Dennis Larsson 02-04-2020 Universität Wien

Assembly using Stacks 2

Introduction

In order to get a reasonable assembly of our RAD datasets a couple of important steps are necessary. But first off, this protocol is a modified version the one written by Rochette and Catchen (https://www.nature.com/articles/nprot.2017.123; doi:10.1038/nprot.2017.123) and it assumes that the reader has read through it at least once. If you haven't read it yet, do so before continuing. It is a brilliant tutorial and contains lots of good information not mentioned here.

This tutorial also assumes that the reader has read and finished the demultiplexing tutorial 'Demultiplexing using illumina2bam and process radtags' (https://github.com/DennisLarsson/demultiplexing).

The scripts used in this tutorial are available at:
https://github.com/DennisLarsson/assembly_using_stacks2

De novo assembly

Stacks is developed around the idea of assembling loci by similarity, read more about this here: (https://onlinelibrary.wiley.com/doi/full/10.1111/mec.15253).
The very nature of denovo assembly is that we don't know the correct parameters and thus have to guess and use key output values to evaluate the 'optimality' of the dataset. This also means that there are as many guess-work methods as there are bioinformaticians. This tutorial will show my way of doing it, but I don't claim it to be the most correct or best way to do it. It is just a way to do it. Ok let get started.

Once you know which species you wish to include in your analysis (you can always output a subsample of those that you used in the assembly, in other words, one assembly can be used for several sub-datasets). Make sure that the fastqs for all of those individuals are present in one and the same folder (preferably, just store all your files in one folder, and use the later described popmap to select which to work on).
We will call this folder *samples*. Then open a new text document and write into it those samples that you wish to use, this will be our *popmap* (step 4 in Rochette et al 2019). The *popmap* should look something like this:

```
spic_1 spic
spic_2 spic
spic_3 spic
pyr_1  pyr
pyr_2  pyr
pyr_3  pyr
```

Where the first column is the same name as the individuals fastq (for example the fastq for the first would be: *spic_1.fq.gz*) and the second column is the population/species to which you would like to assign it. You can also include a third column for yet another hierarchically higher group, but I never do this. But feel free to try it if you think it is meaningful. Each column is separated by a tab.

Parameter optimization

Before we do the full assembly we will do a series of smaller test assemblies to find optimal parameter settings. For this we will prepare a new popmap with a subset of samples called *popmap_test* that we will use for parameter testing (step 14 in Rochette et al 2019). This subsampled popmap should look like the above in structure but only consist of one individual per pop/species or whatever relevant group.

Preferably store the popmaps in a new folder called *RAD_assembly* or something similar and then navigate to that folder using the *cd* command in the terminal. As is mentioned in Rochette et al 2019 under 15A we want to create a number of folders for each of the test assemblies, we do this using the *mkdir* command like this:
mkdir stacks_m3n1M1 stacks_m3n2M2 stacks_m3n3M3 …
Up to the recommended M9n9 as in Rochette et al 2019 (I usually only do up to 6, then check if the number of r80 loci (see more below) have plateaued).

Now we are ready to set up a shell script to run these assemblies automatically without us having to babysit the computer. Create a new text document inside the *RAD_assembly* folder called *testing.sh* and open it. Inside we will write the first command to be run:
denovo_map.pl -M 1 -n 1 -T nrThreads -o ./stacks_m3n1M1/ --samples /path/to/samples --popmap ./popmap_test

Next copy this command and simply change *-M 1 -n 1* to *-M2 -n 2* and *-o ./stacks_m3n1M1/* to *-o ./stacks_m3n2M2/* . replace *nrThreads* with the number of threads (cores) on your computer. Then copy and paste again and repeat until you have the number of command iterations you want to run.

When you are finished, save the script and run it in the terminal (make sure you are in the *RAD_assembly* and that the file is in the same too) using the command:
sh testing.sh

This might take between a day to several days (at most three unless you dataset is very large, in which case you might want to reduce it further, or if your computer is slow) to complete, but you don't have to babysit the computer to run the next assembly run when the previous finishes, instead you can just check in twice or thrice a day to see the progress.
For more information about *denovo_map.pl* and what the different arguments do go to: http://catchenlab.life.illinois.edu/stacks/comp/denovo_map.php where you will also find the many other arguments within *denovo_map.pl*

When the assemblies have finished it is time to check which was the better. We do this by evaluating what is called the r80 loci, that means the loci that passed a filter that removes loci that are not present in at least 80% of the individuals. This filter is given in the software *populations*, which is part of the *stacks 2* package, using the argument *-r 0.8* or *-R 0.8*, where the latter is in case you have multiple populations (15A vii in Rochette et al 2019, but also read: Lost in parameter space: a road map for stacks, by Paris et al 2017; https://doi.org/10.1111/2041-210X.12775). You will be running this for each of the denovo assemblies so once again we will start of by preparing the output folders for *populations* using the command *mkdir* (while the terminal is in the *RAD_assembly* folder):
mkdir populations_m3n1M1 populations_m3n2M2 populations_m3n3M3 ...

Now we are ready to write a similar script as with *denovo_map.pl* but for *populations* software. As previous, first write the first command then copy and paste then change the numbers:
populations --threads nrThreads --in_path ./stacks_m3n1M1 --out_path ./populations_m3n1M1/ --popmap ./popmap_test -R 0.8
populations --threads nrThreads --in_path ./stacks_m3n2M2 --out_path ./populations_m3n2M2/ --popmap ./popmap_test -R 0.8
…

Populations usually run faster than *denovo_map.pl* so all your runs should be finished within a day.
To recover the number of loci for each assembly it can be convenient to use the *grep* command in the terminal. Simply type:
grep "Kept" populations/populations.log*

I have also written a script to extract the loci count called *extract_loci.sh.* **This script has to be run using *bash* so avoid using *sh*. *sh* refers to *dash* by default in Ubuntu and my script is not compatible with *dash*.** Make sure that the script is in the *RAD_assembly* folder, then run it using:
bash extract_loci.sh

The script will output a file with the name of the assembly and the loci count.
The number of snps per locus isn't crucial to the selection of best parameters in my opinion, it is rather a good complement to the r80 loci count, but should you want to make plots like figure 2 in Rochette et al 2019 i also included two scripts to extract the snps per locus as well. they is called *extract_snp_per_locus.sh* and *extract_snp_per_locus2.sh* and should be run like *extract_loci.sh*, with *bash* while in the *RAD_assembly folder*. The *extract_snp_per_locus2.sh* script will create a file in table format ready to import into Excel or Calc. ***Both bash scripts assume that you have followed my above naming convention and folder arrangement, if you divert from it you might have to edit the scripts accordingly.***

Now that we have the count of loci we can now plot them and evaluate which is best. When the number of loci plateau like in figure 2 in Rochette et al 2019 that is where you will find the most optimal parameters.

If you want to optimize your parameters further, you can now explore the parameters around your optimal parameters by increasing m or varying only n or M. However this will most likely not be necessary, but can be done if your full assembly worked poorly later.

Full assembly

Now that you know what the optimal parameters are we can start a full assembly, you can either do as in Rochette et al and do the ustacks separately, which is the fastest approach for assemblies on clusters, but here I will show how to do it using the *denovo_map.pl* wrapper. If you have more than 200 samples it is highly recommended to create the catalog using a subsample of individuals, and then use all in the following sstacks step. How to do this is beyond the scope of this tutorial and will not be covered (although it will only change things marginally and I might write and supplementary to this tutorial if there is interest).

First we will run a denovo assembly like in the testing phase but only for the best parameters and the popmap will all individuals (*popmap)*. The command in the terminal should look something like this (assuming you are standing the *RAD_assembly* folder):
denovo_map.pl -M X -n X -T nrThreads -o ./stacks_full/ --samples /path/to/samples --popmap ./popmap

Where the Xs are replaced with the relevant parameter numbers and nrThreads again replaced with number of threads.

SNP selection using *populations*

Once this has finished we will introduce a number of filters to remove any possible remaining paralogs, which should hopefully be few given that we have the best assembly parameters to begin with. Since we expect erroneously assembled loci to have a high number of variants (any SNPs), we will want to remove them. To do this we will run a shell command that will filter out those that only have 10 snps or less and write their loci number to a whitelist file. This whitelist file will then be used in *populations* during the SNP selection to ensure that we only work with loci with acceptable snp counts. The command looks like this:
cat ./stacks_full/populations.sumstats.tsv | grep -v "^#" | cut -f 1,4 | sort -n | uniq | cut -f 1 | uniq -c | awk '$1 <= 10 {print $2}' > whitelist

If you wish to use a different number for the SNP threshold simply change the ‘10’ inside: *awk '$1 <= 10 {print $2}'* to whatever you think is appropriate.

Now we can use this whitelist plus a few more new arguments in *populations* to produce a vcf file with SNPs ready for any analysis. First create a folder called *populations_final* using *mkdir* then run this command:
populations --in_path ./stacks_full/ --out_path./populations_final --popmap ./popmap -r 0.65 -p X/2 --max_obs_het 0.65 --vcf --threads nrThreads --write_random_snp -W ./whitelist

Where -r is the ratio of individuals within a population that must have a loci present (not missing) for it be regarded as being present in the population (I use 0.65 because I have 3-4 individuals per population and with a threshold of 65% it is required for a loci to be present in at least 3 individuals in the populations with a total of 4 (3/4 = 75%, above the ratio) and 2 for the populations with a total of 3 (2/3 = 66.7%, above the ratio)). This should be changed as appropriate. The *-p X/2* argument means that we require a loci to be present in at least half of the populations, where X is the total number of populations. With X/2 we ensure that a loci has to be present in half of the populations, but it can of course be changed to any whole number if one wants a different ratio. One can also replace -r and -p with the -R argument, which simply require a loci to be present in X% of the individuals, ignoring populations. To enforce a 50% rule one simply writes 0.5. The -R approach would be the best filter approach for a dataset to be used for phylogenetic analyses, since we can maximize the number of SNPs. *--max_obs_het 0.65* removes loci that are heterozygotic in more than 65% of the individuals. This is another way to remove paralogous loci, since we would expect high heterozygosity in loci that are erroneously assembled together into one loci within one individual. *--vcf* simply tells *populations* to output our SNP data in the VCF format. *--write_random_snp* tells *populations* to select one random SNP from each loci, rather than outputting all. This is to avoid linked SNPs, since most population genetics softwares requires unlinked SNPs. this can be removed if one intends to use the dataset in phylogenetic analyses, since link SNPs are not a problem in those cases (except for SNAPP?). Finally we have the *-W ./whitelist* argument to include our whitelist of loci with 10 or less SNPs. for more information about the arguments and what they do please visit: http://catchenlab.life.illinois.edu/stacks/comp/populations.php

Once *populations* has finished you can proceed with your prefered analyses.