Generating and processing high-throughput short read data (eg. RAD-seq, ddRAD): introduction

12 May 2019

Chris Barratt (Flexpool postdoc at iDiv)
Evolution and Adaptation (PI: Renske Onstein)
Sustainability and Complexity in Ape Habitat (PI: Hjalmar Kuehl)

What are we doing today?

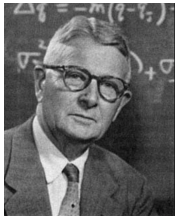
- High-throughput data and RAD-seq background − what and why?
- Sampling design prior to RAD-seq type work
- Bioinformatics to generate output files with a HPC cluster using Stacks

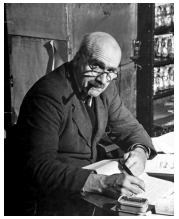
➢ More specific workshops to follow later in the year (e.g. population structure, phylogeny, demographic inference), TBA

Population Genetics

See Wright, Fisher, Haldane in the early-mid 19th century:

- Genetic differentiation within and between populations
- Processes leading to diversity patterns
- Adaptation/speciation



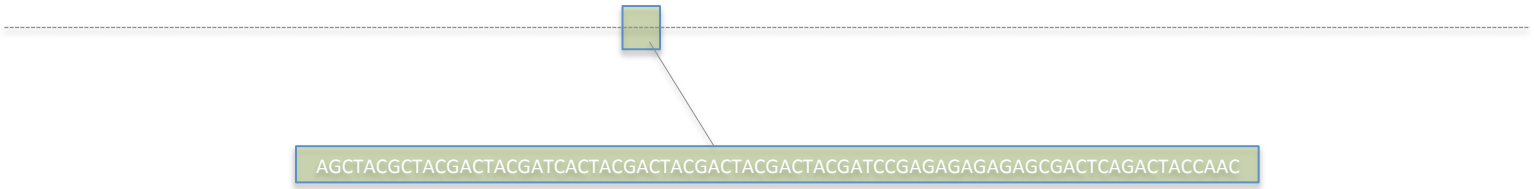


Markers have gradually become more numerous and associated methods more sophisticated:

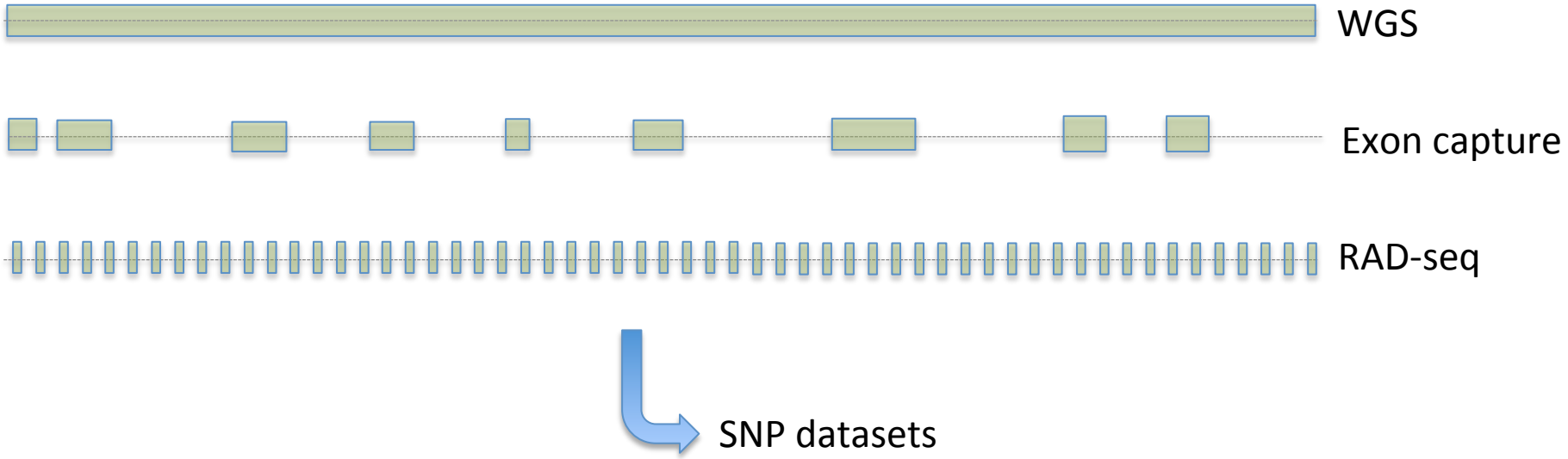
- Allozymes
- AFLPs
- microsatellites
- SNPs

High-throughput data

Genetic data (e.g. nuclear loci, mtDNA, chloroplast DNA)



Genomic data (e.g. WGS, Exon capture, RAD-seq)



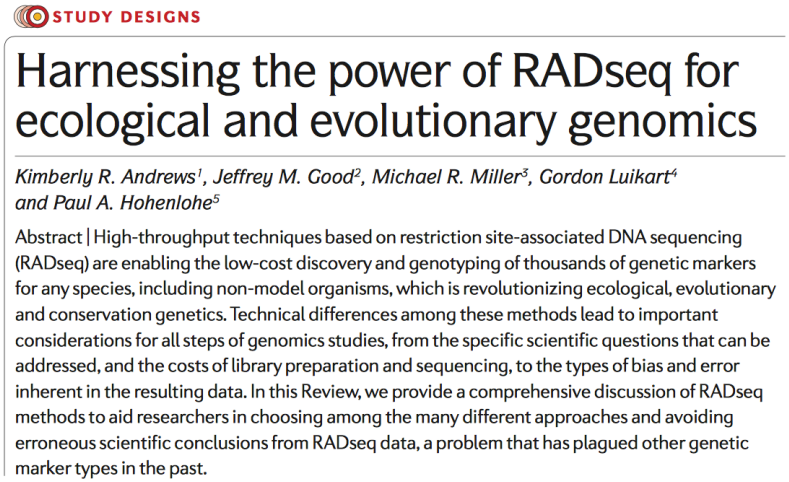
RAD-seq

Positives

- Large data
- Quick data generation
- Pooling of individuals
- High resolution
- Powerful inference
- Cost effective

Negatives

- Laboratory investment (adapters, equipment such as sonicator or Pippin prep)
- Bioinformatic investment – HPC use, also learning how to process data



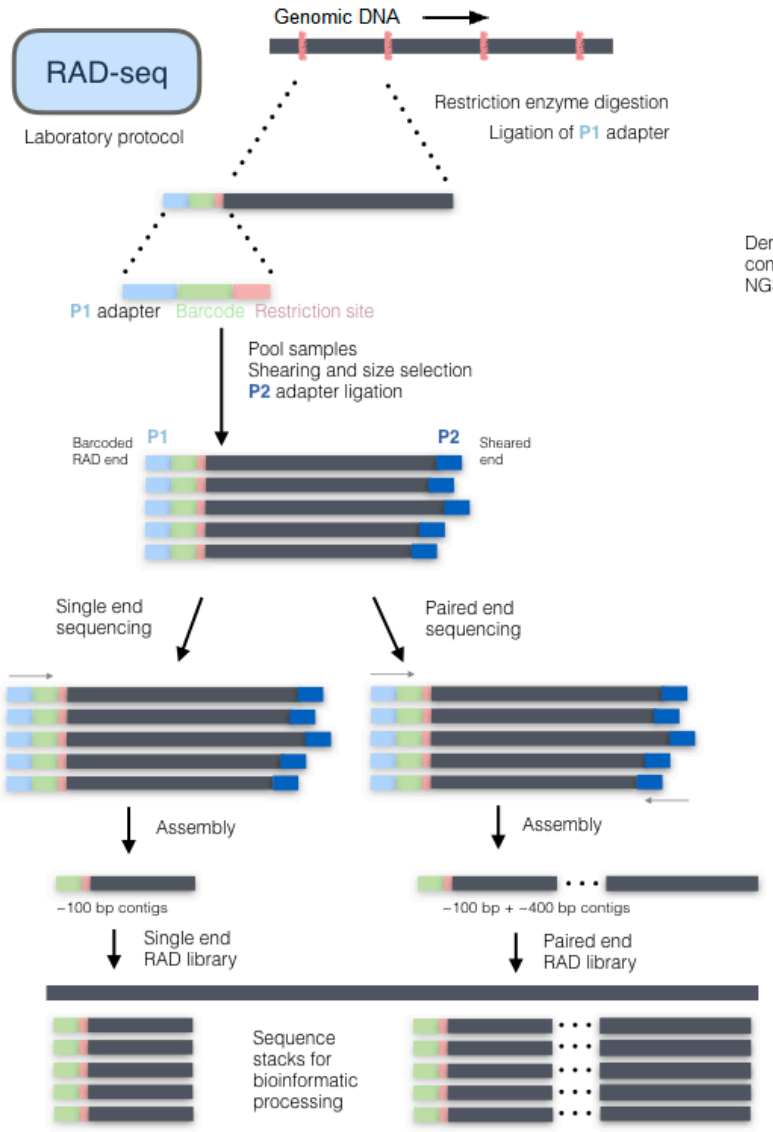
STUDY DESIGNS

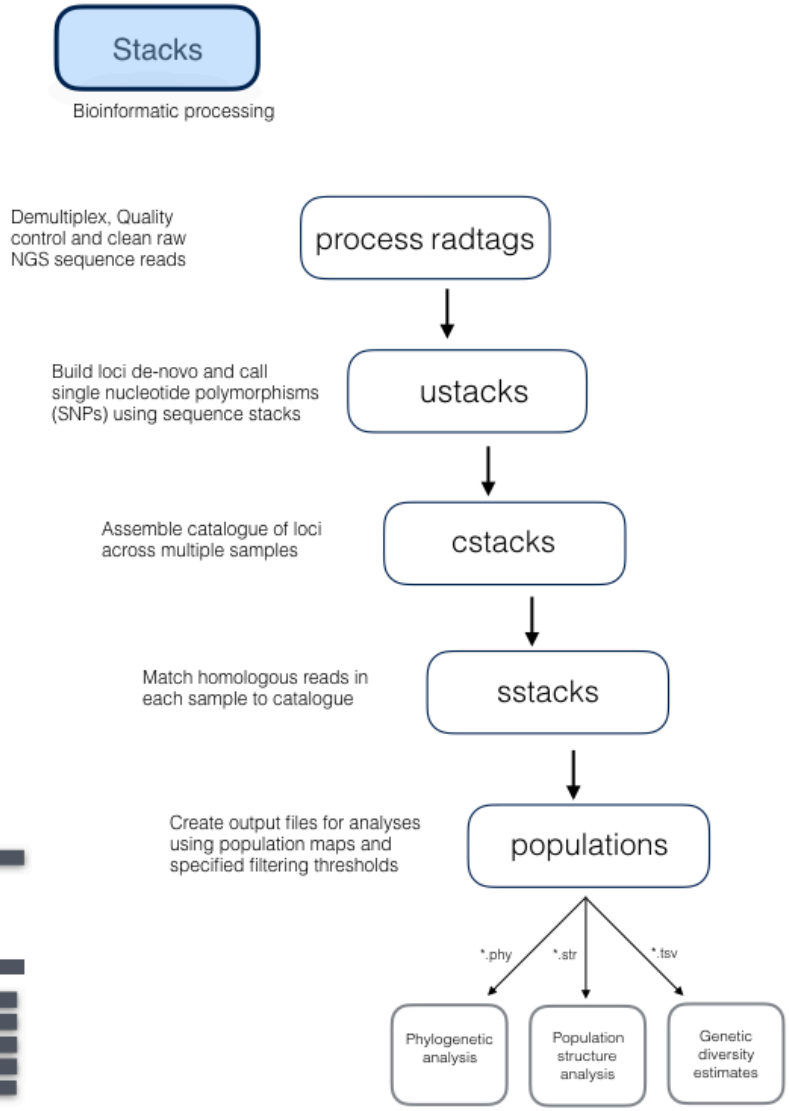
Harnessing the power of RADseq for ecological and evolutionary genomics

Kimberly R. Andrews[1], Jeffrey M. Good[2], Michael R. Miller[3], Gordon Luikart[4] and Paul A. Hohenlohe[5]

Abstract | High-throughput techniques based on restriction site-associated DNA sequencing (RADseq) are enabling the low-cost discovery and genotyping of thousands of genetic markers for any species, including non-model organisms, which is revolutionizing ecological, evolutionary and conservation genetics. Technical differences among these methods lead to important considerations for all steps of genomics studies, from the specific scientific questions that can be addressed, and the costs of library preparation and sequencing, to the types of bias and error inherent in the resulting data. In this Review, we provide a comprehensive discussion of RADseq methods to aid researchers in choosing among the many different approaches and avoiding erroneous scientific conclusions from RADseq data, a problem that has plagued other genetic marker types in the past.

RAD-seq


RAD-seq
Laboratory protocol
Genomic DNA
Restriction enzyme digestion
Ligation of P1 adapter
P1 adapter
Barcode
Restriction site
Pool samples
Shearing and size selection
P2 adapter ligation
Barcoded RAD end
P1
P2
Sheared end
Single end sequencing
Paired end sequencing
Assembly
Assembly
~100 bp contigs
~100 bp + ~400 bp contigs
Single end RAD library
Paired end RAD library
Sequence stacks for bioinformatic processing


Stacks
Bioinformatic processing
Demultiplex, Quality control and clean raw NGS sequence reads
process radtags
Build loci de-novo and call single nucleotide polymorphisms (SNPs) using sequence stacks
ustacks
Assemble catalogue of loci across multiple samples
cstacks
Match homologous reads in each sample to catalogue
sstacks
Create output files for analyses using population maps and specified filtering thresholds
populations
*.phy
*.str
*.tsv
Phylogenetic analysis
Population structure analysis
Genetic diversity estimates

ddRAD

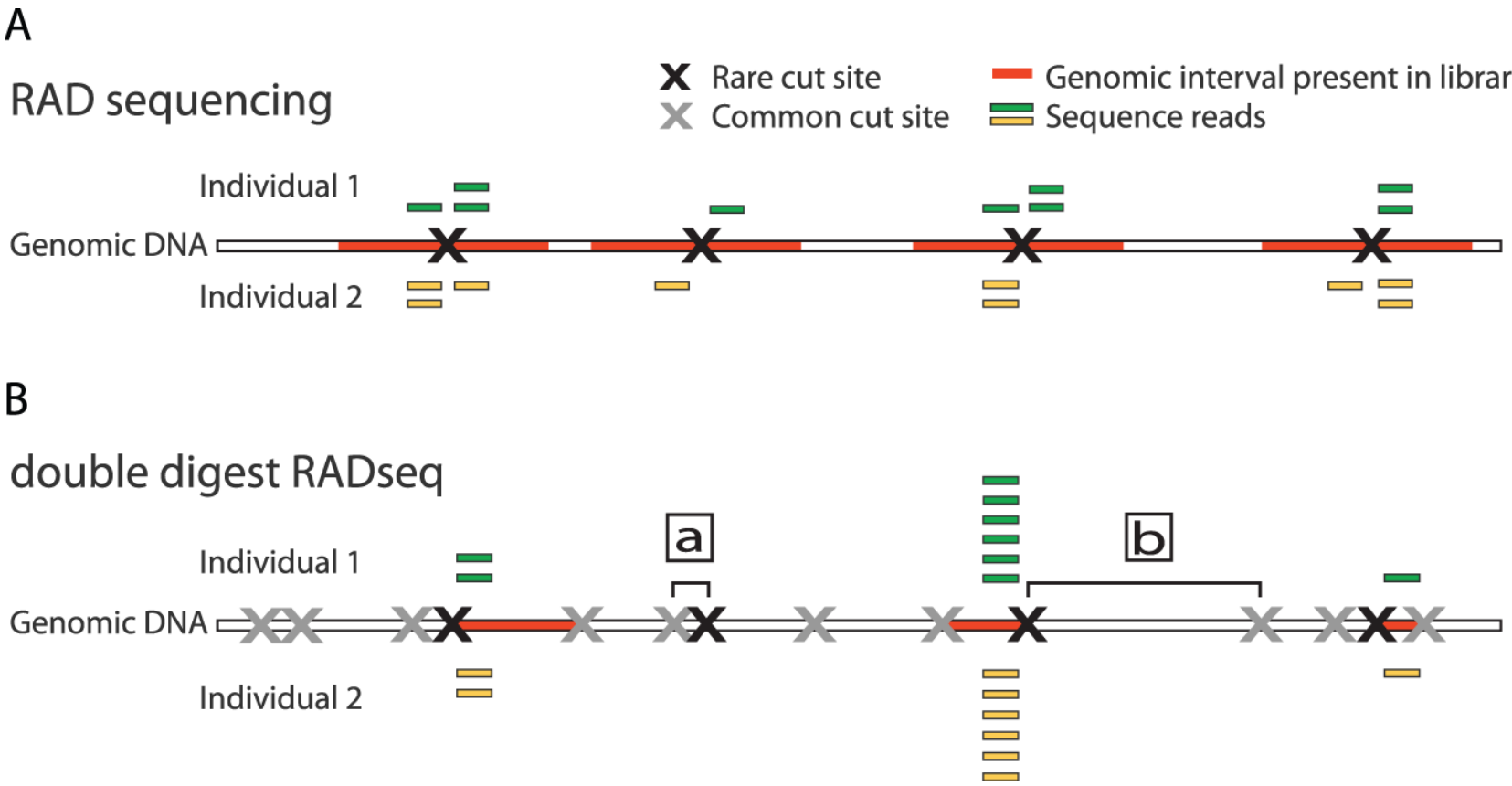


Figure 2. Double digest RAD sequencing improves efficiency and robustness while minimizing cost.
(**A**) Traditional Restriction-Site Associated DNA sequencing (RADseq) uses a single restriction enzyme (RE) digest coupled with secondary random fragmentation and broad size selection to generate reduced representation libraries consisting of all genomic regions adjacent to the RE cut site (red segments). (**B**) Double digest RAD sequencing (ddRADseq), by contrast, uses a two enzyme double digest followed by precise size selection that excludes regions flanked by either [a] very close or [b] very distant RE recognition sites, recovering a library consisting of only fragments close to the target size (red segments). Representation in this library is expected to be inversely proportional to deviation from the size-selection target, thus read counts across regions are expected to be correlated between individuals (yellow and green bars).

Sample design

Genomic considerations: (you can't control this!)

- Genome size, GC content?
- Repetitive elements?
- Ploidy?
- Reference genome?

Sampling considerations: (but you can control this…)

- Conceptualize sequence space…
- Number of samples and populations?
- Restriction enzyme(s) selected
- DNA quality

Genomic considerations

- Smaller is generally better - gives higher coverage of loci as less restriction sites in a smaller genome, so they are sequenced more often per individual
- Few repetitive elements (paralogs) better, genome duplications pose additional bioinformatic challenges
- Diploid organisms generally easier- most software written to deal with this, though there are workarounds

Sampling design considerations

- What are your questions? Pop structure/Phylogeny/ Demography − this will dictate how many individuals/ populations you need to sequence in your sampling strategy

- RAD-seq uses a rare cutter enzyme (e.g. sbfI), ddRAD would introduce a frequent cutter too − think about which to use to maximise the information you will get

- DNA quality is **CRUCIAL**! ddRAD in particular needs very high quality DNA, RAD-seq can deal with degraded samples better. You need **ACCURATE** quantification of **ALL** samples prior to any labwork. Samples will compete for sequence reads so try to have a working concentration (x ng/ul) across all samples. Run samples on a gel and also use Qubit to measure e.g. 3x and take average

In the laboratory (ca. 1 week for RAD-seq)

- Quantify your DNA and samples prior to organising RAD library construction (don't do all together). Bring 96- plates with your DNA aliquots to lab and do protocols there

- Intense concentration needed, but relaxed periods in between (e.g. waiting for PCR / gel steps), so keep alert. One small mistake can ruin hours/days of work!

- Be exceptionally consistent and accurate with pipetting especially at the adapter ligation stages − this will determine if individuals are included in your library when pooling. Also be careful with extracting the gel slices, size selection is very important

Sequencing

- It's wise to run samples on a Bioanalyzer to check the size of the fragments in the final library, sequencing facilities can do this but will charge
- Once your libaries are done you can store them in the fridge/ freezer until ready to take to the sequencing facility
- Enjoy that feeling once it's all done and take a break before…

Bioinformatics

- "Bioinformatics" is an extremely broad subject – covers all types of different analyses. For RAD-seq type data the options are usually Stacks, ipyRAD, dDocent to process data

- Downstream analyses will typically use output files from these programs but there are numerous conversion tools if needed (Plink, PGDSpider)

- Absolutely **NOT** trivial! You need to become familiar with writing code for linux/unix style environments to do the heavy lifting for you

- Also you need to become with High Performance Computing clusters and how to use them as this will be fundamental to parallelising the workload on multiple processors

Bioinformatic workflow using Stacks on a HPC cluster

RAD-seq
raw sequence
read data

HPC cluster

Stacks 1.46
Perl

- Demultiplex
- Cluster reads
- Build catalogue
- Match catalogue
- Assemble datasets

Output files for
pop structure and
phylogenetics

Afrixalus fornasini

- *Afrixalus fornasini* – East African spiny reed frog
- Subset of 17 samples here (for more manageable processing)

Received: 20 February 2018 | Revised: 8 August 2018 | Accepted: 29 August 2018

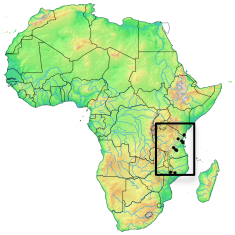
DOI: 10.1111/mec.14862

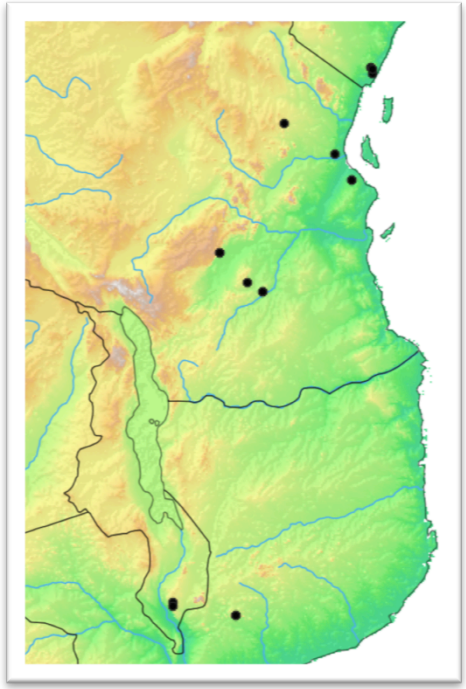
ORIGINAL ARTICLE

WILEY MOLECULAR ECOLOGY

Vanishing refuge? Testing the forest refuge hypothesis in coastal East Africa using genome wide sequence data for seven amphibians

Christopher D. Barratt[1,2] | Beryl A. Bwong[1,3] | Robert Jehle[4] | H. Christoph Liedtke[1,5] | Peter Nagel[1] | Renske E. Onstein[2] | Daniel M. Portik[6,7] | Jeffrey W. Streicher[8] | Simon P. Loader[1,8]





Zoologica Scripta / Volume 48, Issue 2

ORIGINAL ARTICLE | Full Access

Causes and analytical impacts of missing data in RADseq phylogenetics: Insights from an African frog (*Afrixalus*)

Marco Crotti, Christopher D. Barratt, Simon P. Loader, David J. Gower, Jeffrey W. Streicher

First published: 21 January 2019

https://doi.org/10.1111/zsc.12335

Example files – 3 download directories + example out

- **data**
  (where the raw data is held)

```
cd /data/idiv_kuehl/
mkdir chris_data
cd chris_data
wget https://www.dropbox.com/s/9snla69bxs9n43m/data.tar.gz
tar –xzf data.tar.gz
rm data.tar.gz
```

- **submit_scripts**
  (all scripts to process the data)

```
cd ~
mkdir submit_scripts
cd submit_scripts
wget https://www.dropbox.com/s/6pn4brtcdd4jfzj/submit_scripts.tar.gz
tar -xzf submit_scripts.tar.gz
rm submit_scripts.tar.gz
```

- **Stacks**
  (output folder)

```
cd /work/barratt
wget https://www.dropbox.com/s/mlaggiw793jxgjz/Stacks.tar.gz
tar –xzf Stacks.tar.gz
rm Stacks.tar.gz
```

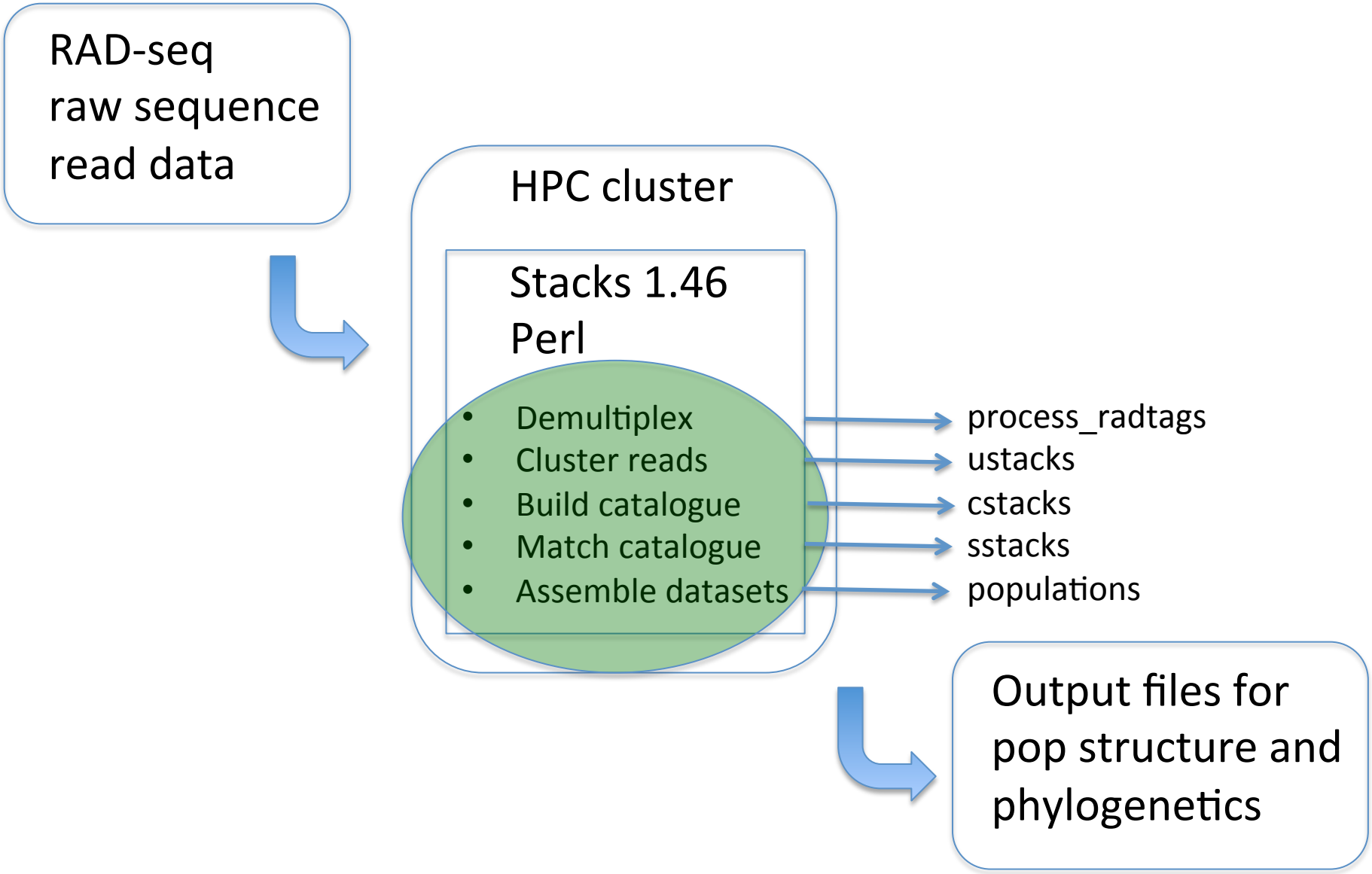
- **Stacks_example_completed**
  (output folder)

```
cd /work/barratt
wget https://www.dropbox.com/s/k7r0xufuot3vx6f/Stacks_example_completed.tar.gz
tar –xzf Stacks_example_completed.tar.gz
rm Stacks_example_completed.tar.gz
```

Stacks (http://catchenlab.life.illinois.edu/stacks/)

- Programmed and maintained by Julian Catchen
- Designed specifically for short read RAD-seq type data
- Written in C++ and Perl
- Well established pipeline which has not changed too much, cited many times
- Good google user group (very useful!)
- Denovo_map.pl, ref_map.pl or modular system (process_radtags, ustacks, cstacks, sstacks, populations) - can rerun any part

Stacks


RAD-seq
raw sequence
read data
HPC cluster
Stacks 1.46
Perl
Demultiplex
Cluster reads
Build catalogue
Match catalogue
Assemble datasets
process_radtags
ustacks
cstacks
sstacks
populations
Output files for
pop structure and
phylogenetics

Shell scripts

- Shell scripts let you send your jobs to the HPC cluster

- Usually one script per individual task, though you can make special scripts which will submit multiple jobs in sequence (see later)

- I have a main script header which I edit for each individual job:
  - Job name
  - Notifications (starts, ends, aborts, pauses etc.) and email address where to send them
  - Input/Output directories and where to place job logfile
  - Requirements − processors needed, amount of memory per processor and runtime (used for queueing)

  - Followed by the main lines of code − load programs and relevant instructions

Shell scripts

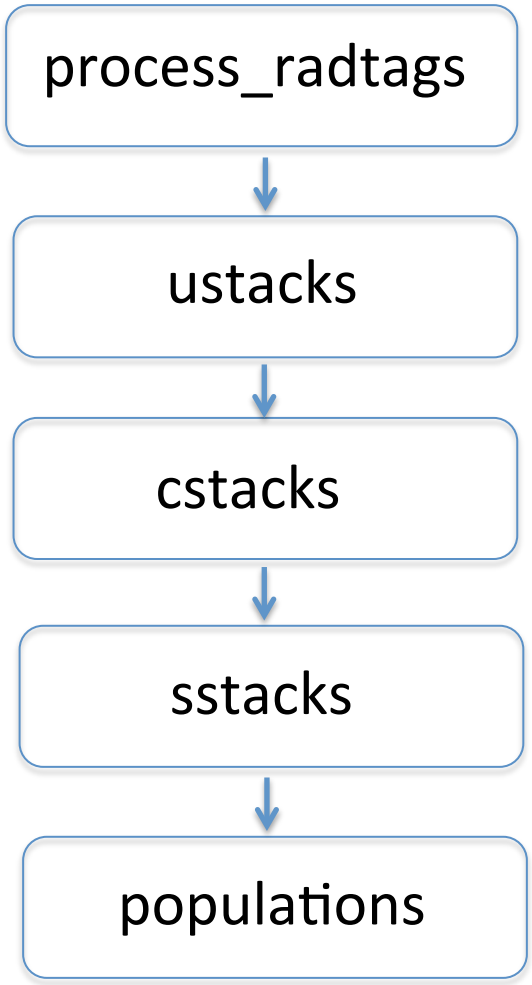
TextWrangler File Edit Text View Search Go Window #! Help

1_process_radtags.sh

/private/var/folders/rp/_4tmp0_d3sv44byf1__k2w5w0000gn/T/0f8aa5cb-2a4a-4d58-a70f-da82442a5887/gpfs0/home/barratt/s.../...

```
1  ####### UFZ cluster job submit #########
2
3  #!/bin/bash
4  #$ -N process_radtags
5  #$ -M christopher_david.barratt@uni-leipzig.de
6  #$ -m beas
7  #$ -wd /work/$USER
8  #$ -S /bin/bash
9  #$ -o /work/$USER/Stacks/Afornasini/job_logs/$JOB_NAME-$JOB_ID.log
10 #$ -j y
11 #$ -pe smp -1
12 #$ -l h_vmem=20G
13 #$ -l highmem
14 #$ -l h_rt=24:00:00
15
16 #get slot information etc. (determine no. of slots with -pe above (can also be a range so
   … that it takes the maximum available slots but will settle for less))
17 echo "Job name: $JOB_NAME"
18 echo "$NSLOTS slots have been dedicated to this job"
19 echo "On following node(s)"
20 cat $PE_HOSTFILE
21 echo "pe host file is $PE_HOSTFILE"
22
23 #load environment
24 source ~/barratt_env/bin/activate
25
26 #load modules, start job
27 cd /work/$USER/Stacks/Afornasini/
28 module load stacks/1.46-1
29 module load perl/5.16/3-1
30
31 # process_radtags - take raw data from sequencer and output sequences per individual using
   … barcodes and restriction enzyme information
32 # -p = file path of raw data from sequencer
33 # -o = output file path of where you want your individual data to go
34 # -b = barcode identifier (one file showing all barcodes for each individual)
35 # -e = restriction enzyme used
36 # -c = clean uncalled sequences
37 # -q - discard low quality sequences
38
39 process_radtags -p /data/idiv_kuehl/chris_data/Stacks/Afornasini/raw_data_from_sequencer/
   … -o /data/idiv_kuehl/chris_data/Stacks/Afornasini/data/ -b
   … /data/idiv_kuehl/chris_data/Stacks/Afornasini/barcodes -e sbfI -c -q
40
```

Stacks



Demultiplex (all data) – feed in adapter barcodes to get *.fq.gz file per individual (only run once per dataset)

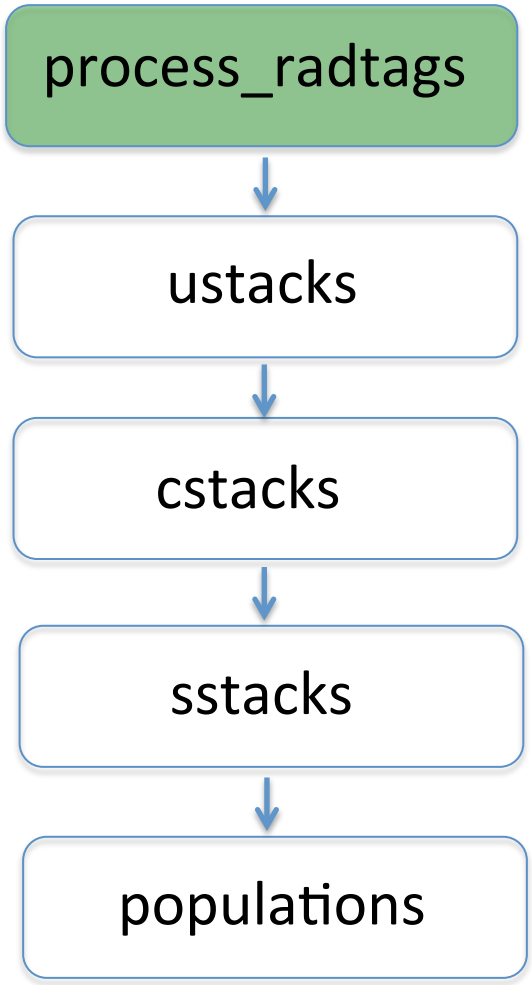
Make “stacks” of common sequences in each individual file – this module is run for each individual sample

Build catalogue library of all stacks from all individuals (run once per dataset)

Match stacks in each individual to the library catalogue – this module is run for each individual sample

Output file generation based on thresholds defined by user and a population map (e.g. only output loci which are present in at least 50% of all samples)

Stacks



Demultiplex (all data) – feed in adapter barcodes to get *.fq.gz file per individual (only run once per dataset)

. . .

- >40 Gigabytes files with all sequence data! You **CANNOT** open this in any editor :-0
- You need to tell Stacks which sequence belongs to which individual using barcode adapter sequences and restriction enzyme cut sequences
- qsub 1_process_radtags.sh (we don't do this step today as our data are already demultiplexed, but I have given you a script)

Stacks

process_radtags

↓

ustacks

↓

cstacks

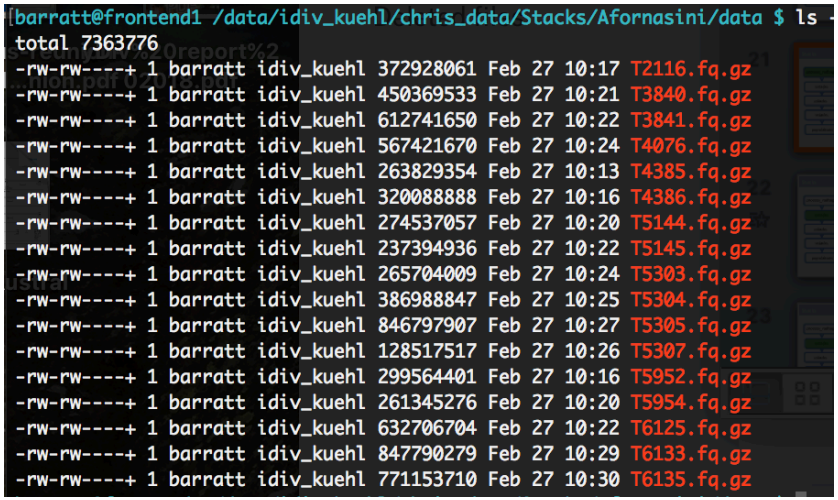
↓

sstacks

↓

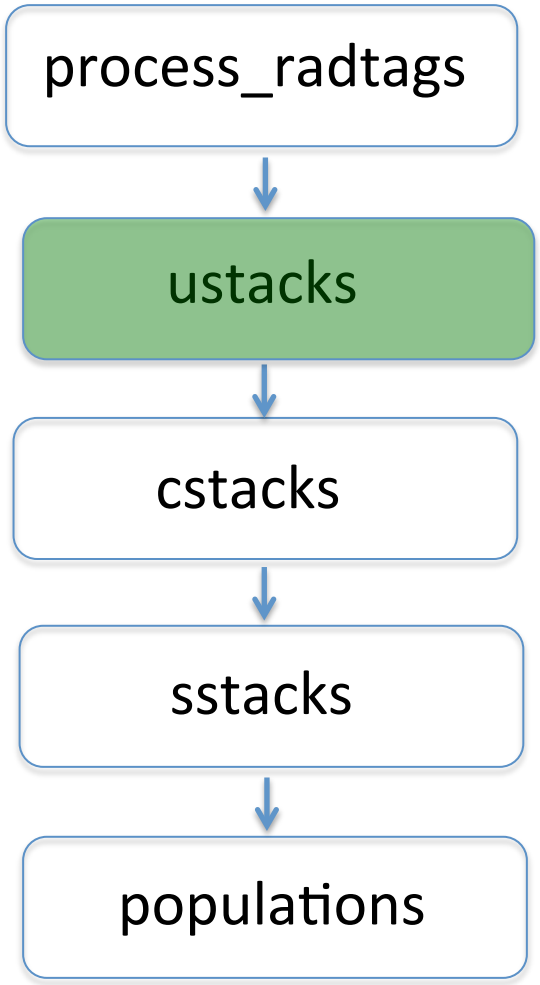
populations

Demultiplex (all data) – feed in adapter barcodes to get *.fq.gz file per individual (only run once per dataset)

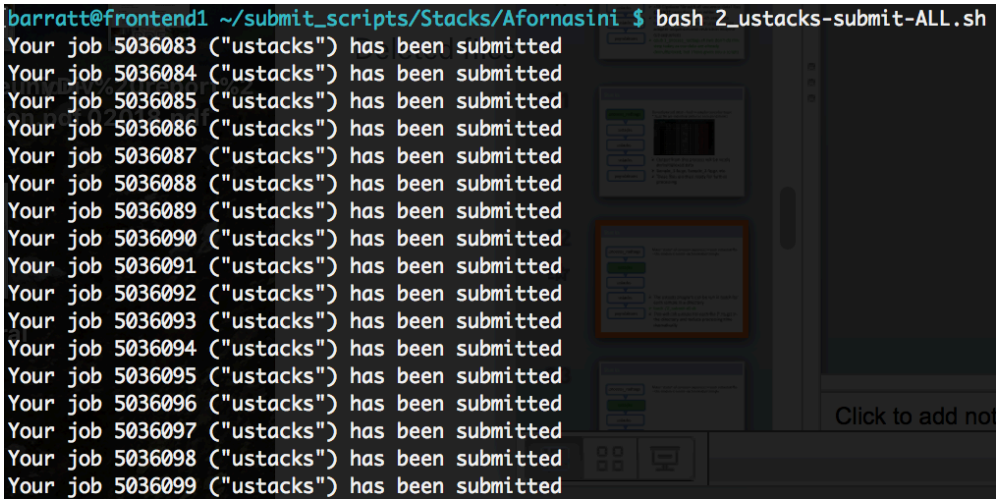


- Output from this process will be nicely demultiplexed data
- Sample_1.fq.gz, Sample_2.fq.gz, etc.

Stacks

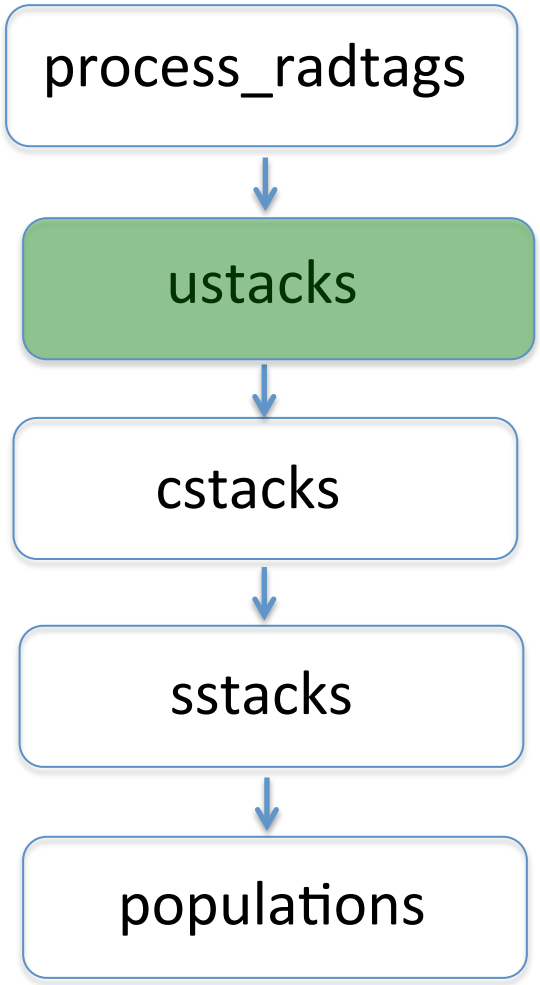


Make “stacks” of common sequences in each individual file – this module is run for each individual sample

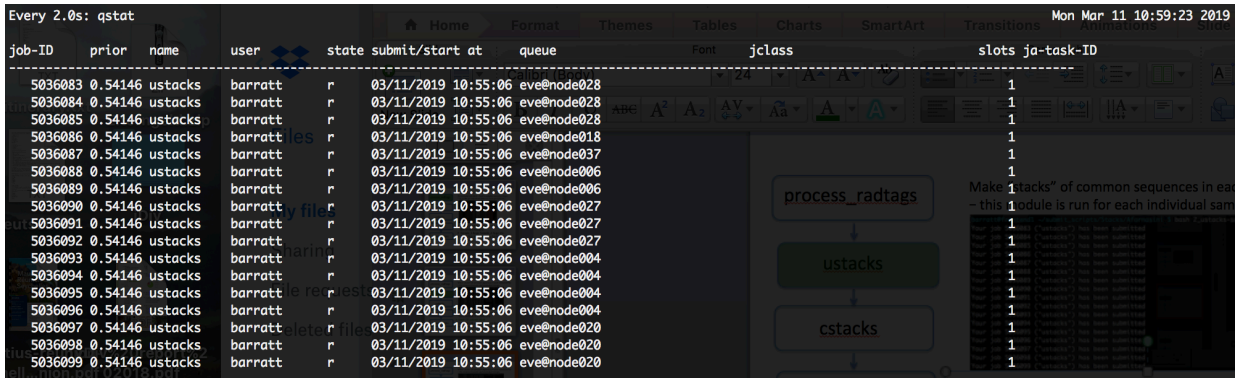


- The ustacks program can be run in batch for each sample in a directory
- bash ./2_ustacks-submit-ALL.sh
- This will call ustacks for each file (*.fq.gz) in the directory and reduce processing time dramatically

Stacks

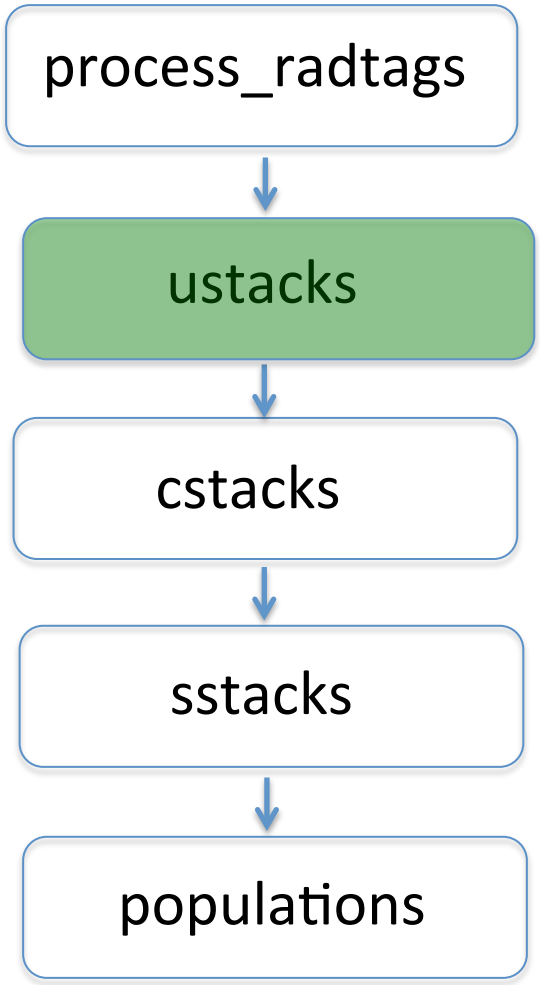


Make “stacks” of common sequences in each individual file – this module is run for each individual sample



- The ustacks program can be run in batch for each sample in a directory
- bash ./2_ustacks-submit-ALL.sh
- Keep track of progress with watch qstat

Stacks

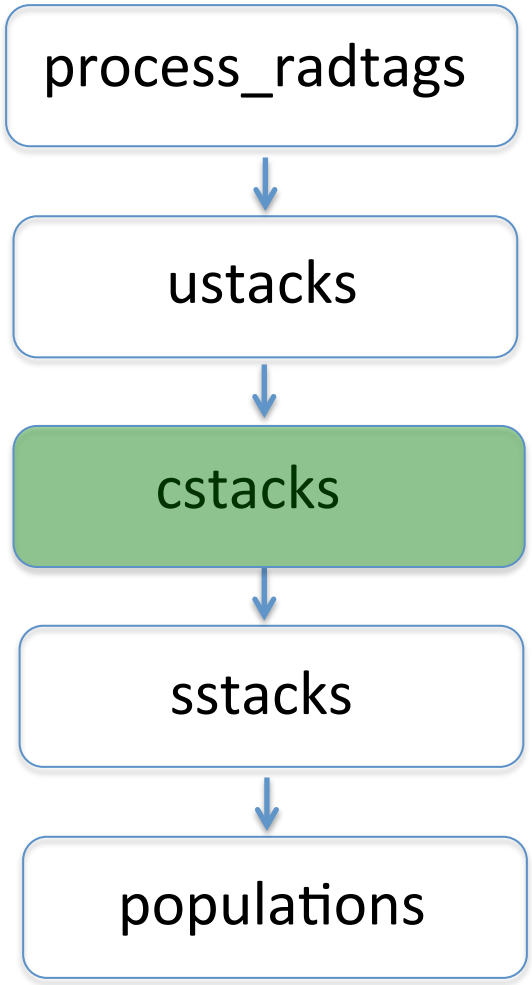


Make “stacks” of common sequences in each individual file – this module is run for each individual sample

```
[barratt@frontend1 /work/barratt/Stacks/Afornasini $ ls -l
total 38016
drwxr-sr-x 2 barratt root     4096 Mar 11 10:55 job_logs
drwxr-sr-x 2 barratt root     4096 Mar 11 10:07 popmaps
-rw-r--r-- 1 barratt root    35325 Mar 11 10:59 T5307.alleles.tsv.gz
-rw-r--r-- 1 barratt root  2672284 Mar 11 10:59 T5307.models.tsv.gz
-rw-r--r-- 1 barratt root 21647229 Mar 11 10:59 T5307.snps.tsv.gz
-rw-r--r-- 1 barratt root 14437608 Mar 11 10:59 T5307.tags.tsv.gz
```

- Output from this process will be 4 files for each individual sample:
  - Sample_1.alleles.tsv.gz
  - Sample_1.models.tsv.gz
  - Sample_1.tags.tsv.gz
  - Sample_1.snps.tsv.gz

Stacks

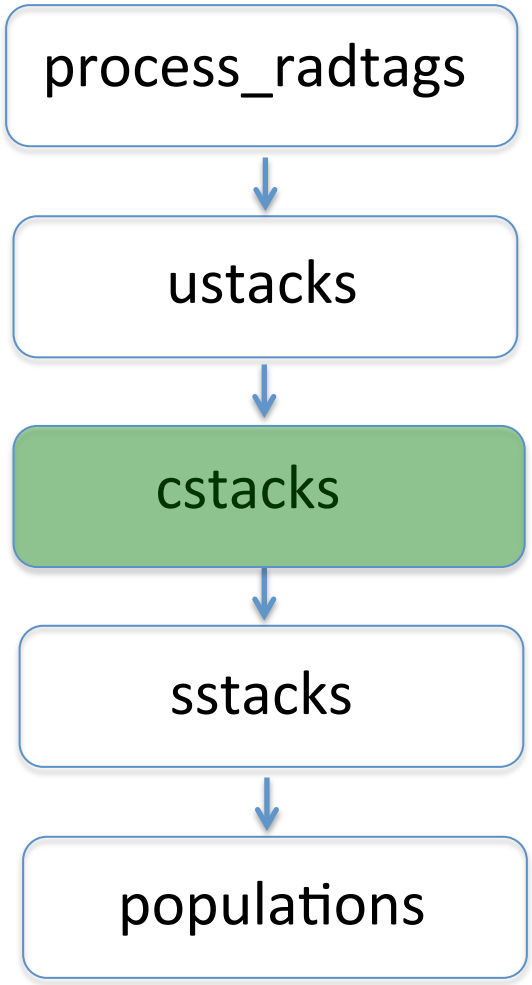


Build catalogue library of all stacks from all individuals (run once per dataset)

```
barratt@frontend1 ~/submit_scripts/Stacks/Afornasini $ qsub 3_cstacks-submit.sh
Your job 5036104 ("cstacks") has been submitted
```

- Now you will build the catalogue library (best to do it from all samples)
- qsub 3_cstacks-submit.sh
- This will take several hours, and should be done on a single processor core

Stacks

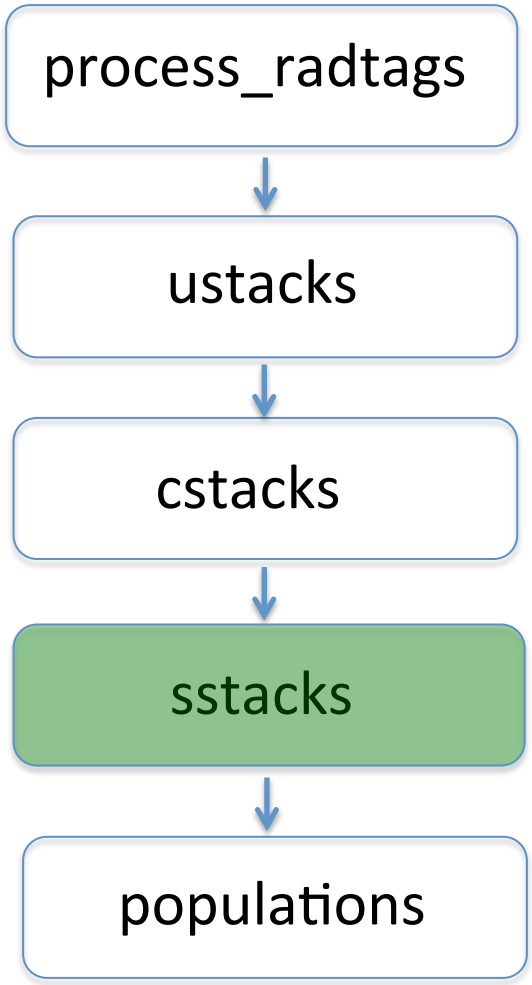


Build catalogue library of all stacks from all individuals (run once per dataset)

```
-rw-r--r-- 1 barratt root   1851976 Mar  1 14:57 batch_1.catalog.alleles.tsv.gz
-rw-r--r-- 1 barratt root   2046139 Mar  1 14:57 batch_1.catalog.snps.tsv.gz
-rw-r--r-- 1 barratt root  63171648 Mar  1 14:57 batch_1.catalog.tags.tsv.gz
```

- Output from this are 3 large files containing all loci from all individuals:
  - batch_1_alleles.tsv.gz
  - batch_1_catalog.tsv.gz
  - batch_1_snps.tsv.gz

Stacks

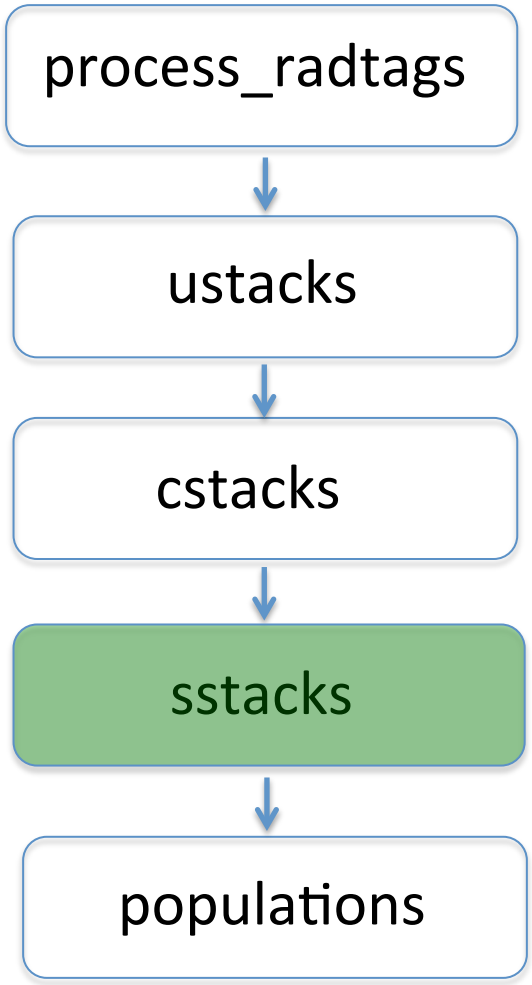


Match stacks in each individual to the library catalogue – this module is run for each individual sample

```
barratt@frontend1 ~/submit_scripts/Stacks/Afornasini $ bash 4_sstacks-submit_ALL.sh
Your job 5038527 ("sstacks") has been submitted
Your job 5038528 ("sstacks") has been submitted
Your job 5038529 ("sstacks") has been submitted
Your job 5038530 ("sstacks") has been submitted
Your job 5038531 ("sstacks") has been submitted
Your job 5038532 ("sstacks") has been submitted
Your job 5038533 ("sstacks") has been submitted
Your job 5038534 ("sstacks") has been submitted
Your job 5038535 ("sstacks") has been submitted
Your job 5038536 ("sstacks") has been submitted
Your job 5038537 ("sstacks") has been submitted
Your job 5038538 ("sstacks") has been submitted
Your job 5038539 ("sstacks") has been submitted
Your job 5038540 ("sstacks") has been submitted
Your job 5038541 ("sstacks") has been submitted
Your job 5038542 ("sstacks") has been submitted
Your job 5038543 ("sstacks") has been submitted
```

- Like ustacks, sstacks can be run in batch for each sample in a directory
- bash ./4_sstacks-submit-ALL.sh
- This will call sstacks for each file (*.fq.gz) in the directory and match all loci to the catalogue, again reducing processing time

Stacks



Match stacks in each individual to the library catalogue – this module is run for each individual sample

```
-rw-r--r-- 1 barratt root    114131 Mar  1 11:24 T2116.alleles.tsv.gz
-rw-r--r-- 1 barratt root    994821 Mar  1 16:11 T2116.matches.tsv.gz
-rw-r--r-- 1 barratt root   6987922 Mar  1 11:24 T2116.models.tsv.gz
-rw-r--r-- 1 barratt root  57001018 Mar  1 11:24 T2116.snps.tsv.gz
-rw-r--r-- 1 barratt root  41478727 Mar  1 11:24 T2116.tags.tsv.gz
-rw-r--r-- 1 barratt root    146455 Mar  1 11:34 T3840.alleles.tsv.gz
-rw-r--r-- 1 barratt root   1175299 Mar  1 16:11 T3840.matches.tsv.gz
-rw-r--r-- 1 barratt root   7918844 Mar  1 11:34 T3840.models.tsv.gz
-rw-r--r-- 1 barratt root  64695279 Mar  1 11:34 T3840.snps.tsv.gz
-rw-r--r-- 1 barratt root  47519361 Mar  1 11:34 T3840.tags.tsv.gz
```

- Output from this process is a matches file, so each individual will have a file like this as well as the other 4 files from ustacks:
- Sample_1.matches.tsv.gz

Stacks

process_radtags

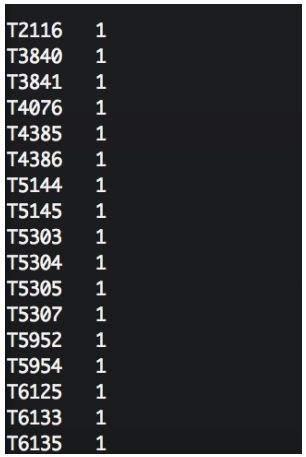
ustacks

cstacks

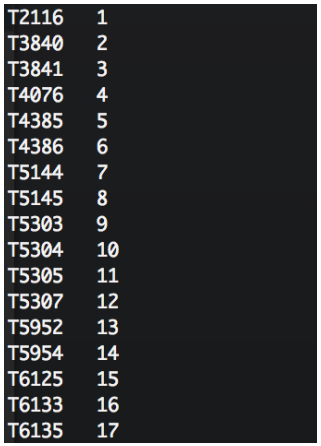
sstacks

populations

Output file generation based on thresholds defined by user (e.g. only output loci which are present in at least 50% of all samples)



T2116	1
T3840	1
T3841	1
T4076	1
T4385	1
T4386	1
T5144	1
T5145	1
T5303	1
T5304	1
T5305	1
T5307	1
T5952	1
T5954	1
T6125	1
T6133	1
T6135	1



T2116	1
T3840	2
T3841	3
T4076	4
T4385	5
T4386	6
T5144	7
T5145	8
T5303	9
T5304	10
T5305	11
T5307	12
T5952	13
T5954	14
T6125	15
T6133	16
T6135	17

- You're ready to generate your output files!
- Here you specify a population map with the samples and their origin. The parameters in the submit script and/or popmap can be adjusted to suit your needs
- qsub 5_populations-structure.sh
- qsub 5_populations-phylogeny.sh

Stacks

process_radtags

ustacks

cstacks

sstacks

populations

Output file generation based on thresholds defined by user (e.g. only output loci which are present in at least 50% of all samples)

```
-rw-r--r-- 1 barratt root    452755 Mar 11 14:02 phylogeny_50.haplotypes.tsv
-rw-r--r-- 1 barratt root   5542530 Mar 11 14:02 phylogeny_50.hapstats.tsv
-rw-r--r-- 1 barratt root     70136 Mar 11 14:02 phylogeny_50.phylip
-rw-r--r-- 1 barratt root    376576 Mar 11 14:02 phylogeny_50.phylip.log
-rw-r--r-- 1 barratt root      4046 Mar 11 14:02 phylogeny_50.populations.log
-rw-r--r-- 1 barratt root      7042 Mar 11 14:02 phylogeny_50.sumstats_summary.tsv
-rw-r--r-- 1 barratt root   8663523 Mar 11 14:02 phylogeny_50.sumstats.tsv
```

```
-rw-r--r-- 1 barratt root    452822 Mar 11 14:06 structure_50.haplotypes.tsv
-rw-r--r-- 1 barratt root    599132 Mar 11 14:06 structure_50.hapstats.tsv
-rw-r--r-- 1 barratt root      1570 Mar 11 14:06 structure_50.populations.log
-rw-r--r-- 1 barratt root    597825 Mar 11 14:06 structure_50.structure.tsv
-rw-r--r-- 1 barratt root       808 Mar 11 14:06 structure_50.sumstats_summary.tsv
-rw-r--r-- 1 barratt root    883884 Mar 11 14:06 structure_50.sumstats.tsv
```

- And here are your output files
- structure_50.structure.tsv – e.g. Structure
- phylogeny_50.phylip – e.g. RAxML
- The output files you need depends on the analyses you will do, but process is the same