Estimating G_{min}

Juan Enciso

January 5, 2023

Follow the instructions to install the required software, prepare the data and run the script. Some lines in the code examples include a \ character at the end of each line; this character breaks single lines of code for making commands look more organised, you do not need to include it if you are typing your own line in the terminal. Although you can do these steps in your computer, I highly recommend doing this in **Compute Canada**.

1 Installing the required software

We will start by installing software that is required for the script to run. The software that we need in this case is `egglib`, a python module with lots of useful resources for population genetics/genomics. We need to do this within a virtual environment, in this case I named my virtual environment `egglib-env`, you can change the name if you want.

Here are the instructions for setting up the virtual environment, and for installing the required modules in this environment.

```
# 1. Load Python 3.8. Other versions may work
# but I have only tested this one so far
module load python/3.8

# 2. Create a Python virtual environment
# the name does not matter as long as you remember
# it. I named mine egglib-env
virtualenv --no-download egglib-env

# 3. Activate the environment you just created
# and only deactivate it after running the script
source egglib-env/bin/activate
```

```
# 4. In the active environment, upgrade pip
pip install --no-index --upgrade pip

# 5. Install the egglib module. This is the module
# that does a lot of the work with genotype data
pip install egglib

# 6. Install the numpy module.
pip install numpy --no-index
```

2 Preparing the input data

For running the script that estimates G_{min}, we need a `vcf` genotypes file that may be obtained with `stacks` or other programs. Remember to filter your genotypes file to retain the sites and individuals with high quality.

Egglib requires that the genotypes are sorted by coordinate in the vcf file. If you used stacks to produce your vcf it is likely that the genotypes are not sorted appropriately. We use the `SortVcf` tool from `picard` to produce a vcf sorted by genomic coordinate.

In addition, egglib requires a simpler vcf format than stacks provides, otherwise it will show errors. We need to exclude the genotype likelihoods field from each genotype record, so that the script can read the data without problems. We use the `annotate` program from `bcftools` for removing genotype likelihoods.

You can find below the code that does these tasks.

```
# 1. sort vcf by coordinate
# load picard tools
module load picard/2.26.3

# Run picard tools with the original vcf file after
# INPUT=, and the output file with the genotypes sorted
# by genomic coordinate after OUTPUT=...
java -jar $EBROOTPICARD/picard.jar SortVcf \
     INPUT=MOCB_GT_161122.vcf \
     OUTPUT=MOCB_GT_161122.sort.vcf

# 2. Get rid of genotype likelihoods
# load bcftools 1.16
```

```
module load bcftools/1.16

# run the annotate program as shown to get rid of the GL field
# from the vcf and re-direct the output to the new vcf file
# using the > operator
bcftools annotate -x FORMAT/GL MOCB_GT_161122.sort.vcf > \
         MOCB_GT_161122.snoGL.vcf
```

3 Running the script

With data ready, we can now run the script. Either copy the script file (`gmin_from_vcf.py`) to a known location using mobaXterm, or open a text editor in computecanada and copy and paste all the code from the script in a new text file. Before running the script we need to change the execution permissions (`+x`).

Note that we pass several pieces of information to the script (7 in total). The first four are **required in the order that they appear**, the last three are optional. The names of the files and populations described below correspond to a specific example ran by me. Your own files are going to be named differently.

- `MOCB_GT_161122.snoGL.vcf` is the vcf file
- `mocb_pops_allopatric2.txt` is the populations file that assigns individuals to populations, just as the populations file used in stacks. Note that you don't have to include all the individuals that are in the vcf, only the ones that you want to compare. Also, the script is designed to compare **only two** populations.
- `MOCH_AL` The name of the first population. This **must** match exactly the name of one of the populations that appears in the second column of the populations file.
- `CBCH_AL` The name of the second population, same requirements as the one above.
- `--num_sites 15` Number of variant sites that must be in a window for it to be analysed. I found 15 to be a good balance of information and number of windows. A very small number will produce many 0 estimates for G_{min}. At the moment I am not completely sure about how to interpret these zeroes, so after running the program I filter out these results. The script has a default of 10 variant sites per window.

- `--step_sites 8` Number of variant sites that the script must advance before creating another window. If you want windows to overlap set it lower than num_sites, if you don't want overlapping windows set them both equal.

- `MOCB_Gmin_allop2.txt` The name of the file that you want to save the results to. The script sends the results to the standard output (the screen), so you need to re-direct that output to the file with the > operator.

```
# 1. Run the script as shown. Remember to set the arguments in the correct
# order.
# This change of permissions only needs to be done ONCE
chmod +x ./gmin_from_vcf.py

./gmin_from_vcf.py MOCB_GT_161122.snoGL.vcf mocb_pops_allopatric2.txt \
                   MOCH_AL CBCH_AL --num_sites 15 --step_sites 8 > \
                   MOCB_Gmin_allop2.txt

# once you are done running the script, deactivate the virtual-env
deactivate
```

Below I include the contents of the populations file `mocb_pops_allopatric2.txt`. In this file you can use any whitespace character to separate the two columns.

```
CBCH_AK111 CBCH_AL
CBCH_AK113 CBCH_AL
CBCH_AK129 CBCH_AL
CBCH_PrBC107 CBCH_AL
CBCH_PrBC110 CBCH_AL
CBCH_PrBC112 CBCH_AL
MOCH_SOR001 MOCH_AL
MOCH_SOR002 MOCH_AL
MOCH_SOR003 MOCH_AL
MOCH_SOR004 MOCH_AL
MOCH_SOR006 MOCH_AL
MOCH_SOR007 MOCH_AL
MOCH_SOR008 MOCH_AL
MOCH_SOR009 MOCH_AL
MOCH_SOR010 MOCH_AL
```

4 Output

The output 5 columns and one line per genomic window analysed, plus a header. The columns have the name of the scaffold or chromosome, the start and end physical positions of each window, the number of variant sites contained within each window (normally equal to `--num_sites`, unless there is not enough information at a site to be included in the estimation). You can load this information into R and explore the distribution of G_{min} values, or how the estimate behaves along a single scaffold/chromosome.

```
CHROM Win_Start Win_End No_Variants Gmin
CM022157.1.CH1 56139 936704 15 0.679
CM022157.1.CH1 833704 2501090 15 0.545
CM022157.1.CH1 938471 2969887 15 0.629
CM022157.1.CH1 2550028 3317809 15 0.643
CM022157.1.CH1 3246263 3849245 15 0.724
CM022157.1.CH1 3325725 3977699 15 0.585
CM022157.1.CH1 3849318 4076225 15 0.730
CM022157.1.CH1 3984229 4195280 15 0.855
CM022157.1.CH1 4083441 4232119 15 0.854
...
```