Contents

1 Why use PyRAD?

Reduced-representation genomic sequence data (e.g., RADseq, GBS) are commonly used to study population-level research questions, and consequently, most software packages for analyzing RADseq data are designed for data with little variation across samples. Phylogenetic analyses typically include species with deeper divergence times (more variable loci across samples) and thus a different approach to clustering and identifying orthologs may perform better. *PyRAD*, the software pipeline described here, is intended to maximize phylogenetic information across disparate samples from RADseq data. Written in Python, the program is flexible and easy to use. The code is human-readable, open-source and may be modified to meet users specific needs.

With respect to constructing phylogenetic data sets, the largest difference between *PyRAD* and alternative programs such as *Stacks*, is in the way that loci are clustered. *Stacks* clusters reads by grouping those with less than N differences between them, but does not allow for indels, such that the presence of even a single indel will cause a frameshift resulting in many base differences between otherwise highly similar sequences. PyRAD instead uses a measure of sequence similarity based on a global alignment of sequences through the program USEARCH . Global alignment allows for indels, and should thus perform better in data sets including more distantly related samples.

An added benefit of this method is the ability to cluster reads that are only partially overlapping. This has proven common in certain types of GBS data where the restriction enzyme cuts both ends of a DNA fragment, as well as in most types of paired-end data. PyRAD can perform reverse complement matching to test for overlap of these reads. This allows building contigs that are longer than individual read lengths, and increases quality by combining reads that have high quality base reads at both ends (quality score decreases towards the end of reads). It also reduces duplication of reads in the data set.

I am very happy to answer questions about the use of PyRAD. Feel free to either email me or post your question to the google group:
`https://groups.google.com/forum/#!forum/pyrad-users`

2 Installation

PyRAD is available for Linux and Mac, and can be downloaded from the following link:
`http://pyrad.googlecode.com`

A few Python modules and external software are required as dependencies. I list how you can find and install each below:

- python 2.5+

- python-numpy
- python-scipy
- muscle
- usearch v.6.0307 (or newer versions) 32 or 64 bit versions

PyRAD will only work on *nix based systems (Mac or Linux), not Windows. These systems will almost always come with Python already installed. *Numpy* and *Scipy* are Python modules/libraries required for performing numerical and scientific calculations. They can be easily installed on Ubuntu-linux through the apt-get commands, for example:

sudo apt-get install python-numpy
sudo apt-get install python-scipy

If you are using a Mac I have found the easiest way to properly install *Numpy* and *Scipy* is to run something called the scipysuperpack script. Depending on your version of OSX you sometimes need to install some odd dependencies and libraries such as a fortran compiler – its a pain. However, this very convenient script checks if you have these other necessitites installed and if you don't it asks you to install them and then does it for you. I've used it and read the code and its a trusted source. There are two versions, one for OSX 10.6 and the other for newer systems, both available at `https://github.com/fonnesbeck/ScipySuperpack/downloads`. Simply download and run the script as:

sh install_superpack.sh

Alternatively both modules can both be downloaded from source at the following link `http://scipy.org/Download`.

The other two required pieces of software are the brilliant programs of Robert Edgar that run the core of *PyRAD*, which includes *MUSCLE* and *USEARCH*. Muscle is once again available through the apt-get command (*sudo apt-get install muscle*). Otherwise, it can be downloaded from the same site where you can download *USEARCH* (`http://www.drive5.com`).

Important: Currently, as of the release date of PyRAD v.1.5, I do not recommend downloading USEARCH v.7.0, as some of the most recent updates have made it incompatible with the paired-end methods of PyRAD. Instead I recommend that you download and continue using USEARCH v.6.0307 for the time being.

3 Using PyRAD – the input files

3.1 the barcodes file

The barcodes file is a simple table linking barcodes to samples. Barcodes can be of varying lengths. Each line should have one name and one barcode, separated by a tab (important: not spaces). An example file, which I call barcodes.txt, can be found with the example data and is formatted like below:

```
sample1    ACACG
sample2    ATTCA
sample3    CGGCATAC
sample4    AAGAACTG
...
```

If your data are already sorted by samples into separate files (de-multiplexed) then a barcode file is not needed.

3.2 the params file

An example parameter file (params.txt) is included in the example data sets. This file lists all of the options or paramater settings necessary for an analysis. The first 13 lines must be filled out, all lines following this are optional and used for more advanced analyses and data filtering. Every line has a parameter (e.g., a number or character string) followed by any number of spaces or tabs and then two hash marks ("##"), after which comments can be added (see the example file).

I highly recommend beginning all analysis by creating an empty template params.txt file with the following command:

```
/home/user/pyRAD/pyRAD -n
```

In the following I describe the params file line by line. Read this to learn how to fill in your template params.txt file. The first 13 lines are the most important, and the rest are optional or define more advanced settings:

3.2.1 required lines 1-13

Line 1: The working directory for your analysis. Three different examples are shown below. This is where all of the output files from the analyses will be placed.

```
                        ##  1. empty uses current as working directory.
./                      ##  1. current location as working directory.
/home/user/RAD/         ##  1. different location for working directory.
```

Line 2: the location of the raw unsorted fastq formatted Illumina sequence data. Use the character * to select multiple files in the same directory. Further directions below for paired-end data.

```
./raw/*.fastq  ##  2. path to raw data (fastq files)
```

Line 3: the location of the barcodes file. If your data are already sorted then lines 2 and 3 can be left blank, but line 18 (see below) must be entered. Example:

```
/home/user/RAD/barcodes.txt       ## 3. path to barcodes files
```

Line 4: the command to call USEARCH. If USEARCH is in your $PATH, then simply list the executable file name here. Otherwise, enter the full path to USEARCH that is required for you to call it from the command line. Example:

```
/home/user/bin/usearch6.0.307_i86linux32    ## 4. usearch path
usearch6.0.307_i86linux32                   ## 4. usearch is in path
```

Line 5: the sequence of the restriction recognition site overhang. If you are using ddRADseq you can enter both restriction cut sites separated by a comma. Example:

```
TGCAG           ##  5. Pst1 cutsite is C|TGCAG
TGCAG,ATTA      ##  5. ddRAD example.
```

Line 6: the number of processors to use. Entering a value that is more than the number of available processors will yield slower results. Example:

```
8               ##  6. n processors.
```

Line 7: the minimum depth of coverage to make a statistical base call at each site in a cluster. Depending on the inferred error rate, five is typically the minimum for a statistical base call. If your data are very low coverage you can instead call majority consensus base calls on reads with coverage as low as 2 by setting option 34 (see below). Example:

```
10              ##  7. mindepth calls.
```

Line 8: the maximum number of undetermined ("N") sites in 'edited' sequences before clustering (see description of step 2 in analysis). This number should be selected with respect to the clustering threshold. A low clustering threshold (.85) can allow more undetermined sites; whereas a high clustering threshold (.92) should exclude reads with too many undetermined sites.

```
4               ##  8. max # Ns in reads
```

Line 9: the similarity threshold to use for global alignment clustering in USEARCH, entered as a decimal. This value is used for both within-sample clustering and

across-sample clustering. I (more or less) require you to use the same threshold for both. Given the way the clustering algorithm works it only makes sense (to me) to do so, otherwise reads that do not cluster together within a sample will later cluster together when you do across-sample clustering, which is undesirable. Anyway, the example:

```
.90               ##  9. clustering threshold
```

Line 10: Indicator of the type of library preparation method. Cut at only one end (RAD or ddRAD) = 0, cut at both ends (GBS) = 1. Also, paired ddRAD or GBS = 1. Filtering is more stringent on the (1) data type, and reverse complement matching is tested for as well. Example:

```
1                 ##  10. RAD(0)/GBS(1)
```

Line 11: Minimum number of (ingroup) samples with data for a given locus to be retained in the final data set (see advanced usage). If you enter a number equal to the full number of samples in your data set then it will return only the loci that have data across all samples. If you enter a lower value, like 4, it will return a more sparse matrix, including any loci for which at least four samples containing data. Example:

```
4                 ##  11. min # of samples in locus
```

Line 12: maximum number (or proportion) of shared polymorphic sites in a locus. Enter a number, or a decimal with the prefix "p" (e.g., p.10 for 10%). This option is used to detect potential paralogs, as a shared heterozygous site across many samples likely represents clustering of paralogs with a fixed difference rather than a true heterozygous site. Example:

```
3                 ##  12. max # of shared heterozygote sites
```

line 13: prefix name for final output files. I usually enter a name indicative of the other parameters in the lines above. Example:

```
c90d10m4p3        ##  13. output name prefix
```

3.2.2 optional lines 14-40

These options are not necessary for most analyses and can be left empty in the params file, in which case PyRAD will use their default values.

Line 14: Is a separator between necessary and optional parameters. xxx.

Line 15: subset selector for steps 2-5. If, for example, you have two data sets that were sequenced together, and in one all of the sample names begin with the letter "A", then you can select only these samples for an analysis by entering the shared prefix "A*" on this line.

```
A*                     ## selects only taxa beginning with 'A'
```

Line 16: Add-on (outgroup) taxa selector. The option on line 11 tells pyRAD to keep only loci with at least n ingroup samples. If you wanted to maximize coverage within a certain clade you could set the other taxa as outgroups, and only loci with at least (line 11 value) ingroup samples will be kept, and any additional matching of 'outgroup' taxa to those will be included in the data, but not count toward the minimum number of samples. List 'outgroup' taxa separated by commas.

```
outg1,outg2          ## outgroup selector
```

Line 17: Exclude taxa. When you are constructing data sets at step 7 (only works at step 7 currently) you can exclude certain taxa in order to maximize coverage among other included samples. For example you may want to exclude taxa that are found to have very few data. To exclude taxa list them comma separated here. Example,

```
sample3,sample4      ## exclude taxa
```

Line 18: If your data are already de-multiplexed into separate files for each barcode then you can skip step 1 in the analysis and go straight to step 2. Now, instead of entering the location of the raw files in line 1, you instead enter here the location of the sorted fastq files. If you have already passed the data through some other program to trim the barcode and cutsite from the sequences then enter the '@' symbol at the beginning of the line before the file location. Examples:

```
/home/user/RAD/fastq/*.fastq      ##  Loc. of sorted files
@/home/user/RAD/fastq/*.fastq     ##  Loc. of sorted & filtered files
```

Line 19: The maximum number of mismatches allowed in a barcode during de-multiplexing. Default is 1, I don't generally recommend going above this.

```
1                      ##  barcode mismatches
```

Line 20: Minimum Phred Quality score (usually offset by 33, but will try to automatically detect if offset is 64) below which base calls are converted to Ns. I think the default value of 20 is reasonable.

```
20               ##  minimum Phred Quality score during step 2 filtering.
```

Line 21: strictness of filtering in step 2. (0) means no filtering for barcodes, adapters and cut sites. (1) looks for these as exact matches and trims them out. (2) tries to detect these things while allowing errors, which enforces very strict filtering. For most data sets that do not include many overlapping and short fragments from GBS or paired end data you do not need to use (2).

```
1                ## strict matching for trimmed sequences
```

Line 22: *a priori* error rate and heterozygosity. Step 4 of the analysis jointly estimates the error rate with heterozygosity, and step 5 uses these values to make base calls. If for some reason you wish to skip the error rate estimate (it is a bit slow, or your samples are

polyploid), then you can enter an *a priori* estimate of the error rate here. Caution: you should let the program estimate it by using step 4, as the error rate here is being measured on data that have already passed through some filters, so error rates estimated elsewhere will not be generally correct. Across several data sets I typically find error rates between 0.0005 - 0.002, depending on length and sequence chemistry. Example:

```
0.0005,0.001      ## E,H
```

Line 23: maximum number of "N"s in a consensus sequence. Clustering across samples can be affected by the number of "N"s, and so this number should be chosen with respect to line 6 and the lengths of reads. Default is 3.

```
3           ## max number Ns in consensus seqs
```

Line 24: maximum number Hs in a consensus sequence. Among several methods to exclude paralogs, you can set a maximum number of heterozygous sites in a consensus sequence. Default is 3.

```
3           ## max number heterozygote sites in consensus seq
```

Line 25: random number seed. Randomization is used at a few points in the analysis. If you set a seed then analyses should be repeatable.

```
112233      ## random number seed.
```

Line 26: Trim right end of consensus sequences. Quality decreases towards the far end of reads. If desired, you can trim off N bases from the right end of reads by setting N here.

```
5           ## cutoff N bases from right side of consensus seqs
```

Line 27: Allow only N haplotypes in a consensus sequence. In diploid organisms, after correcting for sequencing errors, there should be at most two haplotypes making up any consensus genotype call. If there are more the locus will be discarded as containing paralogs. Note default is 2, not well tested for polyploids yet. Example:

```
2           ## diploid 2 haplotypes allowed
```

Line 28: Maximum number of SNPs in a final locus. This can remove potential effects of poor alignments in repetitive regions in a final data set by excluding loci with more than N snps in them.

```
20          ## max number of snps.
```

Line 29: location (path) to call muscle for aligning loci, only needed if call to muscle is not simply 'muscle'.

```
muscle_v.x.x   ## path to call muscle
```

Line 30: Matches paired-end data at step 1 and checks overlap of sequences. However, unless they are found to overlap they are thereafter treated as single reads. Further paired-end steps are still in development.

```
1              ## match paired reads.
```

Line 31: prefix's for two-step clustering. See section on two-step clustering. Much faster for clustering across-samples in very large data sets.

```
A,B,C,D        ## prefix's for hierarchical clustering.
```

Line 32: The minimum number of samples for each clade in a two-step clustering.

```
4,4,4,1        ## min hits for hierarchical clustering.
```

Line 33: Maximum number of insertions/deletions in a within-sample cluster.

```
3              ## max number of insertion/deletions within samples
```

Line 34: Call simple consensus (majority base) on clusters with depth less than 5, (allows you to set option 7 as low as 2).

```
1              ## call consensus on low depth clusters
```

Line 35: Allow overhanging ends of reads. If reads are different lengths or overlap to different degrees, 1,1 will trim to shortest sequence on either side of locus, 0,1 would trim only the right side, 0,0 allows both ends to overhang.

```
1,1            ## overhang/trim ends of different length samples in a locus
```

Line 36: Output formats (u,s,a,n). u = unlinked snps, s = snps, a = alleles, n = nexus. See format section.

```
a,n            ## outputs extra formats of final data if specified.
```

Line 37: The minimum overlap to test for in paired-end or GBS data. I do not recommend below .30.

```
.50            ## minimum overlap of reverse complement matches
```

Line 38: Keep trimmed sequences. This option is only for very messy data, where a size selection method did not work properly such that many fragments were shorter than the read length and thus the adapters were frequently sequenced. This option creates separate data sets for short fragments and for full length reads.

```
0              ## keep trimmed sequences separate.
```

Line 39: the minimum length of kept trimmed reads, only if selected to keep in step 38.

```
50             ## min length of trimmed reads.
```

Line 40: output error locations for within sample base calls. If you are interested in the distribution of sequencing errors in your data you can print it out here. This creates very big files.

```
0               ## print output error locations.
```

Line 41: maximum depth of a within-sample cluster. The default (left empty) is max(meandepth+2*SD, 500), meaning the higher value of either the mean plus two times the standard deviation of cluster depth, or 500. If instead you want to set an absolute value enter it here. Or enter a ridiculously high value for no maxdepth filtering.

```
                ## default maximum depth
50000           ## a set maximum depth
99999999999     ## no maximum depth
```

3.3 the command line

PyRAD must be called from the directory in which it is comes (i.e., with its other dependency .py files). For example, to call *PyRAD* if it had been downloaded to a user's /home directory, type:

```
/home/user/pyRAD/pyRAD -h
```

There are five main options to PyRAD: -h (help screen) -n (generate new params file) -p (designate params file), -s (choose steps), and -d (D-test input file).

The parameter (params.txt) input file should have been filled out as described above. If the params file is properly filled out then the entire analysis – converting RADseq data into a phylogenetic data set – can be done by simply entering:

```
/home/user/pyRAD/pyRAD -p params.txt
```

This will perform all steps (1-7) of the analysis, as described below. If, however, you wish to perform only one, or a few step at a time, then you can select these using the -s (steps) option. To select only the first step:

```
/home/user/pyRAD/pyRAD -p params.txt -s 1
```

Or to select only step 2.

```
/home/user/pyRAD/pyRAD -p params.txt -s 2
```

If you wanted to run your analysis at two different clustering thresholds you can restart from step 3 (clustering) before and after changing the clust threshold in the params file and run the remaining steps with the following command:

```
/home/user/pyRAD/pyRAD -p params.txt -s 34567
```

When first learning new software it often takes a while to learn how to properly fill in the correct parameter settings and options. This can be frustrating with computationally intensive analyses, where one might have to wait many hours (days) before learning that they made a simple mistake. The -s option is an attempt to reduce this annoyance by breaking up the analysis into seven discrete jobs, each of which is performed sequentially. In this way, you will know whether step 1 worked before moving on to step 2. And if step 2 fails, you will retain the results of step 1, and can start again from where the error arose. I also suggest running the example data set which goes quite quickly in order to familiarize yourself with the steps in the analysis. Check out the output files from each step and the statistics output in the stats/ directory.

3.3.1 The seven steps described

Step 1: de-multiplexing. This step uses the barcode map file to split up sequences from your raw fastq files into a separate file for each sample. These are placed in a new directory in your working directory labeled "fastq/".

For paired-end sequences it is necessary that the file names follow a specific format: The first and second read files can begin with any name, but must end with a label for which read it is (e.g., R1 or R2) and a three digit number specifying that file (e.g., 001). Here is an example pair of input files:

```
yourfilename_R1_001.fastq, yourfilename_R2_001.fastq
```

If your data are already demultiplexed this step can be skipped. You will not need to enter a location for your raw data in the params file, but instead you must enter the location of your demultiplexed files into the proper location on line xx of the params file.

Step 2: filtering. This step uses the Phred quality score output by the Illumina platform to remove low quality reads. Sites with a score below a set value are changed into "N"s, and loci with more than the number of allowed "N"s are discarded. The restriction cut site and barcode are removed from reads. Files are written to the "edits/" directory with ending ".edit".

Step 3: within-sample clustering. This step uses a wrapper around the cluster function of USEARCH to cluster reads by sequence similarity and then to align them with MUSCLE. Aligned files are written to a "clust.xx/" directory with the ending ".clustS".

Step 4: error-rate and heterozygosity estimates. This step uses the Likelihood equation of Lynch (20XX) to jointly estimate error rate and heterozygosity from the base counts in

each site across all clusters. Results are written to the Pi_estimate.txt file in the stats/ directory.

Step 5: create consensus sequences. Using the mean error rate and heterozygosity estimated in step 4, this step creates consensus sequences for each cluster. Those which have less than the minimum coverage, more than the maximum number of undetermined sites, or more than the maximum number of heterozygous sites, or more than the allowed number of haplotypes, are discarded.

Step 6. Consensus sequences are clustered across samples using the same settings as in step 3.

Step 7. Alignment, detection of paralogs, output of human readable fasta-like file (.loci), and concatenated loci (.phy), other optional formats can be specified (.nex, .snps, .alleles), and final statistics are written to .stats file. This step is relatively fast, and can be repeated with different values for options 11,12,16,17 to create different data sets that are optimized in coverage for different samples.

3.4 Example data sets

3.4.1 Single-end RAD example

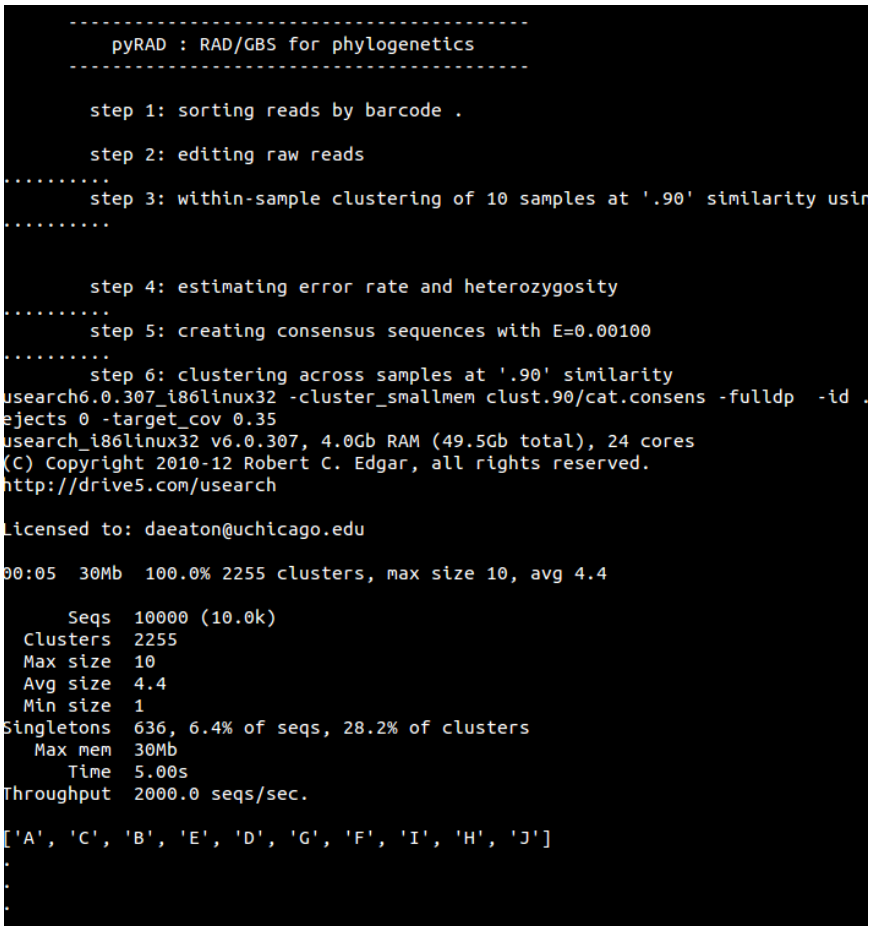
Download the RAD_example.zip directory and perform an analysis on a single-end RAD data set. To change directory into the RAD_example/ directory .

```
cd RAD_example/
```

Here you should see three files: a barcodes file, a params.txt file, and a raw sequence file. Take a look at the params file, you can use this as a template to create your own input file. When running your analysis new directories will be made in your working directory.

```
/home/user/pyRAD/pyRAD -p params.txt
```

This will count up as it proceeds through the seven steps, similar to what is shown below.



This will create several output directories. The fastq/ directory contains the raw data sorted into each sample by their barcodes. The edits/ directory contains the filtered data from step 2. The clust.xx directory contains files with clusters from step 3, and is labeled by replacing "xx" with the clustering threshold. The consensus sequences are also placed in the clust.xx directory. The across-sample clusters of consensus sequences from step 6 are also placed in the clust.xx directory in the file cat.clust_. Finally, step 7 creates output files in a outfiles directory, that are labelled by the output prefix you entered. A stats/ directory is also created, which contains statistics for what happened at each step of the analysis, it is explained further below in the section "Statistics output files".

3.4.2 Paired-end GBS example

...coming soon.

4 Statistics output files

I've tried to put explanations into each statistics file, but some still need further explanation. These files are in the directory stats/ and are created after each step of the analysis.

- s1.sorting.txt – This file lists the number of raw sequences in each file, and the number that were successfully assigned to a sample by the provided barcodes.
- s2.rawedit.txt – This file lists, for each sample, the number of raw sequences, the number that passed filtering, and the number of passed reads for which paired-end reads overlapped, if applicable.
- s3.clusters.txt – This file lists, for each sample, the number of clusters, the mean and SD of their depth of coverage, and then the number of clusters with depth > the minimum set in the params file, and the mean and SD of their coverage.
- Pi_E_estimate.txt – This file lists the ML estimate of error rate (E) and heterozygosity (H) for each sample.
- s5.consens.txt – This file lists the number of consensus for each sample. The number that passed further filtering. The number of sites (n loci x their length). The number of heterozygous sites. And the percentage of heterozygous sites.
- *"output_prefix"*ṡtats.txt – This file shows the number of across-sample clusters, how many are excluded for containing duplicates and the number that were discarded as paralogs (too many shared polymorphisms). It then lists the number of loci recovered for each sample in the data set. It also lists the number of variable sites in the final data set as well as the number of parsimony informative sites (assuming infinite sites mutation).

5 File formats

Here is a description of all of the file types created by PyRAD. They are not all necessarily of interest for you to look at, but they are all human readable.

Files used during the analysis:

- .fq = fastq file sorted into a unique sample
- .edit = fasta formatted file that has passed quality filtering
- .derep = dereplicated .edit file
- .clust = cluster file

- .clustS = aligned cluster file
- .consens = consensus sequences file
- .nex = nexus
- .phy = phylip
- .stats = text file of statistics output

6 D-statistic test for introgression

D-statistic tests can be performed using the (-d) option at the command line and specifying a Dtest input file:

```
/home/user/pyRAD/pyRAD -d D1.1.txt
```

Dtest input files can list a large number of tests to be performed in batch mode. The first three lines specify the following:

Line 1: the number of bootstrap replicates (I recommend 1000).
Line 2: the location of the ".loci" input file, output from a PyRAD analysis (or formatted from elsewhere into my '.loci' format).
Line 3: 4 or 5 taxon test.
Line 4: number of processors to use.
Line 5 -> to the end of file: one test per line.
example:

```
1000
outfiles/c90d10m4p3.loci
4
10
sample1       sample2      sample3      sample4                    ##  test1
[sample1]    [sample2]    [sample3]    [sample4]                  ##  test2
[sample1]    [sample2]    [sample3]    [sample4,sample5,sample6]  ##  test3
```

7 FAQs

1. It says permission denied when calling muscle or usearch?

You may have to set your permission to call the executables of these files using the chmod command, e.g., chmod 555 muscle.

2. My computer ran out of memory during the analysis.

I tried to reduce memory use in most steps of *PyRAD*. Try reducing the number of processors being used (line 5 of params file) as this will reduce the number of files loaded into memory at once.

3. I passed my data through xx and yy programs to filter the data before using PyRAD, will this affect it?

PyRAD has several built in filtering methods, so for most data sets I do not think using other filtering programs is necessary. But it may be. There are instructions above for entering your already de-multiplexed data into the params file, and also for designating whether the barcode and cutsite have already been trimmed from the data.