How to use panicmage

Franz Baumdicker

February 22, 2018

Contents

1 Short summary of panicmage features

The software panicmage simulates distributed genomes according to the *Infinitely Many Genes Model* [2] and can estimate model parameters from observed data. Moreover panicmage computes a p-value for the observed gene frequency spectrum and a given genealogy in the *Infinitely Many Genes Model.* So far p-values for neutral evolution can be computed. In addition panicmage can account for sampling bias under neutral evolution. Given

- a proxy for the true clonal genealogy of the sample

- the gene frequency spectrum of the sample
- the sample size n
- the number of generations to the most recent common ancestor (MRCA) of the sample in millions (optional)

panicmage estimates gene gain and gene loss rates and the number of core genes according to the *Infinitely Many Genes Model*. In addition panicmage computes the p-value of the observed gene frequency spectrum under neutral evolution. Small p-values hint at global selective forces or population growth while very high p-values may appear if gene transfer and/or recombination occur frequently. Finally panicmage provides estimates for the following statistics in a neutral evolving population:

- the average number of genes per individual
- the average total number of genes present in 2, n, 1000 and 10000 individuals for a random clonal genealogy, resp.
- the average total number of genes present in 2 or n individuals for the given clonal genealogy, resp.
- the average number of new genes to be found in the next (n+1-th) sequenced individual

If in addition the number of generations to the MRCA is given panicmage can also estimate:

- The size of the pangenome, that is the total number of different genes present in the whole population.
- The size of the persistant pangenome, that is the total number of different genes present in at least 1% of the whole population.
- The per individual per generation gene gain rate
- The per individual per gene per generation loss rate

If you want to have a closer look at the number of so far sequenced persistant genes and how much more genomes you might have to sequence to get a complete picture of all persistant genes you can use the Mathematica file *howmanygenomes.nb*. You can find this file in the panicmage folder and on the website `www.stochastik.uni-freiburg.de/homepages/baumdicker/software`. See Figure 1 for a screenshot of the file.

2 Installing panicmage

panicmage is written in C and C++ and designed for Linux. Currently there is no manual for Windows or Mac.

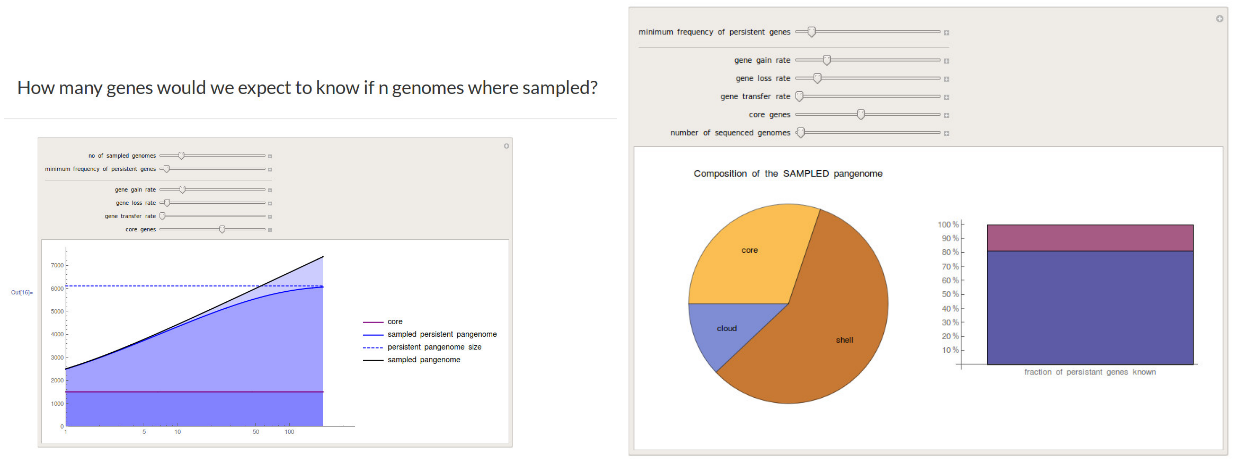
How many genes would we expect to know if n genomes where sampled?
no of sampled genomes
minimum frequency of persistent genes
gene gain rate
gene loss rate
gene transfer rate
core genes
Out[16]=
core
sampled persistent pangenome
persistent pangenome size
sampled pangenome
number of sequenced genomes
Composition of the SAMPLED pangenome
core
cloud
shell
fraction of persistant genes known

Figure 1: A screenshot from howmanygenomes.nb

2.1 dependecies

For panicmage the GNU Scientific Library (GSL) – development package (libgsl0-dev) is needed, please install it, e.g. by typing

```
sudo apt-get install libgsl0-dev
```

In addition panicmage depends on ginac so you might have to install it as well.

```
sudo apt-get install ginac-tools libginac-dev
```

2.2 compiling

In order to install panicmage extract panicmage.zip to any directory, e.g. by typing

```
unzip panicmage.zip
```

there is a binary file *panicmage* which might already work for your system. Otherwise compile it again with:

```
g++ panicmage.c -lm -lgsl -lgslcblas -lcln -lginac -o panicmage
```

In addition the flag -std=c++11 is required for gcc < 6.0. You may now run panicmage from commandline. Type

```
./panicmage
```

to see basic usage.
panicmage requires at least 3 input parameters which have to be prompted in the correct order.

```
./panicmage [TREEFILE] [GFS\_FILE] [INT] ... [OPTIONS]
```

3 Running panicmage

`[TREEFILE]` Has to be the tree your analysis is based on and represents the clonal genealogy of your sample. panicmage accepts only files in Newick format. For more infos on the Newick format have a look at `http://en.wikipedia.org/wiki/Newick_format`. The tree must contain the distances between the nodes and should be rooted. Note that only ultrametric bifurcating trees will give meaningful results! I.e. the distances between each pair of individuals have to equal. However panicmage does not check whether your tree is ultrametric, so be careful. The names/ID's for the individuals can be any string or integer, but 1,...,no. of individuals. It is recommend to use ID's between 1000 and 9999. A simple tree for 3 individuals might look like this: (101:0.6,(103:0.22,102:0.22):0.38);

`[GFS\_FILE]` This file should contain the gene frequency data. The correct format is a simple text file which contains the number of genes per frequency class seperated by whitespaces. To be precise, the first number has to be the total number of unique genes in the population. The second number has to be the number of genes which are present in exactly two individuals. And so on such that the last number is the number of genes present in all individuals (i.e. the number of core genes).

`[INT]` Enter the number of individuals in your sample. This number should equal the number of leafs in your treefile!

optional option

`-g [FLOAT]` Set the number of generations up to the most recent common ancestor (MRCA) for your sample. Optional parameter but, enables panicmage to estimate several statistics of interest (e.g. total pangenome size, per generation gene gain and loss rates). Number is treated in millions, so "-g 1.234" equals 1234000 generations up to the MRCA.

additional options

`-d` less details are printed

`-p` the tree (scaled to the estimated height) is printed

`-n` no neutraliy test is done. Normally panicmage will test whether neutrality holds or not. If this option is set the neutrality test is skipped.

`-s` test for sampling bias is done. Normally panicmage will not test whether the gene frequency spectrum is typical for a neutral evolving population, where samples have not been drawn randomly. If this option is set sampling bias is tested.

`-q [INT]` set the quantity of runs for the tests. Standard value is 10.000 runs.

`-a` all tests and estimations are skipped, only the theoretical results for a population with given parameters are shown. You have to set the parameters theta and rho

`-b` the tree is not rescaled (useful for simulation purposes)

set the parameters to custom values:

`-t [FLOAT]` manually set the value for the parameter theta. If theta is set no estimation is done for any parameter.

`-r [FLOAT]` manually set the value for the parameter rho. If rho is set no estimation is done. The parameter rho has to be larger than zero.

`-c [FLOAT]` manually set the value for the number of core genes. If core is set no estimation is done for any parameter.

Note: if no further options than -t -r -c are set the neutrality test will be done with the given parameters. Otherwise it depends on the additionally set options:

- `-a` the given parameters are only used for the theoretical results; same for -n
- `-s` the given parameters are used for the neutrality test. For the sampling bias test the given parameters are rescaled.
- `-n -s` the given paramters are directly used for the sampling bias test without any rescaling.

4 Preparing the Input for panicmage

4.1 Suggested pipeline to produce the input for panicmage

4.1.1 Estimating the gene frequency spectrum of your sample

- To eliminate differences between organisms due to different methods and thresholds of different authors one should repredict the genes for all considered genomes using the same method. There are several tools available for this e.g. GLIMMER [7] GENEMARK [4], prokka [13] and RAST [1].

1. use the gene prediction software of your choice to produce a fasta file which contains all gene sequences
2. run all-vs-all BLASTP on these sequences
3. use orthomcl [11] or orthagogue [9] to filter the blast results
4. use mcl [10] to generate clustor of ortholog genes based on these filtered blasthits

5. For each cluster count the number of sampled individuals which possess at least one gene from this cluster.

- As an alternative the pipeline roary [12] from `https://sanger-pathogens.github.io/Roary/` can be used for steps 2–4.

4.1.2 Estimating the clonal tree of your sample

The tree should be an indipendent estimate of the true clonal genealogy of the sample and not depend on any dispensable genes. panicmage assumes that the underlying tree is a realization of Kingman's coalescent and idealy the method to estimate the tree accounts for this.
You already should have the set of core genes of your sample from 4.1.1. We suggest to use Clonalframe [8] based on all or a selection of these core genes to estimate the tree.
A possible route to get the tree might be:

- Align the concatenated sequence of all core genes or a selection of several core genes to diminish the influence of possibly existing gene transfer/recombination of single genes. Therefore have a look at Mauve [6] or ClustalO [14]. Mauve produces an alignment in XMFA Format just as required by Clonalframe. As an alternative you can use any other multiple alignment algorithm and convert the alignment to XMFA format. There is a Bioperl script at `http://www.bioperl.org/wiki/HOWTO:AlignIO_and_SimpleAlign` to do so. We will call the final alignment file `alignment.xmfa`.

- Install Clonalframe and type:

  ```
  ./ClonalFrame -G -H alignment.xmfa cf_run1.out > cf_run1.log &
  ./ClonalFrame -G -H alignment.xmfa cf_run2.out > cf_run2.log &
  ./ClonalFrame -G -H alignment.xmfa cf_run3.out > cf_run3.log &
  ./ClonalFrame -G -H alignment.xmfa cf_run4.out > cf_run4.log &
  ```

  to start 4 runs of Clonalframe. Then look at the four consensus trees produced, which are shown in the first line of `cf_run?.out` If these trees are very similar the sampling from the posterior distribution by Clonalframe should be fine. If the consensus tree is bifurcating, i.e. all inner nodes have exactly two children you can use it as input for panicmage, otherwise you will have to estimate a bifurcating tree by another method or trick Clonalframe. You can find a workaround to trick Clonalframe in the appendix A.

4.1.3 Estimating the number of generations to the MRCA for your sample

In order to estimate the effective population size panicmage requires an estimate of the number of generations to the most recent common ancestor of the sample. If available you can base your estimate on archaeological data or use an evolutionary clock. In [3] we used an evolutionary clock, where a 23S rDNA distance of 1% represent about 50 million years of divergence and assumed a generation time of 24 hours, such that 1% 23S rDNA distance represents about 18250 million generations.

5 Citing panicmage

If you used the software panicmage please cite

F. Baumdicker, W. R. Hess, and P. Pfaffelhuber. The infinitely many genes model for the distributed genome of bacteria. *Genome Biology and Evolution*, 4(4):443–456, 2012.

If you want to refer to the paper introducing the *Infinitely Many Genes Model* please cite

F. Baumdicker, W. R. Hess, and P. Pfaffelhuber. The diversity of a distributed genome in bacterial populations. *The Annals of Applied Probability*, 20(5):1567–1606, 2010.

Note that there is an R package designed by Collins and Higgs [5] which is based on the *Infinitely Many Genes Model.* Collins and Higgs extended the model and allow multiple gene classes with different gene gain and loss parameters. The R package is as well able to estimate parameters based on an underlying fixed tree. Neutrality tests are so far only implemented in panicmage.

A Ultrametric bifurcating consensus trees from Clonalframe

Clonalframe computes the majority rule consensus tree from the simulated posterior sample. Especially for a large number of genomes the probability rises that the majority rule consensus tree does not produce a bifurcating tree. In this case you can try to produce an extended majority rule consensus tree instead. Since Clonalframe is not able to produce an extended majority rule consensus tree you may consider the following workaround.
The command

```
grep "((" clonalframe_run1.out | tail -1000 > treesample1.nwk
```

should do the right thing if you sampled 1000 trees from the posterior. You only have to add a ";" to the end of each line in `treesample1.nwk`. Now run phylip consense with Majority rule (extended) for rooted input trees, in our example from `treesample1.nwk`. Phylip produces `outfile` and `outtree`. Have a look at `outfile` to check whether the extended majority rule was able to produce an bifurcating tree. If so you can tell Clonalframe to fix the topology of `outtree`. Therefere modify `outtree` by the following R-script:

```
# install.packages("ape") # if necessary
library(ape)
# read in the outtree where branch lengths correspond to node frequency
phytree <- read.tree(file="outtree")
# Clonalframe requires the tree to be bifurcating and ultrametric
is.binary.tree(outtree)
ultratree = compute.brlen(phytree)
# Clonalframe requires the tip labels to be named "0" to "n-1"
ultratree$tip.label = as.character(as.integer(ultratree$tip.label) - 1)
write.tree(ultratree,file="topology.nwk")
```

You can now run Clonalframe with this fixed topology using the command

```
./ClonalFrame -G -H -w topology.nwk -T alignment.xmfa cf_consense_run1.out
```

The consensus tree suitable for panicmage can be found in line 1 of `cf_consense_run1.out`.

References

[1] R. K. Aziz, D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko. The RAST Server: rapid annotations using subsystems technology. *BMC genomics*, 9:75, 2008.

[2] F. Baumdicker, W. R. Hess, and P. Pfaffelhuber. The diversity of a distributed genome in bacterial populations. *The Annals of Applied Probability*, 20(5):1567–1606, 2010.

[3] F. Baumdicker, W. R. Hess, and P. Pfaffelhuber. The infinitely many genes model for the distributed genome of bacteria. *Genome Biology and Evolution*, 4(4):443–456, 2012.

[4] M. Borodovsky and J. McIninch. GENMARK: Parallel gene recognition for both DNA strands, 1993.

[5] R. E. Collins and P. G. Higgs. Testing the Infinitely Many Genes Model for the Evolution of the Bacterial Core Genome and Pangenome. *Molecular Biology and Evolution*, 29(11):3413–3425, 2012.

[6] A. E. Darling, B. Mau, and N. T. Perna. progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. *PloS one*, 5(6):e11147, 2010.

[7] A. L. Delcher, K. A. Bratke, E. C. Powers, and S. L. Salzberg. Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics*, 23:673–679, 2007.

[8] X. Didelot and D. Falush. Inference of bacterial microevolution using multilocus sequence data. *Genetics*, 175:1251–1266, 2007.

[9] O. K. Ekseth, M. Kuiper, and V. Mironov. orthAgogue: an agile tool for the rapid prediction of orthology relations. *Bioinformatics (Oxford, England)*, 30(5):734–736, mar 2014.

[10] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.

[11] L. Li, C. J. Stoeckert, and D. S. Roos. OrthoMCL: Identification of ortholog groups for eukaryotic genomes. *Genome Research*, 13:2178–2189, 2003.

[12] A. J. Page, C. A. Cummins, M. Hunt, V. K. Wong, S. Reuter, M. T. G. Holden, M. Fookes, D. Falush, J. A. Keane, and J. Parkhill. Roary: Rapid large-scale prokaryote pan genome analysis. *Bioinformatics*, 31(22):btv421, jul 2015.

[13] T. Seemann. Prokka: Rapid prokaryotic genome annotation. *Bioinformatics*, 30(14):2068–2069, 2014.

[14] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega, 2011.