Review and testing of gwasurvivr: an R package for genome-wide survival analysis

Nadia Achatoui Ivy Tumoine Gabriel Abrantes Diaz Piotr Migdałek

2023-10-15

Introduction

Survival analysis has an important place in biomedical research, facilitating the exploration of time-to-event outcomes such as mortality or relapse.

An essential component in exploring the genetic basis of diseases involves investigating Single Nucleotide Polymorphisms (SNPs). These occasional variations in a single letter of DNA can have a significant impact on susceptibility to certain diseases or on response to medical treatments.

Integrating SNPs into survival analysis enables the discovery of genetic factors linked to time-to-event outcomes, shedding light on the genetic factors that influence disease progression and other critical events.

However, the major challenge is that our genome is made up of millions of SNPs, making large-scale survival analysis (GWAS) extremely complex. The existing software options for conducting such analyses are limited in several aspects (the need to interact with raw data, software not suited for survival analysis, and long execution times that hinder scalability). Consequently, researchers often face practical difficulties when conducting large-scale survival analyses.

gwasurvivr, an R/Bioconductor package was designed to surmount these challenges. This library offers a significant advancement in allowing researchers to perform survival analysis on large SNP datasets with remarkable efficiency and accuracy and with multiple file input formats such as VCF, IMPUTE2 or PLINK.

In this paper, we thoroughly examine the functionalities of gwasurvivr and offer an extensive evaluation of its operational mechanisms and effectiveness in unraveling the genetic factors that influence disease survival.

Methods

`gwasurvivr` implements a very common survival analysis model – the Cox proportional hazards regression model. This part focuses on the theory and algorithms behind survival analysis, Cox proportional hazards model and the way the authors implemented it.

Survival Analysis is a set of methods for analyzing time-to-event data, i.e. to estimate the time until an event occurs. We call this amount of time the survival time. One key concept in survival analysis is the hazard function denoted as $h(t)$. This function represents the instantaneous potential at time t for getting the event, given survival up to time t. The hazard function is really useful because it allows us to draw the survival curve denoted as $S(t)$ which is is the probability of the survival time to be greater or equal to t. Most of the time we empirically compute the survival curve from the data, thus obtaining a Kaplan-Meier curve. So for instance if we take t equals to 5 weeks, we will read on the Kaplan-Meier curve the probability of the survival time to be greater or equal to 5 weeks.

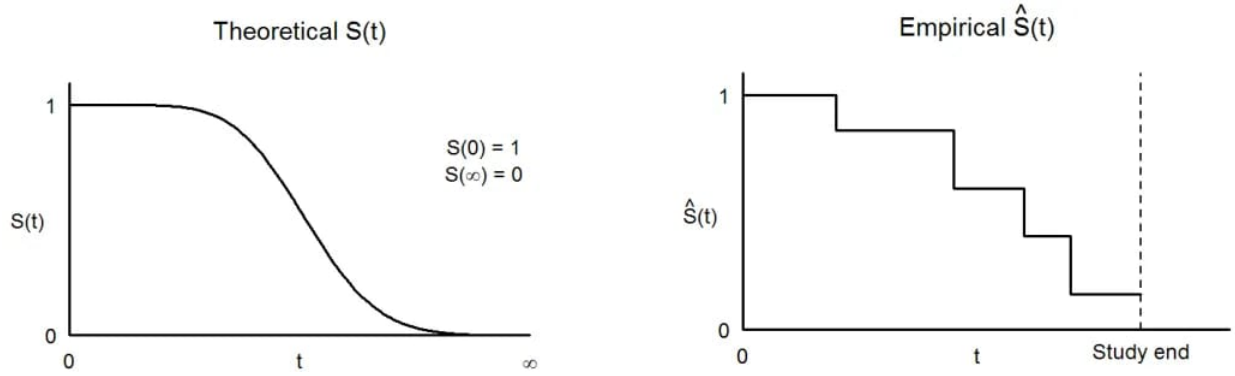

Figure 1: Survival function and Kaplan-Meier estimator.

But in survival analysis we are not so much interested in survival time as in what influences this survival time like taking a drug, smoking, having antecedents... The Cox proportional hazards regression model allows us to do so the new hazard function does not depend only on the time but also on some factors that may influence survival time. We note the new hazard function as follows:

$$h(t, X) = h_0(t) \exp(X\beta)$$

$h_0(t)$ is the baseline hazard depending only on the time, X is the vector of factors of an individual (note that those factors do not depend on time) and β is the vector of the coefficients associated with those factors. For instance, if we have *age* and *sex* as two explanatory factors, the coefficients associated with those factors β_1 and β_2, and β_0 the intercept,the instantaneous potential at time t for getting the event, given survival up to time t and *age* and *sex* is:

$$h(t, X) = h_0(t) exp(\beta_0 + \beta_1.age + \beta_2.sex)$$

In order to estimate β, we use the partial log-likelihood. First let's understand the partial likelihood:

$$PL(\beta) = \prod_{i:\delta_i=1} \frac{exp(\sum_{j=1}^{p} x_{ij}\beta_j)}{\sum_{i':y_{i'} \geq y_i} exp(\sum_{j=1}^{p} x_{i'j}\beta_j)}$$

Where δ is the set of individuals for which an event has occurred at time y_i, and the i' are all the at risk individuals at time y_i. We can thus interpret the fraction as the probability of the individual who died to actually die, as the probability of observing what we observed. The partial likelihood is just the product for each individual died at time y_i. Finally we obtain the partial log-likelihood by taking the log of the partial likelihood:

$$L(\beta) = \sum_{i:\delta_i=1} X_i^T \beta - \log(\sum_{i':y_{i'} \geq y_i} exp(X_{i'}^T \beta))$$

By optimizing this function over β we will maximize the probability of observing the actual observations.

The authors are interesting in conducting survival analyses with million of SNPs, so the optimization of the partial likelihood takes an important number of iterations in the original R-package *survival* (coxph, Therneau and Grambsch, 2000). In order to reduce the number of iterations, the authors decided to modify the *survival* R-package : first the Cox proportional hazard model is fitted with all the non-genetic covariates (i.e. the SNP is not included). Then those estimate parameters are used as initial points for fitting the model with the SNP covariate.

Datasets

`gwasurvivr` provides multiple functions to run a survival analysis. The functions accept a file containing SNPs in different formats which is indicated by the respective name, e.g. `.gds` format is run by the method `gdsCoxSurv`. Each function also needs a file which contains the meta data of patients, like sex, survival time, vital status or age, to fit the survival model.

An example dataset is provided in the extdata folder which is automatically installed with `gwasurvivr` containing all supported dataformats, including `.gds`, `IMPUTE2`, `.vcf` or related formats from the Michigan or Sanger imputation Server or the, `.bed` files or related formats generated by the programme `PLINK` or its successor `PLINK2`.

Results

Benchmarking

The performance analysis was conducted by benchmarking `gwasurvivr` against other software tools (`genipe`, `SurvivalGWAS_SV`, and `GWASTools`) in terms of runtime and scalability. Developers of the package visualized the results of the benchmarking experiments in the original paper, which are presented on the picture below consisting of figures A, B, C, and D.

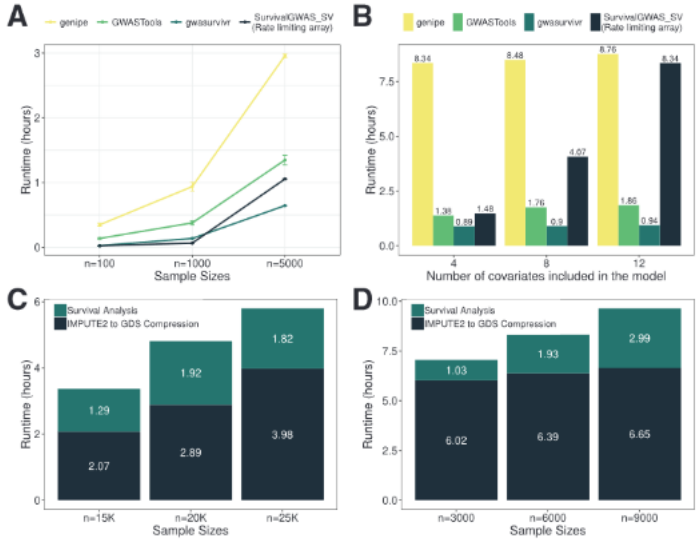

Figure 2: Benchmarking results of the scalability and runtime comparisons of 'gwasurvivr package.'

The performance of gwasurvivr was benchmarked under different conditions, including varying sample sizes, SNP numbers, and the number of covariates (4, 8, 12). It is then compared with other existing software for genome-wide survival analysis.

Analyses were run with identical CPU constraints of 1 node and 8 cores. All runtimes are for 100 batched jobs. The benchmarking experiments included a GWAS with approximately 6 million SNPs for 3000, 6000, and 9000 samples.

The study's findings are illustrated through four key figures. Figure A provides a comprehensive comparison of runtime for survival analyses across various sample sizes (100, 1000, 5000) and 100,000 SNPs, highlighting `gwasurvivr`'s superior performance in most cases, except for scenarios where it lagged slightly behind `SurvivalGWAS_SV` with an N of 1000. In Figure B, the impact of covariate numbers (5, 8, or 12) on runtime

is explored, revealing minimal effects on `gwasurvivr`'s performance compared to other software. Figure C explores how `gwasurvivr`'s performance changes due to compression from IMPUTE2 format to GDS and compares the runtimes obtained by the `SurvivalGWAS_SV` package (Figure D). It's clear that `gwasurvivr` clearly outperforms its main competitor consdiering this task.

The computational efficiency of the package origins from adapting highly efficient `parallel R` package for internal parallelization to fit the Cox PH models. Linux/OS X users can run analyses on a prespecified number of cores by setting the option in the `R` session using command `options("gwasurvivr.cores"=detectCores())`. This option should be defined in the R session before running any of the `gwasurvivr` functions. Unfortunately this option is not available to Windows users, but they can set up cluster object to speed up computing using `makeCluster(detectCores())` syntax.

Gwasurvivr is highlighted as significantly faster and more scalable when dealing with increasing sample sizes, the number of SNPs, and covariates. It is also more flexible and offers different input and output options. Gwasurvivr is noted for overcoming memory limitations typically associated with R by processing data in subsets, enabling genome-wide survival analyses on standard laptop computers. Gwasurvivr emerges as a favorable option for researchers, especially when confronted with large datasets and significant computational requirements. In second part of results analysis we will discuss different use cases for different functions (or data types) and we will test it on the available data. First analyzed data format comes from Michigan Imputation Server.

Use cases

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`). These `vcf.gz` files with addition of `.txt` files representing phenotype are used as an input for cox regression in `gwasurvivr` package.

The exemplary data can be extracted from the package using `system.file` function in according manner.

```
vcf.file <- system.file(package="gwasurvivr",
                        "extdata",
                        "michigan.chr14.dose.vcf.gz")
pheno.fl <- system.file(package="gwasurvivr",
                        "extdata",
                        "simulated_pheno.txt")
```

Simulated phenotype file used as covariate file during regression can be represented in the table as shown below.

Table 1: Simulated phenotype feature dataframe.

ID_1	ID_2	event	time	age	DrugTxYes	sex	group
1	SAMP1	0	12.00	33.93	0	male	control
2	SAMP2	1	7.61	58.71	1	male	experimental
3	SAMP3	0	12.00	39.38	0	female	control
4	SAMP4	0	4.30	38.85	0	male	control
5	SAMP5	0	12.00	43.58	0	male	experimental
6	SAMP6	1	2.60	57.74	0	male	control

Now using phenotype file as covariate file and given loaded VCF file we can run `michiganCoxSurv` function, which is a wrapper for Cox regression model (before running the function, `sex` column was encoded into binary format).

```
michiganCoxSurv(vcf.file=vcf.file,
                covariate.file=pheno.file,
                id.column="ID_2",
                time.to.event="time",
                event="event",
                covariates=c("age", "SexFemale", "DrugTxYes"),
                inter.term=NULL, #interaction term inclusion
                print.covs="only", #defines printing of covariates' statistics
                out.file="michigan_only",
                r2.filter=0.3, #imputation quality score filter
                maf.filter=0.005, #filter for minor allele frequency
                chunk.size=100, #number of variants to proceed per thread
                verbose=F,
                clusterObj=NULL) #for setting up cluster for computations
```

Functions saves the outputed model with the `.coxph` extension as a seperate file as well as SNPs removed (due to low variance or user defined thresholds) in the `.snps_removed` file. Accessed results of the performed regression are showcased below (`print covs = "only"` was chosen as a printing option, which omits some covariates‘ characheristics).

Table 2: Results of the survival analysis using Cox regression model with SNP covariates.

RSID	rs34919020	rs8005305	rs757545375
TYPED	FALSE	FALSE	FALSE
CHR	14	14	14
POS	19459185	20095842	20097287
REF	C	G	A
ALT	T	T	G
AF	0.301263	0.514583	0.519787
MAF	0.301263	0.485417	0.480213
SAMP_FREQ_ALT	0.3428	0.5022	0.5110
SAMP_MAF	0.3428	0.4978	0.4890
R2	0.551952	0.479015	0.480693
ER2	NA	NA	NA
PVALUE	0.2934544	0.3238959	0.2862329
HR	1.5085220	0.7233560	0.7046073
HR_lowerCI	0.7005469	0.3801063	0.3702421
HR_upperCI	3.248374	1.376573	1.340937
Z	1.0505737	-0.9864835	-1.0664221
COEF	0.4111304	-0.3238538	-0.3501147
SE.COEF	0.3913389	0.3282911	0.3283078
N	100	100	100
N.EVENT	42	42	42

To decipher most column names and extract knowledge from the output we included Table 3 in the appendix section, which explains the meaning of certain variables.

Looking at the printed output, column PVALUE holds results for significance test of the SNP covariates (rest of the results is omitted by the printing option). It can be stated that on level of significance 0.05, SNP covariates (all pvalues above 0.05) can be deemed as insignificant as $H_0 : coef_i = 0$. Drawn conclusion seems sensible as phenotype file was generated randomly.

Package offers also adding SNP covariate interactions during the modelling. Setting option `inter.term` to name(s) of the variable(s), in our case `DrugTxYes`, will include these interactions during training of the

regression models. Table 4 with all (`print.covs="all"`) information about covariates is attached in the appendix. Additionally pvalues for the significance test, hazard ratios and confidence intervals for them, Z scores, coefficents and their standard errors can be obtained for all the covariates.

Analyzing the full table of the results gives us a whole picture about performance of the covariates. Looking again at pvalues, now it's clear that only `age` can be deemed as significant in this statistical inference framework.

Additionally, creators of the package included three more modifications of Cox proportional hazard regression for different data types.

`sangerCoxSurv`is the function which handles data formats obtained from Sanger Imputation Server, which pre-phases typed genotypes using either SHAPEIT or EAGLE, imputes genotypes using PBWT algorithm and outputs a .vcf.gz file for each chromosome. This cox regression wrapper 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 output file. Syntax of the function and overall use cases are the same as in previously described `michiganCoxSurv`.

Another format which is handled by the package is IMPUTE2 generated output. IMPUTE2 is a genotype imputation and haplotype phasing program. It outputs 6 files for each chromosome chunk imputed, but only 2 of these files are required for analyses using `gwasurviv`. Loading and pre-processing of the genetic and phenotypic data is conducted as before. To perform survival analysis using IMPUTE2 the function arguments are very similar to `michiganCoxSurv` and `sangerCoxSurv`, however the function now takes a chromosome argument. This is needed to properly annotate the file output with the chromosome that these SNPs are in. Moreover, function `gdsCoxSurv` allows to performs survival analysis on genetic data, which was converted from IMPUTE2 format to GDS format.

Additionaly, authors are also mentioning wrapper `plinkCoxSurv`, which handles data in PLINK format (`.BED`, `.BIM` and `.FAM` files). Pre-processing, syntax and usage of the function is identical to previously described solutions.

Conclusions

`gwasurvivr`, an R package tailored for the analysis of survival outcomes in Genome-Wide Association Studies (GWAS), offers several notable advantages and improvements. It seamlessly integrates GWAS results with survival analysis, making it an accessible tool for researchers seeking to explore the genetic variants' influence on survival outcomes.

Its seems user-friendly at first glance, as use cases are described in the vignette, but syntax and storage of the output in the external file makes it more gimmicky. Moreover, output is just a table with some metrics not a proper model object, which makes it harder to do inference and integrate it with different software concerning survival analysis.

One of its main strengths is inclusion of handling diverse data format used for survival analysis inference, but then no visualization tools were added on top of cox regression wrappers.

Capacity to handle extensive datasets was the main focus of the developers. The goal was clearly achieved as the package is overall fast, has premium options for fast computing, parallelization and setting up clusters to speed up the fitting process.

In conclusion, `gwasurvivr` is a viable options for researchers trying to incorporate genom-wide studies into survival analysis framework due to fast compuation time and fairly easy syntax. However, it has some limitations, primarily focusing on compatibility with other survival analysis frameworks and requiring a working knowledge of R, data preparation, and potentially resource-intensive computations. Users should also keep an eye on updates and maintenance, as with any `R` package.

Appendix

Table 3: Decriptions of the output of the Cox regression fucntions from the `gsurvivr` package

RSID	SNP ID
TYPED	Imputation status: TRUE (SNP IS TYPED)/FALSE (SNP IS IMPUTED)
CHR	Chromosome number
POS	Genomic Position (BP)
REF	Reference Allele
ALT	Alternate Allele
AF	Minimac3 output Alternate Allele Frequency
MAF	Minimac3 output of Minor Allele Frequency
SAMP_FREQ_ALT	Alternate Allele frequency in sample being tested
SAMP_MAF	Minor allele frequency in sample being tested
R2	Imputation R2 score (minimac3 R^2)
ER2	Minimac3 ouput empirical R^2
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

Table 4: Full results of the survival analysis using Cox regression model with SNP covariates and interactions.

RSID	rs34919020	rs8005305	rs757545375
TYPED	FALSE	FALSE	FALSE
CHR	14	14	14
POS	19459185	20095842	20097287
REF	C	G	A
ALT	T	T	G
AF	0.301263	0.514583	0.519787
MAF	0.301263	0.485417	0.480213
SAMP_FREQ_ALT	0.3428	0.5022	0.5110
SAMP_MAF	0.3428	0.4978	0.4890
R2	0.551952	0.479015	0.480693
ER2	NA	NA	NA
PVALUE_INTER.TERM	0.5006863	0.6904295	0.6178695
PVALUE_SNP	0.7678200	0.3564361	0.2919175
PVALUE_age	2.668265e-11	5.951631e-12	3.980446e-12
PVALUE_SexFemale	0.7902958	0.7904299	0.8185354
PVALUE_DrugTxYes	0.8745114	0.7686765	0.6983304
HR_INTER.TERM	1.682057	1.317736	1.411713
HR_SNP	1.1739468	0.6123633	0.5732513
HR_age	1.187492	1.198431	1.198493
HR_SexFemale	1.093165	1.094286	1.080738
HR_DrugTxYes	0.8935289	0.7872530	0.7305965
HR_lowerCI_INTER.TERM	0.3702699	0.3389400	0.3642613
HR_lowerCI_SNP	0.4048147	0.2159322	0.2036739
HR_lowerCI_age	1.128963	1.138204	1.138729
HR_lowerCI_SexFemale	0.5669620	0.5630263	0.5567349
HR_lowerCI_DrugTxYes	0.2209743	0.1598800	0.1493771
HR_upperCI_INTER.TERM	7.641221	5.123111	5.471167
HR_upperCI_SNP	3.404400	1.736605	1.613447
HR_upperCI_age	1.249055	1.261844	1.261393
HR_upperCI_SexFemale	2.107743	2.126832	2.097937
HR_upperCI_DrugTxYes	3.613062	3.876454	3.573314
Z_INTER.TERM	0.6734103	0.3982723	0.4988721
Z_SNP	0.2952276	-0.9221773	-1.0539245
Z_age	6.663806	6.880803	6.937874
Z_SexFemale	0.2659265	0.2657524	0.2294291
Z_DrugTxYes	-0.1579307	-0.2941065	-0.3875753
COEF_INTER.TERM	0.5200172	0.2759148	0.3448041
COEF_SNP	0.1603714	-0.4904296	-0.5564311
COEF_age	0.1718436	0.1810129	0.1810646
COEF_SexFemale	0.08907727	0.09010244	0.07764425
COEF_DrugTxYes	-0.1125765	-0.2392056	-0.3138939
SE.COEF_INTER.TERM	0.7722144	0.6927791	0.6911673
SE.COEF_SNP	0.5432129	0.5318170	0.5279611
SE.COEF_age	0.02578761	0.02630694	0.02609800
SE.COEF_SexFemale	0.3349695	0.3390466	0.3384237
SE.COEF_DrugTxYes	0.7128225	0.8133298	0.8098915
n.sample	100	100	100
n.event	42	42	42