groupedSurv

Efficient Estimation of Grouped Survival Models Using the Exact Likelihood Function

2018-06-28

Introduction

- This document provides examples demonstrating how to use the `groupedSurv` package to estimate the baseline survival rate, covariate, and fixed effect parameters, and compute the efficient score statistic, as well as gene-level statistics for grouped survival models.
- Additional functions can analyze grouped time-to-event data accounting for family structure of related individuals (e.g., trios), and provide estimates of frailty variance. Note, however, that the current implementation of the frailty model is sensitive to departures from model assumptions, and should be considered experimental.
- The major algorithms in this package are written in `C++`, which is ported to `R` by `Rcpp`, to facilitate fast computation.

Model assumptions

Data without family structure [1]

- The input data is assumed to be organized as a `matrix` or `data.frame`, with rows corresponding samples, and columns corresponding to variables of interest (and covariates) for the sample.
- Support is also provided for GenABEL `gwaa.data` objects as alternative inputs.

Data with family structure [2] [Experimental]

- The input `matrix` or `data.frame` is assumed to be organized such that records for each family occur consecutively, and that records for offspring precede those for parents. The variance matrix for the random effects is assumed to be of the form `var`$*$`K`, where `K` is a matrix of kinship coefficients between family members.
- The following family groupings are permitted: (Individual), (Offspring, Offspring), (Offspring, Parent), (Offspring, Parent, Parent), and (Offspring, Offspring, Parent, Parent). Other family structures have not been implemented.

Included functions

Functions for data without family structure

```
thetaEst(Z=NULL, gtime, delta, method="BFGS")
betaEst(x, Z=NULL, alpha, theta=NULL, gtime, delta)
groupedSurv(x, Z=NULL, GenABEL.data=NULL, alpha, theta=NULL,
          gtime, delta, beta=0, nCores=1, reScore=FALSE)
geneStat(x, Z=NULL, GenABEL.data=NULL, alpha, theta=NULL,
          gtime, delta, beta=0, nCores=1,
          FUN=function(Uij, weight){sum((colSums(Uij)*weight)^2)}, geneSet)
```

Functions for data with family structure [Experimental]

```
alphaEstFam(gtime, delta)
betaEstFam(x, fam_group, fam_role, alpha, var, gtime, delta, lower, upper)
varEstFam(x, fam_group, fam_role, alpha, gtime, delta, lower, upper, beta = 0)
groupedSurvFam(x, fam_group, fam_role, alpha, var, gtime, delta, beta=0)
varLLFam(x, fam_group, fam_role, alpha, var, gtime, delta, beta=0)
```

Usage overview

Most users will interface with the package through the `thetaEst()` and `groupedsurv()` functions. The `thetaEst()` function provides maximum likelihood estimates (MLE) for the nuisance parameters, i.e., the baseline survival rate and the parameters for any covariates. The estimates are computed under the null hypothesis, i.e., that the variable of interest has no effect on the time to event. The `thetaEst()` function requires a vector of grouped survival times, `gtime` and a vector of event indicators, `delta`, as arguments. If the model includes covariates, these values, in the form of a `matrix` or `data.frame`, `Z`, are required as well. Optionally, users can specify the method of optimization (`method` = "BFGS" or "CG") which is passed to the `optim` function in `C++`.

The `groupedSurv()` function is the core of the `groupedSurv` package. As inputs, it requires a `matrix` or `data.frame` of variables to be tested for association with the outcome, `x`, a vector of grouped survival times and a vector of event indicators (`gtime` and `delta`), as well as estimates for the nuisance parameters (`alpha` and `theta`). Note that since `thetaEst()` estimates the nuisance parameters under the null hypothesis, these estimates can be reused to calculate the efficient score statistic for any number of variables of interest. If the model includes covariates, the `Z` `matrix` or `data.frame` is also required. In the context of GWAS, covariates and variables of interest can be

provided as part of a GenABEL object (`gwaa.data`), with the arguments `x` and `Z` instead used to specify to the columns of `gwaa.data` to be included in the analyses.

Usage overview

Users also have the option of specifying the number of cores available for parallel processing (`nCores`). The `beta` argument is 0 by default, computing the statistics under the null hypothesis, but users have the option of specifying a different value, e.g., in order to evaluate the statistic under an alternative hypothesis.

By default, the package returns a data frame containing the efficient score statistics, along with the (unadjusted) asymptotic p-values, FWER-adjusted p-values [3] and local FDRs [4, 5] for each of the variables of interest. By setting the optional argument `reScore=TRUE`, the `groupedsurv()` function will also return a matrix of the contribution of each sample to the efficient score statistic for each variable of interest.

A supporting function, `betaEst()`, takes a vector of values of a variable of interest, `x`, along with the same arguments as the `thetaEst()` function, and returns the MLE of the log hazard ratio, β.

Please refer to the individual function manuals for more detailed explanations of the arguments and returned values associated with each function.

A note about coding grouped survival times

Under the grouped failure time model, the continuous survival time, $time \in [0, \infty)$ is not observed. Instead, subjects are assessed for failure only at pre-specified time points, $gtime_1, gtime_2, \ldots, gtime_{r-1}$. These time points form the right end-points of r adjacent intervals, i.e., $[0, gtime_1), [gtime_1, gtime_2), \ldots, [gtime_{r-2}, gtime_{r-1})[gtime_{r-1}, \infty)$.

The contribution of each subject to the likelihood used in the efficient score is composed of the combination of the intervals they survived and, if applicable, that in which the event occurred. For example, a subject who survives the first two intervals (i.e.,has not failed at $gtime_1$ or $gtime_2$) but then has failed by the third observation should be coded as (`gtime`$= gtime_3$, `delta`$= 1$). Similarly, a subject who has not yet failed at the fourth observation time, but who is lost to follow-up before the fifth observation time would be coded as (`gtime`$= gtime_5$, `delta`$= 0$).

Note then that subjects who are censored at the first time point, (`gtime`$= gtime_1$, `delta`$= 0$), contribute no information to the likelihood. Note also that subjects who have not failed by the final observation time point should be considered censored at infinity, and coded (`gtime`$=$`Inf`, `delta`$= 0$).

Simulate grouped survival data

We first simulate continuous survival data:

```
set.seed(111)
n <- 1000
# effect size
beta <- 0.3
# covariate parameters
theta <- c(0.2, 0.2)
# variable of interest associated with outcome
MAF  <- 0.05
x <- matrix(rbinom(n, 2, MAF), ncol = 1)
# additional variables of interest
xMore <- matrix(rbinom(n*100, 2, MAF), ncol = 100)
xMore <- cbind(x, xMore)
# covariate data (centered at 0)
z1 <- rnorm(n)
z2 <- rbinom(n, 1, 0.5) - 0.5
Z <- matrix(cbind(z1, z2), ncol = 2)
# continuous survival time
lam0 <- 1
cmax <- 3
lami <- lam0 * exp(x * beta + Z %*% theta)
stime <- rexp(n, lami)
ctime <- runif(n, 0, cmax)
delta <- stime < ctime
otime <- pmin(stime, ctime)
```

Generate grouped survival data

Then generate grouped survival times from continuous survival data:

```
# grouped observation time points
ntps <- 5
r <- ntps + 1
# last observation time
maxbreakq <- 0.85
maxbreak <- qexp(maxbreakq, lam0)
# grouped survival times
breaks <- (1:ntps) * (maxbreak/ntps)
gtime <- findInterval(otime, breaks) + 1
delta[gtime == r] <- FALSE
dctime <- findInterval(ctime, breaks) + 1
delta[gtime == dctime] <- FALSE
delta <- as.numeric(delta)
gtime[which(gtime == r)] <- Inf
table(gtime, delta)
```

```
##       delta
## gtime   0   1
##   1   124 287
##   2    78 167
##   3    53  81
##   4    51  53
##   5    21  29
##   Inf  56   0
```

Example of thetaEst

Load groupedSurv (after installing its dependent packages):

```
library(groupedSurv)
```

Estimate θ (including baseline survival rate for each time interval and the covariate parameters) under the null hypothesis ($\beta = 0$):

```
thetaest <- thetaEst(Z, gtime, delta)
thetaest


## $alpha
## [1] 0.6754907 0.6723724 0.7182482 0.6553048 0.6465095
##
## $theta
## [1] 0.16915069 0.08762157
```

Examples of groupedSurv and betaEst

Compute the efficient score under the null hypothesis based on the estimated θ :

```
eff <- groupedSurv(x=xMore, Z=Z,  alpha=thetaest$alpha, theta=thetaest$theta,
                   gtime=gtime, delta=delta, beta=0, nCores=1)
head(eff)
```

```
##         stat       pvalue          FDR         FWER
## 1 37.4606427 9.327633e-10 4.318503e-08 9.420909e-08
## 2  0.1676391 6.822186e-01 3.931649e-01 1.000000e+00
## 3  0.1357759 7.125170e-01 3.931649e-01 1.000000e+00
## 4  0.5622443 4.533574e-01 3.572471e-01 1.000000e+00
## 5  0.4115984 5.211593e-01 3.770095e-01 1.000000e+00
## 6  0.6936298 4.049325e-01 3.471767e-01 1.000000e+00
```

One may wish to estimate β for variables of interest found to be associated with the outcome:

```
betaest <- betaEst(x=x, Z=Z, alpha=thetaest$alpha, theta=thetaest$theta,
                   gtime=gtime, delta=delta)
betaest
```

```
## [1] 0.7457616
```

Examples of alternative data sources

One can alternatively input SNP information directly from a GenABEL object.

```
library(GenABEL)
data(srdta)
GenABELdat <- srdta[1:n]
snpsToTest <- GenABELdat@gtdata@snpnames[1:200]
eff <- groupedSurv(x=snpsToTest, Z=Z, GenABEL.data=GenABELdat,
                   alpha=thetaest$alpha, theta=thetaest$theta,
                   gtime=gtime, delta=delta, beta=0, nCores=1)
```

Genotype data can be imported from binary PLINK [6] file using the `BEDMatrix` package.

```
library(BEDMatrix)
path <- system.file("extdata", "example.bed", package = "BEDMatrix")
m <- BEDMatrix(path)
# Extract genotypes for the specified variants
xPLINK <- m[, c("snp0_A", "snp1_C", "snp2_G")]
```

Or, genotype dosage data can also be directly extracted from a VCF file using the `VariantAnnotation` package [7].

```
system("wget ftp://share.sph.umich.edu/minimac3/DosageConvertor/DosageConvertor.v1.0.4.tar.
system("tar -xzvf DosageConvertor.v1.0.4.tar.gz")
library(VariantAnnotation)
exampleVcfFile <- "./DosageConvertor/test/TestDataImputedVCF.dose.vcf.gz"
myvcf <- readVcf(exampleVcfFile, "hg19")
dosedat <- assay(myvcf,"DS")
xVCF <- t(dosedat)
```

Analysis of gene sets

Specify SNP and gene information:

```
geneInfo <- data.frame(gene=c("BRCA1","BRCA2"), chr=c(17,13),
                       start=c(41196312, 32889611), end=c(41277500, 32973805),
                       stringsAsFactors=FALSE)
snpInfo <- data.frame(chr=c(17,17,13,13), pos=c(41211653,41213996,32890026,32890572),
                      rsid=c("rs8176273","rs8176265","rs9562605","rs1799943"),
                      stringsAsFactors=FALSE)
```

Use `snplist` package to create gene sets:

```
library(snplist)

## Loading required package:  RSQLite

setGeneTable(geneInfo)
setSNPTable(snpInfo)
geneset <- makeGeneSet()
```

```
geneset

## $BRCA1
## [1] "rs8176273" "rs8176265"
##
## $BRCA2
## [1] "rs9562605" "rs1799943"
```

Simulate SNP data and compute statistics

Simulate genotyping data:

```
G <- matrix(rbinom(n*nrow(snpInfo), 2, 0.5), ncol=nrow(snpInfo))
colnames(G) <- snpInfo$rsid
```

SNPs can be weighted within gene sets. Generate dummy weights and append them:

```
for(i in seq_len(length(geneset))){
  weight <- rep(1, length(geneset[[i]]))
  geneset[[i]] <- list(geneset[[i]], weight)
}
```

Compute SKAT statistics for each gene set:

```
res <- geneStat(x=G, Z=Z, alpha=thetaest$alpha, theta=thetaest$theta,
                gtime=gtime, delta=delta, geneSet=geneset)
res$stat

## $BRCA1
## [1] 71.49157
##
## $BRCA2
## [1] 422.893
```

Incorporating family structure

Generate grouped survival data:

```
rm(list=ls())
set.seed(111)
m <- 10

# family ID
fgrp <- as.character(rep(1:m, each=3))

# role within family
f_ind <- rep(c('o','f','m'),m)

# grouped survival data
gtimes <- sample(1:4, m*3, replace=TRUE)
deltas <- sample(0:1, m*3, replace=TRUE)

# variable of interest
g <- rbinom(m*3, 2, 0.1)

# parameter search bounds
upper  <- 2
lower  <- 0
```

Examples of alphaEstFam and varEstFam

Estimate baseline survival rates (always under the null hypothesis):

```
alphaest <- alphaEstFam(gtimes, deltas)
alphaest

## [1] 0.8571429 0.6666667 0.3333333 0.0000000
```

Estimate variance under the null by setting beta=0:

```
varest <- varEstFam(x=g, fam_group=fgrp, fam_role=f_ind, alpha=alphaest,
                    gtime=gtimes, delta=deltas, lower, upper, beta=0)
varest

## [1] 0.1476677
```

Examples of groupedSurvFam, PvalueFam, and betaEstFam

Compute the efficient score under the null by setting beta=0, and compute the associated p-values:

```
effFam <- groupedSurvFam(x=g, fam_group=fgrp, fam_role=f_ind, alpha=alphaest,
                         var=varest, gtime=gtimes, delta=deltas, beta=0)
PvalueFam(effFam)

## pi0 is can not be evaluated due to small number of p-values
## FDR and FWER is not calculated.
##        stat    pvalue
## 1 0.7455322 0.3878945
```

Estimate β:

```
betaEstFam(x=g, fam_group=fgrp, fam_role=f_ind, alpha=alphaest,
           var=varest, gtime=gtimes, delta=deltas, lower, upper)

## [1] 2.622524e-05
```

References

[1] Prentice, RL and Gloeckler, LA (1978). Regression analysis of grouped survival data with application to breast cancer data. *Biometrics*, 34(1):57-67.

[2] Ripatti, S and Palmgren, J (2004). Estimation of multivariate frailty models using penalized partial likelihood. *Biometrics*, 56(4):1016-1022.

[3] Bonferroni, CE (1935). Il calcolo delle assicurazioni su gruppi di teste. *Studi in Onore del Professore Salvatore Ortu Carbon*, 13-60.

[4] Storey, JD (2013). The positive false discovery rate: a Bayesian interpretation and the q-value. *Annals of Statistics*, 31(6):2013-2035.

[5] Storey, JD, Taylor, JE, and Siegmund, D (2004). Strong control, conservative point estimation and simultaneous conservative consistency of false discovery rates: a unified approach. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 66:187-205.

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

[7] Obenchain, V, Lawrence, M, Carey, V, Gogarten, S, Shannon, P, and Morgan, M (20014). VariantAnnotation: a Bioconductor package for exploration and annotation of genetic variants *Bioinformatics*, 30(14):2076-2078.

Session Information

- R version 3.4.1 (2017-06-30), `x86_64-pc-linux-gnu`
- Running under: `Debian GNU/Linux 9 (stretch)`
- Matrix products: default
- BLAS: `/usr/lib/openblas-base/libblas.so.3`
- LAPACK: `/usr/lib/libopenblasp-r0.2.19.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: RSQLite 2.0, groupedSurv 1.0.3, knitr 1.17, snplist 0.18
- Loaded via a namespace (and not attached): AnnotationDbi 1.32.3, Biobase 2.30.0, BiocGenerics 0.16.1, DBI 0.7, IRanges 2.4.8, R.methodsS3 1.7.1, R.oo 1.21.0, R.utils 2.5.0, RCurl 1.95-4.8, Rcpp 0.12.12, S4Vectors 0.8.11, XML 3.98-1.9, biomaRt 2.26.1, bit 1.1-12, bit64 0.9-7, bitops 1.0-6, blob 1.1.0, codetools 0.2-15, colorspace 1.3-2, compiler 3.4.1, digest 0.6.12, doParallel 1.0.10, doRNG 1.6.6, evaluate 0.10.1, foreach 1.4.3, ggplot2 2.2.1, grid 3.4.1, gtable 0.2.0, highr 0.6, iterators 1.0.8, lazyeval 0.2.0, magrittr 1.5, memoise 1.1.0, munsell 0.4.3, parallel 3.4.1, pkgconfig 2.0.1, pkgmaker 0.22, plyr 1.8.4, qvalue 2.2.2, registry 0.3, reshape2 1.4.2, rlang 0.1.2, rngtools 1.2.4, scales 0.5.0, splines 3.4.1, stats4 3.4.1, stringi 1.1.5, stringr 1.2.0, tibble 1.3.4, tools 3.4.1, xtable 1.8-2