seqMeta: an R Package for meta-analyzing region-based tests of rare DNA variants

September 9, 2013

Abstract

Region-based tests are becoming a popular tool for analyzing rare genetic variants. In order for these tests to have adequate power, it is often necessary to meta-analyze information from multiple contributing studies, where consent restrictions make it difficult or impossible to share individual level data. We present the R package **seqMeta** for meta-analyzing region based tests, such as SKAT, SKAT-O, and burden tests, as well as single variant tests. The package can accommodate continuous, binary, and survival outcomes for unrelated individuals, and continuous outcomes for related individuals. We also provide convenient functions for conditional analyses and parallelization.

1 Introduction

In association studies of rare genetic variation it is common to pool variants across a gene, region, or pathway to improve statistical power and performance of tests. In order for these test to have adequate power to detect rare variants with modest effects, it is often necessary to use data from multiple contributing studies using a meta-analysis. While meta-analyzing single-variant effect estimates and their standard errors is relatively straightforward, implemented in e.g. METAL, meta-analyzing region-based tests is somewhat more complicated. One popular region based test is SKAT, which is based on a weighted sum of single-variant score tests, using the covariance matrix of the scores to calculate p-values (Wu *et al.*, 2011) . Meta-analyzing SKAT requires meta-analysis of score vectors and their covariance matrices. However, meta-analyzing covariance matrices is cumbersome, as they are 2-dimensional structures of various sizes and cannot be easily stored for a whole study in an array. The purpose of the **seqMeta** package is to facilitate meta-analysis of region-based score tests, such as SKAT, through convenient data structures for handling the components of the tests. There is some software available for similar analyses, for instance the MASS program (`http://dlin.web.unc.edu/software/mass/`). Our implementation is written in R (R Development Core Team, 2009) and supports continuous, binary and survival outcomes for unrelated individuals, continuous outcomes for related individuals, and performs individual variant tests, as well as burden tests, SKAT and SKAT-O.

Use of the package proceeds in two steps. The first is running the study level analyses with the function `prepScores`. This function need only be called once in each study for all genes under consideration. The output is a an R 'list' object where each element corresponds to a gene, and contains the scores and MAFs for that gene, as well as a matrix of the covariance between the scores at all pairs of SNPs within the gene. This list of output is labeled as a `prepScores` object, is saved as an `.Rdata` file, and passed to a central location. The second step combines the output from the study-level analyses, i.e. doing meta-analysis. The meta-analysis functions can create several

different tests including single variant, T1 count (Li and Leal, 2008), Madsen-Browning (Madsen and Browning, 2009), and of course SKAT (Wu *et al.*, 2011).

Meta-analysis functions can be run on a single study alone, and this will produce study level test results. These can be useful for quality control. But the meta-analysis across studies does require the full output from `prepScores`; it cannot be done with just study level test results.

The rest of this document is organized as follows. Section 2 details the functions used at the study level for continuous outcomes in unrelated individuals (Section 2.1), and related individuals (Section 2.2). Section 3 describes the functions used at the meta-analysis level for SKAT (Section 3.1), burden tests (Section 3.2) and single SNP tests (Section 3.4). Section 4 describes modifications for binary and survival outcomes. Section 6 describes how to perform conditional analyses. Section 7 gives some guidance for parallelization on large datasets, and finally Section 8 describes the mathematical details of the implementation. All sections contain example code which should be directly executable in R using the package **seqMeta** available at `http://cran.r-project.org/web/packages/seqMeta/`

2 Study level analysis

We are interested in testing the association of groups of SNPs with a particular outcome. A fundamental ingredient for region-based tests is a common definition across studies of which variants are in which regions. For convenience, we will refer to these regions of SNPs as genes, though in principal they can be any unit of aggregation. All functions in the package require a SNP Information file (argument `SNPInfo`), which links SNPs to genes. By default, the package looks within SNP Information file (a data frame) for the fields `Name` and `gene` for SNP and gene identification respectively. If other fields are desired to indicate SNP names or genes, these must be specified explicitly via the arguments `snpNames` and `aggregateBy`.

Using `prepScores`, users are strongly recommended to include all SNPs in the analysis, even if they are monomorphic in one study. This is for two reasons; firstly, monomorphic SNPs provide information about MAF across all studies; without providing the information we are unable to tell if a missing SNP data was monomorphic in a study, or simply failed to genotype adequately in that study. Second, even if some SNPs will be filtered out of a particular analysis (e.g., because they are intronic or common) constructing prepScores objects using all available SNPs allows greater flexibility in the meta-analysis.

2.1 Unrelated subjects

To create a `prepScores` object containing one study's contributions, use e.g.

```
prepScores(Z=Z, formula =nullModel,
          SNPInfo=SNPInfo, data=pheno)
```

where;

- `Z` is a NSNP rows $\times$ NSAMPLE columns matrix containing the additively coded (typically 0-1-2 for number of minor alleles) genotypes, with column names which match the SNP names used in the SNPInfo file. Rows do not have to be named, but must be in the same order as in the phenotype file (the `data` argument). The `Z` matrix can contain missing genotypes (coded as NA); missing genotypes are imputed to the mean value of the genotype in that study. Because there is no allele-checking functionality, all genotypes should be coded the same way for all studies before running.

- `formula` is a formula object for the null model of the form `outcome ~ covariates`. In the example below, we use the formula `y~sex+bmi` in the example below to analyze an outcome `y`, adjusted for covariates `sex` and `bmi`.
- `SNPInfo` is a matrix or data-frame with columns titled `Name` and `gene`. These columns specify the SNPs, and their associated genes, to be used in the analysis. One can use different columns to designate SNP names and genes with the arguments `Name` and `aggregateBy` respectively.
- `data` is a data frame containing the phenotype information. Phenotypes and genotypes must have the same number of rows and be ordered in the same way, because `prepScores` matches them simply on row order and not by any form of ID variable. The phenos data frame must contain no missing data in the columns used in the analysis.

For example, analysis at a single study, using the example data in the package, might proceed as below.

```
> rm(list=ls())
> library(seqMeta)
> ######### load example data:
> # contains SNPInfo, phenotyes (pheno1, pheno2)
> # genotypes (Z1,Z2), and pedigree information (kins) for pheno2
> data(seqMetaExample)
> ls()

[1] "SNPInfo" "Z1"      "Z2"      "kins"    "pheno1"  "pheno2"

> #Perform study-level analysis:
> c1 <- prepScores(Z1, y~bmi+sex, SNPInfo = SNPInfo, data = pheno1)
> ###save the output, which can be passed to a central location.
> study1.out.file <- tempfile()
> save(c1, file = study1.out.file)
```

In this section we described analysis for continuous outcomes. Binary and survival outcomes for unrelated individuals require only slight modification, and are described in Section 4.

2.2 Related subjects

This analysis proceeds in essentially the same was as for related subjects, but requires pair-wise kinship information in stored in a matrix. This is supplied to `prepScores` via the `kins` argument

```
> c2 <- prepScores(Z2, y~bmi+sex, SNPInfo = SNPInfo, kins = kins,
+                      data = pheno2)
> study2.out.file <- tempfile()
> save(c2, file = study2.out.file)
```

Family structure information is usually stored on disk in a 'makeped' linkage format. To convert this format into to the required kinship matrix, the `makekinship` function in the **kinship2** R! package is recommended; see these packages documentation for details (Therneau *et al.*, 2012). Note that family data is currently only supported for continuous outcomes.

3 Meta-analysis

The results of the one or more individual studies can be meta-analyzed using the functions `skatMeta`, `skatOMeta`, `burdenMeta`, or `singlesnpMeta` for SKAT, SKAT-O, burden tests and individual SNPs respectively. Regardless of the individual analysis (related or unrelated individuals with continuous, binary or survival outcomes) the meta-analysis procedure is the same. Each of these meta-analysis functions has a similar syntax, and contains the arguments

- `...` One or more `prepScores` objects to be meta-analyzed.
- `SNPInfo` The SNP Info file, listing genes and SNPs to be included in the meta-analysis. This is in the same format as the SNPInfo file in `prepScores`, and should use the same naming convention for SNPs and genes.
- `snpNames` The field of SNPInfo where the SNP identifiers are found. Default is `Name`
- `aggregateBy` The field of SNPInfo on which the skat results were aggregated. Default is `gene`. For single snps which are intended only for single variant analyses, it is recommended that they have a unique identifier in this field.
- `mafRange` A range of minor allele frequencies to be included in the analysis. These are based on the pooled MAFs over all studies. By default this is c(0,0.5), which includes all SNPs.
- `verbose` logical, whether or not to print progress bars. Defaults to FALSE.

Each of the functions permits additional arguments, which we detail in their corresponding sections.

3.1 Meta-analyzing SKAT

The SKAT test is a weighted sum of individual score statistics

$$Q = \sum_j w_j^2 U_j^2$$

where w_j is a weight and U_j is the score statistic for for the association between phenotype and variant j. Details are given in Section 8. In the function `skatMeta!` these weights are specified via the `wts` argument, which gives either a function to be applied to the minor allele frequencies or a column of the SNP Information file which can be coerced to a numeric vector.

The additional argument `method` allows the user to specify the method of p-values calculation. The default is `method`'saddlepoint'=, which appears to work well in practice. See the documentation of `pchisqsum` and Section 8 for more details.

For example, below is a meta-analysis of the SKAT test. Here we have use the default testing weights `wts` dbeta(maf,c(1,25))=, often referred to as the 'Wu' weights, which are also the default in the **SKAT** package (Lee *et al.*, 2013).

```
> load(study1.out.file)
> load(study2.out.file)
> meta.results <- skatMeta(c1, c2, SNPInfo = SNPInfo)
> head(meta.results)
```

```
     gene           p     Qmeta      cmaf nmiss nsnps
1   gene1 0.858611740  97982.36 0.2050000   600    15
2  gene10 0.385835255 236430.85 0.3112500  1200    21
3 gene100 0.310105859 201665.04 0.2537500   600    16
4  gene11 0.348999529 298153.58 0.3766667  1800    26
5  gene12 0.005838851 364830.51 0.2175000     0    15
6  gene13 0.716572239 193644.56 0.3095833   600    22
```

Here, `skatMeta` returns the following information

- `gene` The gene name, or other unit of aggregation.
- `p` The p-value from the SKAT test
- `Qmeta` The SKAT Q statistic, defined as $\sum_j w_j U_j^2$, where w_j is the weight given to SNP j, and U_j^2 is associated score statistic.
- `cmaf` The cumulative minor allele frequency. That is $\sum_j \text{MAF}_j$, where MAF_j is the minor allele frequency of variant j, and the sum is over all variants in the gene.
- `nmiss` The number of missing SNPs. For a gene with a single SNP this is the number of individuals which do not contribute to the analysis, due to studies that did not report results for that SNP. For a gene with multiple SNPs, `nmiss` is summed over the gene.
- `nsnps` The number of SNPs in the gene.

It will often be useful to examine results within a study before meta analysis. This can be done by 'meta-analyzing' a single study's results:

```
> study1.results <- skatMeta(c1, SNPInfo = SNPInfo)
```

3.2 Meta-analyzing burden tests

Another commonly used family of tests are the 'burden' tests which regress the phenotype on a weighted sum of genotypes, within each gene. The score test for a weighted sum of genotypes has the form

$$T = \sum_j w_j U_j,$$

where w_j is a weight for SNP j and U_j is the score for SNP j. For instance if MAF_j is the minor allele frequency for SNP j, the Madsen-Browning (Madsen and Browning, 2009) test uses $w_j = (MAF_j(1 - MAF_j))^{-1}$, while the and the T1 count test (Li and Leal, 2008) uses $w_j = \mathbf{1}(MAF_j < 0.01)$.

These types of tests can be computed with the `burdenMeta` function. This has the argument `wts`, which like `skatMeta`, gives either a function to be applied to the minor allele frequencies or a column of the SNP Information file which can be coerced to a numeric vector.

To perform the T1 test one could use the command

```
> meta.t1.results <- burdenMeta(c1, c2, wts = function(maf){maf < 0.01},
+                               SNPInfo = SNPInfo)
```

Equivalently, we could use constant weights and limit the analysis to those SNPs with $MAF < 0.01$:

```
> meta.t1.results <- burdenMeta(c1, c2, wts =1,
+                               mafRange = c(0,0.01), SNPInfo = SNPInfo)
```

Likewise, the Madsen-Browning test could be performed with the command

```
> meta.mb.results <- burdenMeta(c1, c2, wts = function(maf){1/(maf*(1-maf))},
+                               SNPInfo = SNPInfo)
```

Regardless of the weights used, these tests give output of the form

```
> format(head(meta.t1.results),digits=2)
```

	gene	p	beta	se	cmafTotal	cmafUsed	nsnpsTotal	nsnpsUsed	nmiss
1	gene1	0.888	0.029	0.20	0.20	0.0096	15	1	0
2	gene10	0.064	-0.376	0.20	0.31	0.0096	21	1	0
3	gene100	NA	NA	Inf	0.25	0.0000	16	0	0
4	gene11	NA	NA	Inf	0.38	0.0000	26	0	0
5	gene12	NA	NA	Inf	0.22	0.0000	15	0	0
6	gene13	0.214	0.211	0.17	0.31	0.0163	22	2	600

This output is similar to the gene-level summaries reported by `skatMeta`, but also includes additional information. Though we employ a score test, which does not explicitly estimate genetic effects, we do report estimated effects `beta` and their standard error `se`. These can be thought of as one-step approximations to standard maximum likelihood estimates, which may differ slightly when effect sizes are very large (Voorman *et al.*, 2012). The additional columns `cmafTotal` and `cmafUsed`, `nsnpsTotal` and `nsnpsUsed` distinguishing the total cumulative minor allele frequency and number of SNPs, from those that are used in the test. Also, note the genes for which no p-value is returned. In these genes, no SNPs met the inclusion criteria, and thus burden is zero for all individuals, and the test is undefined.

3.3 Meta-analyzing SKAT-O

Recently, the 'optimal' SKAT (SKAT-O, Lee *et al.*, 2012) was proposed to test weighted averages of SKAT and burden tests of the form

$$Q_o(\rho) = (1-\rho)\left(\sum_{j=1}^{p} w_j^{skat} U_j^2\right) + \rho\left(\sum_{j=1}^{p} w_j^{burden} U_j\right)^2$$

When $\rho = 0$ this gives the SKAT test $Q_o = \sum_{j=1}^{p} w_j U_j^2$, and when $\rho = 1$ it gives the burden test $Q_o = \left(\sum_{j=1}^{p} w_j U_j\right)^2$. Note that unlike the version of SKAT-O implemented in the **SKAT** package, we do not assume that the weights used in the burden test are the same as those used in SKAT.

When ρ is fixed, computation of a p-value can be obtained as with SKAT, using that the test statistic $Q_o(\rho)$ is a quadratic form of normally distributed score vector U. Lee *et al.* (2012) propose choosing the value of ρ which minimizes the p-value of the test over a set values of ρ in the interval $[0, 1]$. That is, they find the 'optimal' linear combination of SKAT and burden tests. However, when ρ is chosen to minimize the p-value, the actual p-value reported must reflect the flexibility afforded by this choice. For example, in the simple case where we allow ρ to be either 0 or 1, we would

perform both a burden test and a SKAT test and record the minimum of the p-values. We might correct for multiple testing using a Šidák or Bonferroni correction. However, this is not optimal, since burden tests and SKAT are correlated with each other, especially when the number of variants in a region is small. As shown by Lee *et al.* (2012) it is possible to calculate the distribution of the minimum p-value over any sequence of ρ's, accounting for this correlation between tests. The mathematics are given in the Appendix of that paper, which require individual level information. In Section 8 we give an alternate derivation of the calculations which can be performed at the meta-analysis level.

The syntax of `skatOMeta` is similar to that of `skatMeta` and `burdenMeta`, with the only difference being that weights for SKAT and the burden test must be distinguished, and values of ρ must be specified. These are given in the arguments

- `skat.wts` and `burden.wts` Either a function, or a character string specifying a column in the SNPInfo file, which gives the weights to be used in SKAT and the burden test, respectively. The default is to use the 'beta' weights in SKAT and T1 weights for the burden test.
- `rho` The values of ρ to be used in SKAT-O. The default is `c(0,1)`, which computes SKAT and the burden test, and reports the minimum p-value adjusted for multiple testing.

Here, we illustrate using SKAT-O using'Wu' weights in both the SKAT and the burden test, and choose a sequence of 11 ρ's in $[0, 1]$

```
> meta.skato.results <- skatOMeta(c1, c2, rho=seq(0,1,length=11),
+      burden.wts = function(maf){dbeta(maf,1,25)}, SNPInfo = SNPInfo, method = "int")
> format(head(meta.skato.results),digits=2)
```

```
     gene      p   pmin rho cmaf nmiss nsnps errflag
1   gene1 0.7878 0.5802 1.0 0.20   600    15       0
2  gene10 0.5891 0.3859 0.0 0.31  1200    21       0
3 gene100 0.4044 0.2528 0.5 0.25   600    16       0
4  gene11 0.5416 0.3490 0.0 0.38  1800    26       0
5  gene12 0.0098 0.0058 0.0 0.22     0    15       0
6  gene13 0.9043 0.7166 0.0 0.31   600    22       0
```

The format of the results is similar to the other region-based tests, with the addition of the fields `pmin`, which specifies the minimum p-value among the tests considered, and `rho`, which gives the value of ρ which resulted in `pmin`. Note that p should always be at least as large as `pmin`, since SKAT-O corrects for the multiple tests considered. If there is a single SNP in a gene, or either SKAT or the burden test is undefined, `pmin` will be identical to `p`.

The additional column `errflag` indicates a possibly inaccurate p-values. A value of 0 indicates no error, values larger than 0 indicate potentially inaccurate p-values. Genes where `errflag` is not zero can be re-run with more accurate calculation method, such as `saddlepoint`, which is slower, but more accurate, than `integration`.

```
> table(meta.skato.results$rho)
```

```
  0 0.1 0.2 0.4 0.5 0.6   1
 56   2   2   1   2   1  36
```

The table above displays the value of ρ which gave the smallest p-value among the 11 combinations of SKAT and the burden test. Note that the smallest p-value is typically given by either SKAT or the burden test ($\rho = 0$ or $\rho = 1$), rather than a proper combination of them. For this reason, the default is simply `rhoc(0,1)=` which performs only SKAT and the burden test, and is typically much faster.

Though `skatOMeta` in some sense supersedes `skatMeta` and `burdenMeta`, we recommended running SKAT and burden tests on their own for quality control. For instance, here we calculate SKAT and the burden test that are used SKAT-O, and plot the SKAT-O p-value, to the minimum p-value from SKAT and the burden test.

```
> wu.burden <- burdenMeta(c1, c2, wts = function(maf){dbeta(maf,1,25)},
+                         SNPInfo=SNPInfo)
> pseq <- seq(0,1,length=100)
> plot(y=meta.skato.results$p, x=pmin(wu.burden$p,meta.results$p),
+      xlab ="Minimum of SKAT and Burden", ylab = "SKAT-O")
> abline(0,1,lty=2)
> lines(x=pseq,y=1-(1-pseq)^2,col=2,lty=2,lwd=2)
> legend("bottomright", lwd=2,lty=2,col=2,legend="Sidak correction")
```

We see that the SKAT-O p-values are always larger than the minimum of the SKAT and burden test p-values (the black line), but smaller than the Šidák correction for two independent tests (the red line). In addition to the `errflag` column, plots like these provide a rough check of the SKAT-O p-value accuracy, which can occasionally be hard to achieve for small p-values.

We also note here that the p-values for SKAT-O are close to Šidák correction for the minimum of SKAT and burden tests. As the number of variants increases, the SKAT and burden tests become independent. Thus, in the relatively large genes in this example, little efficiency is gained using the SKAT-O correction relative to the Šidák correction.

3.4 Meta-analyzing single SNPs

While `skatMeta` and `burdenMeta` perform region-based tests, the function `singlesnpMeta` can be used to perform score tests for single SNP associations. The only additional option available is whether or not to report study-specific effects and standard errors (argument `studyBetas`)

```
> meta.ss.results <- singlesnpMeta(c1, c2, SNPInfo = SNPInfo,
+                                  studyBetas = TRUE)
> format(head(meta.ss.results),digits=2)
```

```
   gene    Name    p    maf nmiss ntotal   beta   se beta.c1 se.c1 beta.c2
1 gene1 1000001 0.83 0.0129     0   1200 -0.038 0.18  -0.066  0.25  -0.010
2 gene1 1000002 0.41 0.0146     0   1200 -0.137 0.17  -0.268  0.23   0.013
3 gene1 1000003 0.89 0.0096     0   1200  0.029 0.20  -0.284  0.28   0.354
4 gene1 1000004 0.84 0.0158     0   1200 -0.032 0.16  -0.130  0.24   0.043
5 gene1 1000005 0.50 0.0121     0   1200  0.124 0.18  -0.112  0.28   0.292
6 gene1 1000006 0.47 0.0142     0   1200  0.122 0.17   0.111  0.23   0.134
  se.c2
1  0.25
2  0.24
3  0.29
```

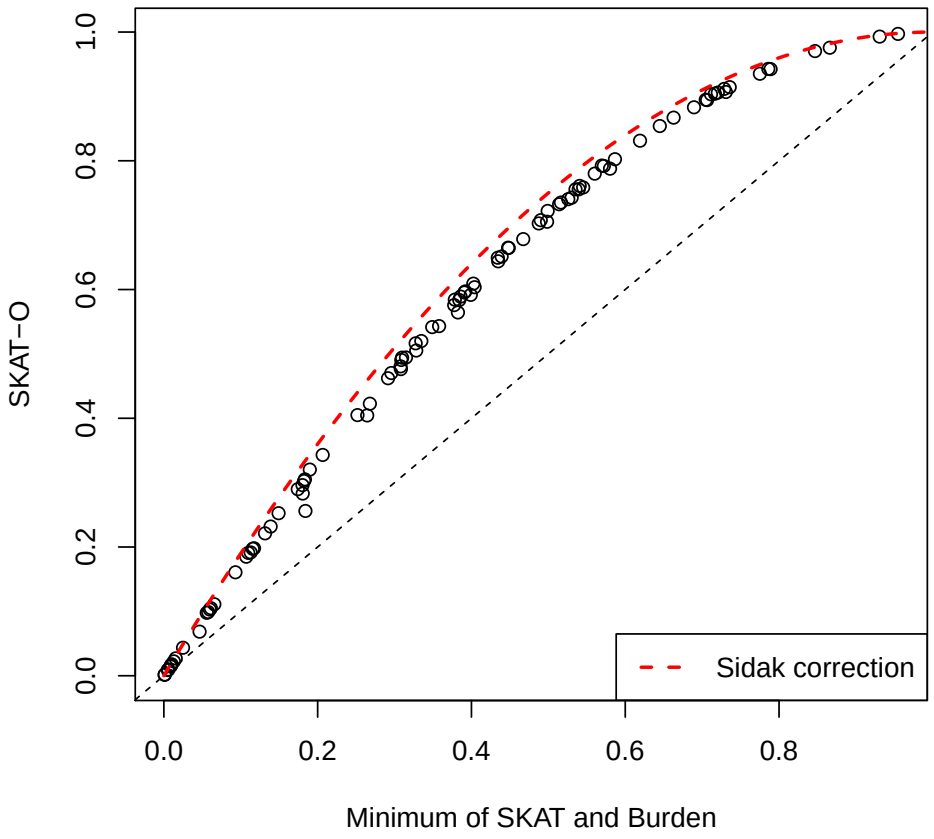

Figure 1: Comparison between SKAT-O p-values, and the minimum of SKAT and burden test p-values. The Šidák correction for two independent tests is $1-(1-p)^2$.

```
4  0.21
5  0.24
6  0.25
```

The output is similar to that of `burdenMeta` and `skatMeta`, but now includes effect estimates for single studies, suffixed by the name given to the corresponding `prepScores` object.

As with the burden test, the coefficients and standard errors can be thought of as one-step approximations to maximum likelihood estimates.

4 Binary and survival outcomes

The **seqMeta** package can also handle binary data and survival outcomes for unrelated populations.

For binary outcomes, this is specified with the `family` argument in the function `prepScores`, which accepts a family object as in the function `glm`.

```
> c1.bin <- prepScores(Z2, ybin~bmi+sex, family = binomial(),
+                      SNPInfo = SNPInfo, data = pheno1)
```

For survival outcomes this fitting process is somewhat more involved, and handled with a separate function `prepCox`. The `formula` syntax is the same as that of `coxph`:

```
> c1.cox <- prepCox(Z=Z1, Surv(time, status) ~ bmi + strata(sex),
+                                  SNPInfo = SNPInfo, data =pheno1)
```

Regardless of model choice, the scores are meta-analyzed in exactly the same way; no modification is necessary of the `skatMeta` function:

```
> study1.bin.results <- skatMeta(c1.bin, SNPInfo = SNPInfo,
+                           aggregateBy = "gene")
> study1.cox.results <- skatMeta(c1.cox, SNPInfo = SNPInfo,
+                           aggregateBy = "gene")
```

5 Filtering

Sometimes it will be desirable to remove SNPs from the analysis, based on minor allele cutoffs or other criteria. Some 'static' criteria, such as SNP annotation, can be handled by subsetting the SNP Info file given to `skatMeta`, `burdenMeta` or `singlesnpMeta`.

A common filtering criteria is minor allele frequency, which in general depend on the individuals contributing to an analysis. For this reason,`skatMeta`, `burdenMeta` or `singlesnpMeta` have an option `mafRange` which includes SNPs within a specified range of minor allele frequencies. For instance, if one only wishes to use only variants with MAF between 0 and 0.05, one would perform the previous analysis by

```
> meta.results <- skatMeta(c1, c2, SNPInfo = SNPInfo,
+                           aggregateBy = "gene", mafRange = c(0,0.05))
> head(meta.results)
```

```
     gene           p     Qmeta      cmaf nmiss nsnps
1   gene1 0.858611740  97982.36 0.2050000   600    15
2  gene10 0.385835255 236430.85 0.3112500  1200    21
3 gene100 0.310105859 201665.04 0.2537500   600    16
4  gene11 0.348999529 298153.58 0.3766667  1800    26
5  gene12 0.005838851 364830.51 0.2175000     0    15
6  gene13 0.716572239 193644.56 0.3095833   600    22
```

Note that the number of SNPs now included is lower than the previous analysis.

6 Conditional analyses

In many cases, it will be desirable to adjust for SNPs in a conditional analysis. This can be done in the package with the function `prepCondScores`. Note that these adjustments are required by the individual studies, and are not performed at the meta-analysis. The syntax and output are identical to `prepScores` , but requires the additional argument `adjustments`. The `adjustments` should be in the same format as the SNPInfo file (i.e. containing a 'gene' and 'Name' column specifying SNP names, or designated alternative columns), and specifies which genes to adjust for which SNPs. For instance, in the below example, we condition 'gene1' analyses on the SNPs labeled 1000001, 1000002, and 1000003, 'gene2' on SNP 1000020, and 'gene13' on SNP 1000100.

```
> adjustments <- SNPInfo[c(1:3, 20,100), ]
> adjustments
```

```
       Name   gene
1   1000001  gene1
2   1000002  gene1
3   1000003  gene1
20  1000020 gene10
100 1000100 gene13
```

```
> ####run on each study:
> c1.adj <- prepCondScores(Z=Z1, y~sex+bmi, SNPInfo = SNPInfo,
+         adjustments=adjustments, data =pheno1)
> c2.adj <- prepCondScores(Z=Z2, y~sex+bmi, SNPInfo = SNPInfo,
+         adjustments=adjustments, kins=kins, data=pheno2)
> SNPInfo.sub <- subset(SNPInfo, (SNPInfo$gene %in% adjustments$gene) &
+         !(SNPInfo$Name %in% adjustments$Name) )
> #skat
> out.skat <- skatMeta(c1.adj,c2.adj, SNPInfo = SNPInfo.sub)
> head(out.skat)
```

```
    gene         p     Qmeta      cmaf nmiss nsnps
1  gene1 0.7638005  86158.87 0.1679167   600    12
2 gene10 0.3234171 235903.90 0.2933333  1200    20
3 gene13 0.7967129 167332.89 0.2920833   600    21
```

The output of `prepCondScores` have class `prepScores`, and contain *only* the genes listed in the `adjustments` argument. Any of the meta-analysis functions can be applied to these objects as before. In the above example, we performed SKAT, using the subset of the full SNPInfo file relevant to the conditional analyses.

7 Parallelization

In most cases, results for individual studies can be computed for whole-exome and exome-chip on several thousand subjects in a few minutes, without use of specialized data structures or parallel processing. However, if either memory limitations or processing speed make this cumbersome, we provide a generic concatenation function `c(...)` to combine multiple `prepScores` objects. This can be employed to break up computation over, e.g. chromosomes. For instance, we can separate computation for the first study over the first and second set of 50 genes. We perform this example in sequence, but it would be a simple matter to run each subset of genes in parallel.

```
> ##subset SNPInfo file to first 50 genes, and second 50 genes:
> SNPInfo1 <- subset(SNPInfo, SNPInfo$gene %in% unique(SNPInfo$gene)[1:50] )
> SNPInfo2 <- subset(SNPInfo, !(SNPInfo$gene %in% unique(SNPInfo$gene)[1:50]) )
> ##subset corresponding genotype files:
> Z1.1 <- subset(Z1, select = colnames(Z1) %in% SNPInfo1$Name)
> Z1.2 <- subset(Z1, select = colnames(Z1) %in% SNPInfo2$Name)
> ##run prepScores separately on each chunk:
> c1.1 <- prepScores(Z1.1, y~bmi+sex, SNPInfo = SNPInfo1, data = pheno1)
> c1.2 <- prepScores(Z1.2, y~bmi+sex, SNPInfo = SNPInfo2, data = pheno1)
> ##combine results:
> c1 <- c(c1.1, c1.2)
> class(c1)
```

```
[1] "seqMeta"
```

We do not provide a method for computing the meta-analysis in parallel. Individual `prepScores` objects are usually under 20 megabytes, and can be processed sufficiently quickly that parallelization is typically not warranted.

8 Method details

In this section we describe the method used in the meta-analysis. The theory for meta-analyzing score tests can be found in e.g. Lin and Zeng (2010).

We first focus on continuous and binary outcomes; modifications for family data and survival outcomes are described subsequently. For study k, denote $y^{(k)}$ as the n-vector of outcomes, $\hat{y}^{(k)}$ as the corresponding vector of fitted values under the null-model (i.e. without genotype), and $G^{(k)}$ as the $n \times p$ matrix of genotypes in one gene, and $X^{(k)}$ as the $n \times q$ matrix of adjustment variables.

8.1 Study level functions

For each gene under consideration, `prepScores` computes the vector of scores $U^{(k)}$ and their corresponding variances $V^{(k)}$. For unrelated individuals these are given by

$$U^{(k)} = G^{(k)T}(y^{(k)} - \hat{y}^{(k)})/\sigma_k^2 \in \mathbb{R}^p$$
$$V^{(k)} = (W^{(k)}G^{(k)})^T(I - H^{(k)})W^{(k)}G^{(k)}/\sigma_k^2 \in \mathbb{R}^{p\times p},$$

where σ_k^2 is the the residual variance for continuous data and is 1 for binary data,

$$W^{(k)} = \begin{cases} \text{diag}(\sqrt{\hat{y}^{(k)}(1-\hat{y}^{(k)})}) & \text{for binary data} \\ I_n & \text{for continuous data} \end{cases},$$

and

$$H^{(k)} = X^{(k)}W^{(k)}((W^{(k)}X^{(k)})^T W^{(k)}X^{(k)})^{-1}(W^{(k)}X^{(k)})^T,$$

is a projection matrix. These can easily be obtained from the output of the `glm` function.

The `prepScores` function computes these for each gene under consideration, and concatenates them in a list. If genotype is missing, it is imputed to the study-specific minor allele frequency. This is done one gene at a time as scores and information are computed, rather than modifying the genotype argument `Z`, which prevents R! from copying potentially large genotype matrices. In addition, the sample size and minor allele count are also stored.

8.2 Meta-analysis

Each function `skatMeta`, `burdenMeta`, and `singlesnpMeta` first begins by meta-analyzing the scores and their variances across all studies, for each unique gene name in the `SNPInfo` file. This follows the simple formulas

$$U = \sum_k U^{(k)}, \quad V = \sum_k V^{(k)}.$$

Under the null hypothesis of no genetic effect, we have that

$$V^{-1/2}U \to_d N_p(0, I_p)$$

or that U is approximately $N_p(0, V)$

Each of the tests statistics computed computed (SKAT, burden, and single snp) are functions of the vector U.

skatMeta For a vector of weights $w = (w_1, \ldots, w_p)^T$ denote $R = \text{diag}(\sqrt{w_1}, \ldots, \sqrt{w_p})$ The SKAT statistic Q is given by

$$Q = \sum_{j=1}^{p} w_j U_j^2 = \|RU\|_2^2,$$

where $S_j = \sum_k {g_j^{(k)}}^T (y^{(k)} - \hat{y}^{(k)})/\sigma_k^2$ and $g_j^{(k)}$, is the genotype for SNP j in study k. Asymptotically

$$Q \sim \sum \lambda_i \chi_1^2,$$

where the λ_i are the eigenvalues of RVR. The distribution function of a sum of eigenvalues can be approximated using the function `pchisqsum` in the **survey** package (Lumley, 2004). By default, we use `method= 'saddlepoint'` for the saddle point approximation, which is implemented in pure R. A slightly faster option is the Davies method, which inverts the characteristic function, and requires the **CompQuadForm** package (Duchesne and de Micheaux, 2010).

burdenMeta and singlesnpMeta The computation for burden tests is much more straightforward. For a weight vector $w = (w_1, \ldots, w_p)^T$, the vector of burdens is given by $G^{(k)}w$, and the score for its association by $w^T G^{(k)T}(y^{(k)} - \hat{y}^{(k)})/\sigma_k^2$. Summing the scores across studies, we get the test statistic

$$T = \sum_k w^T G^{(k)T}(y^{(k)} - \hat{y}^{(k)})/\sigma_k^2 = w^T U$$

which is approximately $N_1(0, w^T V w)$. The function `singlesnpMeta` is a special case of this, where $w_j = 1$ for the variant of interest, and is otherwise zero.

In order to approximate the coefficients and standard errors, we note that the first step of the Fisher scoring algorithm estimates the coefficients and their standard errors from the first and second derivatives of the likelihood, which are given by the score and its variance. We plug in the meta-analyzed score and variance to obtain the one-step approximations

$$\hat{\beta} = \frac{w^T U}{w^T V w}, \text{ and } \quad \hat{se}(\hat{\beta}) = \frac{1}{\sqrt{w^T V w}}.$$

If we apply the Wald test to these coefficients, the resulting χ^2 test statistic is exactly the χ^2 test statistic for the score test.

SKAT-O Denote the SKAT-O statistics

$$Q_o(\rho) = (1-\rho)\|UR\|_2^2 + \rho(w^T U).$$

Our goal is to find the value of ρ which gives the smallest p-value among a sequence of ρ's (given in `rho`), and correct for the flexibility afforded by this choice.

Denote the sequence of ρ's $0 \leq \rho_1 < \cdots < \rho_q \leq 1$ at which we calculate p-values for $Q_o(\rho)$. In the simplest case, we can choose $\rho_1 = 0$ and $\rho_2 = 1$ to perform the SKAT and burden tests. We then calculate the p-values for each of these tests. For fixed ρ , $Q_o(\rho_i)$ follows a mixture of χ_1^2 variables, where the mixing parameters are given by the eigenvalues of $W^{1/2} V W^{1/2}$, where $W = ((1-\rho) \cdot RR + \rho \cdot ww^T)$. To calculate $W^{1/2}$ we use the eigenvalue decomposition of the matrix $((1-\rho) \cdot RR + \rho \cdot ww^T)$.

Now, let p_i be the p-value for $Q_o(\rho_i)$. We then select the minimum p-value $p_{min} = \min\{p_1, \ldots, p_q\}$. To find the actual p-value we wish to report, we must compute the probability that of observing a p_{min} smaller than what was observed under the null hypothesis. Unfortunately, it is difficult to directly write down a distribution of p_{min} under the null hypothesis using well-known distributions. However, the distribution of p_{min} is numerically tractable if we break the distribution of $Q_o(\cdot)$ into the conditional distribution of the SKAT statistic given the value of the burden test, and then average over values of the burden test. In order to do this, first note that if T_i is the $1 - p_{min}$ quantifier of the distribution of $Q_o(\rho_i)$, we know that the test using ρ_i will give a p-value less than p_{min} when $(1-\rho_i)\|UR\|_2^2 + \rho_i (w^T U)^2 > T_i$, i.e. when

$$\|UR\|_2^2 > \frac{T_i - \rho_i (w^T U)^2}{1 - \rho_i}.$$

Thus, given the value of the burden test, we observe a smaller p-value than p_{min} when

$$\|UR\|_2^2 > \min_{i=1\ldots q} \left\{ \frac{T_i - \rho_i (w^T U)^2}{1 - \rho_i} \right\}.$$

So, it suffices to find the conditional distribution of UR given $w^T U$, calculate $Pr(\|UR\|_2^2 > \min_{i=1\ldots q} \left\{ \frac{T_i - \rho_i (w^T U)^2}{1-\rho_i} \right\})$, and integrate it over the distribution of $w^T U$. Since UR and $w^T U$ are linear combinations of a Normal vector, we can write down their joint distribution. Standard algebra gives that

$$UR \mid \{w^T U = a\} \sim N_p \left(\frac{RVw}{w^T V w} a, \quad R^T V R - \frac{RVww^T VR}{w^T V w} \right).$$

We can again determine the distribution of $\|UR\|_2^2$ using normal quadratic forms. These differ slightly from the calculations used in SKAT, due to the presence of the mean vector $\frac{RVw}{w^T V w} a$.

Instead, the distribution of $\|UR\|_2^2 \mid \{w^T U = a\}$ is the sum of non-central χ^2 distributions, whose distribution functions are available in any of the functions in the **CompQuadForm** package (Duchesne and de Micheaux, 2010). For speed, we recommend `method=`integration'`, which first attempts the Davies method, and in the case where this gives an error, it uses the `farebrother` method. Using `method=`saddlepoint'` is slower, but has higher relative accuracy.

Thus, if we denote $\phi(\cdot)$ as the marginal distribution of $w^T U$, which is $N(0, w^T V w)$, we can find the true significance of p_{min} using the following algorithm

1. For each ρ_i, calculate p_i based on the quadratic forms $Q_o(\rho_i)$.
2. Calculate $p_{min} = \min\{p_1, \ldots, p_q\}$.
3. Calculate T_i's, the $(1-p_{min})$ quantiles of the $Q_o(\rho_i)$. Denoting F_{Q_i} as the distribution function of $Q_o(\rho_i)$ found in Step 1, we do this by solving $F_{Q_i}(x) = p_{min}$ for x using `uniroot`. Potential inaccuracies here result in `errflag=2`.
4. Form the conditional distribution $UR \mid \{w^T U = a\}$, and the conditional probability
$$f_S(a) = Pr\left(\|UR\|_2^2 > \min_{i=1\ldots q}\left\{\frac{T_i - \rho_i (w^T U)^2}{1-\rho_i}\right\} \mid w^T U = a\right).$$
5. Numerically integrate
$$p_{actual} = \int_{-\infty}^{\infty} f_S(a)\phi(a)da$$
and report p_{actual}.

8.3 Family data

This framework extends easily to the case of family data. A more detailed description of this method is given in Chen *et al.* (2013). In the pooled-data analysis, the related individuals would have a different form of the likelihood, corresponding to a mixed-model. But, since likelihoods are on the same scale, the scores from related and un-related data can be combined as usual. We can think of all studies using a mixed-model, but where un-related individuals have no random effects.

Here, instead of outcomes y having variance $\sigma_y^2 I_n$, they have variance $\sigma_y^2 \Omega = \sigma_y^2(\theta_1 \Theta + \theta_2 I_n)$, where Θ is twice the kinship matrix, and $\theta_1 + \theta_2 = 1$. In unrelated individuals $\Theta = I_n$. The scores for this model are
$$U^{(k)} = G^{(k)^T} \Omega^{-1}(y^{(k)} - \hat{y}^{(k)}).$$
If the matrix of covariates is X, the variance of the score is
$$V^{(k)} = G^{(k)T}\left(\Omega^{-1} - \Omega^{-1}X(X^T\Omega^{-1}X)^{-1}X^T\Omega^{-1}\right)G^{(k)}.$$
We calculate these using the `lmekin` function in the **coxme** package (Therneau, 2012). These can be combined as usual with scores and variances from other studies.

Unfortunately, the score test for generalized linear mixed models does not exist in closed form, and no available implementation of generalized linear mixed models allows large used-defined variance components, such as a kinship matrix. We therefore leave the extension of binary traits for family data for future work.

8.4 Survival outcomes

We apply a similar procedure to obtain test statistics for Cox proportional hazards regression. In this case, the score test is anti-conservative, so we instead use a signed likelihood ratio statistic Lumley and Scott (2013). Specifically, let S_j^2 is the likelihood ratio test statistic for the including of the single variant j in the null model, β_j be the corresponding maximum likelihood estimate, and V be the variance of the usual score test. we use

$$U_j = \text{sign}(\beta_j) S_j / \sqrt{V_{jj}},$$

which is asymptotically equivalent to the score test, but enjoys superior finite sample performance.

In order to implement this procedure efficiently, we use a modified version of the`coxph.fit` function in the **survival** package (Therneau, 1999).

9 Summary

In this paper, we have described the basic usage and syntax of the **seqMeta** R package. This includes functions for computing study level results with the functions `prepScores` for continuous and binary traits and `prepCox` for survival outcomes. We also describe the functions that pool study level results for meta-analysis, including `skatMeta`, `burdenMeta`, `skatOMeta` and `singlesnpMeta`.

References

Chen H, Meigs J, Dupuis J (2013). "Sequence Kernel Association Test for Quantitative Traits in Family Samples." *Genetic Epidemiology*, **37**(2), 196–204.

Duchesne P, de Micheaux PL (2010). "Computing the distribution of quadratic forms: Further comparisons between the Liu-Tang-Zhang approximation and exact methods." *Computational Statistics and Data Analysis*, **54**, 858–862.

Lee S (2013). *MetaSKAT: Meta analysis for SNP-set (Sequence) Kernel Association Test.* R package version 0.27, URL `http://CRAN.R-project.org/package=MetaSKAT`.

Lee S, Miropolsky L, Wu M (2013). *SKAT: SNP-set (Sequence) Kernel Association Test.* R package version 0.82, URL `http://CRAN.R-project.org/package=SKAT`.

Lee S, Wu MC, Lin X (2012). "Optimal tests for rare variant effects in sequencing association studies." *Biostatistics*, **13**(4), 762–775.

Li B, Leal SM (2008). "Methods for detecting associations with rare variants for common diseases: application to analysis of sequence data." *The American Journal of Human Genetics*, **83**(3), 311–321.

Lin D, Zeng D (2010). "On the relative efficiency of using summary statistics versus individual-level data in meta-analysis." *Biometrika*, **97**(2), 321–332.

Lumley T (2004). "Analysis of complex survey samples." *Journal of Statistical Software*, **9**(1), 1–19.

Lumley T, Brody J, Dupuis J, Cupples A (2012). "Meta-analysis of a rare-variant association test." *Technical report*, University of Auckland. URL `http://stattech.wordpress.fos.auckland.ac.nz/files/2012/11/skat-meta-paper.pdf`.

Lumley T, Scott A (2013). "Partial likelihood ratio tests for the Cox model under complex sampling." *Statistics in Medicine*, **32**(1), 110–123.

Madsen BE, Browning SR (2009). "A groupwise association test for rare mutations using a weighted sum statistic." *PLoS genetics*, **5**(2), e1000384.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL `http://www.R-project.org`.

Therneau T (2012). *coxme: Mixed Effects Cox Models*. R package version 2.2-3, URL `http://CRAN.R-project.org/package=coxme`.

Therneau T, Atkinson E, Sinnwell J, Matsumoto M, Schaid D, McDonnell S (2012). *kinship2: Pedigree functions*. R package version 1.3.7, URL `http://CRAN.R-project.org/package=kinship2`.

Therneau TM (1999). "A package for survival analysis in S." *Technical report*, Mayo Foundation. URL `http://www.mayo.edu/hsr/people/therneau/survival.ps`.

Voorman A, Rice K, Lumley T (2012). "Fast computation for genome-wide association studies using boosted one-step statistics." *Bioinformatics*, **28**(14), 1818–1822.

Wu M, Lee S, Cai T, Li Y, Boehnke M, Lin X (2011). "Rare-variant association testing for sequencing data with the sequence kernel association test." *The American Journal of Human Genetics*, **89**(1), 82–93.