gwasurvivr Vignette

Abbas Rizvi[1]***, Ezgi Karaesmen***[1]***, Martin Morgan***[2]***, and Lara Sucheston-Campbell***[1]

[1]The Ohio State University, Columbus, OH
[2]Roswell Park Comprehensive Cancer Center, Buffalo, NY

May 02, 2018

Package

gwasurvivr 0.99.1

Contents

1 Introduction

`gwasurvivr` can be used to perform survival analyses of imputed genotypes from Sanger and Michigan imputation servers and IMPUTE2 software. This vignette is a tutorial on how to perform these analyses. This package can be run locally on a Linux, Mac OS X, Windows or conveniently batched on a high performing computing cluster. `gwasurvivr` iteratively processes the data in chunks and therefore intense memory requirements are not necessary. `gwasurvivr` package comes with three main functions to perform survival analyses using Cox proportional hazard (Cox PH) models depending on the imputation method used to generate the genotype data:

1. `michiganCoxSurv`: Performs survival analysis on imputed genetic data stored in compressed VCF files generated via Michigan imputation server.

2. `sangerCoxSurv`: Performs survival analysis on imputed genetic data stored in compressed VCF files generated via Sanger imputation server.

3. `impute2CoxSurv`: Performs survival analysis on imputed genetic data from IMPUTE2 output.

All functions fit a Cox PH model to each SNP including other user defined covariates and will save the results as a text file directly to disk that contains survival analysis results. `gwasurvivr` functions can also test for interaction of SNPs with a given covariate. See examples for further details.

1.1 Main input arguments

All three functions require the following main arguments:

- `vcf.file`: A character string giving the path to genotype data file (`impute.file` for IMPUTE2)
- `covariate.file`: A data frame comprising sample IDs (that match to the genotype data), phenotype (time, event) and additional covariate data
- `id.column`: A character string providing exact match to sample ID column in covariate.file
- `time.to.event`: A character string that matches time column name in covariate.file
- `event`: Character string that matches event column name in covariate.file
- `covariates`: Character vector with matching column names in covariate.file of covariates of interest
- `out.file`: A character string giving output name

Further arguments can be passed depending on the user preference. For example, user can define minor allele frequency (MAF) or info score threshold to filter out SNPs that have low MAF or info score to avoid false-positive signals and to reduce computing time. User can also define a subset of samples to be analyzed by providing the set of sample IDs. Users can also control how chunk size – the number of rows (SNPs) to include for each iteration.

IMPORTANT: In the `covariate.file`, categorical variables need to be converted to indicator (dummy) variables and be of class `numeric`. Ordinal variables represented as characters, ie "low", "medium" and "high" should be converted to the appropriate numeric values as well.

1.2 Main output format

The output for the 3 main functions in `gwasurvivr` are very similar but with subtle differences. In general the output includes the following main fields: RSID, TYPED, CHR, POS, REF, ALT, Allele Frequencies*, INFO*, PVALUES, HRs, HR confidence intervals, coefficient estimates, standard errors, Z-statistic, N, and NEVENT. Allele frequencies and INFO differ by the input imputation software.

Note: Invoking the `inter.term` argument for any of the functions will make the PVALUE and HRs and HR confidence intervals represent the INTERACTION term and not the SNP alone.

The non-software specific fields are summarized below:

Column	Description
RSID	SNP ID
CHR	Chromosome number
POS	Genomic Position (BP)
REF	Reference Allele
ALT	Alternate Allele
SAMP_FREQ_ALT	Alternate Allele frequency in sample being tested
SAMP_MAF	Minor allele frequency in sample being tested
PVALUE	P-value of single SNP or interaction term
HR	Hazard Ratio (HR)
HR_lowerCI	Lower bound 95% CI of HR
HR_upperCI	Upper bound 95% CI of HR
COEF	Estimated coefficient of SNP
SE.COEF	Standard error of coefficient estimate
Z	Z-statistic
N	Number of individuals in sample being tested
NEVENT	Number of events that occurred in sample being tested

The software specific fields are summarized below:
1. `michiganCoxSurv` unique output columns are AF, MAF, INFO, ER2. They are summarized below.

Column	Description
TYPED	Imputation status: TRUE (SNP IS TYPED)/FALSE (SNP IS IMPUTED)
AF	Minimac3 output Alternate Allele Frequency
MAF	Minimac3 output of Minor Allele Frequency
INFO	Imputation INFO score (minimac3 $R^2$)
ER2	Minimac3 ouput empirical $R^2$

Please see Minimac3 Info File for details on output

2. `sangerCoxSurv`

Column	Description
TYPED	Imputation status: TRUE (SNP IS TYPED)/FALSE (SNP IS IMPUTED)
RefPanelAF	HRC Reference Panel Allele Frequency
INFO	Imputation INFO score from PBWT

3. `impute2CoxSurv`

Column	Description
TYPED	'—' is imputed, repeated RSID is typed
A0	Allele coded 0 in IMPUTE2
A1	Allele coded 1 in IMPUTE2
exp_freq_A1	Expected allele frequency of alelle code A1
INFO	Imputation INFO score computed based on ratio of empircal and expected variance in dosage format

More statistics can be printed out by invoking the `print.covs` argument and setting it to `print.covs=all` (single SNP/SNP*covariate interaction) or `print.covs=some` (SNP*covariate ineraction). These options are available mainly for modeling purposes (covariate selection) and aren't suggested for very large analyses as it will greatly increase the number of columns in the output, depending on how many covariates users are adjusting for.

2 Getting started

Install `gwasurvivr` from the Sucheston-Campbell Lab Github repository using `devtools`.

```
devtools::install_github("suchestoncampbelllab/gwasurvivr")
```

2.1 Dependencies

Note: This package depends on `GWASTools` which uses netcdf framework on linux. Therefore, for linux users, please install `libnetcdf-dev` and `netcdf-bin` before installing `gwasurvivr`. These linux libraries may already installed on an academic computing cluster.

CRAN packages:
1. `ncdf4`
2. `matrixStats`
3. `parallel`
4. `survival`

```
install.packages(c("ncdf4", "matrixStats", "parallel", "survival"))
```

Bioconductor packages:
1. `GWASTools`
2. `VariantAnnotation`
3. `SummarizedExperiment`

```
source("https://bioconductor.org/biocLite.R")
biocLite("GWASTools")
biocLite("VariantAnnotation")
biocLite("SummarizedExperiment")
```

Load `gwasurvivr`.

```
library(gwasurvivr)
```

2.2 User settings: parallelization setup

`gwasurvivr` uses `parallel` package for its internal parallelization to fit the Cox PH models. Users are not required to define a parallelization setup, by default `gwasurvivr` functions will detect the user's operating system and set the cluster object to `FORK` if the platform is Linux/OS X and to `SOCK` if Windows. However, parallelization settings can be modified by the user if needed. Users are given two ways to define their cluster settings:

1. Setting the number of cores to be used:

Linux/OS X users can run analyses on a prespecified number of cores by setting the option in the R session as shown below. This option should be defined in the R session before running any of the `gwasurvivr` functions. Here we decide to use 4 cores. This option is not available to Windows users.

```
options("gwasurvivr.cores"=4)
```

2. Providing a user defined cluster object

To modify more settings, users can also provide a "cluster object" to any of the `gwasurvivr` functions. The cluster object can be generated via `makeCluster`, `makePSOCKcluster`, `makeForkCluster` functions from `parallel` package or similar cluster object functions from `snow` or `snowfall` packages. This method can be applied on any operating system. User can create a cluster object before running any of the functions and pass the cluster object to the `clusterObj` argument as shown below. For further details see `??parallel::parallel`.

```
library(parallel)
cl <- makeCluster(detectCores())

impute2CoxSurv(..., clusterObj=cl)
sangerCoxSurv(..., clusterObj=cl)
michiganCoxSurv(..., clusterObj=cl)
```

3 R Session Examples

While we use `tidyverse` and `magrittr` packages in these example for data preparation/manipulation purposes, any of the data pre-processing steps can be done in base R.

```
library(tidyverse)
library(magrittr)
```

3.1 Michigan Imputation Server

Michigan Imputation Server pre-phases typed genotypes using HAPI-UR, SHAPEIT, or EAGLE (default is EAGLE2), imputes using Minimac3 imputation engine and outputs Blocked GNU Zip Format VCF files (`.vcf.gz`). Just as with Sanger these `.vcf.gz` files are used as input for `gwasurvivr`. The function, `michiganCoxSurv` uses a modification of Cox proportional hazard regression from the R library `survival`. Minimac uses slightly different metrics to assess imputation quality (R^2 versus info score) and complete details as to minimac output are available on the Minimac3 Wikipage.

The function, `michiganCoxSurv` uses a modification of cox proportional hazard regression from the R library `survival`. Built specifically for genetic data, `michiganCoxSurv` allows the user to filter on info score (imputation quality metric) and minor allele frequency from the reference panel used for imputation using `RefPanelAF` as the input arguement for `maf.filter`. Users are also provided with the sample minor allele frequency in the `sangerCoxSurv` output.

Samples can be selected for analyses by providing a vector of `sample.ids`. The output from Sanger imputation server returns the samples as `SAMP1, ..., SAMPN`, where `N` is the total number of samples. The sample order corresponds to the sample order in the vcf file you used for imputation. Note, sample order can also be found in the `.fam` file if genotyping data were initially in `.bed`, `.bim` and `.fam` (PLINK) format prior to conversion to VCF. If no sample list is specified all samples are included in the analyses.

```
vcf.file <- system.file(package="gwasurvivr",
                        "extdata",
                        "michigan.chr14.dose.vcf.gz")
pheno.fl <- system.file(package="gwasurvivr",
                        "extdata",
                        "simulated_pheno.txt")
pheno.file <- read_delim(pheno.fl, delim=" ", col_names=TRUE)
pheno.file %>%
    head()
## # A tibble: 6 x 8
##    ID_1 ID_2  event  time   age DrugTxYes sex    group
##   <int> <chr> <int> <dbl> <dbl>     <int> <chr>  <chr>
## 1     1 SAMP1     0 12     33.9         0 male   control
## 2     2 SAMP2     1  7.61  58.7         1 male   experimental
## 3     3 SAMP3     0 12     39.4         0 female control
## 4     4 SAMP4     0  4.3   38.8         0 male   control
## 5     5 SAMP5     0 12     43.6         0 male   experimental
## 6     6 SAMP6     1  2.6   57.7         0 male   control
# recode sex column and remove first column
pheno.file <- pheno.file %>%
        mutate(SexFemale=if_else(sex=="female", 1L, 0L))

# select only experimental group sample.ids
sample.ids <- pheno.file %>%
        filter(group=="experimental") %$%
        ID_2
head(sample.ids)
## [1] "SAMP2"  "SAMP5"  "SAMP7"  "SAMP9"  "SAMP11" "SAMP12"
```

In this example, we will select samples from the `experimental` group and will test survival only on these patients. The first column in the `pheno.file` are sample IDs (we will match on these). We include `age`, `DrugTxYes`, and `sex` in the survival model as covariates.

We perform the analysis using the `experimental` group to demonstrate how one may want to prepare their data if not initially all samples are patients or cases (i.e. a case-control study and survival of cases is of interest). We also are showing how the IDs (`sample.ids`) need to be a vector of class `character`. The `chunk.size` refers to size of each data chunk read in and is defaulted to 10,000 rows, users can customize that to their needs. The larger the `chunk.size` the more memory (RAM) required to run the analysis. The recommended `chunk.size=500` and probably should not exceed `chunk.size=5000`.

By default survival estimates and pvalues for the SNP adjusted for other covariates are outputted (`print.covs='only'`), however users can select `print.covs=all` to get the coefficient estimates for covariates included in the model. Depending on the number of covariates included this can add substantially to output file size. Next we run `michiganCoxSurv` with the default, `print.covs="only"`, load the results into R and provide descriptions of output by column. We will then run the analysis again using `print.covs="all"`. `verbose=TRUE` is used for these examples so the function display messages while running.

Use `?michiganCoxSurv` for argument specific documentation.

3.1.1 Single SNP analysis

`print.covs="only"`

```
michiganCoxSurv(vcf.file=vcf.file,
                covariate.file=pheno.file,
                id.column="ID_2",
                sample.ids=sample.ids,
                time.to.event="time",
                event="event",
                covariates=c("age", "SexFemale", "DrugTxYes"),
                inter.term=NULL,
                print.covs="only",
                out.file="michigan_only",
                info.filter=0.3,
                maf.filter=0.005,
                chunk.size=500,
                verbose=TRUE,
                clusterObj=NULL)
```

```
## Analysis started on 2018-05-02 at 15:55:07
## Covariates included in the models are: age, DrugTxYes, SexFemale
## 253 samples are included in the analysis
## Analyzing chunk 0-500
## Analyzing chunk 500-1000
## Analysis completed on 2018-05-02 at 15:55:10
## 240 SNPs were removed from the analysis for not meeting the threshold criteria.
## List of removed SNPs can be found in /var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/michigan
## 260 SNPs were analyzed in total
## The survival output can be found at /var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/michigan_
```

Here we load the data and glimpse the first few values in each column outputted from the SNP*interaction survival analyses using `print.covs="only"`.

```
michigan_only <- read_tsv("michigan_only.coxph")
```

```
michigan_only %>%
    head() %>%
    glimpse()
## Observations: 6
## Variables: 21
## $ RSID          <chr> "rs201487625", "rs28881575", "rs77206417", "rs1388...
## $ TYPED         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
## $ CHR           <int> 14, 14, 14, 14, 14, 14
## $ POS           <int> 19258506, 19264217, 19264589, 19264800, 19264875, ...
## $ REF           <chr> "A", "A", "C", "C", "G", "T"
## $ ALT           <chr> "G", "G", "T", "T", "A", "A"
## $ AF            <dbl> 0.7533720, 0.9868190, 0.0463043, 0.0105328, 0.3109...
## $ MAF           <dbl> 0.2466280, 0.0131810, 0.0463043, 0.0105328, 0.3109...
## $ SAMP_FREQ_ALT <dbl> 0.7861, 0.9822, 0.0363, 0.0159, 0.3101, 0.0199
## $ SAMP_MAF      <dbl> 0.2139, 0.0178, 0.0363, 0.0159, 0.3101, 0.0199
## $ INFO          <dbl> 0.925998, 0.647238, 0.487405, 0.378574, 0.583201, ...
## $ ER2           <dbl> NA, NA, NA, NA, NA, NA
## $ PVALUE        <dbl> 0.5221423, 0.3859810, 0.7960395, 0.5338763, 0.8391...
## $ HR            <dbl> 0.8965579, 2.0522850, 1.1445003, 1.5619517, 1.0395...
## $ HR_lowerCI    <dbl> 0.6417414, 0.4039371, 0.4112794, 0.3832609, 0.7145...
## $ HR_upperCI    <dbl> 1.252554, 10.427054, 3.184893, 6.365620, 1.512361,...
## $ COEF          <dbl> -0.10919241, 0.71895381, 0.13496810, 0.44593611, 0...
## $ SE.COEF       <dbl> 0.1706007, 0.8293112, 0.5221687, 0.7168242, 0.1912...
## $ Z             <dbl> -0.6400466, 0.8669288, 0.2584761, 0.6220997, 0.202...
## $ N             <int> 253, 253, 253, 253, 253, 253
## $ NEVENT        <int> 100, 100, 100, 100, 100, 100
```

3.1.2 SNP with covariate interaction

A SNP*covariate interaction can be implemented using the `inter.term` argument. In this example, we will use `DrugTxYes` from the covariate file as the covariate we want to test for interaction with the SNP.

`print.covs="only"`

```
michiganCoxSurv(vcf.file=vcf.file,
                covariate.file=pheno.file,
                id.column="ID_2",
                sample.ids=sample.ids,
                time.to.event="time",
                event="event",
                covariates=c("age", "SexFemale", "DrugTxYes"),
                inter.term="DrugTxYes",
                print.covs="only",
                out.file="michigan_intx_only",
                info.filter=0.3,
                maf.filter=0.005,
                chunk.size=500,
                verbose=FALSE,
                clusterObj=NULL)
```

Here we load the data and glimpse the first few values in each column outputted from the SNP*interaction survival analyses using `print.covs="only"`

```
michigan_intx_only <- read_tsv("michigan_intx_only.coxph")
```

```
michigan_intx_only %>%
    head() %>%
    glimpse()
## Observations: 6
## Variables: 21
## $ RSID          <chr> "rs201487625", "rs28881575", "rs77206417", "rs1388...
## $ TYPED         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
## $ CHR           <int> 14, 14, 14, 14, 14, 14
## $ POS           <int> 19258506, 19264217, 19264589, 19264800, 19264875, ...
## $ REF           <chr> "A", "A", "C", "C", "G", "T"
## $ ALT           <chr> "G", "G", "T", "T", "A", "A"
## $ AF            <dbl> 0.7533720, 0.9868190, 0.0463043, 0.0105328, 0.3109...
## $ MAF           <dbl> 0.2466280, 0.0131810, 0.0463043, 0.0105328, 0.3109...
## $ SAMP_FREQ_ALT <dbl> 0.7861, 0.9822, 0.0363, 0.0159, 0.3101, 0.0199
## $ SAMP_MAF      <dbl> 0.2139, 0.0178, 0.0363, 0.0159, 0.3101, 0.0199
## $ INFO          <dbl> 0.925998, 0.647238, 0.487405, 0.378574, 0.583201, ...
## $ ER2           <dbl> NA, NA, NA, NA, NA, NA
## $ PVALUE        <dbl> 0.45928864, 0.02343794, 0.29679339, 0.31659650, 0....
## $ HR            <dbl> 1.331283e+00, 4.124811e+01, 3.413024e-01, 7.002755...
## $ HR_lowerCI    <dbl> 6.239217e-01, 1.653024e+00, 4.530005e-02, 2.304321...
## $ HR_upperCI    <dbl> 2.840604e+00, 1.029269e+03, 2.571462e+00, 2.128114...
## $ COEF          <dbl> 0.2861432, 3.7196052, -1.0749863, 11.1566440, -0.3...
## $ SE.COEF       <dbl> 0.3866702, 1.6413260, 1.0303372, 11.1401955, 0.426...
## $ Z             <dbl> 0.7400187, 2.2662196, -1.0433345, 1.0014765, -0.71...
## $ N             <int> 253, 253, 253, 253, 253, 253
## $ NEVENT        <int> 100, 100, 100, 100, 100, 100
```

3.2 Sanger Imputation Server

Sanger Imputation Server pre-phases typed genotypes using either SHAPEIT or EAGLE, imputes genotypes using PBWT algorithm and outputs a `.vcf.gz` file for each chromosome. These `.vcf.gz` files are used as input for `gwasurvivr`. The function, `sangerCoxSurv` uses a modification of cox proportional hazard regression from the R library `survival`. Built specifically for genetic data, `sangerCoxSurv` allows the user to filter on info score (imputation quality metric) and minor allele frequency from the reference panel used for imputation using `RefPanelAF` as the input arguement for `maf.filter`. Users are also provided with the sample minor allele frequency in the `sangerCoxSurv` output.

Samples can be selected for analyses by providing a vector of `sample.ids`. The output from Sanger imputation server returns the samples as `SAMP1, ..., SAMPN`, where `N` is the total number of samples. The sample order corresponds to the sample order in the vcf file you used for imputation. Note, sample order can also be found in the `.fam` file if genotyping data were initially in `.bed`, `.bim` and `.fam` (PLINK) format prior to conversion to VCF. If no sample list is specified all samples are included in the analyses.

```
vcf.file <- system.file(package="gwasurvivr",
                        "extdata",
                        "sanger.pbwt_reference_impute.vcf.gz")
```

```
pheno.fl <- system.file(package="gwasurvivr",
                        "extdata",
                        "simulated_pheno.txt")
pheno.file <- read_delim(pheno.fl,
                         delim=" ",
                         col_names=TRUE)
## Parsed with column specification:
## cols(
##   ID_1 = col_integer(),
##   ID_2 = col_character(),
##   event = col_integer(),
##   time = col_double(),
##   age = col_double(),
##   DrugTxYes = col_integer(),
##   sex = col_character(),
##   group = col_character()
## )
pheno.file %>%
    head()
## # A tibble: 6 x 8
##    ID_1 ID_2  event  time   age DrugTxYes sex    group
##   <int> <chr> <int> <dbl> <dbl>     <int> <chr>  <chr>
## 1     1 SAMP1     0 12     33.9         0 male   control
## 2     2 SAMP2     1  7.61  58.7         1 male   experimental
## 3     3 SAMP3     0 12     39.4         0 female control
## 4     4 SAMP4     0  4.3   38.8         0 male   control
## 5     5 SAMP5     0 12     43.6         0 male   experimental
## 6     6 SAMP6     1  2.6   57.7         0 male   control
```

In this example, we will select samples from the `experimental` group and will test survival only on these patients. The first column in the `pheno.file` are sample IDs (we will match on these). We include `age`, `DrugTxYes`, and `sex` in the survival model as covariates.

```
# create new sex variable where Females are 1 and males are 0
pheno.file <- pheno.file %>%
        mutate(SexFemale=if_else(sex=="female", 1L, 0L))

# select only experimental group sample.ids
sample.ids <- pheno.file %>%
        filter(group=="experimental") %$%
        ID_2
sample.ids %>%
    head()
## [1] "SAMP2"  "SAMP5"  "SAMP7"  "SAMP9"  "SAMP11" "SAMP12"
```

We perform the analysis using the `experimental` group to demonstrate how one may want to prepare their data if not initially all samples are patients or cases (i.e. a case-control study and survival of cases is of interest). We also are showing how the IDs (`sample.ids`) need to be a vector of class `character`. The `chunk.size` refers to size of each data chunk read in and is defaulted to 10,000 rows, users can customize that to their needs. The larger the `chunk.size` the more memory (RAM) required to run the analysis. The recommended `chunk.size=500` and probably should not exceed `chunk.size=5000`.

By default survival estimates and pvalues for the SNP adjusted for other covariates are outputted (`print.covs='only'`), however users can select `print.covs=all` to get the coefficient estimates for covariates included in the model. Depending on the number of covariates included this can add substantially to output file size. Next we run `sangerCoxSurv` with the default, `print.covs="only"`, load the results into R and provide descriptions of output by column. We will then run the analysis again using `print.covs="all"`. `verbose=TRUE` is used for these examples so the function display messages while running.

Use `?sangerCoxSurv` for argument specific documentation.

```
sangerCoxSurv(vcf.file=vcf.file,
              covariate.file=pheno.file,
              id.column="ID_2",
              sample.ids=sample.ids,
              time.to.event="time",
              event="event",
              covariates=c("age", "SexFemale", "DrugTxYes"),
              inter.term=NULL,
              print.covs="only",
              out.file="sanger_only",
              info.filter=0.3,
              maf.filter=0.005,
              chunk.size=500,
              verbose=TRUE,
              clusterObj=NULL)
```

```
## Analysis started on 2018-05-02 at 15:55:14
## Covariates included in the models are: age, DrugTxYes, SexFemale
## 253 samples are included in the analysis
## Analyzing chunk 0-500
## Analyzing chunk 500-1000
## Analysis completed on 2018-05-02 at 15:55:17
## 240 SNPs were removed from the analysis for not meeting the threshold criteria.
## List of removed SNPs can be found in /var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/sanger_
## 260 SNPs were analyzed in total
## The survival output can be found at/var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/sanger_on
```

3.2.1 Single SNP analysis

`print.covs="only"`

Here we load the data and glimpse the first few values in each column from the survival analyses.

```
sanger_only %>%
    head() %>%
    glimpse()
## Observations: 6
## Variables: 19
## $ RSID          <chr> "rs201487625", "rs28881575", "rs77206417", "rs1388...
## $ TYPED         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
## $ CHR           <int> 14, 14, 14, 14, 14, 14
## $ POS           <int> 19258506, 19264217, 19264589, 19264800, 19264875, ...
## $ REF           <chr> "A", "A", "C", "C", "G", "T"
## $ ALT           <chr> "G", "G", "T", "T", "A", "A"
## $ RefPanelAF    <dbl> 0.7533720, 0.9868190, 0.0463043, 0.0105328, 0.3109...
## $ SAMP_FREQ_ALT <dbl> 0.7861, 0.9822, 0.0363, 0.0159, 0.3101, 0.0199
## $ SAMP_MAF      <dbl> 0.2139, 0.0178, 0.0363, 0.0159, 0.3101, 0.0199
## $ INFO          <dbl> 0.925998, 0.647238, 0.487405, 0.378574, 0.583201, ...
## $ PVALUE        <dbl> 0.5221423, 0.3859810, 0.7960395, 0.5338763, 0.8391...
## $ HR            <dbl> 0.8965579, 2.0522850, 1.1445003, 1.5619517, 1.0395...
## $ HR_lowerCI    <dbl> 0.6417414, 0.4039371, 0.4112794, 0.3832609, 0.7145...
## $ HR_upperCI    <dbl> 1.252554, 10.427054, 3.184893, 6.365620, 1.512361,...
## $ COEF          <dbl> -0.10919241, 0.71895381, 0.13496810, 0.44593611, 0...
## $ SE.COEF       <dbl> 0.1706007, 0.8293112, 0.5221687, 0.7168242, 0.1912...
## $ Z             <dbl> -0.6400466, 0.8669288, 0.2584761, 0.6220997, 0.202...
## $ N             <int> 253, 253, 253, 253, 253, 253
## $ NEVENT        <int> 100, 100, 100, 100, 100, 100
```

Column names with descriptions from the survival analyses with covariates, specifying the default `print.covs="only"`

`print.covs="all"` In this example, `sangerCoxSurv` is used with `print.covs="all"`. The output is described below. `verbose=FALSE` is used for these examples so the function does not display messages while running.

```r
sangerCoxSurv(vcf.file=vcf.file,
              covariate.file=pheno.file,
              id.column="ID_2",
              sample.ids=sample.ids,
              time.to.event="time",
              event="event",
              covariates=c("age", "SexFemale", "DrugTxYes"),
              inter.term=NULL,
              print.covs="all",
              out.file="sanger_all",
              info.filter=0.3,
              maf.filter=0.005,
              chunk.size=500,
              verbose=FALSE,
              clusterObj=NULL)
```

3.2.2 SNP with covariate interaction

A SNP*covariate interaction can be implemented using the `inter.term` argument. In this example, we will use `DrugTxYes` from the covariate file as the covariate we want to test for interaction with the SNP.

`print.covs="only"`

```r
sangerCoxSurv(vcf.file=vcf.file,
              covariate.file=pheno.file,
              id.column="ID_2",
              sample.ids=sample.ids,
              time.to.event="time",
              event="event",
              covariates=c("age", "SexFemale", "DrugTxYes"),
              inter.term="DrugTxYes",
              print.covs="only",
              out.file="sanger_intx_only",
              info.filter=0.3,
              maf.filter=0.005,
              chunk.size=500,
              verbose=TRUE,
              clusterObj=NULL)
```

```
## Analysis started on 2018-05-02 at 15:55:17
## Covariates included in the models are: age, DrugTxYes, SexFemale
## Models will include interaction term: SNP*DrugTxYes
## 253 samples are included in the analysis
## Analyzing chunk 0-500
## Analyzing chunk 500-1000
## Analysis completed on 2018-05-02 at 15:55:22
## 240 SNPs were removed from the analysis for not meeting the threshold criteria.
## List of removed SNPs can be found in /var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/sanger_i
## 260 SNPs were analyzed in total
## The survival output can be found at/var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/sanger_int
```

Here we load the data and glimpse the first few values in each column outputted from the SNP*interaction survival analyses using `print.covs="only"`

```
sanger_intx_only <- read_tsv("sanger_intx_only.coxph")
```

```
sanger_intx_only %>%
    head() %>%
    glimpse()
## Observations: 6
## Variables: 19
## $ RSID          <chr> "rs201487625", "rs28881575", "rs77206417", "rs1388...
## $ TYPED         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
## $ CHR           <int> 14, 14, 14, 14, 14, 14
## $ POS           <int> 19258506, 19264217, 19264589, 19264800, 19264875, ...
## $ REF           <chr> "A", "A", "C", "C", "G", "T"
## $ ALT           <chr> "G", "G", "T", "T", "A", "A"
## $ RefPanelAF    <dbl> 0.7533720, 0.9868190, 0.0463043, 0.0105328, 0.3109...
## $ SAMP_FREQ_ALT <dbl> 0.7861, 0.9822, 0.0363, 0.0159, 0.3101, 0.0199
## $ SAMP_MAF      <dbl> 0.2139, 0.0178, 0.0363, 0.0159, 0.3101, 0.0199
## $ INFO          <dbl> 0.925998, 0.647238, 0.487405, 0.378574, 0.583201, ...
## $ PVALUE        <dbl> 0.45928864, 0.02343794, 0.29679339, 0.31659650, 0....
## $ HR            <dbl> 1.331283e+00, 4.124811e+01, 3.413024e-01, 7.002755...
## $ HR_lowerCI    <dbl> 6.239217e-01, 1.653024e+00, 4.530005e-02, 2.304321...
## $ HR_upperCI    <dbl> 2.840604e+00, 1.029269e+03, 2.571462e+00, 2.128114...
## $ COEF          <dbl> 0.2861432, 3.7196052, -1.0749863, 11.1566440, -0.3...
## $ SE.COEF       <dbl> 0.3866702, 1.6413260, 1.0303372, 11.1401955, 0.426...
## $ Z              <dbl> 0.7400187, 2.2662196, -1.0433345, 1.0014765, -0.71...
## $ N              <int> 253, 253, 253, 253, 253, 253
## $ NEVENT         <int> 100, 100, 100, 100, 100, 100
```

Column names with descriptions from the survival analyses of SNP*covariate and specifying `print.covs="only"`.

3.3 IMPUTE2 Imputation

IMPUTE2 is a genotype imputation and haplotype phasing program (Howie et al 2009). IMPUTE2 outputs 6 files for each chromosome chunk imputed (usually 5 MB in size). Only 2 of these files are required for analyses using `gwasurvivr`:

- Genotype file (`.impute`)
- Sample file (`.sample`)

More information can be read about these formats

We are going to load in and pre-process the genetic data and the phenotypic data (`covariate.file`).

```
impute.file <- system.file(package="gwasurvivr",
                            "extdata",
                            "impute_example.impute2.gz")
sample.file <- system.file(package="gwasurvivr",
                            "extdata",
                            "impute_example.impute2_sample")
covariate.file <- system.file(package="gwasurvivr",
                        "extdata",
                        "simulated_pheno.txt")
covariate.file <- read_delim(covariate.file, delim=" ")
covariate.file <- covariate.file %>%
        mutate(SexFemale=if_else(sex=="female", 1L, 0L))
covariate.file %>%
    head()
## # A tibble: 6 x 9
##    ID_1 ID_2  event  time   age DrugTxYes sex    group        SexFemale
##   <int> <chr> <int> <dbl> <dbl>     <int> <chr>  <chr>            <int>
## 1     1 SAMP1     0 12     33.9         0 male   control              0
## 2     2 SAMP2     1  7.61  58.7         1 male   experimental         0
## 3     3 SAMP3     0 12     39.4         0 female control              1
## 4     4 SAMP4     0  4.3   38.8         0 male   control              0
## 5     5 SAMP5     0 12     43.6         0 male   experimental         0
## 6     6 SAMP6     1  2.6   57.7         0 male   control              0
sample.ids <- covariate.file %>%
        filter(group=="experimental") %$%
        ID_2
```

To perform survival analysis using IMPUTE2 the function arguments are very similarto `michiganCoxSurv` and `sangerCoxSurv`, however the function now takes a chromosome arguement. This is needed to properly annotate the file output with the chromosome that these SNPs are in. This is purely an artifact of IMPUTE2 and how we leverage `GWASTools` in this function.

3.3.1 Single SNP analysis

First we will do the analysis with no interaction term, followed by doing the analysis with the interaction term. The recommended output setting for single SNP analysis is `print.cov="only"`.

```
impute2CoxSurv(impute.file=impute.file,
               sample.file=sample.file,
               chr=14,
               covariate.file=covariate.file,
               id.column="ID_2",
               sample.ids=sample.ids,
               time.to.event="time",
               event="event",
               covariates=c("age", "SexFemale", "DrugTxYes"),
               inter.term=NULL,
               print.covs="only",
               out.file="impute_example_only",
               chunk.size=500,
               maf.filter=0.005,
               info.filter=0.3,
               flip.dosage=TRUE,
               verbose=FALSE,
               clusterObj=NULL)
```

```
## Covariates included in the models are: age, DrugTxYes, SexFemale
## 253 samples are included in the analysis
## Analysis started on 2018-05-02 at 15:55:22
## 238 SNPs were removed from the analysis for not meeting
## the given threshold criteria or for having MAF = 0
## List of removed SNPs are saved to /var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/impute_exam
## 262 SNPs were included in the analysis
## The Cox model results output was saved to /var/folders/1w/bb5rrzjn4v9bvq_hlptzq4bc0000gn/T//RtmpKaoPCr/imp
## Analysis completed on 2018-05-02 at 15:55:24
```

Here we load the data and glimpse the first few values in each column output.

```
impute2_res_only <- read_tsv("impute_example_only.coxph")
impute2_res_only %>%
    head() %>%
    glimpse()
```

```
## Observations: 6
## Variables: 18
## $ RSID        <chr> "rs201487625", "rs28881575", "rs77206417", "rs138864...
## $ TYPED       <chr> "---", "---", "---", "---", "---", "---"
## $ CHR         <int> 14, 14, 14, 14, 14, 14
## $ POS         <int> 19258506, 19264217, 19264589, 19264800, 19264875, 19...
## $ A0          <chr> "A", "A", "C", "C", "G", "G"
## $ A1          <chr> "G", "G", "T", "T", "A", "A"
## $ exp_freq_A1 <dbl> 0.7861, 0.9822, 0.0363, 0.0159, 0.3101, 0.0055
## $ SAMP_MAF    <dbl> 0.2139, 0.0178, 0.0363, 0.0159, 0.3101, 0.0055
## $ INFO        <dbl> 1.000, 0.744, 0.448, 0.563, 0.586, 0.403
## $ PVALUE      <dbl> 0.5221423, 0.3859810, 0.7960396, 0.5338763, 0.839192...
## $ HR          <dbl> 0.8965579, 2.0522850, 1.1445002, 1.5619516, 1.039573...
## $ HR_lowerCI  <dbl> 0.641741430, 0.403937073, 0.411279389, 0.383260879, ...
## $ HR_upperCI  <dbl> 1.252554, 10.427054, 3.184893, 6.365620, 1.512361, 4...
## $ COEF        <dbl> -0.10919241, 0.71895381, 0.13496807, 0.44593609, 0.0...
## $ SE.COEF     <dbl> 0.1706007, 0.8293112, 0.5221687, 0.7168242, 0.191255...
## $ Z           <dbl> -0.64004663, 0.86692884, 0.25847600, 0.62209965, 0.2...
## $ N           <int> 253, 253, 253, 253, 253, 253
## $ NEVENT      <int> 100, 100, 100, 100, 100, 100
```

3.3.2 SNP covariate interaction

Now we will perform a SNP*covariate interaction survival analysis using `impute2CoxSurv`.

```
impute2CoxSurv(impute.file=impute.file,
               sample.file=sample.file,
               chr=14,
               covariate.file=covariate.file,
               id.column="ID_2",
               sample.ids=sample.ids,
               time.to.event="time",
               event="event",
               covariates=c("age", "SexFemale", "DrugTxYes"),
               inter.term="DrugTxYes",
               print.covs="only",
               out.file="impute_example_intx",
               chunk.size=500,
               maf.filter=0.01,
               info.filter=0.3,
               flip.dosage=TRUE,
               verbose=FALSE,
               clusterObj=NULL)
```

Here we load the data and glimpse the first few values in each column outputted from the SNP*interaction survival analyses using `print.covs="only"`.

```
impute2_res_intx <- read_tsv("impute_example_intx.coxph")
impute2_res_intx %>%
    head() %>%
    glimpse()
```

```
## Observations: 6
## Variables: 18
## $ RSID        <chr> "rs201487625", "rs28881575", "rs77206417", "rs138864...
## $ TYPED       <chr> "---", "---", "---", "---", "---", "---"
## $ CHR         <int> 14, 14, 14, 14, 14, 14
## $ POS         <int> 19258506, 19264217, 19264589, 19264800, 19264875, 19...
## $ A0          <chr> "A", "A", "C", "C", "G", "T"
## $ A1          <chr> "G", "G", "T", "T", "A", "A"
## $ exp_freq_A1 <dbl> 0.7861, 0.9822, 0.0363, 0.0159, 0.3101, 0.0199
## $ SAMP_MAF    <dbl> 0.2139, 0.0178, 0.0363, 0.0159, 0.3101, 0.0199
## $ INFO        <dbl> 1.000, 0.744, 0.448, 0.563, 0.586, 0.487
## $ PVALUE      <dbl> 0.45928865, 0.02343794, 0.29679328, 0.31659654, 0.47...
## $ HR          <dbl> 1.331283e+00, 4.124811e+01, 3.413023e-01, 7.002818e+...
## $ HR_lowerCI  <dbl> 6.239217e-01, 1.653024e+00, 4.530004e-02, 2.304297e-...
## $ HR_upperCI  <dbl> 2.840604e+00, 1.029269e+03, 2.571461e+00, 2.128174e+...
## $ COEF        <dbl> 0.2861432, 3.7196052, -1.0749866, 11.1566529, -0.305...
## $ SE.COEF     <dbl> 0.3866702, 1.6413260, 1.0303372, 11.1402054, 0.42643...
## $ Z           <dbl> 0.7400187, 2.2662196, -1.0433348, 1.0014764, -0.7159...
## $ N           <int> 253, 253, 253, 253, 253, 253
## $ NEVENT      <int> 100, 100, 100, 100, 100, 100
```

4 Batch Examples

4.1 Batch Example sangerCoxSurv

Batch jobs for multiple analyses and different subsets are easy to implement using `gwasurvivr`. These types of analyses should be reserved for usage on a UNIX-based high performance computing cluster. This is facilitated by the package `batch`, which can internalize R variables from bash. First write an R script (e.g. `mysurvivalscript.R`) to pass in bash.

```r
## mysurvivalscript.R
library(gwasurvivr)
library(batch)
library(tidyverse)
parseCommandArgs(evaluate=TRUE)

options("gwasurvivr.cores"=4)
# recode sex column
pheno.file <- pheno.file %>%
        mutate(SexFemale=if_else(sex=="female", 1L, 0L))
# select only experimental group
sample.ids <- pheno.file %>%
        filter(group=="experimental") %$%
        ID_2

## -- unlist the covariates
## (refer below to the shell script as to why we are doing this)
covariates <- covariates %>%
        str_split("_") %>%
        unlist()
```

```
sangerCoxSurv(vcf.file=vcf.file,
              covariate.file=pheno.file,
              id.column="ID_2",
              sample.ids=sample.ids,
              time.to.event=time,
              event=event,
              covariates=covariates,
              inter.term=NULL,
              print.covs="only",
              out.file="sanger_only",
              info.filter=0.3,
              maf.filter=0.005,
              chunk.size=500,
              verbose=TRUE,
              clusterObj=NULL)
```

Now we can run a shell script. This can be used well with manifest files to set up multiple runs with different outcomes and different subsets of samples. We define a manifest file has columns that corresond to the functions that the user wants to pass and each row is a separate analysis that a user may want to run. The covariates are separated by an underscore (`"_"`). This is so it can be passed properly, and also why we used `str_split` to split the covariates.

```
#!/bin/bash
PATH=/path/to/dir/impute_chr

R --script ${PATH}/survival/code/mysurvivalscript.R -q --args \
        vcf.file ${PATH}/chr14.vcf.gz \
        pheno.file ${PATH}/phenotype_data/pheno.txt \
        covariates DrugTxYes_age_SexFemale\
        time.to.event time \
        event event \
        out.file ${PATH}/survival/results/sanger_example_output
```

The file paths above are completely arbitrary and were just used as an example of how file structure may be and where desirable output would be stored.

4.2 Batch Example impute2CoxSurv

Exactly the same as for `sangerCoxSurv` but this time with the input arguments for `impute2CoxSurv`. See `?impute2CoxSurv` for help

4.3 Batch Example michiganCoxSurv

Exactly the same as for `sangerCoxSurv` but this time with the input arguments for `michiganCoxSurv`. See `?michiganCoxSurv` for help

Session info

Here is the output of `sessionInfo()` on the system that this document was compiled:

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2    magrittr_1.5      forcats_0.3.0     stringr_1.3.0
##  [5] dplyr_0.7.4       purrr_0.2.4       readr_1.1.1       tidyr_0.8.0
##  [9] tibble_1.4.2      ggplot2_2.2.1     tidyverse_1.2.1   gwasurvivr_0.99.1
## [13] knitr_1.20        BiocStyle_2.8.0
##
## loaded via a namespace (and not attached):
##   [1] colorspace_1.3-2           rprojroot_1.3-2
##   [3] DNAcopy_1.54.0             XVector_0.20.0
##   [5] GenomicRanges_1.32.0       GWASTools_1.26.0
##   [7] rstudioapi_0.7             mice_2.46.0
##   [9] MatrixModels_0.4-1         bit64_0.9-7
##  [11] AnnotationDbi_1.42.0       lubridate_1.7.4
##  [13] xml2_1.2.0                 splines_3.5.0
##  [15] mnormt_1.5-5               jsonlite_1.5
##  [17] logistf_1.22               Rsamtools_1.32.0
##  [19] broom_0.4.4                compiler_3.5.0
##  [21] httr_1.3.1                 backports_1.1.2
##  [23] assertthat_0.2.0           Matrix_1.2-14
##  [25] lazyeval_0.2.1             cli_1.0.0
##  [27] htmltools_0.3.6            quantreg_5.35
##  [29] prettyunits_1.0.2          tools_3.5.0
##  [31] gtable_0.2.0               glue_1.2.0
##  [33] GenomeInfoDbData_1.1.0     reshape2_1.4.3
##  [35] Rcpp_0.12.16               Biobase_2.40.0
##  [37] cellranger_1.1.0           Biostrings_2.48.0
##  [39] nlme_3.1-137               rtracklayer_1.40.0
##  [41] psych_1.8.3.3              lmtest_0.9-36
##  [43] xfun_0.1                   rvest_0.3.2
##  [45] XML_3.98-1.11              zlibbioc_1.26.0
##  [47] MASS_7.3-50                zoo_1.8-1
##  [49] scales_0.5.0               BSgenome_1.48.0
##  [51] VariantAnnotation_1.26.0   hms_0.4.2
##  [53] gdsfmt_1.16.0                parallel_3.5.0
##  [55] SummarizedExperiment_1.10.0 sandwich_2.4-0
##  [57] SparseM_1.77                yaml_2.1.19
##  [59] memoise_1.1.0               biomaRt_2.36.0
##  [61] rpart_4.1-13                stringi_1.1.7
##  [63] RSQLite_2.1.0               S4Vectors_0.18.0
##  [65] GenomicFeatures_1.32.0      BiocGenerics_0.26.0
##  [67] BiocParallel_1.14.0         GenomeInfoDb_1.16.0
##  [69] rlang_0.2.0                 pkgconfig_2.0.1
##  [71] matrixStats_0.53.1          bitops_1.0-6
##  [73] evaluate_0.10.1             lattice_0.20-35
##  [75] bindr_0.1.1                 GenomicAlignments_1.16.0
##  [77] bit_1.1-12                  plyr_1.8.4
##  [79] bookdown_0.7                R6_2.2.2
##  [81] IRanges_2.14.0              DelayedArray_0.6.0
##  [83] DBI_1.0.0                   pillar_1.2.2
##  [85] haven_1.1.1                 foreign_0.8-70
##  [87] mgcv_1.8-23                 GWASExactHW_1.01
##  [89] survival_2.42-3             RCurl_1.95-4.10
##  [91] nnet_7.3-12                 modelr_0.1.1
##  [93] crayon_1.3.4                utf8_1.1.3
##  [95] rmarkdown_1.9               progress_1.1.2
##  [97] grid_3.5.0                  readxl_1.1.0
##  [99] quantsmooth_1.46.0          blob_1.1.1
## [101] digest_0.6.15               stats4_3.5.0
## [103] munsell_0.4.3
```

References

1. Terry M. Therneau and Patricia M. Grambsch (2000). Modeling Survival Data: Extending the Cox Model. Springer, New York. ISBN 0-387-98784-3.

2. Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2017). SummarizedExperiment: SummarizedExperiment container. R package version 1.6.3.

3. Gogarten SM, Bhangale T, Conomos MP, Laurie CA, McHugh CP, Painter I, Zheng X, Crosslin DR, Levine D, Lumley T, Nelson SC, Rice K, Shen J, Swarnkar R, Weir BS and Laurie CC (2012). "GWASTools: an R/Bioconductor package for quality control and analysis of genome-wide association studies." Bioinformatics, 28(24), pp. 3329-3331. doi: 10.1093/bioinformatics/bts610.

4. B. N. Howie, P. Donnelly and J. Marchini (2009) A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. PLoS Genetics 5(6): e1000529

5. Das S, Forer L, Schönherr S, Sidore C, Locke AE, Kwong A, Vrieze S, Chew EY, Levy S, McGue M, Schlessinger D, Stambolian D, Loh PR, Iacono WG, Swaroop A, Scott LJ, Cucca F, Kronenberg F, Boehnke M, Abecasis GR, Fuchsberger C. **Next-generation genotype imputation service and methods.** Nature Genetics 48, 1284–1287 (2016). 27571263

6. Efficient haplotype matching and storage using the Positional Burrows-Wheeler Transform (PBWT)", Richard Durbin Bioinformatics 30:1266-72 (2014).