Supplementary Documentation for gwasurvivr: an R package for genome wide survival analysis

Abbas A Rizvi, Ezgi Karaesmen, Martin Morgan, Junke Wang, Michael Sovic, Theresa Hahn, Lara Sucheston-Campbell

October 9, 2018

Contents

`gwasurvivr` is an R package that can be used to conduct survival analysis (Cox proportional hazards model) on imputed GWAS data from either IMPUTE2 (Howie, et al., 2009) or VCF files from the Michigan and/or Sanger imputation servers. `gwasurvivr` can also be used on directly typed data in plink format (`.bed`, `.bim` and `.fam` files).

Herein, we detail our implementation of the Cox model, generation of the simulated data and survival benchmarking and graphically report the correlation of `gwasurvivr` beta coefficient estimates, minor allele frequencies (MAF) and p-values with those produced from SurvivalGWAS_SV, genipe, and GWASTools.

To reproduce the data and create Figure 1 and Supplementary Figures 2-4, the data is available on the gwasurvivr manuscript repository. GitHub Large File Storage (LFS).

To clone the whole repository:

```
git lfs clone https://github.com/suchestoncampbelllab/gwasurvivr_manuscript.git
```

Table 1: **Supplementary Table 1**: Description of arguments that are built manually in gwasurvivr and passed directly to survival::coxph.fit, bypassing the main survival::coxph function

variables	description
X	matrix of predictors
Y	Surv object
STRATA	vector containing stratification, we set to NULL
OFFSET	offset vector, we set to NULL
INIT	initial values for coefficients
CONTROL	result of a call to survival:::coxph.control
WEIGHTS	vector of weights that we set to NULL
METHOD	efron method used for handling ties
ROWNAMES	rownames that we set to NULL

Implementation of Survival Model in gwasurvivr

Modifying coxph

We decrease computation time by decreasing the number of Newton-Raphson iterations used to optimize the partial likelihood function in the Cox proportional hazard models. To do this, a survival model was fit using only non-genetic covariates (i.e. the SNP is not included and only covariates are fit); `survival::coxph` (Therneau and Grambsch, 2000) is modified such that gwasurvivr manually creates the objects found in the helper function (`survival::coxph.fit`) that fits the Cox model.

These variables are then passed to `survival::coxph.fit`.

Benchmarking with survival package

To assess if providing initial estimates from covariates versus using the survival function as implemented in the survival package improves computational time, we tested a dataset of 500 individuals at 7255 SNPs with 1, 2, or 3 covariates. These data are a subset of the simulated data described in detail below.

The helper function `gwasurvivr:::coxParam`, adjusted for this Supplementary documentation is labeled `gcoxph`. In `gcoxph_model.R` we fit the model without the SNP and the parameter estimates are then used as initial points for all subsequent models and applied over all SNPs in the dataset. If there were no covariates, the initial estimates would be null. The function `coxph_model.R` implements a `survival` model (survival package, Therneau and Grambsch, 2000) without using the optimization starting point obtained from including covariates in the model.

To test the package runtime over a pre-specified number of iterations and including 1, 2, or 3 covariates the `microbenchmark` package in R was used. The code for Supplementary Figure 1) is available.

By leveraging an initalization point from the analyses with covariates `gwasurvivr` (gcoxph) is several seconds faster than the survival analyses function as implemented in `survival` (coxph, Therneau and Grambsch, 2000) in R (**Supplementary Figure 1**). While this is a small test dataset, in practice this would be an appreciable difference when testing across several thousands of samples and millions of SNPS. In the `gwasurvivr` package, we opted to use `parallel::parApply` instead of `base::apply` as shown above to compute across multiple cores.

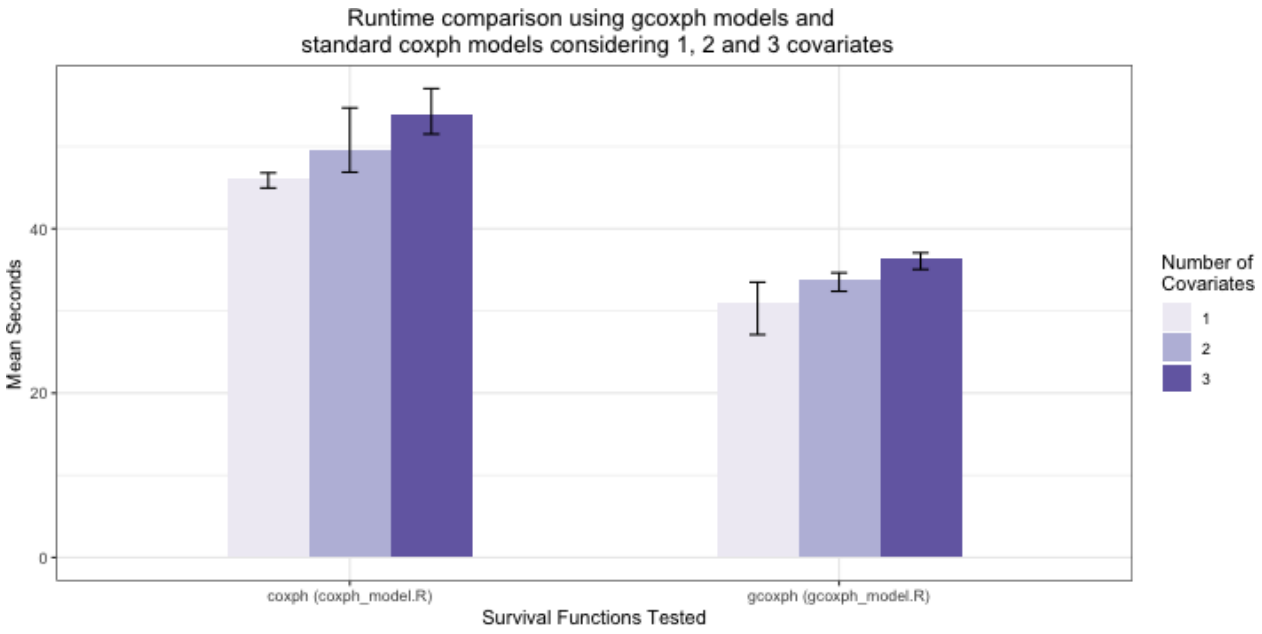

Figure 1: **Supplementary Figure 1:** The x-axis are the survival functions tests, coxph and gcoxph. The y-axis are mean seconds from three iterations of each function. The error bars represent the maximum and minimum run time from three iterations with the barplots showing the mean runtimes of either coxph or gcoxph for 1, 2, or 3 covariates. gcoxph, which fits initial points based on parameter estimates from covariates alone, runs on average faster than the traditional coxph model for 1, 2, and 3 non-genetic covariates and shows a decreasing min to max range as the number of covariates increase.

Computational Experiments

We used the University at Buffalo Computational Center for Research (UB CCR) academic cluster for our benchmarking analyses. Each analysis was run exclusively on node CPU-L5520 with the same system specifications, controlling the computational resources for each run. The UB CCR uses Simple Linux Utility for Resource Management (SLURM) scheduling for jobs. SLURM scripts to run the analyses were generated using shell scripts below. Benchmarking was performed using identical CPU constraints, 1 node (2.27 GHz Clock Rate) and 8 cores with 24 GB of RAM, on the University at Buffalo Center for Computational Research supercomputer. With the exception of the larger sample size tests, these were run using the same node but 12 CPUs. genipe (Lemieux Perreault, et al., 2016), SurvivalGWAS_SV (Syed, et al., 2017), and GWASTools (Gogarten, et al., 2012) were performed as specified by the authors on available online documentation. We performed the following benchmarking runtime experiments either against existing software or against time with varying N and SNP numbers that were performed:
Simulation 1. Compare gwasurvivr against genipe, GWASTools and SurvivalGWAS_SV - varying sample sizes (n=100, n=1000, n=5000) and 100,000 SNPs (m=100000) and 3 non-genetic covariates
Simulation 2. Comparison of gwasurvivr, genipe, GWASTools and SurvivalGWAS_SV with N=5,000 and 100,000 SNPs (m=100,000) with 4 covariates (age, drug treatment, sex and 1 PC), 8 covariates (age, drug treatment, sex and 5 PCs) and 12 covariates (age, drug treatment, sex and 9 PCs)
Simulation 3. Increasingly larger sample sizes (N=15K, 20K and 25K) tested on Chromosome 22
Simulation 4. Full autosomal GWAS with varying sample sizes (N=3K, 6K and 9K)

Simulating Genotypes and Phenotypes

Genotypes

HAPGENv2 (Su, et al., 2011) was used to generate simulated genetic datasets from 1000 Genomes Project CEU data (NCBI Build 36) for all benchmarking experiments. To replicate simulations the 1000 Genomes Project CEU data should be downloaded in its entirety (only a subset is available on our GitHub repo). The code for all HAPGENv2 simulations are available on our GitHub.

Phenotypes

For each sample size tested, survival events (alive/dead) were simulated as two separate datasets. For the dead dataset, time to event and covariates were simulated using a normal distribution. For the alive dataset, time was simulated by randomly sampling weighted probabilities for times to simulate few samples being censored, covariates were simulated from a normal distribution. Principal components (PCs) were simulated using random normal distributions with decreasing variance for each additional PC. Furthermore, the `.sample` file from IMPUTE2 includes 4 columns (ID_1, ID_2, missing, and sex) which link individuals with their respective genotypes. For SurvivalGWAS_SV and GWASTools, the simulated phenotypes were appended to column 5 onward in the `.sample` file.

The following genotypes and phenotypes were simulated:
Simulations 1 and 2. Subset of chromosome 18 for 100,000 SNPs 1) varying N and 3 covariates done in triplicate and 2) with 4, 8 and 12 covariates

- genotype code
- phenotype code
- PCs phenotype code

Simulation 3. chromosome 22 (~117,000 SNPs) for larger sample sizes (N=15000-25000)

- genotype code

Simulation 4. Full GWAS for N=9000 (the smaller subsets were just parsed from the data during analyses)

- genotype code
- phenotype code
- simulate sample ids code

Benchmarking with other software capable of GWAS coxph survival analysis

We benchmarked `gwasurvivr` with GWAS survival analysis software, genipe, SurvivalGWAS_SV and GWASTools using simulated phenotype and genotype data. Genetic data were formatted as output from IMPUTE2 software (.GEN). Genipe, SurvivalGWAS_SV, and GWASTools do not directly take VCF data output from Sanger or Michigan imputation servers. SurvivalGWAS_SV does accept VCF files as an input but uncompressed and not explicity the same format that Sanger and Michigan imputation servers output, rendering additional steps to be taken. The benchmarking with IMPUTE2 was done for (1) varying sample sizes and (2) varying additional non-genetic covariates. Both are described here.

gwasurvivr

The following scripts were used to run gwasurvivr using `impute2CoxSurv`. These R scripts are run using a shell script (SLURM script) that pass the system variables into R (facilitated by the R package `batch`).

N=100, 1000 and 5000 with M=100K SNPs + 3 non-genetic covariates in triplicate:
- `run_gwasurvivr.R`
- `create_gwasurvivr_scripts.sh`

N=5,000 and M=100K with 4, 8 and 12 covariates:
- `run_gwasurvivr_covs.R`
- `gwasurvivr_cov4.sh`
- `gwasurvivr_cov8.sh`
- `gwasurvivr_cov12.sh`

genipe

For genipe, the shell scripts was used to generate SLURM scripts for genipe and each sample and SNP set. We used specific settings for OPENBLAS that are suggested on genipe's website to ensure that computational efficiency was maximized.

varying sample sizes + 3 non-genetic covariates:
- `create_genipe_scripts.sh`

additional covariates:
- `genipe_cov4.sh`
- `genipe_cov8.sh`
- `genipe_cov12.sh`

SurvivalGWAS_SV

To maximize the performance of SurvivalGWAS_SV, these jobs were run using "array" jobs as recommended by the authors. An [example batch script]((https://www.liverpool.ac.uk/media/livacuk/instituteoftranslationalmedicine/biostats/batchexample.sh), provided in the SurvivalGWAS_SV documentation, was converted from PBS to SLURM. 24GB of ram was not needed on all runs, however was used to ensure each run remained uniform. The jobs were split into array sets of 1000 SNPs for m=100,000, totaling 100 batched jobs in a single array. We define rate-limiting array as the array index that had the longest runtime. In the main manuscript, we report SurvivalGWAS_SV runtimes as the rate-limiting array

runtime. This is an important caveat and bears consideration when using SurvivalGWAS_SV. Depending on availability on the computing cluster, the analyses could be completed as quickly as the longest individual array job (which is shown in Figure 1), or potentially the entire runtime could be equal to the summation runtime of all of the array indices if these cannot be run simultaneously (or if there are failures with any of the array indices). The shell script below was used to generate SLURM scripts for SurvivalGWAS_SV for each sample and SNP set.

N=100, 1000 and 5000 with M=100K SNPs + 3 non-genetic covariates in triplicate:
- `create_sv_scripts.sh`

N=5,000 and M=100K with 4, 8 and 12 covariates:
- `sv_cov4.sh`
- `sv_cov8.sh`
- `sv_cov12.sh`

GWASTools

For GWASTools, the files are converted to GDS format and survival is run using `GWASTools::assocCoxPH` within `gwastools_survival.R`. The R script was passed to the SLURM scripts using the script `create_gwastools_scripts.sh`. GWASTools does not run in parallel across multiple cores on a single computing processor internally, however experienced users could code this themselves.

N=100, 1000 and 5000 with M=100K SNPs + 3 non-genetic covariates in triplicate:
- `gwastools_survival.R`
- `create_gwastools_scripts.sh`

N=5,000 and M=100K with 4, 8 and 12 covariates:
- `gwastools_survival_covs.R`
- `gwastools_cov4.sh`
- `gwastools_cov8.sh`
- `gwastools_cov12.sh`

Runtime large N chromosomes to test size limitations

We tested chr22 with different sample sizes of N=15,000; N=20,000; N=25,000 using `gwasurvivr::impute2CoxSurv`. The code for all of the runs can be found here. The R script called from the shell scripts to run these analyses is labeled `run_bigNs.R`.

Runtime GWAS with different sample sizes

We performed three GWAS (chr1-chr22) with different sample sizes (n=3000; n=6000; n=9000) using `gwasurvivr::impute2CoxSurv`. The code to simulate the GWAS is available on our repository. The R script used to run these analyses is `run_fullgwas.R`. The shell script run these scripts on SLURM can be found here.

Time Plots

Figure 1

To generate Figure 1 times from the computation runtime were pulled from SLURM log files and collected using the perl scripts, which can be found in each of the log folders on our manuscript GitHub repository, compiled and Figure 1 was generated using the R code shown here.

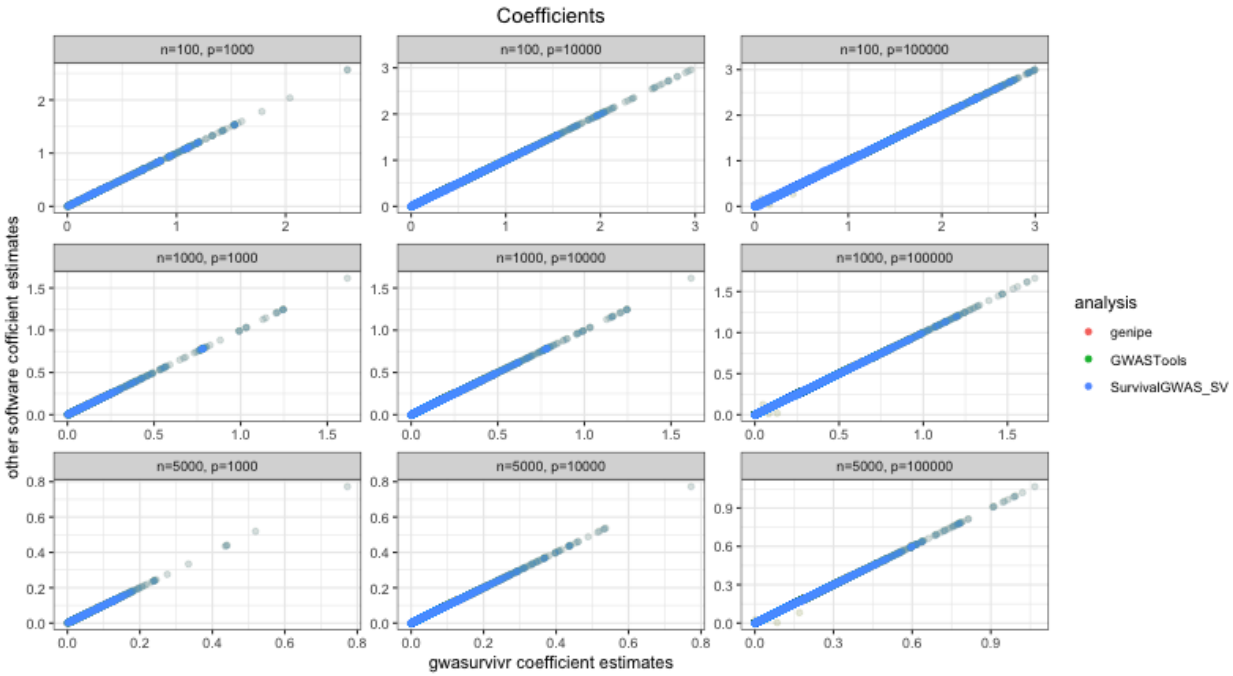

Figure 2: **Supplementary Figure 2.** Correlation of beta coefficient estimates between gwasurvivr and other software. The x-axis are gwasurvivr coefficients and the y-axis are coefficient estimates from the other software. The points are colored to indicate the software being used where red is genipe, green is GWASTools alone, and blue is SurvivalGWAS_SV. Each panel represents different simulations with varying sample sizes (n) and number of SNPs (p) included in the provided imputed genetic dataset. The estimates are near perfectly correlated and thus not all colors are visible in each plot.

Diagnostic Plots

Supplementary Figures 2, 3 and 4 below show the correlation of the coefficient estimates, minor allele frequency and p-values, respectively between gwasurvivr and all other software assessed. The correlations show excellent agreement. The R code used to generate supplemental figures 2-4 can be found here.

Coefficient Estimates

Minor Allele Frequency (MAF)

P-value Estimates

Full GWAS Runtimes

gwasurvivr calculations

Minor Allele Frequency (MAF)

For a given SNP with alleles A and B, where n_{AB} and n_{BB} are the number of individuals with AB and BB genotype respectively, and N is the sample size, the expected allele frequency of allele B ($freq_B$) be can be

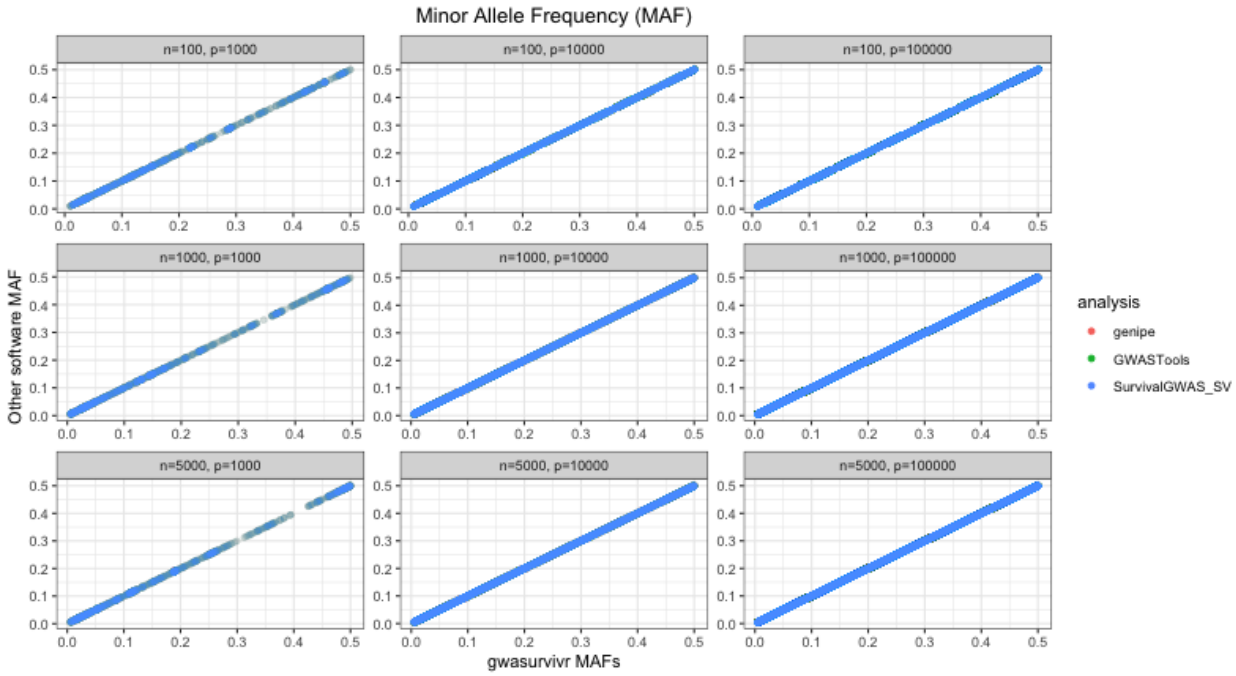

Figure 3: **Supplementary Figure 3.** Correlation of estimated minor allele frequency (MAF) between gwasurvivr and other GWAS survival analysis software. The x-axis are gwasurvivr MAFs and the y-axis are MAFs from the other software. The points are colored to indicate the software being used where red is genipe, green is GWASTools alone, and blue is SurvivalGWAS_SV. Each panel represents a different simulation with varying sample sizes (n) and number of SNPs (p) included in the provided imputed genetic dataset. The estimates are near perfectly correlated and thus not all colors are visible in each plot.

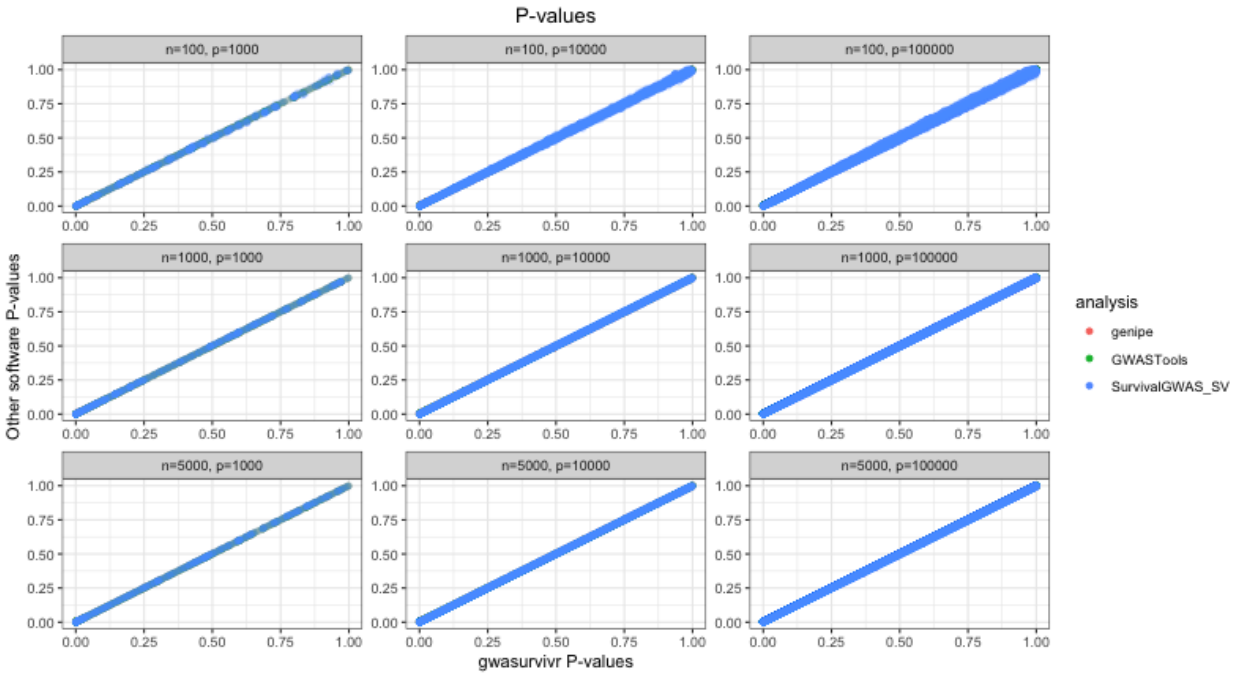

Figure 4: **Supplementary Figure 4.** Correlation of p-values between gwasurvivr and other GWAS survival analysis software. The x-axis are gwasurvivr p-values and the y-axis are p-values from the other software. The points are colored to indicate the software being used where red is genipe, green is GWASTools alone, and blue is SurvivalGWAS_SV. Each panel represents different simulations with varying sample sizes (n) and number of SNPs (p) included in the provided imputed genetic dataset. The estimates are near perfectly correlated and thus not all colors are visible in each plot.

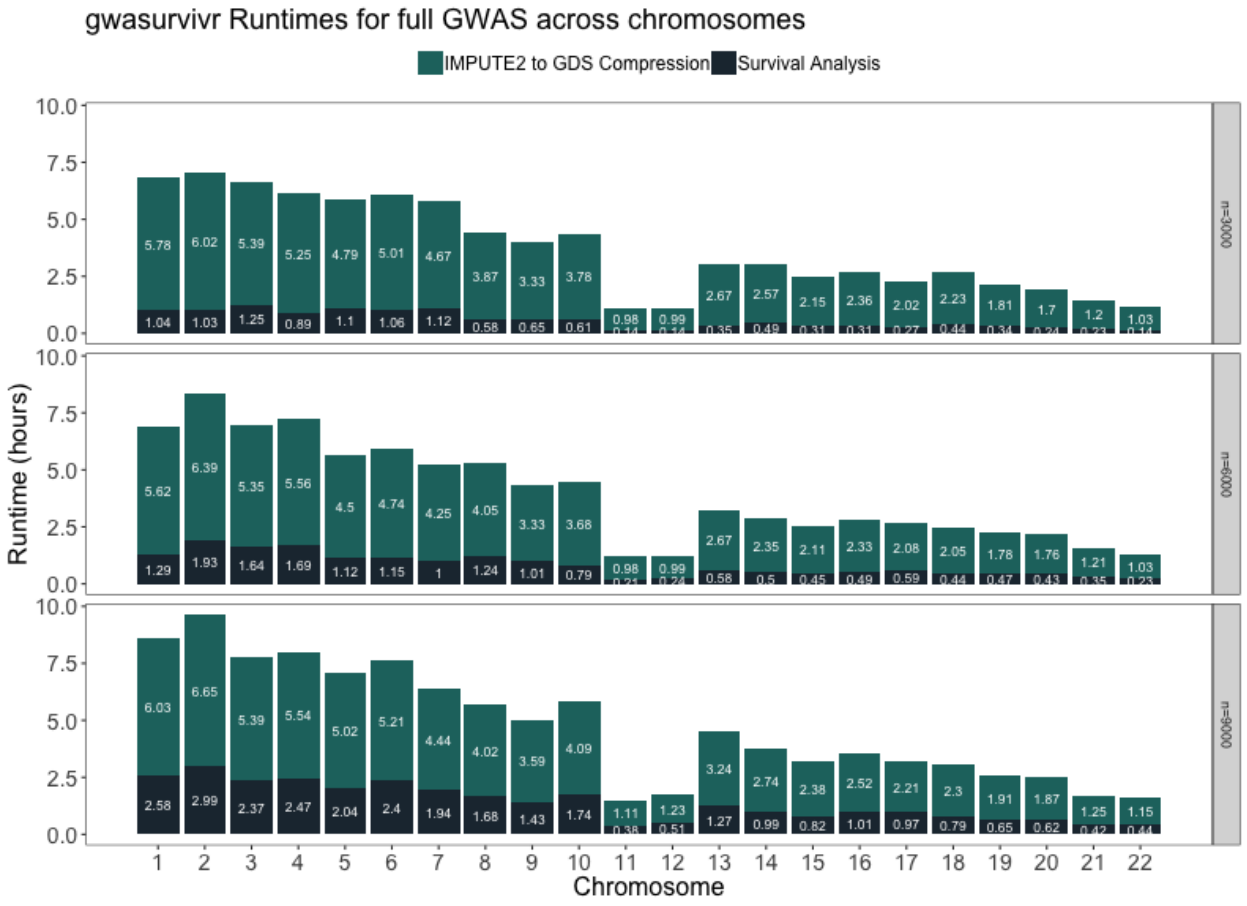

Figure 5: **Supplementary Figure 5.** Full GWAS run times for all chromosomes using gwasurvivr impute2CoxSurv. The three panels represent different sample sizes of N=3000 (top), N=6000 (middle) and N=9000 (bottom). The dark blue shaded area is elapsed time for compressing IMPUTE2 to GDS format and the dark green shaded areas are the computational time to run the survival analysis alone. On a computing cluster, each chromosome can be scheduled to run as individual jobs for best performance. Each GWAS was run on the UB CCR supercomputer with on the same node with 24 GB of RAM and 4 CPUs per node.

calculated as:

$$freq_B = \frac{n_{AB} + 2n_{BB}}{2N}$$

For individual i, the allele dosage of SNP j (D_{ij}) with alleles A and B, where allele B is the effect allele and p_{AB} and p_{BB} are the posterior genotype probabilities as computed by the imputation, is calculated as:

$$D_{ij} = p_{AB_{ij}} + 2 \cdot p_{BB_{ij}}$$

For SNP j The estimated allele frequency of an effect allele B (θ_{B_j}) can therefore be calculated as:

$$\theta_{B_j} = \frac{\sum_{i=1}^{N} D_{ij}}{2N}$$

This was coded in R as follows:

```
# calculate MAF
# genotypes variable is a matrix of dosages,
## where each column is a sample and each row is a SNP
exp_freq_A1 <- round(matrixStats::rowMeans2(genotypes)*0.5,4)
MAF <- ifelse(exp_freq_A1 > 0.5,
                  1-exp_freq_A1,
                  exp_freq_A1)
```

Imputation quality metric

Michigan Imputation Server

For the Michigan imputation server, imputation is performed using the minimac3 algorithm (Das et al., 2016). minimac3 computes and outputs an imputation quality metric known as R^2. R^2 is the estimated value of the squared correlation betwen imputed genotypes and true, unobserved genotypes (Das et al, 2016). The R^2 value is extracted directly from the Michigan imputation output VCF in `gwasurvivr::michiganCoxSurv`

Sanger Imputation Server

For the Sanger imputation server, we grab the `INFO` field directly from the VCF file in `gwasurvivr::sangerCoxSurv`. The INFO field is the IMPUTE2 (Howie, et al., 2009) score as calculated by the `bcftools + impute-info` plugin from posterior genotype probabilities (McCarthy et al., 2016).

IMPUTE2 Imputation

The INFO score for IMPUTE2 (Howie, et al., 2009) results are not calculated in `gwasurvivr` internally, instead we use the INFO scores that are provided in a separate file after performing imputation (`.info` file). Users select SNPs from the `.info` file to remove based on preferred criterion (ie INFO $< .8$) these are then used in the argument `exclude.snps` in `impute2CoxSurv` to filter out the SNPs prior to analysis.

References

Das S, Forer L, Schönherr S, Sidore C, Locke AE et al. Next-generation genotype imputation service and methods. Nature Genetics 2016; 48, 1284–1287 (2016) doi:10.1038/ng.3656%5B1%5D

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

McCarthy, S et al. (2016) A reference panel of 64,976 haplotypes for genotype imputation, Nature Genetics. 48(10):1279-83 [27548312]

Su, Z., Marchini, J. and Donnelly, P. HAPGEN2: simulation of multiple disease SNPs. Bioinformatics 2011;27(16):2304-2305.

Therneau, T.M. and Grambsch, P.M. Modeling Survival Data: Extending the Cox Model. Springer; 2000.