.2282156 0.7177063
```

As a typical GWAS study may span thousands of SNPs and SNP sets, `summary()` allows for the succinct listing of p-values for the most significant results. The returned data frame can be saved for future reference or reporting.

```
ttesum <- summary(ttepvals, sort="pB", nrows=5, dropcols=c("m","Q","QB"))
ttesum
```

```
##               W rank         p  pB
## XYZ1 55.944016   45 0.1270435 0.1
## XYZ8  7.782686    8 0.4549809 0.2
## XYZ3  3.619096    3 0.3056411 0.4
## XYZ4 12.586511    9 0.1822259 0.4
## XYZ6 22.151629   18 0.2253228 0.4
```

References

[1] Andrews D.K.W. Asymptotic Results for Generalized Wald Tests. *Econometric Theory*, 3:348-358, 1987.

[2] Tsiatis A.A. *Semiparametric Theory and Missing Data.* Springer Science+Business Media, LLC, New York, NY, 2006.

[3] `snplist` is available under the GNU General Public License from the Comprehensive R Archive Network (CRAN) web site.