**Guide for genetic load analysis using Genomic Evolutionary Rate Profiling**

**Rebecca S. Taylor,1,3,4\* Micheline Manseau,1 Peng Lui 1 and Paul J. Wilson2**

1Landscape Science and Technology, Environment and Climate Change Canada, Colonel By Drive, Ottawa, K1S 5B6, Canada

2Biology Department, Trent University, East Bank Drive, Peterborough, K9L 1Z8, Canada

3Technical contact

4Lead contact

\*Correspondence: rebecca.taylor@ec.gc.ca

**SUMMARY**

This protocol describes how to undertake an analysis of genetic load in wildlife species using Genomic Evolutionary Rate Profiling (GERP). Focusing on caribou (*Rangifer tarandus*), we detail the steps involved including automation of input file production, as well as make available a modified version of the GERP program which creates more user-friendly outputs and streamlines the extraction of sites of interest from a VCF file as known derived alleles using three outgroup species to enable the measurement of genetic load. For complete details on the use and execution of this protocol, please refer to Taylor et al.1

**BEFORE YOU BEGIN**

The protocol describes how genetic load was calculated in a study of caribou but can be applied to any species as long as appropriate taxa are used for the evolutionary rate profiling steps. We assume that an appropriately filtered variant calling format (VCF) file with high quality genotypes has been made from genomes of the species where genetic load is being measured. This protocol is designed for Linux and was tested on Linux CentOS 7, but may also work on MacOSX and WSL if the necessary software is installed. Our custom R script to extract derived SNPs from a VCF file requires the following R packages: vcfR, tidyverse, data.table, magrittr, and parallel.

KEY RESOURCES TABLE

|  |  |  |
| --- | --- | --- |
| REAGENT or RESOURCE | SOURCE | IDENTIFIER |
| Deposited data | | |
| Reference genome sequences from multiple species | NCBI or other genome database | https://www.ncbi.nlm.nih.gov/data-hub/genome/?taxon=40674 |
| Software and algorithms | | |
| BBmap 38.86 | Bushnell et al.3 | https://github.com/BioInfoTools/BBMap |
| Java version 7 or above | Arnold et al.4 | www.java.com |
| BWA-MEM 0.7.17 | Li5 | <https://github.com/lh3/bwa> |
| Samtools 1.5 | Li et al.6 | <https://github.com/samtools/samtools> |
| htsbox | N/A | https://github.com/lh3/htsbox |
| BCFtools 1.19 | Li7 | <https://samtools.github.io/bcftools/> |
| genomic evolutionary rate profiling (GERP) | Davydov et al.2 | <https://github.com/BeckySTaylor/GERP/tree/main> |
| R statistical software 4.2.2 | R Core Team8 | <https://www.r-project.org/> |

**STEP-BY-STEP METHOD DETAILS  
Reference genomes of species used for evolutionary rate profiling**

**Timing: 2.0 h**

1. To find sites which are evolutionarily conserved in caribou, we compare the genomes of 48 cervid species. We first need a tree which has appropriate branch lengths reflecting the evolutionary distance between species which is done using TimeTree: <http://www.timetree.org/>. Copy and paste the list of the species being used and download the resulting tree in newick (.nwk) format. The species must have available reference genomes (Figure 1):
2. Next download the reference genome fasta file for each species, here is the link for available mammal genomes publicly available on the National Centre for Biotechnology Information (NCBI) where the genomes used for the caribou work were sourced: https://www.ncbi.nlm.nih.gov/data-hub/genome/?taxon=40674.

**Alignment to the caribou reference genome and production of input alignment files**

**Timing: Using 32 cores, 1 hour for indexing reference genome (if needed) and ~45 minutes per species included, so 37 hours for 48 species.**

1. To identify sites which are evolutionarily conserved in caribou, we align all species to our caribou reference genome and create a fasta file per scaffold containing the sequence for each of the species (a multi-fasta alignment) which is used as input for the GERP analysis. We have created a script named ‘Pipeline\_input\_alignment\_GERP.sh’ to run all of the steps with only four parameters needing to be input at the start, and is available on GitHub (<https://github.com/BeckySTaylor/GERP/tree/main>). The first step is to make a directory where you wish to run the script and put all of the downloaded reference genomes into that directory. Then, within the same directory, make a simple control file which contains the name of each genome file, followed by the species name after a tab space. If you wish to do a test run to check everything works, given how long it takes with a full dataset, we recommend running the pipeline using only a few species at first to ensure the correct outputs are made.

#Example control file, ensure not to use a space in the species name

GCF\_002863925.1\_EquCab3.0\_genomic.fasta Equus\_caballus

GCF\_002288925.2\_Delphinapterus\_leucas.fasta Delphinapterus\_leucas

GCF\_011762595.1\_Tursiops\_truncatus.fasta Tursiops\_truncatus

….

1. Then run the bash script provided on the GitHub page, for which we have provided explanations of the settings for each step below. At the top within the ‘Analysis Options’ section, you must input the path to the directory where the analysis will run and where the genomes and control file are stored, the name of the control file, and the name of the reference genome fasta file that will be used (for your focal species). If you wish to use the caribou reference genome, which is the example included below, you can remove the ‘#’ where indicated to download and unzip it. If you already have a bwa indexed version of the focal reference genome you can also put those files into the directory and put a ‘#’ next to the ‘bwa index’ line to save run time. You must also input the number of chromosomes (or scaffolds) for your reference genome that you wish to use. Note the number starts at 0 so if you want to use 35 chromosomes, you must enter ‘34’.
2. Input parameter description:

Step 2.2 to generate fastq files from the downloaded genomes using the reformat.sh script included with BBmap:

qfake - quality value used for fasta to fastq reformatting

fastareadlen - break fasta files into reads of at most this length

qout - ASCII offset for output quality, 64 for Illumina

addcolon - append ' 1:' and ' 2:' to read names, if not already present.

trimreaddescription - trim the names of reads after the first whitespace

int - determines whether input file is considered interleaved

Step 2.3 to align to the reference genome and generate a BAM file, removing reads aligning to more than one genomic location as well as supplementary reads:

-t – number of threads to use

-B - mismatch penalty

-O - gap open penalties for deletions and insertions

-F will remove reads with a particular flag, in this case the flag 2048 will remove supplementary reads

-b will ensure output is in BAM format

-q will remove reads below a chosen quality, removing very low-quality reads here should remove reads that are not uniquely mapped.

-h will keep the header row in the output file.

Step 2.4 to convert each bam into a fasta file and then split into one fasta file per scaffold. The script also renames the header line of each resulting fasta file to contain the species name instead of the scaffold name, otherwise they are all the same across species.

-f - reference genome

-Q minimum base quality

-q minimum mapping quality

-l minimum query length

-s drop alleles with depth less than this value

-R random allele fasta

Step 3 to concatenate each of the desired scaffolds and reformat so one multi-alignment fasta file per scaffold, containing all of the species one per line, is created. Note that scaffold one is number 0, scaffold 2 is number 1 etc. The resulting files are used as input for the GERP analysis.

**Run GERPcol function in GERP++**

**Timing: 40 min per scaffold**

The next step involves running the GERPcol function2 on each scaffold to output the rejected substitution (RS) scores for evolutionary conservation at each site. The goal of the analysis, for being able to measure genetic load, is to find sites with high conservation scores. The assumption is that a mutation at a site which is highly conserved over the phylogenetic tree (and thus millions of years of evolution) is likely to be deleterious. Finding how many sites which have mutations at these highly derived locations can give a measure of genetic load1,2. It is best to exclude the focal species, here caribou, from the analysis as this can lead to biases. However, this leads to missing data in the alignment which makes it difficult to interpret the output files which, in the original version of the program, don’t print which site the score pertains to. We modified the code for the gerpcol function in a couple of ways to make the outputs more user friendly and to help with extracting derived alleles of the evolutionarily conserved sites from the caribou genomes in VCF format, the important next step for getting a measure of genetic load in our re-sequenced individuals. Our modified script prints out the position for each score, as well as the allele for three specified sister species to enable the detection of derived alleles in caribou (see below). Here we output the alleles for the white-tailed deer, the moose, and the red deer, the closest species to caribou in the multi-species alignment, used to determine the ancestral alleles for each site. The modified version of the program is best downloaded as a tar file (named ‘gerp\_modified.tar.gz’) from:<https://github.com/BeckySTaylor/GERP/tree/main> . However, we describe the use of the GERPcol script in detail here.

Firstly, it is crucial to calculate the appropriate input parameters. As the newick file downloaded from TimeTree has the tree branch lengths in how many millions of years they have been separated (e.g., 4 for 4 million years), the numbers need to be scaled as GERPcol requires them to be substitutions per site. Given that the average mammal mutation rate is 2.2 × 10−9 per base pair per year3, we can calculate the input parameter to be 0.0022 mutations per million years on average at a site. If working on a different taxonomic group, a different value should be used.

The program also needs a transition/transversion rate (-r parameter), the default is 2 but we can also calculate this easily from a VCF file.

1. Calculate the transition/transversion ratio using BCFtools

>bcftools stats Input\_VCF.recode.vcf.gz

For our caribou genomes this comes to 2.06 (this value should be between 2.0 and 2.1 in a mammal species)4. The GERPcol function can now be run for each scaffold. The first species listed after ‘-e’ will be the specified sister species used for the analysis and should be written as named in the alignment file. It will print the alleles at all positions for the other two specified species as well, listed with a dash (-) in between, but this is purely for the next step to extract derived alleles where we want to compare against three outgroup species (see below).

1. Compile and run the modified GERPcol function

#Decompress the downloaded GERP tar file in desired location, and then compile:

>cd gerp

>make clean && make

#Then the program is ready to run, here for scaffold 1, the ‘-a’ parameter specifies that the input alignment is in mfa format

>./ gerpcol -t TimeTree\_species\_website.nwk -s 0.0022 -f Scaffold1\_GERP\_formatted.mfa -a -e Odocoileus\_virginianus-Alces\_alces-Cervus\_elaphus -v -r 2.06

The function will output 6 columns of data (including a header row). The first is the position on the scaffold ‘Pos’, V1 values are the calculated neutral rate for each site which will vary depending on the number of species in your tree with missing data. The V2 values are the calculated RS scores. A value of -1 indicates too many species in the alignment have missing data. Then the next three columns are the alleles for the three specific sister species which is to facilitate the next step to extract derived alleles (below). Table 1 shows output for six sites as an example.

**Table 1. Example of a few lines of output from the modified GERPcol function from scaffold 34**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pos | V1 | V2 | Odocoileus\_virginianus | Alces\_alces | Cervus\_elaphus |
| 48417 | 0.0805 | 0.0805 | G | G | G |
| 48418 | 0.471 | 0.471 | A | A | A |
| 48419 | 0.471 | -0.941 | T | T | T |
| 48420 | 0.471 | 0.471 | T | T | T |
| 48421 | 0.471 | 0.471 | T | T | T |
| 48422 | 0.471 | 0.471 | T | T | T |

It is important to note that the range of possible scores from a GERP analysis varies depending on your input species and phylogenetic tree and so the score ranges are different between studies and not directly comparable. The larger the phylogenetic tree (with a longer evolutionary distance between taxa), the higher the possible range of scores. For this study, the maximum RS score is 2.48.

**Calculate genetic load**

**Timing: 15 minutes per scaffold**

To get a measure of genetic load in our re-sequenced caribou genomes (filtered into the commonly used VCF format), we pull out those derived alleles with RS scores of interest. For this, we use the alleles from the three sister species to infer the derived allele in caribou using our custom R script. The script pulls out alleles when they are not found in any of the three sister species, and thus are likely to be derived in caribou. Firstly, split the VCF file by scaffold as we are running the analysis for each separately. This is done using BCFtools.

1. Split the VCF file by scaffold, here for scaffold 1

# -r indicates which scaffold you wish to retain in the output

#-O indicates the desired output type, in this case the z will indicate a compressed VCF file as output

#-o indicated the desired name of the output VCF file

>bcftools view -r Scaffold\_1 -Oz -o Input\_VCF\_Scaffold1.vcf.gz Input\_VCF.recode.vcf.gz

For the caribou analysis, we compared the derived alleles at different score ranges to see if there was evidence of purging the most putatively deleterious alleles (i.e. SNPs at those sites with the highest evolutionary conservation scores). We therefore pulled out all derived alleles, all of those with a positive RS score, and all of those with a score over 2 which is at the top end of the range in this dataset. This can be customized in our R script (named ‘Derived\_alleles’) which is also available for download here:<https://github.com/BeckySTaylor/GERP/tree/main> .

1. Pull out derived alleles, given the three outgroup species, at sites with a high conservation score

#Run this R script inputting the path to the VCF file in the first line, the path to the GERP output file for the appropriate scaffold in the ‘sca=fread’ line, and the desired output file suffix in the ‘fwrite’ line. If multiple computing cores are available, input the desired number after ‘nThread=’ and after ‘mc.cores=’

library(vcfR)

library(tidyverse)

library(data.table)

library(magrittr)

library(parallel)

vcf <- read.vcfR("./Input\_VCF\_Scaffold1.vcf.gz")

vcf\_field\_names(vcf, tag = "FORMAT")

Z <- vcfR2tidy(vcf, format\_fields = c("GT", "DP"))

gt2 = Z$gt

gt2 %<>%

filter(!str\_detect(gt\_GT\_alleles, "\\\*")) %>%

mutate("ref" = str\_remove(gt\_GT\_alleles, "(/|\\|).$"),

"tar" = str\_remove(gt\_GT\_alleles, "^.(/|\\|)"));

sca = fread("./Scaffold15.fasta.rates", nThread = 8) %>%

mutate(Pos = Pos + 1) %>%

filter(V2 > 2)

sams = gt2$Indiv %>% unique();

out = parallel::mclapply(seq\_len(length(sams)),

FUN = function(i){

system(sprintf('echo "processing %s\n"', sams[i]));

gt2 %>%

filter(Indiv == sams[i]) %>%

select(POS, ref, tar) -> tmp;

sca %>%

filter(Pos %in% tmp$POS) -> tmp2;

left\_join(tmp2, tmp, by = c("Pos" = "POS")) %>%

filter((tar != Odocoileus\_virginianus &

tar != Alces\_alces &

tar != Cervus\_elaphus) |

(ref != Odocoileus\_virginianus &

ref != Alces\_alces &

ref != Cervus\_elaphus)) %>%

mutate("sam" = sams[i]) -> out;

fwrite(out, paste0(i, "\_Scaffold15\_over2.txt"));#write to drive

return(out);

}, mc.preschedule = F, mc.cores = 8)

The line ‘filter(V2 > 2)’ specifies the score range of interest, here specifying only those scores over 2. To get all positive scores this can be changed to ‘filter(V2 > 0)’, or for all derived alleles regardless of score, then change to ‘filter(V2 != -1)’ to include all but those that are missing data.

The three outgroup species are also specified in the script, twice per species. If using different outgroup species than these can also be changed. The script outputs one file per individual in CSV format listing the position on the scaffold, the V1 and V2 scores, the alleles for the three outgroup species, and the alleles for the caribou from the VCF file (ref and tar), as well as the sample ID (Table 2).

**Table 2. Example output from the derived alleles with a RS score over 2**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pos | V1 | V2 | Odocoileus\_virginianus | Alces\_alces | Cervus\_elaphus | ref | tar | sam |
| 191655 | 2.36 | 2.36 | C | C | C | C | T | 28575 |
| 191989 | 2.28 | 2.28 | C | C | C | C | T | 28575 |
| 301301 | 2.26 | 2.26 | G | G | G | G | C | 28575 |
| 306839 | 2.4 | 2.4 | C | C | C | C | A | 28575 |

**EXPECTED OUTCOMES**

The resulting CSV file gives the derived alleles, based on three outgroup species, at sites which have specified evolutionary conservation scores. These can be used to calculate measures of genetic load. For example, for each individual you can add the number of derived alleles in different score ranges, or similarly calculate the average RS score (V2 column) in different score ranges (Figure 2). The CSV files can easily be imported into R and manipulated as desired.

**LIMITATIONS**

This method detects sites in the genome with high evolutionary conservation scores across millions of years, and we thus assume that derived SNPs at these sites are deleterious. However, we do not know the function of these sites and so the derived alleles are only putatively deleterious. This method uses a lot of computing resources and produces a number of large files and so does require access to reasonable computational power such as from cloud computing or high-performance computers. It is also likely better to use another independent method to measure genetic load to check for a consistent pattern, for example using a genome annotation method if a high-quality annotation, preferably done using RNA sequencing data, is available.

**TROUBLESHOOTING**

**Problem 1**

Reference genome of focal species is highly fragmented with many small scaffolds.

**Potential solution**

Our bash script automates the process and so creating the input files for GERP is simple, although the run time may be affected with many scaffolds. The GERP analysis does have to be run once per scaffold, however.

**Problem 2**

The bash script fails due to not finding the needed commands.

**Potential solution**

It is important to download all programs listed in the table if not already done so, and it is also important to ensure they are accessible from the specified path which is input into the bash script.

**Problem 3**

You already have a multispecies alignment you wish to use as into GERPcol to run step 8.

**Potential solution**

It is fine to start our protocol at step 8 if you already have an alignment you wish to use. However, this does need to be split into one alignment per scaffold/chromosome as the output lists the position but not the scaffold information.

**Problem 4**

Error running modified GERPcol script during step 8 due to dependencies ‘not found’.

**Potential solution**

The GERPcol function is written in C++ and our modified version requires some dependencies to be available. These are usually already installed on Linux systems but if this error occurs ensure dependencies are downloaded: libm.so.6: version `GLIBC\_2.29', libc.so.6: version `GLIBC\_2.34', libstdc++.so.6: version `GLIBCXX\_3.4.20’, libstdc++.so.6: version `CXXABI\_1.3.9', libstdc++.so.6: version `GLIBCXX\_3.4.29', and libstdc++.so.6: version `GLIBCXX\_3.4.21'. It will list missing dependencies which need to be installed as an error message when you try to run the script. If the dependencies are there but still do not work, it is possible that the version being used is the issue. If so, a solution is to recompile the code with the commands ‘make clean && make’, and it is necessary to recompile the program on every new computer you run it on.

**Problem 5**

R script doesn’t read in VCF file correctly for step 10.

**Potential solution**

We optimized this script using our VCF file which was created using GATK4. We assume it will work from VCF files produced by other programs given that it is a standard file format, but this has not been tested.

**RESOURCE AVAILABILITY**

**Lead contact**

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Rebecca S. Taylor ([rebecca.taylor@ec.gc.ca](mailto:rebecca.taylor@ec.gc.ca)).

**Materials availability**

This study did not generate new unique reagents.

**Data and code availability**

All scripts and code are available at: <https://github.com/BeckySTaylor/Phylogenomic_Analyses/tree/main>

**ACKNOWLEDGMENTS**

We are thankful to the Shared Hierarchical Academic Research Computing Network (SHARCNET: www.sharcnet.ca), Compute Canada, and Amazon Cloud Computing for high-performance computing services. We are also thankful to Broderick Crosby for help with writing the bash script. Funding from this research was provided by the Genomic Applications Partnership Program of Genome Canada, Environment and Climate Change Canada, the Government of Canada's Genomics Research and Development Initiative (GRDI).

**AUTHOR CONTRIBUTIONS**

R.S.T. Developed the pipeline and wrote the manuscript, M.M. secured funding and edited the manuscript, P.L. wrote the scripts for the GERP analysis and ancestral allele detection and edited the manuscript, and P.J.W. secured funding and edited the manuscript.

**DECLARATION OF INTERESTS**

The authors declare no competing interests.

**REFERENCES**

1. Taylor, R.S., Manseau, M., Keobouasone, S., Lui, P., Solmundson. K., Kelly, A., Larter, N.C., Gambery, M., Schwantje, H., Thacker, C., et al. (2024). High genetic load without purging in caribou, a diverse species at risk. Curr. Biol. *34*, 1234-1246.
2. Davydov, E.V., Goode, D.L., Sirota, M., Cooper, G.M., Sidow, A., and Batzoglou, S. (2010). Identifying a high fraction of the human genome to be under selective constraint using GERP++. PLoS Comput. Biol. *6*, e1001025.
3. Bushnell, B., Rood, J. and Singer, E. (2017). BBMerge—Accurate paired shotgun read merging via overlap. PLoS ONE*,* *12*, e0185056.
4. Arnold, K., Gosling, J., and Holmes, D. (2005). *The Java programming language*. Addison Wesley Professional.
5. Li H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. Preprint at [arXiv:1303.3997v2](http://arxiv.org/abs/1303.3997).
6. Li, H., et al. (2009). The sequence alignment/map format and samtools. Bioinformatics *25*, 2078–2079.
7. Li, H. (2011). A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequence data. Bioinformatics, *27*, 2987-2993.
8. R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL: <https://www.R-project.org/>

**Figure Legends:**

**Figure 1. The TimeTree of the species used for the GERP analysis**.

**Figure 2. Individual genetic load.** Results from one caribou (ID 28575) showing the total numbers of all derived alleles, alleles with a positive score, and those with a score over two (A), as well as the average scores of all derived alleles, alleles with a positive score, and those with a score over two (B).