

Stacks Analiz sırası	
1- Process_radtag	Genomik kütüphane önce <i>processle radtag</i> ile demultiplex edilecek. Analizlerin en baştaki belki de en önemli basamağı, örneklerin barkod eşleştirmesini ya da isimlendirmeleri titiz bir şekilde düzenlenmeli hata yapılmamalıdır.
2- demultiplex sonrası okumalardaki kalite kontrolü	Analizi yapılacak tüm örnekler ait okumaları bir araya topla. Eğer birden fazla kütüphanede dağılmış örneklerin varsa tek bir klasörde birleştir. Aynı şekilde process-radtag çıktısındaki örneklerine ait filtrasyon ve okuma bilgilerini bir excelde topla ve kontrol et. Çok düşük okumada olan bireyleri veriden ele ve klasöründen sil diğerleri ile lokus inşasına başla.
3-ustacks veya pstacks	Eldeki örneklerin okumalarıyla ustacks basamağını gerçekleştir önce default parametreleri dene sonra tüm basamaklar bittikten soran parametre değişimini (-m, -M ve gap) veri setinden en iyi verimi alacağın şekilde ayarlamaya çalış. Bu basamaktaki output dosyasındaki coverege ve standart sapma değerlerini excele geçir, farklı parametre uygulamalarında karşılaştırma için gerekli ve yayında tablo olarak vereceksin. Eğer referans genomun varsa pstacks kullanacaksın
4-cstacks	Bireylere ait okumalardan katalog oluşturulacak ve tüm basamaklar içerisinde en uzun olanıdır. Burada sadece -n ve gap parametreleri değiştirilebilir.
5-sstacks	Bu aşamada bireylere ait lokuslar kataloglarla karşılaştırılır, lokuslar ve haplotipler inşa edilir. Parametreleri değiştirerek (-m, -M ve gap) denemelerle veri setinden en verimli şekilde yararlanılacak parametre değeri belirlenecek.
6- Filtration-I Populations +R	Filtrasyon. Bu basamakta population programı ile -p -m -r parametreleri ile ilk filtrasyonu sonra oluşan vcf file ile R da sondaki bazlar ve thetaya göre aşırı değişken lokuslar filtre edilecek ve whitelist oluşturulacak.
7- Filtration-II analizler için veri dosyalarının oluşturulması Populations	Oluşturulan whitelist yapılmak istenen analize göre ikinci filtrasyon yapılarak ya da direk olarak analizler için veri dosyaları oluşturulacak.
8- Plink ile ileri filtrasyon	Oluşturulan veri dosyalarından analizlere uygun filtrasyonlar yapılacak.

Stacks (http://catchenlab.life.illinois.edu/stacks/)
Manuel http://catchenlab.life.illinois.edu/stacks/manual/#wl http://catchenlab.life.illinois.edu/stacks/faq.php
Other pipelines are available to produce genotype information in groups of individuals. Two of the most widely used are SAMtools/BCFtools (Li et al. 2009) and the Genome Analysis Toolkit (GATK, McKenna et al. 2010). These tools are meant to operate on top of a genome, for example by detecting nucleotide variants through matches to the reference sequence. GATK, in particular, is highly optimized to work on the human genome. In contrast, Stacks was developed to have at its core a catalogue that works as an internal reference for each project regardless of the presence of a genome. Even when a reference genome is used to stack reads, nucleotide variants are still identified de novo. The catalogue approach is particularly useful for the majority of organisms for which a reference genome does not exist or is in a draft state. Furthermore, SAMtools/BCFtools and GATK can call SNPs in multiple samples and can generate allele frequencies, but there is no built-in concept of populations. Instead, populations are managed by hand as collections of BAM and VCF files, as compared to the integrated way that this occurs in Stacks.
This pseudo-testcross format requires a large number of genetic markers, such as is provided by the RAD genotyping platform
Dataset: single-end ddRADseq

1- process_radtags

Ayrıntılı bilgi için: http://catchenlab.life.illinois.edu/stacks/comp/process_radtags.php
<http://catchenlab.life.illinois.edu/stacks/manual/#wl>

In a typical analysis, data will be received from an Illumina sequencer, or some other type of sequencer as FASTQ files. The first requirement is to demultiplex, or sort, the raw data to recover the individual samples in the Illumina library. While doing this, we will use the [Phred](#) scores provided in the FASTQ files to discard sequencing reads of low quality. These tasks are accomplished using the process_radtags program

The process_radtags program can:

- handle data that is barcoded, either inline or using an index, or unbarcoded.
- use combinatorial barcodes.
- check and correct for a restriction enzyme cutsite for single or double-digested data.
- filter adapter sequence while allowing for sequencing error in the adapter pattern.
- process individual files or whole directories of files.
- directly read/write gzipped data
- filter reads based on Illumina's Chastity filter.
- name output files according to their sample names instead of barcode names, if supplied.

Aşağıda verisetinin demultipleks edilmesi için bizim kullanacağımız komut bloğu

Not1: analizler truba/scratch/.. dizini içerisinde çalıştırılacak

Not2: işi kümeye yüklemeyen önce hazırlanan komut bloğunu önce interaktif çalıştır, sorun var mı, çalışıyor mu diye gör, bir süre devam etsin sonra ctrl+c ile sonlandır. Bu işlemten sonra iş olarak kümeye yükle.

Not3: Kütüphaneyi indirdikten sonra burada dikkat edeceğin nokta R1 kısaltmasını değiştirmemen gerek çünkü program R1 ve R2 yi paired end olarak algılar.

```
process_radtags -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/1-  
process_radtag/Lib_1_P_lusch_srp_R1.fastq.gz -i gzfastq -b  
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/1-process_radtag/lusch_barcode.txt -o  
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/1-process_radtag/outstacks/ -e ecoRI -q -r -E phred33  
-D --inline_null --adapter_1 AGATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG --adapter_mm 1 --barcode_dist 1 2 --  
filter_illumina
```

-f	path to the input file if processing single-end sequences
-i	input file type, either 'bustard' for the Illumina BUSTARD format, 'bam', 'fastq' (default), or 'gzfastq' for gzipped FASTQ.
-b	path to a file containing barcodes for this run.
-o	path to output the processed files.
-q	discard reads with low quality scores
-r	rescue barcodes and RAD-Tags
-E	specify how quality scores are encoded, 'phred33' (Illumina 1.8+, Sanger, default) or 'phred64' (Illumina 1.3 - 1.5).
-D	capture discarded reads to a file

Barcode options:

--inline_null barcode is inline with sequence, occurs only on single-end read (default)

Restriction enzyme options:

-e [enz], --renz_1 [enz] provide the restriction enzyme used (cut site occurs on single-end read)

Adapter options:

--adapter_1 [sequence] provide adaptor sequence that may occur on the single-end read for filtering

--adapter_mm [mismatches] number of mismatches allowed in the adapter sequence

Advanced options:

--barcode_dist provide the distance between barcodes to allow for barcode rescue (default 2)

--filter_illumina discard reads that have been marked by Illumina's chastity/purity filter as failing.

The barcodes file looks like this:

```
% more ./barcodes/barcodes
CGATA<tab>ACGTA<tab>sample_01
CGGCG      ACGTA      sample_02
CGATA      TAGCA      sample_03
CGGCG      TAGCA      sample_04
```

Bizim örneğimizde kullandığımız barkod listesi

```
-bash-4.2$ cat lusch_barcode.txt
ACGAGTGCCTC led-1
AGCACTGTAGC orb-1
CTCGCTGTGCC tun-1
CATAGTAGTGC egr-1
TCACGTACTAC heller-1
TGTACTACTCC birter-1
TAGAGACGAGC birbak-1
CACGCTACGTC birkem-1
TCTAGCGACTC lusedavz-1
CGCGTATACAC luslagn-1
ACACATACGCC luspat-1
ACTCGCGCACC chobakd-1
CGTCGATCTCC chobtylos-1
TAGACTGCACC chobesn3-1
-bash-4.2$
```

Process radtag ile demultiplex işlemini aşağıdaki slurm komut bloğu ile birlikte sisteme yükledik

```
#!/bin/bash : klasik Batch başlangıcı, olması şart
#SBATCH -p mercan : işin yüklendiği partition/sunucu adı
#SBATCH -A sarkaya : kullanıcı adı
#SBATCH -J process_radtag : yüklenen işin adı
#SBATCH -N 1 : kaç adet nod istediğini gösterir burada bir adet
#SBATCH -n 24 : işe kaç adet çekirdek atadığını gösterir (mercanda her çekirdek 5gb 5x 24=120Gb ram atadık)
#SBATCH --time=05:00:00 : işin tahmini ne kadar sürede biteceğini belirttik
#SBATCH --mail-type=ALL : başlama, sonlanma ve oluşabilecek sorunlara yönelik e-posta gönder
#SBATCH --mail-user=kaya_sarp@hotmail.com : bu e-posta adresine gönder

export OMP_NUM_THREADS=24

echo "SLURM_NODENAME $SLURM_NODENAME"
echo "NUMBER OF CORES $SLURM_NTASKS"

cd /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/1-process_radtag :analizin gerçekleştirildiği yeri gösterir
```

Analizin yapılacağı stacks komut bloğu

```
process_radtags -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/1-
process_radtag/Lib_1_P_lusch_srp_R1.fastq.gz -i gzfastq -b /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/1-
process_radtag/lusch_barcode.txt -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/1-process_radtag/outstacks/
-e ecoRI -q -r -E phred33 -D --inline_null --adapter_1 AGATCGGAAGAGCTCGTATGCCGCTCTCTGCTTG --adapter_mm 1 --
barcode_dist_1 2 --filter_illumina 2>&1
exit
```

İşi kümeye yüklemeyen önce eğer stack kümeye admin tarafından yüklenmişse stack modülünü yüklememiz gerekir. Bazı durumlarda modül yükleme komutunu slurm iş yükleme komut bloğuna eklemek gerekir.

1- öncelikle stacks kümede yüklü mü ona bakın (eğer kendi home dizininizde yüklü değilse)

module av stacks : bu komutu kopyalayıp terminale yapıştırıp enter yap (inreraktif kullanım)

2- sonrasında stacks programını çağırırız

module load stacks/1.46 : bu komutu kopyalayıp terminale yapıştırıp enter yap (inreraktif kullanım)

3- komut bloğunun çalışıp çalışmadığını bir sorun olup olmadığına bakarız. Bunun için stacks komut bloğunu interaktif olarak çalıştırırız

Sorun yoksa ctrl+c ile işlemi sonlandır ve işi sisteme sbatch ile yükle

Sonrasında process_radtag.slurm iş yükleme komut bloğu **sbatch** komutu ile kümede sıraya yüklenir

Submitted batch job 5619284

Sonrasında **squeue** komut ile işin koşturulmaya başlayıp başlamadığı kontrol edilir

İşi mercan79 nolu nodda koşturulmakta

```
-bash-4.2$ ls
Lib_1_P_lusch_srp_R1.fastq.gz lusch_barcode.txt outstacks process_radtag.slurm
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$ sbatch process_radtag.slurm
Submitted batch job 5619284
-bash-4.2$
-bash-4.2$
-bash-4.2$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
5619284 mercan process_ sarkaya R 0:06 1 mercan79
-bash-4.2$
-bash-4.2$
```

Not: işi kümeye yüklemeyen önce aşağıdaki komutla hangi partititonların boş olup olmadığını görerek o kuyruğa işi gönderebilirsin

sinfo --states=IDLE

```
-bash-4.2$ sinfo --states=IDLE
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
cuda      up 15-00:00:0  2    idle levrek[134,137]
single    up 15-00:00:0  0    n/a
short     up  4-00:00:0  0    n/a
mid1      up  4-00:00:00  0    n/a
mid2      up  8-00:00:00  0    n/a
long      up 15-00:00:0  0    n/a
levrekv2  up 15-00:00:0  0    n/a
mercan*   up 15-00:00:0  33   idle mercan[24,35,37-38,43,47,57-59,70-78,89-92,104,124,127-129,136,141,160-161,178,185]
smp       up  8-00:00:00  0    n/a
sardalya  up 15-00:00:0  0    n/a
-bash-4.2$
```

Görüldüğü gibi mercanda 33 nod boş cuda da 2 tane diğer sunuculardaki nodlar dolu.

Process rad tag sonuçlarının elde edilen dosyalar

```
-bash-4.2$
-bash-4.2$ ls outstacks/
birbak_1.fq.gz  chobakd_1.fq.gz  egr_1.fq.gz  Lib_1_P_lusch_srp_R1.fastq.gz.discards  luspap_1.fq.gz  tun_1.fq.gz
birkem_1.fq.gz  chobesn3_1.fq.gz  heller_1.fq.gz  lusedavz_1.fq.gz  orb_1.fq.gz
barter_1.fq.gz  chobtylos_1.fq.gz  led_1.fq.gz  luslagn_1.fq.gz  process_radtags.1-process_radtag.log
-bash-4.2$
-bash-4.2$
```

Analiz yaklaşık 4dk sürdü, bu veri single end eğer veri paired end olsaydı bunun iki katı veri olacaktır. Analiz sonrası oluşturulan **process_radtags.log** dosyası excelde açılır. Bu dosyada veri hakkında bilgileri ne kadar dizi olduğunu ne kadarının temiz ne kadarının filtre edildiğini ve her bir bireydeki okuma miktarlarını verir.

Process_radtag dan sonra outstacks dosyası içerisinde bulunan **process_radtags.log** dosyasını excel'de aç. Bu dosyada demultipleks ve filtrasyon sonrası hem kütüphanedeki total okumalara hemde her bir bireye ait okumalara ait bazı istatistikler bulunur. Sonuçları excelde histogram ya da pasta grafik yaparak daha anlaşılır kılabilirsin.

Burada ne kadar okumanın yeterli olacağı ya da alt sınır okuma sayısının ne olacağını belirlemek canlıların genomuna kütüphaneyi hazırlayanın deneyimine ve okutulan cihaza bağlı olarak değişir. Ancak bazı verilerde en az 124000 okuma gibi bir sayı fena değildir ancak 500bin altında okuması gelen birey çok verimli olmayabilir.

Bu basamakta okuması çok düşük bireyler analizlerden uzaklaştırılabilir ancak okumaları az olsa da çok düşük olmayanlar ilerleyen basamaklarda elde edilen lokus sayısına bakılarak veriden uzaklaştırılması daha iyi olur.

2- Ustacks (unique stacks)

Ayrıntılı bilgi için: <http://catchenlab.life.illinois.edu/stacks/comp/ustacks.php>

<http://catchenlab.life.illinois.edu/stacks/manual/#w>

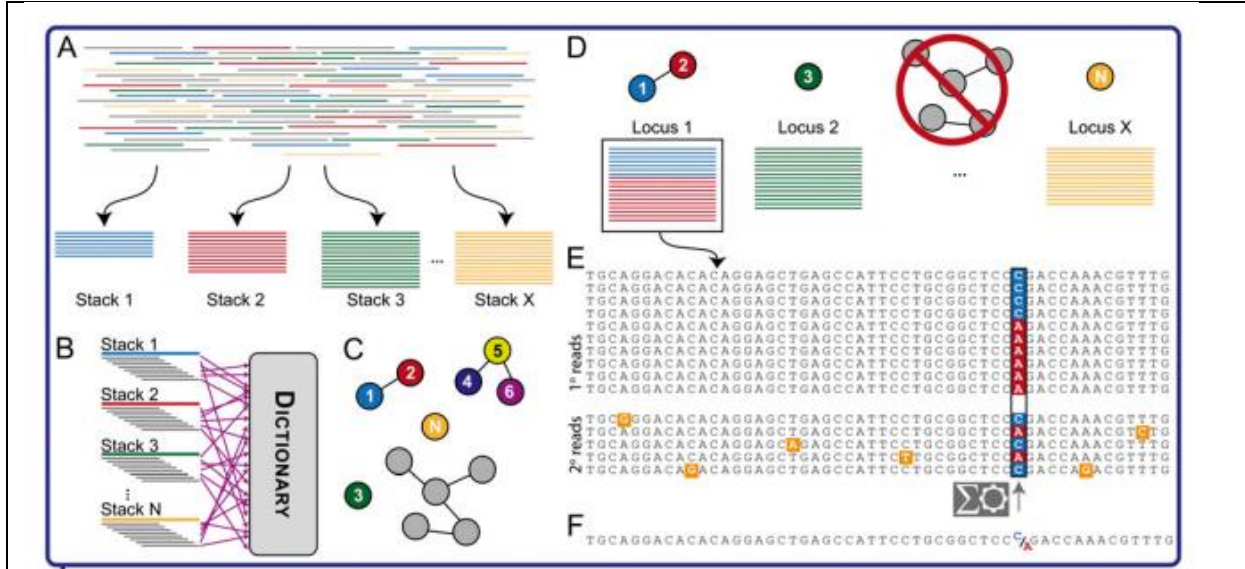
<http://catchenlab.life.illinois.edu/stacks/comp/ustacks.php>

http://catchenlab.life.illinois.edu/stacks/param_tut.php

Builds loci de novo and detects haplotypes in one individual, The Stacks core component program is ustacks, which identifies unique loci de novo.

The unique stacks program will take as input a set of short-read sequences and align them into exactly-matching stacks. Comparing the stacks it will form a set of loci and detect SNPs at each locus using a maximum likelihood framework.

The ustacks (unique stacks) program reads cleaned sequences and distills data into unique, exactly matching stacks by loading reads into a hash table (Figure 1A). Unique stacks that contain fewer reads than a configurable threshold (the stack-depth parameter) are disassembled, and the reads are set aside because these stacks are indistinguishable from stacks generated with sequencing error. Reads in a stack are primary reads, and reads that are set aside are secondary reads. The ustacks program calculates the average depth of coverage, then identifies stacks that are two standard deviations above the mean and excludes them, along with all stacks that are one nucleotide apart from these extremely deep (lumberjack) stacks, which usually represent repetitive elements.



Not: Bu aşamada barkodların yazılı olduğu dosyadan eğer sildiğin dosyalar varsa-düşük okumadan dolayı- adlarını çıkarmayı unutma.

Program Options

ustacks -t file_type -f file_path [-d] [-r] [-o path] [-i id] [-m min_cov] [-M max_dist] [-p num_threads] [-R] [-H] [-h]

Kullanılan komut bloğundan biri

ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/birbak-1.fq.gz -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/ -i 3 -m 3 -p 8 -M 2 -d -r --bound_high 0.1 --model_type bounded 2>&1

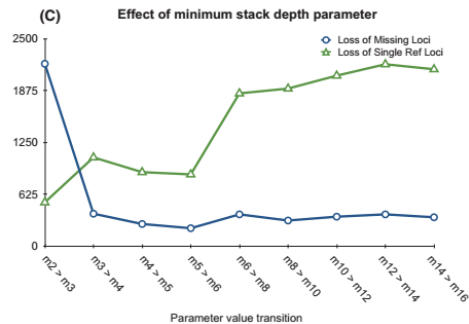
-t input file Type. Supported types: fasta, fastq, gzfasta, or gzfastq.

-f input file path

-o output path to write results.

-i MSQ ID to insert into the output to identify this sample
The -i flag specifies a sample ID for each individual and is passed through to the database. In later parts of the pipeline, such as with sstacks, samples are referred to by this sample ID. For each individual, three files will be produced:

-m Minimum depth of coverage required to create a stack (default 2).



c) As we increase the minimum number of raw reads required to form a stack, we see a trade-off between the number of false loci removed from our data set (blue line) vs. the number of true loci lost due to low coverage of the locus (green line) Catchen et al. 2013

Eğer bu parametreyi yükseltirsen düşük derinlikteki okumalarını kaybedersin ki RADseq de genelde okuma derinliği 3-6 aralığındadır. Eğer çok düşürürsen hatalı ve güvenilir olmayan okumaları yetersiz derinlikle birlikte hesaba katmış olursun. Eğer verideki okuma derinliği yüksekse bu değeri yüksek tutabilirsin. Okuma derinliğinin düşükse düşürebilirsin

The minimum stack depth parameter controls the number of raw reads required to form an initial stack. If the depth of coverage for a particular stack is below this value, then an allele will not be formed and those reads are temporarily set aside by the algorithm (they are used later in the algorithm, see below). Raw reads that are placed in a stack are referred to as primary reads, while those that are set aside are referred to

as secondary reads.

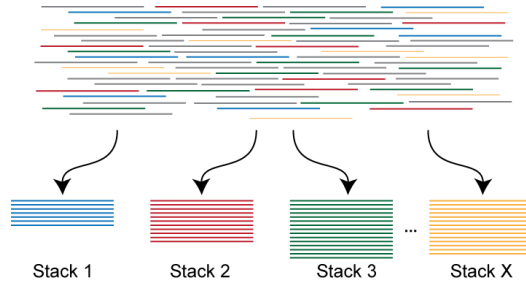


Figure 1. The initial stage of the ustacks de novo assembly algorithm forms exactly matching stacks from raw short-reads.

Some things to consider when setting this parameter value:

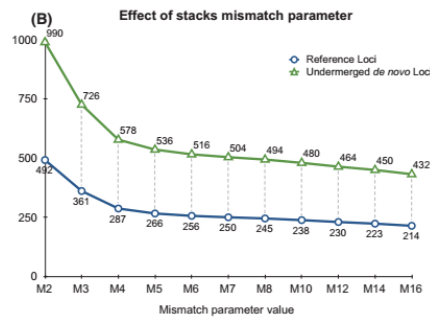
- If set to a value of 3 then three or more identical reads must be found to consider those reads a stack. If a stack is formed with only two reads, then those reads are set aside and a stack is not constructed.
- If this parameter is set too low, then reads with convergent sequencing errors are likely to be erroneously labeled as stacks.
- If this parameter too high, then true alleles will not be recorded and will drop out of the analysis.
- If you have low sequencing depth for your samples, you will have to set this parameter to a relatively low value. Conversely, if you have very high sequencing coverage, you will want to increase this parameter.

If you have a high error rate in your sequencing lane, then you are likely to see convergent sequencing or PCR errors (errors that occur independently at the same nucleotide position in the same read) and should increase the minimum stack depth.

-M

Maximum distance (in nucleotides) allowed between stacks (default 2).

Stacks will, through the program ustacks, use a k-mer search algorithm to merge alleles into loci. First, exactly matching reads are formed into stacks using a hashing algorithm. Stacks are subsequently decomposed into k-mers (subsequences of length k) that are compared among stacks to find matching alleles (see Catchen et al. 2011 for more detail). In the previous version of Stacks, this process was controlled by two parameters. The stack depth parameter (-m) controls the number of raw reads required to form a stack, and the mismatch parameter (-M) specifies the number of allowed nucleotide mismatches between two stacks to merge them into a locus.



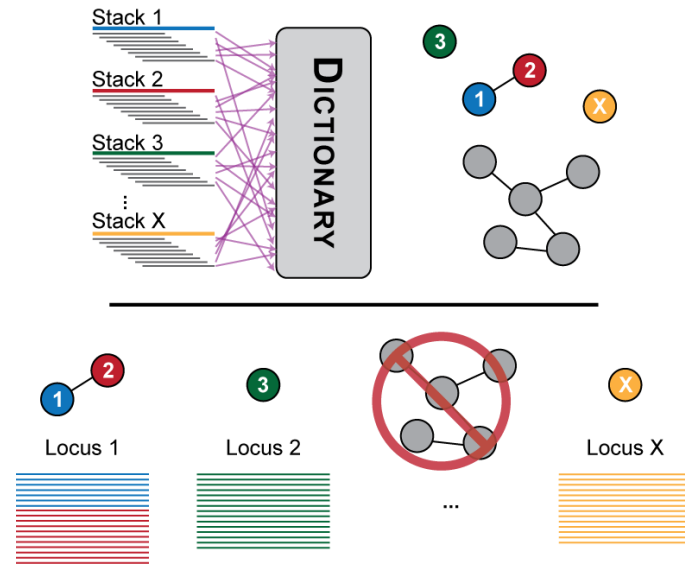
(B) Allowing two mismatches between stacks (equivalent to nucleotide distance) results in 990 de novo loci that should be merged into 492 loci according to the reference genome. Increasing -M reduces these undermerged loci, although the rate of reduction decreases after -M 4 Catchen et al. 2013

Enfazla lokusu M2 değeriinde elde etmiş çünkü değeri büyütürsen farklı lokuslar birleşiyor ve gerçekte olandan dahaz lokus elde ediyorsun

Eğer M değerini artırırsan okuma sayısını arttırabilirsin ancak bu kez birbirinden az farklılık taşıyan lokusların birleşmesine neden olarak az sayıda lokus elde etmene neden olur. Eğer M değerini çok düşük girensen bu kez de tek bir lokusu çok sayıda az okumalı lokusa bölersin buda diğer bireylerle eşleşmeyen çok sayıda yapay lokusa neden olur ve veri kaybına neden olur.

Once a set of exactly matching stacks has been generated, the second stage of the algorithm seeks to

match putative alleles together into a locus. The distance allowed between stacks parameter represents the number of nucleotides that may be different between two stacks in order to merge them. These nucleotide differences may be due to polymorphisms present between two alleles, or they may be due to sequencing error.



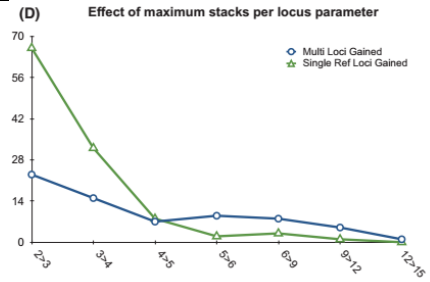
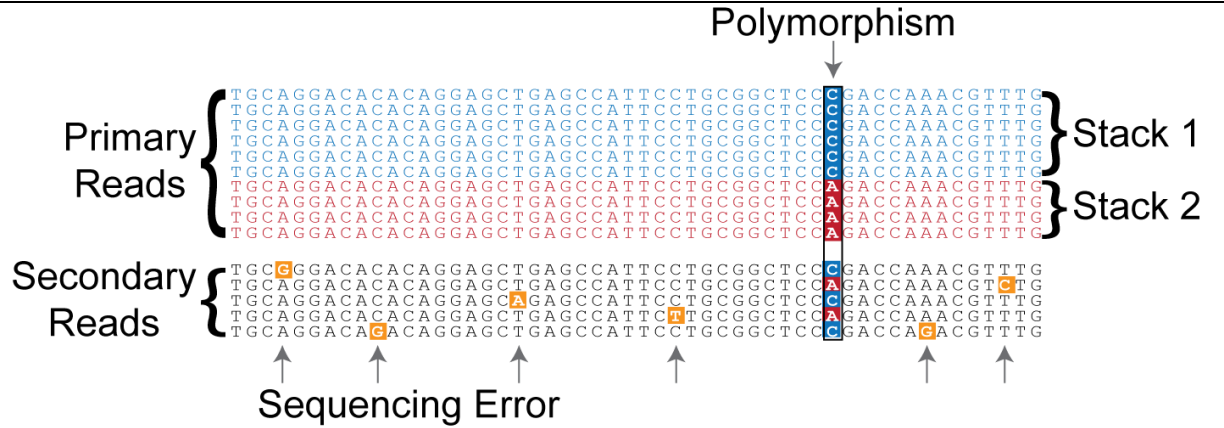
In Figure 2, Stack 1 and Stack 2 are found to have fewer nucleotide mismatches than allowed by the distance parameter and are merged into polymorphic Locus 1. Stack 3 and Stack X are found to be monomorphic and are converted to individual Locus 2 and Locus X. The large grey set of stacks represents a set of repetitive sequence that has too many alleles to be biologically correct. The pipeline detects these loci and blacklists them from the rest of the pipeline.

Some things to consider when setting this parameter value:

- If you set this parameter too low, then some loci will fail to be reconstructed. This means the SNPs contained in that locus will not be identified and this locus will appear as two loci to the remainder of the pipeline.
- Setting this parameter too high will allow repetitive sequence to chain together in to large, nonsensical loci. For example, if stack A is one nucleotide apart from stack B, which is one nucleotide apart from stack C, which is one nucleotide apart from stack D, then A, B, C, and D will be merged into a locus despite A and D being four nucleotides apart. These loci are not useful to the pipeline and at several points the pipeline will try to detect these and set them aside.

You will want to experiment with several different values of this parameter to see how many polymorphic loci you can construct.

-N	Maximum distance allowed to align secondary reads to primary stacks (default: M + 2).
-R	retain unused reads
-H	disable calling haplotypes from secondary reads
-p	enable parallel execution with num_threads threads.
-h	display this help message.
Stack assembly options:	
-r	enable the Removal algorithm, to drop highly-repetitive stacks (and nearby errors) from the algorithm.
-d	enable the Deleveraging algorithm, used for resolving over merged tags
d ve r parametreleri ile mitokondri ve its'ler gibi genomda çok bulunan ve fazla okunan diziler de uzaklaştırılmış olur	
--max_locus_stacks [num]	maximum number of stacks at a single <i>de novo</i> locus (default 3)

<p>(D) Effect of maximum stacks per locus parameter</p>  <p>D) As we increase the number of stacks allowed to exist at a single locus, we see a trade-off between the number of true loci added to the data set (green line) vs. the number of collapsed, false loci we add to the data set (blue line). Catchen et al. 2013</p>	
--k_len [len]	specify k-mer size for matching between alleles and loci (automatically calculated by default).
Gapped assembly options:	
--gapped	perform gapped alignments between stacks.
--max_gaps	number of gaps allowed between stacks before merging (default: 2).
--min_aln_len	minimum length of aligned sequence in a gapped alignment (default: 0.80).
Model options	
--model_type [type]	either 'snp' (default), 'bounded', or 'fixed'
	The bounded-error model can be selected in both ustacks and pstacks by specifying the --model_type bounded option. The bounds can be set by specifying the --bound_high and --bound_low options to ustacks and pstacks.
For the SNP or Bounded SNP model:	
--alpha [num]	chi square significance level required to call a heterozygote or homozygote, either 0.1, 0.05 (default), 0.01, or 0.001.
For the Bounded SNP model:	
--bound_low [num]	lower bound for epsilon, the error rate, between 0 and 1.0 (default 0).
--bound_high [num]	upper bound for epsilon, the error rate, between 0 and 1.0 (default 1).
For the Fixed model	
--bc_err_freq [num]	specify the barcode error frequency, between 0 and 1.0
<p>The specific values of the mismatch distance (-M), minimum stack depth (-m) and maximum stacks allowed per locus (--max_locus_stacks) chosen by the researcher represent a trade-off between leaving undermerged loci in the data set and confounding loci in the data by overmerging them. The optimal values for these parameters depend on the rate of polymorphism, the amount of sequencing error and the depth of sequencing performed. We therefore strongly encourage researchers to test a range of values for each parameter when approaching a data set for the first time</p> <p>Once the loci are formed, the secondary reads are brought back into the analysis and are aligned against the assembled loci using a more permissive nucleotide mismatch value (you can control this value with the -N parameter to ustacks or denovo_map.pl). This process provides more depth which aids the SNP calling model in detecting polymorphisms. A locus with a single polymorphism is outlined in Figure 3.</p>	
<p style="text-align: center;">Polymorphism</p>  <p>Figure 3. The major components in a Stacks Locus</p> <p>Once the loci are formed, the secondary reads are brought back into the analysis and are aligned against the assembled loci using a more permissive nucleotide mismatch value (you can control this value with the -N parameter to ustacks or</p>	

denovo_map.pl). This process provides more depth which aides the SNP calling model in detecting polymorphisms. A locus with a single polymorphism is outlined in Figure 3.

Bu komut bloğunu excelde hazırlamak istersen

Bunun için =BİRLEŞTİR (concatenated) formülünü kullan basamaklar aşağıdaki gibi

- 1- tüm komutlarda aynı olan path ve parametreleri aynı hücre içerisine yaz
- 2- örneklerin eğer 100 adetse process_radtag sonuç dosyasından örneklerini listesini elde edebilirsin eğer 100den çoksa örnek isimlerinden oluşan bir liste hazırla ve örnek adı girilecek hücrelere yapıştır
- 3-farklı girilecek parametreleri farklı hücrelere gir ve farklı hücreler arasında boşluk olmasını istiyorsan "" işaretini kullan
- 4- sonra kaç adet örneğin varsa her hücreyi o kadar kopyala aşağı çekerek hücre kenarını
- 5-sonra =BİRLEŞTİR formülü kullan ve tek tek hücreleri shift tuşuna basılı olarak seç enterla
- 6- sonra oluşan hücreyi aşağı doğru çekerek her bir örnek için çoğalt
- 7- oluşan komut bloğunu sonra kopyala ve slurm bloğuna yapıştır

Anaiz sonunda her bir birey için **.tags.tsv.gz**; **.alleles.tsv.gz**; **models.tsv.gz**; **.snps.tsv.gz** dosyaları oluşturulur; Bu dosyalarla cstacks aşamasından katalog oluşturulacak.

.tags.tsv	contains the stack that composes each loci, including a consensus sequence, and the output for each nucleotide from the SNP calling model.
.snps.tsv	contains SNPs that were identified for each locus
.alleles.tsv	contains alleles observed at each locus. If more than one SNP occurred at a locus, these alleles are recorded as haplotypes at the locus.

Sonrasında ustacks.slurm adlı slurm komut bloğunu aşağıdaki gibi ayarladım

```
#!/bin/bash
#SBATCH -p mercan
#SBATCH -A sarkaya
#SBATCH -J ustacks
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --time=05:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=kaya_sarp@hotmail.com

export OMP_NUM_THREADS=1

echo "SLURM_NODELIST $SLURM_NODELIST"
echo "NUMBER OF CORES $SLURM_NTASKS"

cd /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks

ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/birbak-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 1 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/birter-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 2 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/chobesn3-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 3 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/egr-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 4 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/led-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 5 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/luslagn-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 6 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
```

```

bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/orb-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 7 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/birkem-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 8 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/chobakd-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 9 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/chobtylos-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 10 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/heller-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 11 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/lusdavz-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 12 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/luspat-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 13 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1
ustacks -t gzfastq -f /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks/tun-1.fq.gz -o
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/outustacks -i 14 -m 3 -p 24 -M 2 -d -r --bound_high 0.1 --model_type
bounded 2>&1

```

exit

Not-1: çekirdek kullanımına üç yerde aynı rakam yazmalı eğer stacks komutu içerirse yukarda yazdığın çekirdek sayısını yazmazsan sen 10 seçsen bile komutta 8 yazılıysa 8 kullanır.

Not-2: .slurm iş betiğini P. lusch klaösür içerisinde aşağıdaki komutla oluştur ve yukardaki komut bloğunu kopyala ve yapıştır.

Cat > ustacks.slurm

Not: eğer internet bağlantın iyi değilse cat komutu ile doyaı oluşturup komutları yapıştırmada sorun çıkıyor ve hepsini yapıştırmıyor bu nedenle cat le doyaı oluşturtultan sonra nano ile açıp sadece komutu yapıştır.

Aşağıdaki komutla kuyrukların yoğunluğunu kontrol ettikten sonra

sinfo --states=IDLE

Aşağıdaki komut ile slurm dosyasını hazırladım ve sbatch komutu ile sardalyada analizi başlattım
cat > ustacks.slurm

```

-bash-4.2$
-bash-4.2$ squeue
          JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
          5626946     mercan  ustacks  sarkaya  R           0:01         1 mercan81
-bash-4.2$
-bash-4.2$
-bash-4.2$

```

Analiz sırasında ne kadar memory kullanıldığını görmek için

-bash-4.2\$ scontrol show jobid -dd 5626946

```

JobId=5626946 Name=ustacks
  UserId=sarkaya(3771) GroupId=trgridd(9004)
  Priority=20011 Nice=0 Account=sarkaya QOS=normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
  DerivedExitCode=0:0
  RunTime=00:22:39 TimeLimit=05:00:00 TimeMin=N/A
  SubmitTime=2018-01-28T14:52:48 EligibleTime=2018-01-28T14:52:48
  StartTime=2018-01-28T15:18:53 EndTime=2018-01-28T20:18:53
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
  Partition=mercan AllocNode:Sid=levrek1:101663

```

```
ReqNodeList=(null) ExcNodeList=(null)
NodeList=mercan81
BatchHost=mercan81
NumNodes=1 NumCPUs=24 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=0
Nodes=mercan81 CPU_IDs=0-23 Mem=120000
MinCPUsNode=1 MinMemoryCPU=5000M MinTmpDiskNode=0
Features=(null) Gres=(null) Reservation=(null)
Shared=OK Contiguous=0 Licenses=(null) Network=(null)
Command=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/ustacks.slurm
WorkDir=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks
StdErr=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/slurm-5626946.out
StdIn=/dev/null
StdOut=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/2-ustacks/slurm-5626946.out
BatchScript=
```

Burada her bir dosyadan 5 adet yeni dosyanın oluştuğu görülüyor. Bu dosyalar **.tags.tvsgz; .alleles.tvsgz; models.tvsgz; .snps.tsvgz**; Bu dosyalarla cstucks analizi gerçekleştirilecek

Analiz sonunda her bir birey için **.tags.tvsgz; .alleles.tvsgz; models.tvsgz; .snps.tsvgz** dosyaları oluşturulur; Bu dosyalarla cstucks aşamasından katalog oluşturulacak.

.tags.tsv	contains the stack that composes each loci, including a consensus sequence, and the output for each nucleotide from the SNP calling model.
.snps.tsv	contains SNPs that were identified for each locus
.alleles.tsv	contains alleles observed at each locus. If more than one SNP occurred at a locus, these alleles are recorded as haplotypes at the locus.

Analizler süresince oluşan sonuç istatistikleri önemli olan process_radtag ve ustacks basamakları bu nedenle bu iki basamak sonucu elde edilen sonuçları Excel sayfasında kaydet. Böylece analizler süresince değiştirdiğin -m ya da -M gibi parametrelerin veri setine etkisi gözleyebileşin. Ancak tüm bu parametrelerin lokus sayısına ve SNPs performansına etkisin gözlemek için en son veri dosyalarını görmen gerekiyor. Bu nedenle önce standart parametrelerle başla bunların sonucu gör sonrasında farklı parametreler dene.

Ustacks analiz sonucu oluşan ve değerlerin yazılı olduğu dosyadaki bazı değeri Excel sayfasına yaz ve parametreleri değiştirdiğinde bunlardaki değişimi gözleyerek en olası veri seti parametre uyumunu yakala. Bunun için aşağıdaki komutu kullan. Bu komutla masa üstündeki slurm-5310909-M2.out dosyası içerisindeki After remainders merged, coverage depth Mean değerlerini yeni oluşturulacak meancoverage.txt dosyasına yaz diyosun. Böylece istediğin verileri hatasız seçip excele ekleyebilirsin aynı sıradaki.

grep "After remainders merged, coverage depth Mean" slurm-5310909-M2.out > meancoverage.txt

eğer örnek isimlerinide istiyorsan aşağıdaki komutu kullan
grep "Parsing" slurm-5310909-M2.out > ind-coverage.txt

Oluşturduğun bu Excel dosyası önemli çünkü buradaki istatistikleri inceleyerek
 1- işlerin yolunda olup olmadığını yani process-radtag sonucu elde ettiğin sonuçlarla ustacks sonucu elde etiklerini karşılaştıır eğer uyumsuzluk varsa biryerlerde hata var.
 2- parametre değişimlerini yarattığı lokus artış azalışlarının farklılıkları
 3- kütüphane kalitesi vb.

cstacks: Merges loci from multiple individuals to form a catalog.

Not1: bu analiz için veri büyüklüğüne bağlı olarak değişmekle birlikte 100gb altında ram kullanma. Eğer düşük memory girersen analiz düşük memory uyarısı verir ve öldürölür.

```
Slurmd[levrek5]: Job 5308191 exceeded memory limit (34601880 > 33554432), being killed
slurmd[levrek5]: Exceeded job memory limit
slurmd[levrek5]: *** JOB 5308191 CANCELLED AT 2017-05-03T20:22:48 ***
```

Not2: Bu analize başlamadan önce ustacks ile oluşturduğun klasörü yedekle çünkü ilk denemde örnek sayına ve veri büyüklüğüne bağlı olarak memory yetmeyebilir oluşan hata mevcut klasörü etkiliyor ve analiz yeterli

memory versen bile ilerlemeye biliyor. Böylesi durumlarda eğer klasörleri yedekleme yapmadıysan dönüp `process_radtag`'dan analizlere başlaman gerek.

job submit ederken dikkat edilecek bir diğer hususta partititons (kuyruklar, hesaplama noktaları) dır. Partititon, yapacağın analize göre değişir ve partititonları etkin bir şekilde kullanman analizini kuyrukta fazla beklemeden kısa sürede bitirmene, zaman ve memory kısıtlamasından kaynaklı sıkıntılara engel olur.

İş göndereceğinde Güncel duruma ve her bir kümedeki kuyruğun node, time, processor, memory ve kullanılabilirlik durumlarını görmek için `sinof -l` veya `sinfo -Nel` komutlarını kullanabilirsin *idle* yazılı olanlar kullanıla bilir boşta olan kuyruklar. Boşta olan sunucuları derli toplu görmek açısından `sinfo --states=IDLE` komutu oldukça kullanışlı aşağıda kuyrukların kullanılabilirlik durumları zaman limitleri, nod sayıları listesi görünüyor.

```
-bash-4.1$ sinfo --states=IDLE
PARTITION AVAIL TIMELIMIT NODES STATE MODELIST
cuda up 15-00:00:0 0 n/a
single up 15-00:00:0 2 idle levrek[5-6]
short up 4-00:00:0 0 n/a
mid1 up 4-00:00:0 0 n/a
mid2 up 8-00:00:0 0 n/a
long up 15-00:00:0 0 n/a
levrekv2 up 15-00:00:0 0 n/a
mercan* up 15-00:00:0 20 idle mercan[19,21-22,42-44,92,105-106,109-111,113-114,118,122,125,157,189,191]
smp up 8-00:00:0 1 idle orkinos1
sardalya up 15-00:00:0 62 idle sardalya[19,30-31,59,61,79-124,127,132-134,137-139,146-149]
```

Daha detaylı bilgi için `sinfo -Nel` komutu kullanır. Bu komutla ise aşağıdaki gibi daha detaylı olarak kullanılacak partititonların kapasiteleri, kullanılabilirlik durumları görünüyor. Trubada da yüksek performans gerektiren işlerde farklı partititonları kullanman gerek.

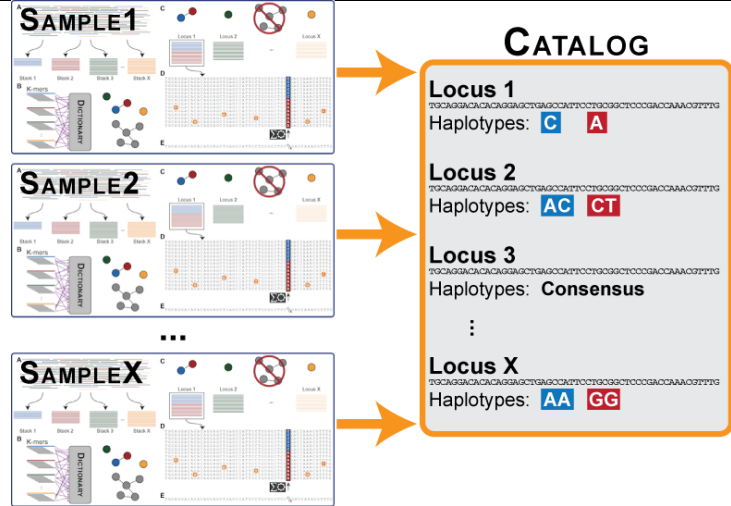
```
-bash-4.1$ sinfo -Nel
Wed May 3 22:59:03 2017
MODELIST
levrek2 1 cuda drained 16 2:8:2 256000 0 1 (null) test
levrek[3-4] 2 cuda allocated 16 2:8:2 256000 0 1 (null) none
levrek[5-6] 5 single mixed 16 2:8:2 256000 0 1 (null) none
levrek[10-14,65-128] 69 long mixed 16 2:8:2 256000 0 1 (null) none
levrek[10-14,65-128] 69 mid2 mixed 16 2:8:2 256000 0 1 (null) none
levrek[10-14,65-128] 69 mid1 mixed 16 2:8:2 256000 0 1 (null) none
levrek[10-14,65-128] 69 short mixed 16 2:8:2 256000 0 1 (null) none
levrek[129-132,136-144] 14 cuda mixed 24 2:12:2 256000 0 1 (null) none
levrek[134-135] 2 cuda down* 24 2:12:2 256000 0 1 (null) Not responding
levrek[145-175,177-178,180-188,190-192] 45 levrekv2 mixed 24 2:12:2 256000 0 1 (null) none
levrek[176,179,189] 3 levrekv2 completing 24 2:12:2 256000 0 1 (null) none
mercan[9-39,41,43,45-103,121-123,125-127,130-146,148-181,183-192] 159 mercan* mixed 24 4:6:1 128000 0 1 (null) none
mercan[40,42,44,104-120,124,128-129] 23 mercan* idle 24 4:6:1 128000 0 1 (null) none
mercan147 1 mercan* down 24 4:6:1 128000 0 1 (null) Node unexpectedly re
mercan182 1 mercan* drained 24 4:6:1 128000 0 1 (null) batch job complete f
orkinos1 1 smp idle 224 16:14:1 412810 0 1 (null) none
sardalya[2-3,8-20,27,29-31,49-57,62-65,73,125-128,150-151,153] 40 sardalya mixed 28 2:14:2 256000 0 1 (null) none
sardalya[4-7,22-26,28,32,58-61,67-72,74-124,129-149,152] 94 sardalya idle 28 2:14:2 256000 0 1 (null) none
sardalya[21,33-48,66] 18 sardalya down* 28 2:14:2 256000 0 1 (null) Not responding
```

Not: kuyruk (partititons) tayin ederken kuyruğun özelliklerini (node, processor, memeory) doğru gir yoksa hata verir. Eğer analiz çok ram istiyorsa ve MPI destekliyse birden fazla node kullanabilirsin

After the ustacks now, we want to create a catalog. We know that every allele we observe in the progeny must be present in the parents, so we will create the catalog from the two parents with cstacks:

A catalog can be built from any set of samples processed by the ustacks or pstacks programs. It will create a set of consensus loci, merging alleles together. In the case of a genetic cross, a catalog would be constructed from the parents of the cross to create a set of all possible alleles expected in the progeny of the cross.

Stacks can be used on any set of samples, including parents/progeny from a mapping cross or samples from a set of populations. The Stacks catalog can be loaded with any number of individuals. For any particular set of samples, the catalog is meant to hold all the alleles segregating in the population. In a mapping cross we can be sure that the parents of the cross will contain all the alleles and so those are the only samples that need to be loaded into the catalog. When analyzing a population we need to load all individuals into the catalog, or in a very large analysis, at least enough individuals to capture all the major alleles segregating



CATALOG

Locus 1
Haplotypes: C A

Locus 2
Haplotypes: AC CT

Locus 3
Haplotypes: Consensus

⋮

Locus X
Haplotypes: AA GG

The stacks program will be executed on each individual sample in the data set to build loci. Once this is complete, the data from each individual will be merged into a catalog (by the cstacks program), which is meant to contain all the loci and alleles in the population. In the case of a mapping cross, then the catalog can be built solely from the parental loci. In the case of a population, the catalog will be constructed from the loci in each individual in the population.

Rarely in the case of a mapping cross, but frequently in the case of a population, there will be monomorphic, or fixed loci in two or more individuals in the population. However, if you compare these loci to one another, you will find that they are differentially fixed versions of the same locus and should be merged into a single locus in the catalog. If the distance between catalog loci parameter is greater than 0, then cstacks will use the consensus sequence from each locus to attempt to merge loci together across samples.

cstacks -b 1 -o /nfs/turbo/lsa-knowles/temp_nfs/kayas/lemniscatus3/ex1 -s /nfs/turbo/lsa-knowles/temp_nfs/kayas/lemniscatus3/ex1/1A_lemniscatus3_STLL10.... -p 8 -n 2 2>&1

-p	enable parallel execution with num_threads threads.
-b	MySQL ID of this batch -b 1 bu komuttaki tüm örnekler tek bir kataloğa ait demek
-s	TSV file from which to load radtags.
-o	output path to write results.
-m	include tags in the catalog that match to more than one entry.
-n	number of mismatches allowed between sample tags when generating the catalog.
-g	base catalog matching on genomic location, not sequence identity.
-h	display this help message

Some things to consider when setting this parameter value: -n

- As with nucleotide mismatch parameter, if you set this parameter too low (or leave it at the default value of 0) there will be loci across individuals that are represented independently in the catalog that are truly the same locus. This will cause you to miss fixed differences in a population analysis, for example. If you plan to build a phylogenetic tree from these data, loci with fixed differences are the most important ones.
- If you set this parameter too high, you will again allow loci close together in sequence space to chain together and create big, erroneous loci in the catalog.

Not1: Bu analiz yüksek memory ister. Eğer triba da iş yükleme betiğine eğer istediğin memory miktarını girmezsen Levrek kuyruklarında sana çekirdek başına maksimum 16Gb kullanım veriyor. Ancak bu analiz için veri büyüklüğüne bağlı olarak değişmekle birlikte 100gb altında ram kullanma.

Not2: ayrıca ustacks sonucu elde ettiğin sonuç klasörünü kopyala yedekle ve cstacks analizine öyle başla. Çünkü cstacks analizi sırasında bir sorun olursa özellikle ram yetmezliğinden kaynaklı aynı klasör ile cstacks analizini ram artırarak tekrar denersen de problem veriyor. Bu durum muhtemelen analiz sırasında klasörün içerisine atılan dosyalardan kaynaklanıyor. Bu nedenle ustacks sonuç klasörünü başka bir isimle yedekle çünkü gerekli memory durumunu başta kestiremeyebilirsin aksi taktirde yeniden process_radtag basamağından başlaman gerek.

Not3: ustacks ve cstacks aşamasında iş için ne kadar süre gerektiğini belirlemek için önce interaktif olarak komutu çalıştır bir süre takip et bir bireyin analizi ne kadar sürede bitiyor onu belirle ona göre iş betiğine süreyi hesapla. Eğer işe yeterli süreyi vermezsen analiz bitmeden iş sistem tarafından öldürülür.

```
#!/bin/bash
#SBATCH -p sardalya
#SBATCH -A sarkaya
#SBATCH -J ustacks
#SBATCH -N 1
#SBATCH -n 28
#SBATCH --time=05:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=kaya_sarp@hotmail.com

export OMP_NUM_THREADS=28

echo "SLURM_NODELIST $SLURM_NODELIST"
echo "NUMBER OF CORES $SLURM_NTASKS"

cd /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/

cstacks -b 1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/birbak-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/birter-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/chobesn3-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/egr-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/led-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/luslagn-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/orb-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/birkem-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/chobakd-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/chobtylos-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/heller-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/lusdavz-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/luspat-1 -
s/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/outcstacks/tun-1 -p 28 -n 4 --gapped --
max_gaps 2 2>&1

exit
```

```
-bash-4.2$ squeue
          JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
          5628278  sardalya    ustacks  sarkaya  R        1:13      1 sardalya103
-bash-4.2$
```

Analiz için kullanılan memoriye bakamak istiyorsak

```
-bash-4.2$ scontrol show jobid -dd 5628278
```

```
JobId=5628278 Name=ustacks
  UserId=sarkaya(3771) GroupId=trgridd(9004)
  Priority=20010 Nice=0 Account=sarkaya QOS=normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
  DerivedExitCode=0:0
  RunTime=00:03:00 TimeLimit=1-05:00:00 TimeMin=N/A
  SubmitTime=2018-01-29T00:59:33 EligibleTime=2018-01-29T00:59:33
  StartTime=2018-01-29T01:00:03 EndTime=2018-01-30T06:00:03
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
```



```

Partition=sardalya AllocNode:Sid=levrek1:123248
ReqNodeList=(null) ExcNodeList=(null)
NodeList=sardalya103
BatchHost=sardalya103
NumNodes=1 NumCPUs=28 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=0
  Nodes=sardalya103 CPU_IDs=0-27 Mem=252000
MinCPUsNode=1 MinMemoryCPU=9000M MinTmpDiskNode=0
Features=(null) Gres=(null) Reservation=(null)
Shared=OK Contiguous=0 Licenses=(null) Network=(null)
Command=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/cstacks.slurm
WorkDir=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks
StdErr=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/slurm-5628278.out
StdIn=/dev/null
StdOut=/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/3-cstacks/slurm-5628278.out
BatchScript=

```

Not: -p parametresi de çekirdek sayısı ile aynı olmalı yoksa sen 14 versen bile stacs 8 kullanır
Cstack analizi sonrası oluturlan kataloglar outcstacks içerisinde görülmekte

```

-bash-4.2$ ls outcstacks
batch_1.catalog.alleles.tsv.gz  birter-1.alleles.tsv.gz  chobesn3-1.snps.tsv.gz  heller-1.fq.gz  lUSDavz-1.tags.tsv.gz  orb-1.models.tsv.gz
batch_1.catalog.snps.tsv.gz    birter-1.fq.gz          chobesn3-1.tags.tsv.gz  heller-1.models.tsv.gz  lUSDagn-1.alleles.tsv.gz  orb-1.snps.tsv.gz
batch_1.catalog.tags.tsv.gz    birter-1.models.tsv.gz  chobtylos-1.alleles.tsv.gz  heller-1.snps.tsv.gz  lUSDagn-1.fq.gz          orb-1.tags.tsv.gz
birbak-1.alleles.tsv.gz       birter-1.snps.tsv.gz    chobtylos-1.fq.gz       heller-1.tags.tsv.gz  lUSDagn-1.models.tsv.gz  tun-1.alleles.tsv.gz
birbak-1.fq.gz                birter-1.tags.tsv.gz    chobtylos-1.models.tsv.gz  heller-1.tags.tsv.gz  lUSDagn-1.snps.tsv.gz    tun-1.fq.gz
birbak-1.models.tsv.gz        chobakd-1.alleles.tsv.gz  chobtylos-1.snps.tsv.gz  led-1.alleles.tsv.gz  lUSDagn-1.tags.tsv.gz    tun-1.models.tsv.gz
birbak-1.snps.tsv.gz          chobakd-1.fq.gz         chobtylos-1.tags.tsv.gz  led-1.models.tsv.gz  lUSDagn-1.tags.tsv.gz    tun-1.snps.tsv.gz
birbak-1.tags.tsv.gz          chobakd-1.models.tsv.gz  egr-1.alleles.tsv.gz    led-1.snps.tsv.gz    lUSDagn-1.tags.tsv.gz    tun-1.tags.tsv.gz
birkem-1.alleles.tsv.gz       chobakd-1.snps.tsv.gz    egr-1.fq.gz             led-1.tags.tsv.gz    lUSDpat-1.fq.gz          tun-1.tags.tsv.gz
birkem-1.fq.gz                chobakd-1.tags.tsv.gz    egr-1.models.tsv.gz     lUSDavz-1.alleles.tsv.gz  lUSDpat-1.models.tsv.gz
birkem-1.models.tsv.gz        chobesn3-1.alleles.tsv.gz  egr-1.snps.tsv.gz       lUSDavz-1.fq.gz      lUSDpat-1.tags.tsv.gz
birkem-1.snps.tsv.gz          chobesn3-1.fq.gz         egr-1.tags.tsv.gz       lUSDavz-1.models.tsv.gz  orb-1.alleles.tsv.gz
birkem-1.tags.tsv.gz          chobesn3-1.models.tsv.gz  heller-1.alleles.tsv.gz  lUSDavz-1.snps.tsv.gz  orb-1.fq.gz
-bash-4.2$
-bash-4.2$
-bash-4.2$

```

Analiz sonunda üç tane catalog dosyası oluşturur bunlar

```

batch_1.catalog.alleles.tsv.gz
batch_1.catalog.snps.tsv.gz
batch_1.catalog.tags.tsv.gz

```

batch_1.catalog.tags.tsv	contains the stack that composes each loci, including a consensus sequence
batch_1.catalog.snps.tsv	contains SNPs that were identified for each locus
batch_1.catalog.alleles.tsv	contains alleles observed at each locus

sstacks: Matches loci from an individual against a catalog

Sets of stacks constructed by the [ustacks](#) or [pstacks](#) programs can be searched against a catalog produced by [cstacks](#). In the case of a genetic map, stacks from the progeny would be matched against the catalog to determine which progeny contain which parental alleles.

-p	enable parallel execution with num_threads threads
-b	MySQL ID of this batch.
-c	TSV file from which to load the catalog loci
-s	TSV file from which to load sample loci
-o	output path to write results
-g	base catalog matching on genomic location, not sequence identity
-x	don't verify haplotype of matching locus.
-v	print program version.
-h	display this help message

```

#!/bin/bash
#SBATCH -p sardalya
#SBATCH -A sarkaya
#SBATCH -J sstacks
#SBATCH -N 1
#SBATCH -n 28
#SBATCH --time=1-05:00:00

```

```
#SBATCH --mail-type=ALL
#SBATCH --mail-user=kaya_sarp@hotmail.com

export OMP_NUM_THREADS=28

echo "SLURM_NODLIST $SLURM_NODLIST"
echo "NUMBER OF CORES $SLURM_NTASKS"

cd /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/

sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/birbak-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/birter-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/chobesn3-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/egr-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/led-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/luslagn-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/orb-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/birkem-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/chobakd-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/chobtylos-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/heller-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/lusdazv-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/luspat-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
sstacks -b 1 -c /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/batch_1 -s
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-sstacks/outsstacks/tun-1 -o /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/4-
sstacks/outsstacks/ -p 28 --gapped 2>&1
```

```
exit
```

```
-bash-4.2$
-bash-4.2$ squeue
              JOBID PARTITION   NAME   USER  ST       TIME  NODES NODELIST(REASON)
              5630327  sardalya  sstacks sarkaya  R       13:30      1 sardalya136
-bash-4.2$
-bash-4.2$
```

Analiz sardalya sunucusunda gerçekleştirildi bu sunucu ailesinde her noda 9Gb'lık 28ram bulunmakta toplam 252Gb ram bulunmakta.

Analiz 72Gb memory ile yaklaşık 2 saat sürdü.

Bu analizle klasör içerisine kataloglarla eşleştirme sonucu oluşturulan **matches.tvsgz** dosyaları oluşturulur. Bu catalogue-matched data populasyon analizinde veri olarak kullanılacak.

```

-bash-4.2$
-bash-4.2$ ls outstacks/
batch_1.catalog.alleles.tsv.gz  birter-1.alleles.tsv.gz  chobesn3-1.models.tsv.gz  heller-1.alleles.tsv.gz  lusdavz-1.models.tsv.gz  orb-1.alleles.tsv.gz
batch_1.catalog.snps.tsv.gz    birter-1.fq.gz          chobesn3-1.snps.tsv.gz   heller-1.fq.gz          lusdavz-1.snps.tsv.gz    orb-1.fq.gz
batch_1.catalog.tags.tsv.gz    birter-1.matches.tsv.gz chobesn3-1.tags.tsv.gz   heller-1.matches.tsv.gz lusdavz-1.tags.tsv.gz    orb-1.matches.tsv.gz
birbak-1.alleles.tsv.gz       birter-1.models.tsv.gz  chobtylos-1.alleles.tsv.gz heller-1.models.tsv.gz  luslagn-1.alleles.tsv.gz orb-1.models.tsv.gz
birbak-1.fq.gz                birter-1.snps.tsv.gz    chobtylos-1.fq.gz       heller-1.snps.tsv.gz   luslagn-1.fq.gz          orb-1.snps.tsv.gz
birbak-1.matches.tsv.gz       birter-1.tags.tsv.gz    chobtylos-1.matches.tsv.gz heller-1.tags.tsv.gz   luslagn-1.matches.tsv.gz orb-1.tags.tsv.gz
birbak-1.models.tsv.gz        birter-1.tags.tsv.gz    chobtylos-1.models.tsv.gz heller-1.tags.tsv.gz   luslagn-1.models.tsv.gz tun-1.alleles.tsv.gz
birbak-1.snps.tsv.gz          birter-1.tags.tsv.gz    chobtylos-1.snps.tsv.gz  heller-1.tags.tsv.gz   luslagn-1.snps.tsv.gz   tun-1.fq.gz
birbak-1.tags.tsv.gz          birter-1.tags.tsv.gz    chobtylos-1.tags.tsv.gz  heller-1.tags.tsv.gz   luslagn-1.tags.tsv.gz   tun-1.matches.tsv.gz
birkem-1.alleles.tsv.gz       chobakd-1.alleles.tsv.gz chobtylos-1.tags.tsv.gz  led-1.fq.gz            luspat-1.alleles.tsv.gz  tun-1.models.tsv.gz
birkem-1.fq.gz                chobakd-1.snps.tsv.gz   egr-1.alleles.tsv.gz     led-1.matches.tsv.gz   luspat-1.fq.gz          tun-1.snps.tsv.gz
birkem-1.matches.tsv.gz       chobakd-1.snps.tsv.gz   egr-1.fq.gz              led-1.models.tsv.gz    luspat-1.matches.tsv.gz tun-1.tags.tsv.gz
birkem-1.models.tsv.gz        chobakd-1.tags.tsv.gz   egr-1.matches.tsv.gz     led-1.snps.tsv.gz      luspat-1.models.tsv.gz
birkem-1.snps.tsv.gz          chobesn3-1.alleles.tsv.gz egr-1.models.tsv.gz     led-1.tags.tsv.gz      luspat-1.snps.tsv.gz
birkem-1.tags.tsv.gz          chobesn3-1.fq.gz        egr-1.snps.tsv.gz        lusdavz-1.alleles.tsv.gz luspat-1.tags.tsv.gz
-bash-4.2$

```

Buraya kadar yapılanlar	
Lib_1_P_lusch_srp_R1.fastq.gz	Çalıştığımız örnek. Sing end ddRadSeq sekansları kütüphane veri seti. Çalışmada 14 örnek kullanıldı ve bu örneklerle ait tüm okumalar tek yönlü barkodlanmış olarak bu veri dosyası içerisinde.
Process_radtags	Lib_1_P_lusch_srp_R1.fastq.gz genomik kütüphane içerisindeki ve lusch_barcode.txt dosyası içerisinde barkodları ve bireyleri içeren dosyaya göre bireyler demultiplex edildi.
ustacks	ustacks stacks pipeline'nin core programıdır. Bu aşamada (uniques stacks) her bir bireye ait olan diziler kendi içerisinde hizalanır. Olası lokusları ve haplotipleri saptar. Bu işlemin sonunda 4 yeni dosya oluşur Data from each individual are grouped into loci, and polymorphic nucleotide sites are identified (ustacks or pstacks for unaligned or aligned data, respectively). Data from each individual are grouped into loci, and polymorphic nucleotide sites are identified (ustacks or pstacks for unaligned or aligned data, respectively).
cstacks	Bu aşamada her bir bireyin lokusları ele alınır ve birleştirilerek kaç parental lokuslar oluşturularak katalog oluşturulur. Bu işlemin sonunda 3 yeni dosya oluşur. Stacks employs a Catalog to record all loci identified in a population and matches individuals to that Catalog to determine which haplotype alleles are present at every locus in each individual. Loci are grouped together across individuals and a catalogue is written (cstacks)
sstacks	cstacks ile lokusların kataloglarının oluşturulmasından sonra sstacks (set of catalogs) ile bu lokusların bireylerdeki dağılımı gerçekleştirilir. Loci from each individual are matched against the catalogue to determine the allelic state at each locus in each individual (sstacks)

Veri filtrelemesi:	
Populations http://catchenlab.life.illinois.edu/stacks/comp/populations.php https://github.com/enormandeau/stacks_workflow/blob/master/00-scripts/stacks_4_populations.sh	
<p>The list of sampling populations are supplied to the populations program in a population map file, which contains the individual sample in one column and an integer representing the population in another column. Once the first four stages of Stacks have completed, populations can be run on these processed reads repeatedly using the same catalogue-matched data, but using different parameters or population maps. Researchers can thus evaluate the sensitivity of results on different parameters and divide samples in various ways geographically or by phenotype).</p>	
<p>Populasyon aşaması çok opsiyonlu ve farklı amaçlar için defalarca tekrarlanacak bir basamak. Bunlar</p> <p>i) filtrasyon</p> <p>ii) farklı analizler için veri dosyalarının oluşturulması</p> <p>ii) populasyon genetiği analizlerinin yapılması</p>	
<p>Bu aşamada ikinci dosyaya ihtiyacın var bu da popmap. Eğer birey ya da populasyon çıkarmak istiyorsan popmap dosyasından çıkaracaksın</p>	
<p>populations -b 1 -P /nfs/turbo/lsa-knowles/temp_nfs/kayas/lemniscatus/stacksout -M /nfs/turbo/lsa-knowles/temp_nfs/kayas/lemniscatus/stacksout/lemnisc_popMap1.txt -p 3 -m 3 -r 0.5 -t 8 --vcf 2>&1</p>	
<p>The populations program has a number of filtering parameters that allow one to control execution. For example, for each locus, a researcher can set</p> <p>-r: a minimum percentage of individuals within a population,</p> <p>-p: a minimum number of populations,</p> <p>-m: a minimum depth of coverage for each individual and</p> <p>-a: a minimum allele frequency</p>	
-p [int]	minimum number of populations a locus must be present in to process a locus
-r [float]	minimum percentage of individuals in a population required to process a locus for that population.
--min_maf [float]	specify a minimum minor allele frequency required to process a nucleotide site at a locus (0 < min_maf < 0.5)
--max_obs_het [float]	specify a maximum observed heterozygosity required to process a nucleotide site at a locus.
-m [int]	specify a minimum stack depth required for individuals at a locus.
--lnl_lim [float]	filter loci with log likelihood values below this threshold.
--write_single_snp	restrict data analysis to only the first SNP per locus
--write_random_snp	restrict data analysis to one random SNP per locus.
-B	path to a file containing Blacklisted markers to be excluded from the export.
-W	path to a file containing Whitelisted markers to include in the export
File output options:	
--ordered_export	if data is reference aligned, exports will be ordered; only a single representative of each overlapping site
--genomic	output each nucleotide position (fixed or polymorphic) in all population members to a file (requires --renz).
--fasta	output full sequence for each unique haplotype, from each sample locus in FASTA format, regardless of plausibility.
--fasta_strict	output full sequence for each haplotype, from each sample locus in FASTA format, only for biologically plausible loci
--vcf	output SNPs in Variant Call Format (VCF).
--vcf_haplotypes	output haplotypes in Variant Call Format (VCF).
--genepop	output results in GenePop format
--structure	output results in Structure format.
--phase	output genotypes in PHASE format.
--fastphase	output genotypes in fastPHASE format.
--beagle	output genotypes in Beagle format
--beagle_phased	output haplotypes in Beagle format
--plink	output genotypes in PLINK format
--hzar	output genotypes in Hybrid Zone Analysis using R (HZAR) format.
--phylip	output nucleotides that are fixed-within, and variant among populations in Phylip

	format for phylogenetic tree construction.
--phylip_var	include variable sites in the phylip output encoded using IUPAC notation.
--phylip_var_all	include all sequence as well as variable sites in the phylip output encoded using IUPAC notation.
--treemix	output SNPs in a format useable for the TreeMix program (Pickrell and Pritchard).
Population Map	
The Population Map	
A population map is a text file containing two columns: the prefix of each sample in the analysis in the first column, followed by an integer in the second column indicating the population. This simple example shows six individuals in two populations (the integer indicating the population can be any, unique number, it doesn't have to be sequential):	
<pre>% more popmap indv_01 6 indv_02 6 indv_03 6 indv_04 2 indv_05 2 indv_06 2</pre>	
Alternatively, we can use strings instead of integers to enumerate our populations	
<pre>% more popmap indv_01 fw indv_02 fw indv_03 fw indv_04 oc indv_05 oc indv_06 oc</pre>	
Whitelists and Blacklists	
The populations program allows the user to specify a list of catalog locus IDs (also referred to as markers) to the program. In the case of a whitelist, the program will only process the loci provided in the list, ignoring all other loci. In the case of a blacklist, the listed loci will be excluded, while all other loci will be processed. These lists apply to the entire locus, including all SNPs within a locus if they exist.	
The populations and genotypes programs allow the user to specify a list of catalog locus IDs (also referred to as <i>markers</i>) to the two programs. In the case of a whitelist, the program will only process the loci provided in the list, ignoring all other loci. In the case of a blacklist, the listed loci will be excluded, while all other loci will be processed. These lists apply to the entire locus, including all SNPs within a locus if they exist.	
A whitelist or blacklist are simple files containing one catalog locus per line, like this:	
<pre>% more whitelist 3 7 521 11 46 103 972 2653 22</pre>	
SNP-specific Whitelists	
In the populations program it is possible to specify a whitelist that contains catalog loci and specific SNPs within those loci. This is useful if you have a specific set of SNPs for a particular dataset that are known to be informative; perhaps you want to see how these SNPs behave in different population subsets, or perhaps you are developing a SNP array that will contain a specific set of data.	
To create a SNP-specific whitelist, simply add a second column (separated by tabs) to the standard whitelist where the second column represents the column within the locus where the SNP can be found. Here is an example:	

```
% more whitelist
1916<tab>12
517      14
517      76
1318
1921      13
195       28
260       5
28        44
28        90
5933
19369     18
```

You can include all the SNPs at a locus by omitting the extra column, and you can include more than one SNP per locus by listing a locus more than once in the list (with a different column). The column is a zero-based coordinate of the SNP location, so the first nucleotide at a locus is labeled as column zero, the second position as column one. These coordinates correspond with the column reported in the batch_X.sumstats.tsv file as well as in several other output files from populations.

Whitelists and blacklists are processed by the populations and genotypes programs directly after the catalog and matches to the catalog are loaded into memory. All loci destined not to be processed are pruned from memory at this point, before any calculations are made on the remaining data. So, once the whitelist or blacklist is implemented, the other data is no longer present and will not be seen or interfere with any downstream calculations, nor will they appear in any output files.

Stacks populations filtrasyon

Bu filtrasyonları yaparken amaç en az missing data ve en fazla birey ve lokus dengesini kurmak. Bu yönde filtrasyonu yapmak.

I. filtrasyon: lokusların eldesi ve vcf file a yazdırılması

Amaç elimizdeki dizilerde aşırı varyasyon gösteren bölgelerin ilk eliminasyonunu yapmak. Stacks da populations ile filtrasyonu yaparken major üç parametre var bunlar -p 3 (en az 3 popülasyonda ortak olan lokusları koru), -m 3 (coverage miktarı; en az 3 okuma bir lokus için yeterli) ve -r 0.5 (popülasyon içerisindeki örneklerin %50'si tarafından paylaşılan lokusları tut) için komut bloğundaki parametreleri değiştireceksin. İlk aşamada-popülasyon-I aşamasında tüm lokusların büyük bir kısmını elde etmek en iyi yoldur.

II. R ile filtrasyon

Sonrasında R betiği ile Populastin-I basamağında elde edilen vcf dosyasının filtrasyonu yapılır

Bununla iki filtrasyon yapıyor

- i) aşırı varyasyon gösteren sonlardaki bazların uzaklaştırılması ve
- ii) theta parametresine göre aşırı varyasyonel lokusların filtrasyonu.

R'da filtrasyon yaptıktan sonra iki dosya oluşturursun birisi white diğeri black. White senin filtre ettiğin temizlediğin verileri içeren, filtrasyon sonrası uzaklaştırılan veriler, black list ise içerisinde aşırı varyasyon ya da kirlilik barındıran okumalar bulunur (örneğin blacklist dekiler mtgenoma ait diziler olabilir)

III. Filtrasyon:

1- filogenetik analizler için stacks populations da filtrasyon, veri dosyalarının oluşturulması ve PLINK ile missing data kontrolü

Filogenetik analizler için stacks da global filtrasyon yaparak missing data miktarı %50 civarına düşürülecek. Sonrasında oluşturulacak phylip file ile analizler yapılacak.

2- popülasyon genetiği analizleri için PLINK ile filtrasyon

Elde edilen white dosyasıyla popülasyonda vcf ya da plink dosyası oluşturuyorsun bu kez plink programı kullanılarak popülasyon genetiği analizleri için filtrasyon yapacaksın. Öte yanda bazı bireylerde oldukça düşük okumalar var onları uzaklaştırmak gerekebilir bu yapacağın analizlere bağlı.

Popülasyon genetiği analizleri için <%25 missing data kullanımı analizlerin güvenilirliği için iyidir.

Stacks da ilk filtrasyonu yaparken
İlk filtrasyon populasyon stacks amaç ilkin filtrasyonu yapıp VCF file oluşturmak
Daha önceden process radtags, uestacks, cstacks ve sstacks ile oluşturduğın dosyalara, popmap dosyasına (lusch_popMap.txt) ve iş yükleneceği slurm dosyasına ihtiyacın var. Hepsini stacksout doyası içerisine dizilerin olduğu yere koyduk.
Populasyon da ilk filtrasyonu yapmak ve vcf dosyasını oluşturmak için 14 örneklik PopMap dosyası aşağıdaki şekilde oluşturuldu.
<pre> led-1 led-1 orb-1 orb tun-1 tun-1 egr-1 egr-1 heller-1 heller-1 birter-1 birter-1 birbak-1 birbak-1 birkem-1 birkem-1 ludavz-1 ludavz-1 luslagn-1 luslagn-1 luspat-1 luspat-1 chobakd-1 chobakd-1 chobtylos-1 chobtylos-1 chobesn3-1 chobesn3-1 </pre>
<p>Bu aşamada istersen üç parametre ile filtrasyon da yapabilirsin istersen sadece plink ya da R da filtrasyon için dosya oluşturabilirsin. Eğer herhangi bir filtrasyon yapılmayacaksa r ve p değerlerini düşük ya da sıfır tutarak-tüm veriyi istiyorum anlamında dosyaları üretebilirsin</p> <p>-r 0 populasyon içindeki bireylerdeki tüm snpsleri istiyorum demek. -p 2 en az 2 populasyonda paylaşılan lokusları tut -m 5 coverage, derinlik --min_maf 0 tüm locileri tut hatta varyasyonel olmasa bile --max_obs_het 0.5 tüm aleleri tut dersin çünkü diploit bireyler için, yeni versiyonlarda devamlı değişebilir önce kontrol et -t 12 core --vcf --plink 2>&1</p> <p>Detaylı bilgiyi http://catchenlab.life.illinois.edu/stacks/comp/populations.php linkten bulabilirsin.</p> <p>Eğer küdeme yüklüyse stacks modülleri yüklemeyi unutma. Aşağıdaki komut bloklarını okuttuktan sonra</p>
<pre> module av stacks module load stacks/1.41 </pre>
aşağıdaki komut bloğunu ile işi yükle
<pre> #!/bin/bash #SBATCH -p mercan #SBATCH -A sarkaya #SBATCH -J popl #SBATCH -N 1 #SBATCH -n 24 #SBATCH --time=01-05:00:00 #SBATCH --mail-type=ALL #SBATCH --mail-user=kaya_sarp@hotmail.com export OMP_NUM_THREADS=24 echo "SLURM_NODELIST \$SLURM_NODELIST" echo "NUMBER OF CORES \$SLURM_NTASKS" cd /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/5-populations-l/popl/ populations -b 1 -P /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/5-populations-l/popl/ -M /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/5-populations-l/popl/popMap.txt -p 2 -m 3 -r 0.5 -t 24 --min_maf 0 -- max_obs_het 0.5 --vcf 2>&1 exit </pre>
Not: Bu aşamada genellikle parametreleri düşük tutarak yüksek miktarda lokus elde edilip sonrasında yapılacak analizlere

göre missing data miktarı belirlenir ve değer değiştirilebilir.

Missign data özellikle filogenetik analizlerde kullanışlı olur dış grubu ayırmak için

Analiz sonunda dosyaların görünümü aşağıdaki gibi

```
-bash-4.2$ ls
1-process_radtag 2-ustacks 3-cstacks 4-sstacks 5-populations-I 6-populations-II RAXML structure
-bash-4.2$
-bash-4.2$ cd 5-populations-I/
-bash-4.2$ ls
pop1 pop1-slurm
-bash-4.2$ cd pop1/
-bash-4.2$ ls
batch_1.catalog.alleles.tsv.gz birkem-1.fq.gz chobakd-1.tags.tsv.gz egr-1.models.tsv.gz lUSDavz-1.fq.gz luspat-1.tags.tsv.gz
batch_1.catalog.snps.tsv.gz birkem-1.matches.tsv.gz chobesn3-1.alleles.tsv.gz egr-1.snps.tsv.gz lUSDavz-1.matches.tsv.gz orb-1.alleles.tsv.gz
batch_1.catalog.tags.tsv.gz birkem-1.models.tsv.gz chobesn3-1.fq.gz egr-1.tags.tsv.gz lUSDavz-1.models.tsv.gz orb-1.fq.gz
batch_1.haplotypes.tsv birkem-1.snps.tsv.gz chobesn3-1.matches.tsv.gz heller-1.alleles.tsv.gz lUSDavz-1.snps.tsv.gz orb-1.matches.tsv.gz
batch_1.hapstats.tsv birkem-1.tags.tsv.gz chobesn3-1.models.tsv.gz heller-1.fq.gz lUSDavz-1.tags.tsv.gz orb-1.models.tsv.gz
batch_1.populations.log birter-1.alleles.tsv.gz chobesn3-1.snps.tsv.gz heller-1.matches.tsv.gz luslagn-1.alleles.tsv.gz orb-1.snps.tsv.gz
batch_1.sumstats_summary.tsv birter-1.fq.gz chobesn3-1.tags.tsv.gz heller-1.models.tsv.gz luslagn-1.fq.gz orb-1.tags.tsv.gz
batch_1.sumstats.tsv birter-1.matches.tsv.gz chobtylos-1.alleles.tsv.gz heller-1.snps.tsv.gz luslagn-1.matches.tsv.gz popMap.txt
batch_1.vcf birter-1.models.tsv.gz chobtylos-1.fq.gz heller-1.tags.tsv.gz luslagn-1.models.tsv.gz tun-1.alleles.tsv.gz
birbak-1.alleles.tsv.gz birter-1.snps.tsv.gz chobtylos-1.matches.tsv.gz led-1.alleles.tsv.gz luslagn-1.snps.tsv.gz tun-1.fq.gz
birbak-1.fq.gz birter-1.tags.tsv.gz chobtylos-1.models.tsv.gz led-1.fq.gz luslagn-1.tags.tsv.gz tun-1.matches.tsv.gz
birbak-1.matches.tsv.gz chobakd-1.alleles.tsv.gz chobtylos-1.snps.tsv.gz led-1.matches.tsv.gz luspat-1.alleles.tsv.gz tun-1.models.tsv.gz
birbak-1.snps.tsv.gz chobakd-1.fq.gz chobtylos-1.tags.tsv.gz led-1.models.tsv.gz luspat-1.fq.gz tun-1.snps.tsv.gz
birbak-1.tags.tsv.gz chobakd-1.matches.tsv.gz egr-1.alleles.tsv.gz led-1.snps.tsv.gz luspat-1.matches.tsv.gz tun-1.tags.tsv.gz
birkem-1.alleles.tsv.gz chobakd-1.models.tsv.gz egr-1.fq.gz led-1.tags.tsv.gz luspat-1.models.tsv.gz
-bash-4.2$
```

Populasyonla ilk filtrasyonu yaptık ve içlerinde vcf fileda olduğu 5 farklı dosya elde ettik.

```
batch_1.haplotypes.tsv
batch_1.hapstats.tsv
batch_1.populations.log
batch_1.sumstats_summary.tsv
batch_1.sumstats.tsv
batch_1.vcf
```

bu dosyaların isimlerini ilk kısımları katalog ile aynı olduğu için bu dosyaları aşağıdaki komutla bir klasör yarattıp ve onun içerisine attın.

Bu dosyalardan batch.vcf (bu dosyayı oluturmak için komut bloğuna --vcf yazdık) R da filtrasyon için kullanacağız.

```
mkdir pop1_r0.5m3p3_310517 && mv batch_1.[h,p,s,v]* pop1_r0.5m3p3_310517
```

Sonrasında batch_1.vcf dosyasını aşağıdaki komutla masa üstüne indirdim

```
scp -r
sarkaya@levrek1.ulakbim.gov.tr:/truba_scratch/sarkaya/lusch_130517/outstacks/pop1_r0.5m3p3_310517/batch_1.vcf
/home/mobaxterm/Desktop/
```

İlk filtrasyon sonucu 263878 lokus elde edildi 14 bireyden

R ile II. filtrasyon whitelist oluşturma

R da filtrasyon için **reformat_vcfOK_mod_startingZERO.r** dosyası kullanılacak. Bu dosya içerisindeki baştaki adrese çalıştığınız dosyaların adres yolunu yazın ve bu dosyadaki komutlarla veriyi filtre edip whitelist dosyası oluşturun. Sonrasında whitelist dosya ile III. filtrasyon için gerekli plink dosyaları oluşturulacak.

Reformat_vcfOK_mod_startingZERO.r dosyası

her bir komutu tek tek okut

R scripti ile iki filtrasyon gerçekleştirirsin

3. aşırı varyasyonel olan son kısımdaki bazıları uzaklaştırırsın,

ii) thetaya göre aşırı varyasyonel lokuslar uzaklaştırılır.

Elde edilen vcf dosyasında R da ilk filtrasyonu yapmak için **reformat_vcfOK_mod_startingZERO_DEAsrpEm.r**

#November 17, 2015 – Andrea Thomaz

#read .vcf file from stacks output (WITHOUT write_random_SNP flag) for:

#plot the frequency of variable sites per position along all loci

#calculate theta based on number of segregating sites and individuals to create blacklist to delete very variable loci.

Require(plyr)

require(pegas)

```

#READ VCF
setwd("C:/Users/user/Desktop/aa") dosyanın path'i
data <- read.table('batch_1.vcf', header = FALSE, sep = "\t") populations-l basamağı sonunda ilk filtrasyonla
oluşturulan vcf dosyası ve üzerinde çalışılacak dosya

head(data,10)

#SEQUENCE LENGTH
seq_len <- 140 #MODIFY HERE according the length of the sequence (deletes positions at the end of seq, 10 less
than orig seq)

#selecting loci ID and position from column $V3
#this is for ID column problem in stacks v.1.46
loci_num<-as.numeric(sub('_', '', data$V3))
pos2 <- as.numeric(sub('.', '', data$V3))

#creates dataframe with loci ID, the variable positions and the number of individuals in each loci
new_data <- data.frame(loci_ID = loci_num,
  pos_vcf1 = data[,2],
  pos = pos2,
  #pos_dea = (data[,2] - seq_len*(loci_num-1))-2,
  ind = rowSums(data[,10:length(data)] != "0"))

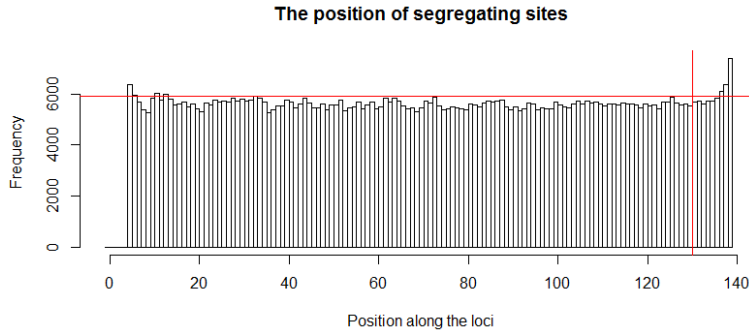
head(new_data)
min(new_data$pos)#should always be position 5 (first positions are the adapters)
max(new_data$pos)#should always be position 139
length(unique(new_data$loci_ID))#how many loci do I have

yukardaki komutları okuttuğunda aşağıdaki sonuçları elde dersin
  loci_ID pos_vcf1 pos ind
1      14      1840  18   7
2      14      1851  29   7
3      14      1852  30   7
4      14      1858  36   7
5      14      1859  37   7
6      14      1870  48   7
> min(new_data$pos)#should always be position 5 (first positions are the adapters)
[1] 5
> max(new_data$pos)#should always be position 139
[1] 139
> length(unique(new_data$loci_ID))#how many loci do I have
[1] 162831 hatırlarsan popuations sonunda 176897 locusun var diyordu burada bu kadar göstermesinin sebebi büyük
oranda en sonraki bazda varolan yüksek varyasyonun dikkate alınmaması

par(mar = rep(2, 4)) bunu okutmayınca bazen plot sorun çıkarabiliyor
#saving graph with frequency of variable sites along the loci
#pdf("./SNPdistr_pos140bp.pdf") path'i göster bununla grafiğin pdf sini elde edersin böyle kapalı
hist(new_data$pos, xlim = c(-1,seq_len), breaks = c(seq(-1, seq_len-1, by=1)), xlab = 'Position along the loci',
main = 'The position of segregating sites');
abline(5900, 0, col = "red")#helps to find where starts to increase toward the end, last positions have strong
increase
abline(v = 130, col = "red")#helps to figure out where to cut off before increase in bad calls
#move the lines around to visualize depending on my case
#dev.off()

```

Yukardaki komutları okuttuğunda aşağıdaki plotu elde ederiz.



Genomdaki mutasyon eğiliminin nötral olmasından dolayı normalde tüm lokuslarda mutasyon dağılımının aşağı yukarı benzer olmasını bekleriz. Grafikte de bu görülüyor ancak özellikle sondaki pozisyonlarda aşırı bir mutasyon varlığı görülüyor bunun sebebi dizileme sırasında son kısımlarda gözlenen hatalı okumalar ya da hizalamadır. Bunların veriden uzaklaştırılması gerekir. Bu grafiğe göre son kısımlarda 6. Pozisyonda aşırı varyasyon görülüyor. İşaretleme sondan 10 baz olarak görülüyor ancak sondan 5 bazın silinmesi daha iyi. Ancak filtrasyonu 10 ile devam ettirdim.

#BASE ON THE GRAPH, CHOOSE HOW MANY POSITION TO DELETE FROM THE END

to_del <- 10 #how many sites to delete in the end of the sequence

#10 is based on the 130 I chose 24000 he ab line above

seq_len_cut <- seq_len - to_del

yukardaki komutla sondan kaç tane bazı silmeye karar verdiysen o rakamı yazarak sondan o kadar baz silinir.

Aşağıdaki komutla silinene baza göre whitelist oluşturulur ve silinmiş pozisyonları gösteren histogram oluşturulur

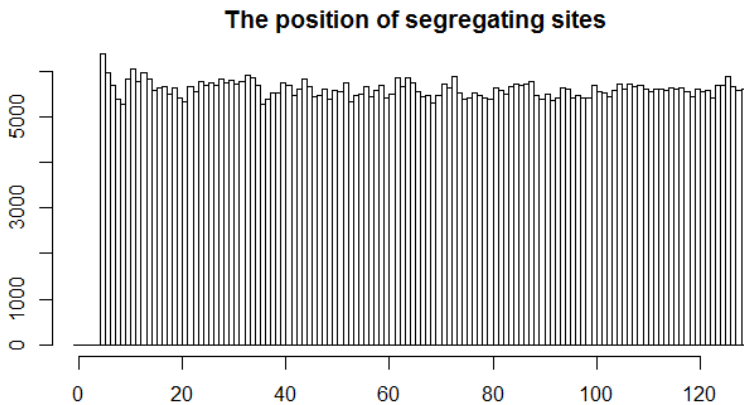
#create a whitelist to exclude those 10 (to_del) positions

whitelist <- subset(new_data, pos < seq_len_cut)[,c(1,3,4)]

#pdf("./SNPdistr_pos_cutto125bp.pdf") path'i göster bununla grafiğin pdf sini elde edersin böyle kapalı

hist(whitelist\$pos, xlim = c(0,seq_len_cut), breaks = c(seq(-1, seq_len_cut -1 , by=1)), xlab = 'Position along the loci', main = 'The position of segregating sites');

#dev.off()



Aşağıdaki komutlarla ikinci filtrasyona başlarsın

#calculating theta for all loci

var.sites <- count(whitelist, "loci_ID")

length(var.sites\$loci_ID)

max(var.sites\$freq)

theta_calc <- merge(unique(whitelist[,2]), var.sites, by = "loci_ID")

theta_calc\$theta <- 0

head(theta_calc)

for (i in 1:length(theta_calc\$theta)){

theta_calc[i,4] <- theta.s(theta_calc\$freq[i], theta_calc\$ind[i])/seq_len_cut

```

}

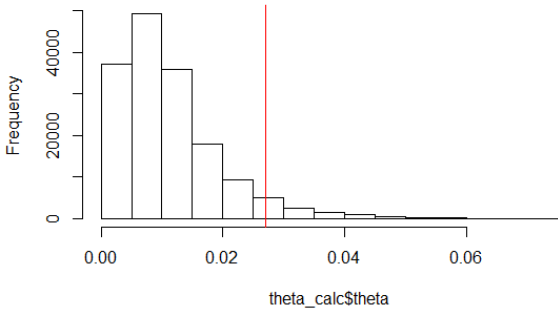
#calculating the 95% quantile to exclude loci extremely variable
quant <- quantile(theta_calc$theta, probs=0.95) #set the value to be
quant

yukardaki komutları okuttuğumuzda aşağıdaki değerleri elde ederiz.
> length(var.sites$loci_ID)
[1] 160195   sondaki varyasyonel bazlar uzaklaştırıldıktan sonra elde edile lokus sayısı
> max(var.sites$freq)
[1] 43   mevcut lokuslar içerisinde lokus içerisinde görülen maksimum varyasyon miktarı 43. Bunu
var.sites$freq komutunu okutarak görebilirsin.
> theta_calc <- merge(unique(whitelist[,2]), var.sites, by = "loci_ID")
> theta_calc$theta <- 0
> head(theta_calc)
  loci_ID ind freq theta
1      14 100   6     0
2      38 100   3     0
3      43 100  23     0
4      49 100   3     0
5      55 100  15     0
6      61 100   7     0
> for (i in 1:length(theta_calc$theta)){
+   theta_calc[i,4] <- theta.s(theta_calc$freq[i], theta_calc$ind[i])/seq_len_cut
+ }
> quant <- quantile(theta_calc$theta, probs=0.95) #set the value to be
> quant
      95%
0.0270943
theta değeri veri içerisindeki aşırı varyasyonel lokusların varlığı hakkında bilgi verir ve elde
edilen %95'lik güven dağılımının dışında kalanlar veriden uzaklaştırılır.
Bu varyasyonlar okuma hatlarından, pseudo genlerden ve olası mtgenom bulaşığından kaynaklanıyor büyük
oranda bu nedenle genomdan uzaklaştırılır.

#pdf("./theta125bp.pdf")   pah'i göster bununla grafiğin pdf sini elde edersin böyle kapalı
hist(theta_calc$theta)
abline(v = quant, col="red")
#dev.off()

Yukarıdaki komutla aşağıdaki histogram elde edilir
Histogram of theta_calc$theta

```



```

#what is the maximum number of mutations in a loci
max(theta_calc$freq) #max theta before
x <- subset(theta_calc, theta < quant)
max(x$freq) #max theta after, make sure is realistic for a 140 bp sequence
#think about what mutation rate the spp might have

> max(theta_calc$freq) #max theta before
[1] 43   bu değer theta ile filtrasyon öncesi görülen lokus içi maksimum varyasyon miktarı
> x <- subset(theta_calc, theta < quant)
> max(x$freq) #max theta after, make sure is realistic for a 140 bp sequence
[1] 17   bu değer ise theta ile filtrasyon sonrası lokus içi maksimum varyasyon miktarı. Aşırı varyasyonel lokusl
arı temizledik.

```

```
#saving whitelist for re-run populations in stacks
blacklist <- subset(theta_calc, theta > quant)[,1]
#write.table(blacklist, file="blacklist.txt", sep = '\n', row.names = F, col.names = F) black list gerekliyse aç
```

Yukardaki komutla theta ile uzaklaştırılan lokuslar black liste aktarılır.

Aşağıdaki komutlar ile de filtrasyon sonrası elde edilen lokuslar whiteliste yazdırılır.

```
#removes the blacklist from the whitelist and write off white list
whitelist$blacklist <- match(whitelist$loci_ID, blacklist, nomatch = 0)
whitelist_final <- subset(whitelist, blacklist == 0)
length(unique(whitelist_final$loci_ID)) #number of unique loci, this is the number I need to get out with "write random loci"
length(whitelist_final$loci_ID) #number of snps
write.table(whitelist_final[,1:2], file="whitelist_BRACHY.txt", sep = '\t', row.names = F, col.names = F)
```

yukardaki komutları run ettiğimizde sonuç olarak elimizde 152837 lokus ve bunlardaki SNPs sayısı.

```
> whitelist$blacklist <- match(whitelist$loci_ID, blacklist, nomatch = 0)
> whitelist_final <- subset(whitelist, blacklist == 0)
> length(unique(whitelist_final$loci_ID)) #number of unique loci, this is the number I need to get out with "write random loci"
[1] 152192 mevcut lokusu sayın filtrasyonlar sonrası
> length(whitelist_final$loci_ID) #number of snps
[1] 567979
> write.table(whitelist_final[,1:2], file="whitelist_BRACHY.txt", sep = '\t', row.names = F, col.names = F)
```

Bu işlem sonunda white list **whitelist_BRACHY.txt**

Sstacks sonrası population ile ilk sonrasında R da ikinci filtrasyonu yaptıktan sonra elde ettiğimiz **whitelist_BRACHY.txt** dosyası ile popülasyon aşamasına gider ve analiz yapmak istediğimiz veri dosyalarını uygun filtrasyonlara göre elde etmek için filtrasyona gidersin ve sonunda analizler için gerekli dosyaları elde edersin.

3. Filtrasyon stacks da dosya dönüştürme ve plink filtrasyon

3.1. Filogenetik analizler için stacks populations ile veride filtrasyon ve PLINK ile kontrol

Bunun için PopMap.txt dosyasını aşağıdaki gibi her bire bir popülasyon gibi düzenleyeceksin.

İsimlerini gir ama phylip file maksimum 5 karaktere kadar bu nedenle 5 karaktere göre yeni isimleri belirle

```
led-1 led
orb-1 orb
tun-1 tun
egr-1 egr
heller-1 heller
birter-1 bir_1
birbak-1 bir_2
birkem-1 bir_3
lusdavz-1 lus_1
luslagn-1 lus_2
luspat-1 lus_3
chobakd-1 chob_1
chobtylos-1 chob_2
```

aşağıdaki komutla global filtrasyon yapıp filogenetik analizler için phylip ve kontrol içinde plink dosyalarını oluşturacaksın. Bu filtrasyon için P ve r parametrelerini değiştir.

Filogenetik analizler için phylip dosyası oluşturmak için komut

```
populations -b 1 -P /truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/6-populations-II/popII-2/ -M
/truba_scratch/sarkaya/lusch_2_250717/EGBKO2018_srp/6-populations-II/popII-2/popMap.txt -W whitelist_BRACHY.txt -r
```



```
0.5 -p 2 -m 3 --min_maf 0 --max_obs_het 0.5 --write_random_snp -t 28 --phylip_var_all --phylip --plink 2>&1
```

materyal metoda belirteceğin için excelde iyi notlarını tut analizlerin
filogenetik analizler için missing data miktarı <%50 olmalı. İki adet veri seti elde edersen birisi her lokus başına SNP ve diğer tüm dizin. Elde ettiğin phylip dosyalarını RaxML de eğer PGDSpider programı ile Nexus a dönüştürsen PAUP4 de NJ ya da SVDquartetts analizi yapabilirsin

-W ile whitelist dosyası okutulur

3.2. populasyon gentiği analizleri için Plink de filtrasyon

Stacks da ve R ile filtrasyonda missing datayı birey ya da lokus bazında kontrol edemiyorsun bunlarla global filtrasyon yapıyorsun. Filtrasyon aşamasında plink programından da yararlanılacak bu nedenle stacks komut bloklarına plink flagını ekle.

İlk populasyon aşamasında filtrasyon yaptın ve vcf file oluşturduğun, sonrasında R'da II. filtrasyonu yapıp whitelist'i oluşturduktan, sonra tekrar populations ile plink dosyalarını oluşturacağız ve bu dosyalarla plink'te filtrasyonu yapacağız.

Bu filtrasyonları yaparkenki mantık en az missing data ve en fazla veri dengesini kurmak. Bu yönde filtrasyonu yapmak. Ayrıca yapacağın analize görede filtrasyonları da missign data miktarını değiştirmelisin

Populasyon için popmap.txt dosyan populasyonlara göre aşağıdaki gib olmalı

```
led-1 led
orb-1 orb
tun-1 tun
egr-1 egr
heller-1 heller
birter-1 bir
birbak-1 bir
birkem-1 bir
lusdavz-1 lus
luslagn-1 lus
luspat-1 lus
chobakd-1 chob
chobtylos-1 chob
```

İlk olarak aşağıdaki komutla whitelist i trubaya yükledim

```
scp -r /home/mobaxterm/Desktop/whitelist_BRACHY.txt
sarkaya@levrek1.ulakbim.gov.tr:/truba_scratch/sarkaya/lusch_130517/outstacks
```

Plink dosyasını elde etmek için aşağıdaki gibi slurm dosyasını hazırladım ve job olarak yükledim ve yaklaşık 15dk sürdü

```
#!/bin/bash
#SBATCH -p sardalya
#SBATCH -A sarkaya
#SBATCH -J pop-llus
#SBATCH -N 1
#SBATCH -n 14
#SBATCH --time=10:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=kaya_sarp@hotmail.com
```

```
export OMP_NUM_THREADS=14
```

```
echo "SLURM_NODELIST $SLURM_NODELIST"
echo "NUMBER OF CORES $SLURM_NTASKS"
```

```
cd /truba_scratch/sarkaya/lusch_130517/outstacks/
```

```
populations -b 1 -P /truba_scratch/sarkaya/lusch_130517/outstacks/ -M
/truba_scratch/sarkaya/lusch_130517/outstacks/popMap.txt -W whitelist_BRACHY.txt -r 0.5 -p 2 -m 3 --min_maf 0 --
```

```
max_obs_het 0.5 --write_random_snp -t 14 --plink 2>&1
```

```
exit
```

--write_random_snp her lokusdan rastgele SNPs leri structur dosyasına yazdırılır

Burada tüm parametreleri aynı tutup değiştirmedığımız için filtrasyon yapmadık sadece yeni dosyaları oluşturduk.

Sonrasında Plinke ait map ve ped dosyalarını masa üstüne aşağıdaki komutlarla indirdim

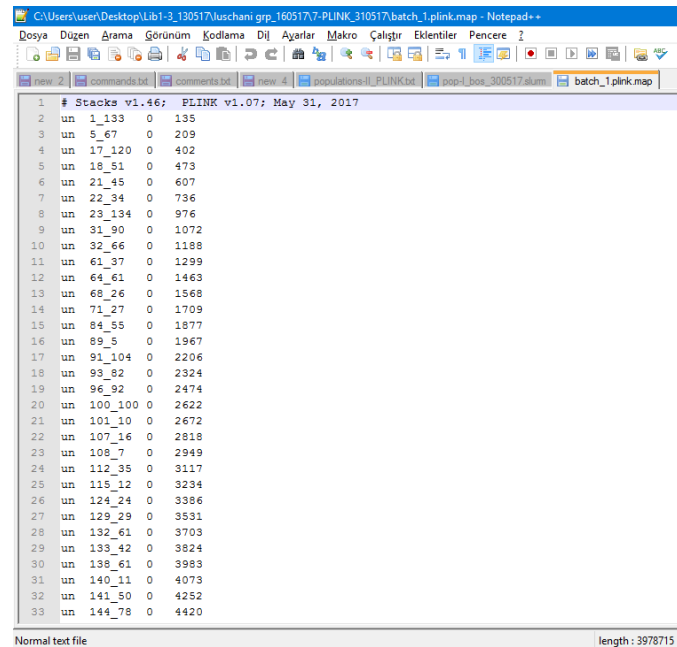
```
scp -r sarkaya@levrek1.ulakbim.gov.tr:/truba_scratch/sarkaya/lusch_130517/outstacks/pop2_afterwhite_r0.5-p2-m3_310517/batch_1.plink.map /home/mobaxterm/Desktop/
```

PLINK

<http://zzz.bwh.harvard.edu/plink/summary.shtml#missing>

Plink programı ile elde edilen dosyadan bireyler ya da lokuslar çıkarılarak veri setinde missing data miktarı azaltılacak. Bu yolla filtrasyon yapılarak sonunda structure ve arlequin de analizler için str. dosyası oluşturulacak.

Oluşturulan batch_1.plink.map ve batch_1.plink.ped dosyalarının içerinse baktığımızda aşağıdaki gibi olduğu görülür.



```
C:\Users\user\Desktop\Lab1-3_130517\luschani_grp_160517\7-PLINK_310517\batch_1.plink.map - Notepad++
Dosya Düzen Arama Görünüm Kodlama Dil Ayarlar Makro Çalıştır Eklentiler Pencere ?
new 2 commands.txt comments.txt new 4 populations-II_PLINK.txt pop-I_box_300517_alum batch_1.plink.map
1 # Stacks v1.46; PLINK v1.07; May 31, 2017
2 un 1_133 0 135
3 un 5_67 0 209
4 un 17_120 0 402
5 un 18_51 0 473
6 un 21_45 0 607
7 un 22_34 0 736
8 un 23_134 0 976
9 un 31_90 0 1072
10 un 32_66 0 1188
11 un 61_37 0 1299
12 un 64_61 0 1463
13 un 68_26 0 1568
14 un 71_27 0 1709
15 un 84_55 0 1877
16 un 89_5 0 1967
17 un 91_104 0 2206
18 un 93_82 0 2324
19 un 96_92 0 2474
20 un 100_100 0 2622
21 un 101_10 0 2672
22 un 107_16 0 2818
23 un 108_7 0 2949
24 un 112_35 0 3117
25 un 115_12 0 3234
26 un 124_24 0 3386
27 un 129_29 0 3531
28 un 132_61 0 3703
29 un 133_42 0 3824
30 un 138_61 0 3983
31 un 140_11 0 4073
32 un 141_50 0 4252
33 un 144_78 0 4420
Normal text file length: 3978715
```

Verideki missing data miktarını görmek için trubadan indirdiğim batch_1.plink.map ve batch_1.plink.ped dosyalarını P. luschani grp analizi yaptığım klasör içerisine koydum. Ayrıca bu klasör içerisine plink.exe executable dosyasına da kopyaladım. Sonrasında terminali açtım ve verilerin olduğu kalasöre girdim. Bu klasör içerisinde trubadan indirdiğim batch_1.plink.map ve batch_1.plink.ped dosyaları ve plink.exe bulunmaktadır. Verideki missing data miktarına yönelik değerleri görmek için aşağıdaki komutu kullandım

```
./plink --file batch_1.plink --allow-extra-chr --missing --out plusch_miss_stat
```

```
[2017-05-31 16:26:48] ~/Desktop/Lib1-3_130517/luschani_grp_160517/7-PLINK_310517
[user.sarpus] > ls
PLINK_comments.txt  batch_1.plink.map  batch_1.plink.ped  plink.exe  prettify.exe

[2017-05-31 16:26:49] ~/Desktop/Lib1-3_130517/luschani_grp_160517/7-PLINK_310517
[user.sarpus] > ./plink --file batch_1.plink --allow-extra-chr --missing --out plusch_miss_stat
PLINK v1.90b4.3 64-bit (9 May 2017) www.cog-genomics.org/plink/1.9/
(C) 2005-2017 Shaun Purcell, Christopher Chang GNU General Public License v3
Logging to plusch_miss_stat.log.
Options in effect:
--allow-extra-chr
--file batch_1.plink
--missing
--out plusch_miss_stat

3988 MB RAM detected; reserving 1994 MB for main workspace.
.ped scan complete (for binary autoconversion).
Performing single-pass .bed write (165973 variants, 119 people).
--file plusch_miss_stat-temporary.bed + plusch_miss_stat-temporary.bim +
plusch_miss_stat-temporary.fam written.
165973 variants loaded from .bim file.
119 people (0 males, 0 females, 119 ambiguous) loaded from .fam.
Ambiguous sex IDs written to plusch_miss_stat.nosex.
Using 1 thread (no multithreaded calculations invoked).
Before main variant filters, 119 founders and 0 nonfounders present.
Calculating allele frequencies... done.
Total genotyping rate is 0.0969265.
--missing: Sample missing data report written to plusch_miss_stat.imiss, and
variant-based missing data report written to plusch_miss_stat.lmiss.

[2017-05-31 16:27:51] ~/Desktop/Lib1-3_130517/luschani_grp_160517/7-PLINK_310517
[user.sarpus] >
```

Bu komut 4 adet dosya oluşturur

miss_stat.imiss bireylere ait (individuals) missing data'yı verir
miss_stat.lmiss lokuslara ve snps miktarlarına yönelik bilgiler verir
miss_stat.log analizin sürecine ait bilgiler verir
miss_stat.nosex oluşturduğunuz popMap dosyasının aynısıdır
miss_stat.imiss dosyasına baktığımızda

FID	Family ID (populasyon adı burada)
IID	Individual ID
MISS_PHENO	Missing phenotype? (Y/N)
N_MISS	Number of missing SNPs
N_GENO	Number of non-obligatory missing genotypes (toplam lokus sayısı)
F_MISS	Proportion of missing SNPs

	FID	IID	MISS_PHENO	N_MISS	N_GENO	F_MISS
1						
2	birbak	birbak-1	Y	125632	165973	0.7569
3	birbak	birbak-2	Y	130475	165973	0.7861
4	birbak	birbak-3	Y	132046	165973	0.7956
5	birbak	birbak-4	Y	124339	165973	0.7492
6	birbak	birbak-5	Y	125249	165973	0.7546
7	birbak	birbak-6	Y	164303	165973	0.9899
8	birkem	birkem-1	Y	161416	165973	0.9725
9	birkem	birkem-2	Y	165200	165973	0.9953
10	birkem	birkem-3	Y	160557	165973	0.9674
11	birkem	birkem-4	Y	161599	165973	0.9736
12	birkem	birkem-5	Y	161159	165973	0.971
13	birkem	birkem-6	Y	165218	165973	0.9955
14	birolm	birolm-1	Y	155743	165973	0.9384
15	birolm	birolm-2	Y	155916	165973	0.9394
16	birolm	birolm-3	Y	156204	165973	0.9411
17	birolm	birolm-4	Y	165890	165973	0.9995
18	birolm	birolm-5	Y	162452	165973	0.9788
19	birolm	birolm-6	Y	165356	165973	0.9963
20	birtah	birtah-1	Y	128984	165973	0.7771
21	birtah	birtah-2	Y	125436	165973	0.7558
22	birtah	birtah-3	Y	127488	165973	0.7681
23	birtah	birtah-4	Y	126600	165973	0.7628
24	birtah	birtah-5	Y	121303	165973	0.7309
25	birtah	birtah-6	Y	165118	165973	0.9948
26	birter	birter-1	Y	135464	165973	0.8162
27	birter	birter-2	Y	133125	165973	0.8021
28	birter	birter-3	Y	133418	165973	0.8039
29	birter	birter-4	Y	136326	165973	0.8214
30	birter	birter-5	Y	142669	165973	0.8596
31	birter	birter-6	Y	164787	165973	0.9929
32	chobakd	chobakd-1	Y	156986	165973	0.9459
33	chobakd	chobakd-2	Y	139825	165973	0.8425

Sonrasında bu dosyadaki verileri Excel sayfasına kopyalayıp yapıştırdığımızda (noktalari virgül yap) sıralayarak ve grafiklerle bireylerdeki ve popülasyonlardaki missing veri miktarını görebiliriz. Bu sonuçlara göre veriden bireyleri ya da popları çıkarabiliriz.

miss_stat.imiss dosyasına baktığımızda

SNP	SNP identifier
CHR	Chromosome number
N_MISS	Number of individuals missing this SNP
N_GENO	Number of non-obligatory missing genotypes
F_MISS	Proportion of sample missing for this SNP

	CHR	SNP	N_MISS	N_GENO	F_MISS
1					
2	un	1_133	93	119	0.7815
3	un	5_67	102	119	0.8571
4	un	17_120	99	119	0.8319
5	un	18_51	106	119	0.8908
6	un	21_45	106	119	0.8908
7	un	22_34	110	119	0.9244
8	un	23_134	99	119	0.8319
9	un	31_90	107	119	0.8992
10	un	32_66	110	119	0.9244
11	un	61_37	100	119	0.8403
12	un	64_61	106	119	0.8908
13	un	68_26	107	119	0.8992
14	un	71_27	112	119	0.9412
15	un	84_55	109	119	0.916
16	un	89_5	113	119	0.9496
17	un	91_104	75	119	0.6303
18	un	93_82	104	119	0.8739
19	un	96_92	108	119	0.9076
20	un	100_100	97	119	0.8151
21	un	101_10	107	119	0.8992
22	un	107_16	112	119	0.9412
23	un	108_7	113	119	0.9496
24	un	112_35	109	119	0.916
25	un	115_12	111	119	0.9328
26	un	124_24	113	119	0.9496
27	un	129_29	99	119	0.8319
28	un	132_61	106	119	0.8908
29	un	133_42	111	119	0.9328
30	un	138_61	112	119	0.9412
31	un	140_11	112	119	0.9412
32	un	141_50	104	119	0.8739
33	un	144_78	97	119	0.8151

Burada ise lokuslara ve SNPs yönelik missign data bilgilerini bulabilirsin. Bu veriyi de Excel sayfasına aktarip orda fitler la küçükten büyüğe sıralayarak hangi lokus ve snpslerin ne kadar birey tarafından paylaşıldığını görebilirsin. Özellikle Fastsimcoal analiz gibi missing data içermeyen verilerle çalışan programlarda yapacağın analizlerde tüm bireyler tarafından paylaşılan SNPs buradan seçerek veri dosyasını oluşturabilirsin.

SNPs or Loci extraction

Örneğin miss_stat.imiss dosyasını excelle kopyaladık ve bireylerde en fazla missing görülen SNPs veriden uzaklaştırmak istiyoruz. Bunun için önce excelde en az sayıdaki SNPs belirleriz ve aşağıdaki gibi kopyalar yeni bir text file yapıştırıp

kaydederiz. Aşağıda 30SNPs seçildi veri setinden uzaklaştırılması için

1	CHR	SNP	N_MISS	N_GENO	F_MISS
2	0	49_65	95	100	0.95
3	0	68_67	94	100	0.94
4	0	87_100	95	100	0.95
5	0	104_44	89	100	0.89
6	0	116_39	95	100	0.95
7	0	125_21	89	100	0.89
8	0	126_33	96	100	0.96
9	0	143_38	63	100	0.63
10	0	164_95	92	100	0.92
11	0	191_42	93	100	0.93
12	0	272_48	71	100	0.71
13	0	285_52	83	100	0.83
14	0	298_27	90	100	0.9
15	0	318_81	87	100	0.87
16	0	377_34	95	100	0.95
17	0	393_11	89	100	0.89
18	0	397_43	95	100	0.95
19	0	459_88	94	100	0.94
20	0	466_11	95	100	0.95
21	0	487_7	93	100	0.93
22	0	516_58	94	100	0.94
23	0	522_37	88	100	0.88
24	0	629_112	95	100	0.95
25	0	630_42	96	100	0.96
26	0	631_80	91	100	0.91
27	0	643_31	94	100	0.94
28	0	652_52	95	100	0.95
29	0	729_35	92	100	0.92
30	0	761_15	88	100	0.88

Sonrasında aşağıdaki komutla bu SNPs veri setinden uzaklaştırırız

```
./plink --file batch_1.plink --exclude excludesnps.txt --allow-extra-chr 0 --recode --out new_file_name
```

Komutu çalıştırdığımızda

```

/home/mobaxterm/Desktop/Plink/Stable (beta 4.3, 9 May)
1.sarkaya@evrek1.ulakbim.gov.tr 2./home/mobaxterm
Error: Failed to open excludesnps.txt.

[2017-05-22 15:06.43] ~/Desktop/Plink/Stable (beta 4.3, 9 May)
[user.sarpus] > ./plink --file batch_1.plink --exclude exclude.txt --allow-extra-chr 0 --recode --out new_file_name
PLINK v1.90b4.3 64-bit (9 May 2017) www.cog-genomics.org/plink/1.9/
(C) 2005-2017 Shaun Purcell, Christopher Chang GNU General Public License v3
Logging to new_file_name.log.
Options in effect:
--allow-extra-chr 0
--exclude exclude.txt
--file batch_1.plink
--out new_file_name
--recode

3988 MB RAM detected; reserving 1994 MB for main workspace.
.ped scan complete (for binary autoconversion).
Performing single-pass .bed write (67998 variants, 100 people).
--file: new_file_name-temporary.bed + new_file_name-temporary.bim +
new_file_name-temporary.fam written.
67998 variants loaded from .bim file.
100 people (0 males, 0 females, 100 ambiguous) loaded from .fam.
Ambiguous sex IDs written to new_file_name.nosex.
--exclude: 67969 variants remaining.
Using 1 thread (no multithreaded calculations invoked).
Before main variant filters, 100 founders and 0 nonfounders present.
Calculating allele frequencies... done.
Total genotyping rate is 0.0981796.
67969 variants and 100 people pass filters and QC.
Note: No phenotypes present.
--recode ped to new_file_name.ped + new_file_name.map ... done.

[2017-05-22 15:08.17] ~/Desktop/Plink/Stable (beta 4.3, 9 May)
[user.sarpus] >

[2017-05-22 15:23.30] ~/Desktop/Plink/Stable (beta 4.3, 9 May)
[user.sarpus] >

```

Bu SNPs veri setinden uzaklaştırıldı. Bu filtrasyon sonucunda 4 farklı dosya oluştu bunlar:

new_file_name.log analiz hakkında bilig verir
new_file_name.map filtrasyon sonrası oluşturulan yeni map file
new_file_name.ped filtrasyon sonrası oluşturulan yeni ped file
new_file_name.nosex bireylerin ve popların simini içeren popMap dosyası

Sonrasında seçilen SNPs uzaklaştırıldığını kontrol etmek için aşağıdaki komutu kullandık

```
./plink --file new_file_name --allow-extra-chr --missing --out miss_stat
```

Yeni oluşturulan map ve pad dosyalarına göre elde edilen miss_stat.lmiss dosyasına baktığımızda seçilen SNPs veriden uzaklaştırıldığı görülür.

SNPs or Loci keeping

Fastsimcoal gibi analizler için veri setinde hiç missing data olmamalı bu nedenle verisetindeki tüm bireyler tarafından paylaşılan lokusları seçmen gerek. Bunun için ilk olarak aşağıdaki komutla elindeki structure dosyasından miss_stat.lmiss

dosyasını ve ona bakrak lokuslara ait missing verileri elde edeceksin.

```
./plink --file batch_1.plink --missing --out miss_stat --allow-extra-chr 0
```

Sonrasında bu verileri excele aktar ve küçükten büyüğe sıralatarak tüm bireylerde paylaşılan (N_MISS sıfır olacak) lokusları seçip kopyala ve txt file oluştur. Sonrasında bu text file ile mevcut map ve ped dosyalarından yeni map ve ped dosyaları oluştur bunun için aşağıdaki komutu kullan.

```
./plink --file batch_1.plink --extract keep.txt --allow-extra-chr --recode --out new_file_name --make-bed
```

Bu komutla sadece istenilen lokusları içeren .bim, .fam, .bed, map ve ped dosyaları oluşturulur.

Sonrasında yeni oluşturulan new_file_name.map ve ped dosyalarında istenilen lokusların olup olmadığını görmek için aşağıdaki komutla miss_stat.lmiss dosyasını oluşturduk. Excele bu dosyayı açıp baktığımızda seçilen lokusların olduğu görülür.

```
./plink --file new_file_name --missing --out miss_stat --allow-extra-chr 0
```

Bu veri dosyasından structure file oluşturmak için aşağıdaki komutu kullanman gerekir (bfile veya file kullan)

```
./plink --bfile new_file_name --allow-extra-chr --recode-structure
```

exclude individuals

Bunun için whitelist sonrası oluşturulan plink dosyaları aç bireylerden hangilerinde missing data a hangilerini çıkarmak istiyorsan belirle ve bu bireylerin olmadığı aşağıdaki gibi bir popmap text dosyası oluştur. Bu dosyayı oluşturan .nosex uzantılı dosyadada kopyalayarak oluşturabilirsin

```
led      1.1_leder
led      1.2_leder
led      1.3_leder
orb      2.1_orbe
orb      2.2_orbe
orb      2.3_orbe
tun      3.1_tun
tun      3.2_tun
tun      3.3_tun
egr      4.1_egr
egr      4.2_egr
egr      4.3_egr
hel      5.1_hel
hel      5.2_hel
hel      5.3_hel
birter   8a-1_lsh_bir
birter   8a-2_lsh_bir
birter   8a-3_lsh_bir
birter   8a-4_lsh_bir
birter   8a-5_lsh_bir
birbak   8b-1_lsh_bir
birbak   8b-2_lsh_bir
....
```

Sonrasında mevcut plink dosyalarından çıkarılacak bireylerin listelendiği text dosyası ve aşağıdaki komut kullanılarak bireylerin çıkarıldığı yeni plink dosyaları oluşturulur.

Make bed komutunu çıkara bilirsin bed dosyalarını gerek yoksa

```
./plink --file batch_1.plink --remove exindividuals.txt --out exind --make-bed --allow-extra-chr --recode
```

Bu komuttan sonra oluşan exind uzantılı yeni map ve ped fileları yine aşağıdaki komutla açarak yeni missing lokus dağılımını gözleyebilirsin

```
./plink --file exind.plink --missing --out miss_stat --allow-extra-chr 0
```

exclude loci and individuals

Whitelist sonrası populations ile oluşturulan plink dosyasından birey ve lokus filtrasyonunu aynı anda yapmak istiyorsan

çıkarılacak lokus ve bireylere yönelik iki ayrı text dosyası oluşturacaksın --exclude komutu ile lokus textindekileri --remove komutu ile birey textindekileri veriden uzaklaştırılacak.

```
./plink --file batch_1.plink --exclude ListRareAlleles.txt --remove STTTORVGP_3indsToExclude.txt --allow-extra-chr --recode-structure
```

plink file iki tane map ve ped

plink ile veri setinden istediğin bireyleri ve alelleri/lokusları veriden uzaklaştırabilirsin

Plink ile populations daki r ve p komutlarının yaptığı işi tüm veri çapında değil birey ve lokus düzeyinde yapabiliyorsun.

Birey başına missign data özellikle populasyon analizleri için çok önemli

Çok lokus için çok birey, çok birey için de çok lokus silersin dolayısıyla denge lazım

PCA bireylerdeki missing datanın yaratabileceği sıkıntıları gözlemlemeden iyi bir yol böylece bireyleri gözlemleyebilirsin

Filogenetik analizler için maksimum %50 missign

Populasyon genetiği analizleri için maksimum %20 missign

fastsimcoal gibi analizler için veri setinde hiç missing data olmamalı

RAxML'de filogenetik analiz

Filogenetik analizler u.snps.phy veri dosyası kullanarak yapıldı. Analizler için sadece tuncayi ve luschani bireylerinden oluşan 61 örneklik bir veri dosyası ve tüm türlerin bulunduğu 119 örneklik diğer bir veri dosyası ile analizler yapıldı.

lusc_61_inv.u.snps.phy 48701 lokus 61 birey

lusc_61_inv.u.snps.phy dosyası ile aşağıdaki komut ile 200 bottstraplık RAxML analizi yapıldı.

Kullanılan veri snps ler lokus olduğu için bu model kullanıldı. Bu komutta köşeli parantez içerisindeki diğer bireyin yazılmasın bence gerek yok teki yeter eğer farklı bir tür varsa onu gir. Bu şekilde diğer bireyinde diğer takson olarak aldı ve monofiletik bağlamadı

Bu veri seti tüm bireyleri ve bu bireylerdeki her lokusdan bir SNP içermektedir. Analiz için aşağıdaki slurm komutu kullanıldı

```
#!/bin/bash
#SBATCH -p sardalya
#SBATCH -A sarkaya
#SBATCH -J RAxMLus61_u.snps
#SBATCH -N 1
#SBATCH -n 28
#SBATCH --time=08-05:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=kaya_sarp@hotmail.com
```

```
export OMP_NUM_THREADS=28
```

```
echo "SLURM_NODELIST $SLURM_NODELIST"
```

```
echo "NUMBER OF CORES $SLURM_NTASKS"
```

```
cd /truba/home/sarkaya/RAxML-fluxCompiled/lsuch/
```

```
../raxmlHPC-PTHREADS-SSE3 -s /truba/home/sarkaya/RAxML-fluxCompiled/lsuch/lusc_61_inv.u.snps.phy -n
```

```
lusc_61_u.snps.txt -m GTRGAMMA -O -T 28 -f a -x 12345 -p 12345 -# 200 -o tun-1 [, tun-3]
```

```
exit
```

Bu analiz sonunda aşağıdaki dosyalar oluştu bu dosyalardan RAxML_bipartitions.lusc_61.snps.txt içerisinde bootstraplu ağaç var

RAxML_bestTree.lusc_61.snps.txt

RAxML_bipartitions.lusc_61.snps.txt RAxML_info.lusc_61.snps.txt

RAXML_bipartitionsBranchLabels.lusc_61.snps.txt RAXML_bootstrap.lusc_61.snps.txt

Analiz sonucunda oluşan ağaca baktığımda özellikle biroylm-4 bireyinin missgn datadan dolayı dış grupla çıktığı görülüyor bu nedenle yüksek missing içeren aşağıdaki bireyleri veri dosyasından uzaklaştırarak tekrar analiz başlatacağım.

Structure

Plinkte filtrasyon sonucu 16 tür bulunduğu %35 missing lokus içeren 157 bireylik 1864 unlinked lokus ile structure analizi yapılacak.

Plink str file da modifikasyonlar

1- Veri setini plinkte oluşturduktan sonra bu dosyayı kopyala bu ana file kalsın sonra kopya plink plink file in uzantsını.str olarak değiştir

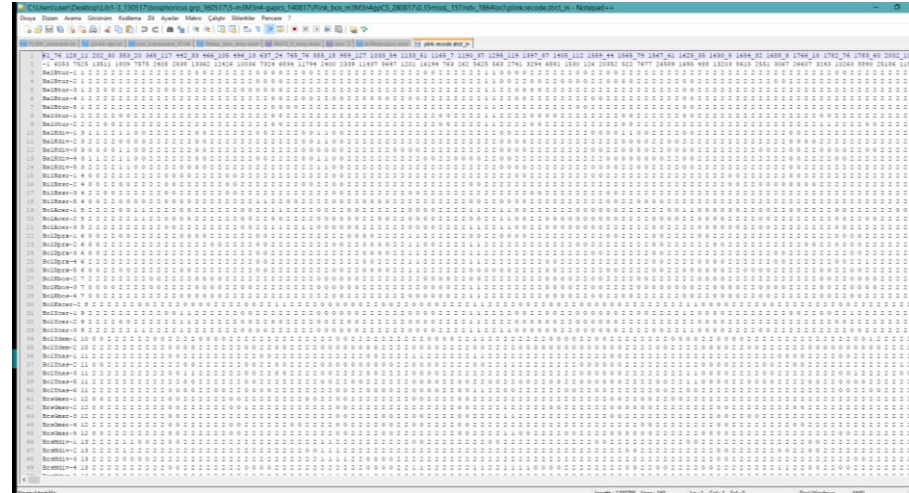
2- bu dosyayı Notepad++ aç ve üste iki satırda bulunan lokus numaralarını sil.

3- bireylere ait pop numara modifikasyonlarını Notepad++ da yap.

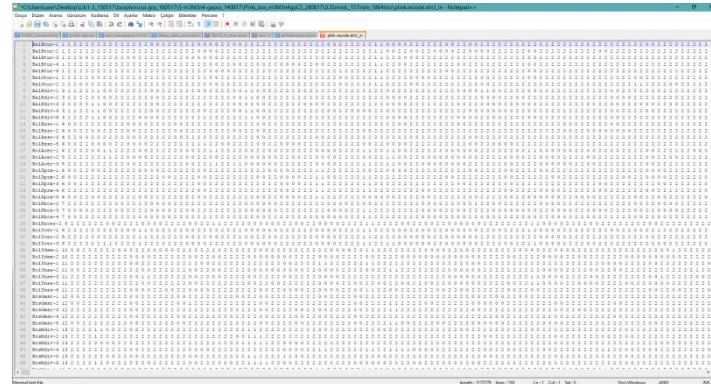
4- tüm düzenlemeleri yaptıktan sonra notepad++ da düzen----EOL dönüştürme----unix biçimine dönüştür diyerek kaydet. Sonra kümeye at.

Not: Sakın Excelde açıp sonra tekrar başka bir yere atma bir sürü değişiklik yapıyor ve structure bu file ı okumuyor. Excele sadece lokus sayısına bakmak için açabilirsin oda 16bin üzerindeki bakamazsın

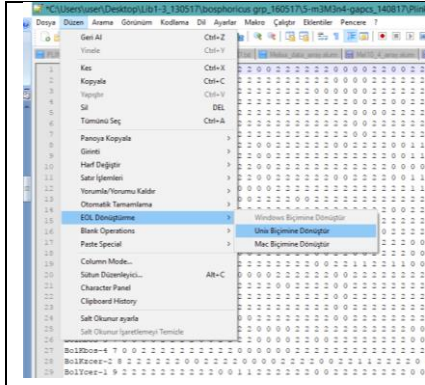
Plink dosyası aşağıdaki gibi



Sonra ilk iki satısı sil



Sonra unix formunda kaydet



Analiz için Mainparams file 'a ihtiyacın var. Bu dosyaların sayısı kaç K parametresinde deneme yapmak istiyorsan o kadar olacaklar.

Mainparams file da veri dosyası plinkte hazırlandığı için ONEROWPERIND 1 olarak girildi. Eğer stacks dan ya da ipyrad dan elde edilseydi 0 girilecektir.

Mainparams dosyasına bakacak olursak en baştaki satırda dosya adı

```
#define OUTFILE ochotona_k2      dosya adı, burada K2 ye göre analiz yapılacak
#define INFILE ochotona-structure.tsv    analizde kullanılacak veri dosyasının adı
#define NUMINDS 56      veri dosyasındaki birey sayısı, kaçtane bireyin var
#define NUMLOCI 4857    veri dosyasındaki lokus sayısı
#define LABEL 1         bireylerin adlarının bulunduğu sütünün kaç tane eğer plink se tek
#define POPDATA 1       popülasyonlar ait sütun kaçtan
#define POPFLAG 0       popflag yok anlamında
#define LOCDATA 0       lokalite bilgisi yok
#define PHENOTYPE 0     fenotip bilgisi yok
#define MARKERNAMES 0
#define MAPDISTANCES 0
#define ONEROWPERIND 0   eğer plink dosyası ise 1, yani lokus iki sutunda birey tek satırda ise 1
#define PHASEINFO 0
#define PHASED 0
#define RECESSIVEALLELES 0
#define EXTRACOLS 0
#define MISSING -9      missing data -9 olarak işaretli anlamında, plink dosyasında 0
#define PLOIDY 2         diploit olduğu için 2
#define MAXPOPS 2        kaç adet popülasyon var bu değer = denemek istenen K parametresi, burada k 2
#define BURNIN 5000      bunları verisetine göre büyüte birlisin uzun süre
#define NUMREPS 20000    aynı şekilde veri setine göre arttırabilirsin coverage sağlamak için mcmc de

#define NOADMIX 0
#define LINKAGE 0
#define USEPOPINFO 0

#define LOCPRIOR 1       this tells the program to use PopData to set the locations
#define INFERALPHA 1     Infer the value of the model parameter  $\alpha$  from the data; otherwise  $\alpha$  is fixed at the value ALPHA
                        which is chosen by the user
#define ALPHA 1.0
#define POPALPHAS 0
#define UNIFPRIORALPHA 1
#define ALPHAMAX 10.0
#define ALPHAPROPSD 0.025

#define FREQSCORR 1.0

#define INFERLAMBDA 0
```

```
#define POPSPECIFCLAMBDA 0
#define LAMBDA 1.0
#define COMPUTEPROB 1
#define PFROMPOPFLAGONLY 0
#define ANCESTDIST 0
#define STARTATPOPINFO 0
#define METROFREQ 10
```

```
#define UPDATEFREQ 1000
```

K1 için Main param file aşağıdaki gibi girildi

```
#define OUTFILE Mel_10_4_k1
#define INFILE Mel_10_4.str
#define NUMINDS 157
#define NUMLOCI 1864
#define LABEL 1
#define POPDATA 1
#define POPFLAG 0
#define LOCDATA 0
#define PHENOTYPE 0
#define MARKERNAMES 0
#define MAPDISTANCES 0
#define ONEROWPERIND 1
#define PHASEINFO 0
#define PHASED 0
#define RECESSIVEALLELES 0
#define EXTRACOLS 0
#define MISSING 0
#define PLOIDY 2
#define MAXPOPS 1
#define BURNIN 200000
#define NUMREPS 500000
```

```
#define NOADMIX 0
#define LINKAGE 0
#define USEPOPINFO 0
```

```
#define LOCPRIOR 1
#define INFERRALPHA 1
#define ALPHA 1.0
#define POPALPHAS 0
#define UNIFPRIORALPHA 1
#define ALPHAMAX 10.0
#define ALPHAPROPSD 0.025
```

```
#define FREQSCORR 1.0
```

```
#define INFERRLAMBDA 0
#define POPSPECIFCLAMBDA 0
#define LAMBDA 1.0
#define COMPUTEPROB 1
#define PFROMPOPFLAGONLY 0
#define ANCESTDIST 0
#define STARTATPOPINFO 0
#define METROFREQ 10
```

```
#define UPDATEFREQ 1000
#define RANDOMIZE 0
```

İlk denemede 10 K parametresi ile 5 tekrarlı olarak analiz yapıldı. Bunun için slurm iş yükleme dosyasını aşağıdaki gibi düzenledim.

Burada 5 farklı array.slurm dosyası oluştur ve bunları ayrı ayrı yükle. Amaç bu ayrı komut dosyaları

ile ayrı ayrı 10 tekrarlı 5 dosyayı başlatıp hızlıca analizi bitirmek. Array yaparken çevre değişkenlerine yeni komutlar ekliyorsun aşağıdaki gibi.
Seed numarasına dikkat et çünkü eğer aynı anda başlarsa analizler aynı seed verilerse sonuçlar aynı olur.

Not: Sardalyad array çalışmadı ama long ve meracanda sorun olmadı

```
#!/bin/bash
#SBATCH -p long
#SBATCH -A sarkaya
#SBATCH -J 1.bos_str
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --time=14-05:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=kaya_sarp@hotmail.com
#SBATCH --array=1-10
#SBATCH --output=slurm-%A_%a.out

export OMP_NUM_THREADS=8

echo "SLURM_NODELIST $SLURM_NODELIST"
echo "NUMBER OF CORES $SLURM_NTASKS"

echo "SLURM_JOBID: " $SLURM_JOBID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID

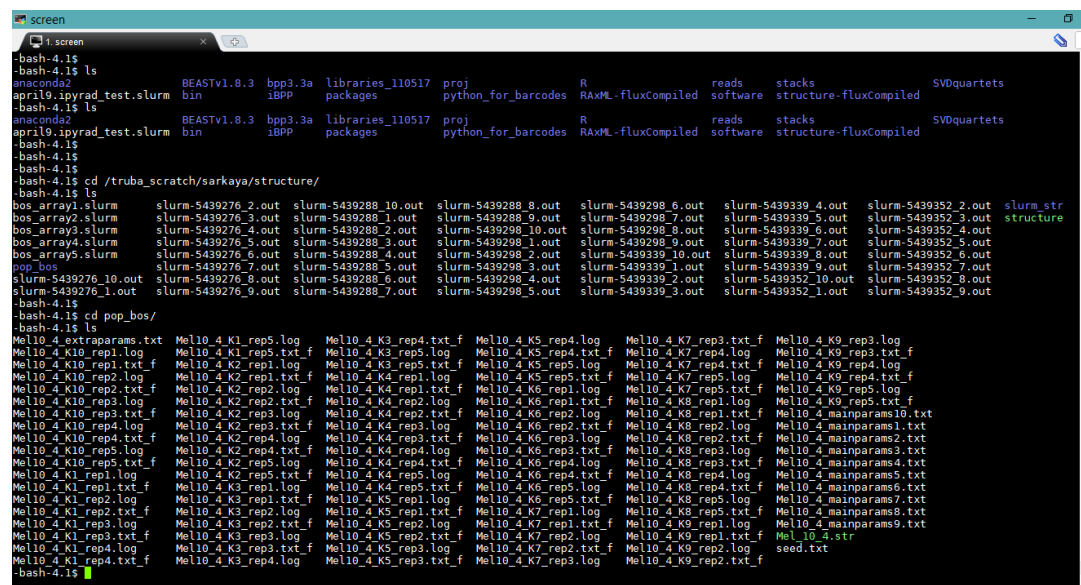
cd /truba_scratch/sarkaya/structure/pop_bos

structure -m Mel10_4_mainparams "${SLURM_ARRAY_TASK_ID}.txt" -e Mel10_4_extraparams.txt -K
"${SLURM_ARRAY_TASK_ID}" -D 11111 "${SLURM_ARRAY_TASK_ID}" -o Mel10_4_K "${SLURM_ARRAY_TASK_ID}"_rep1.txt
> Mel10_4_K "${SLURM_ARRAY_TASK_ID}"_rep1.log

exit
```

Analizleri bu şekilde başlat.

Analiz sonunda oluşan log ve txt_f dosyaları aşağıdaki gibi



```
screen
-bash-4.1$
-bash-4.1$ ls
anaconda2 BEASTv1.8.3 bpp3.3a libraries_110517 proj python_for_barcodes R RAXML-FluxCompiled reads stacks structure-FluxCompiled SVDquartets
-bash-4.1$ ls
anaconda2 BEASTv1.8.3 bpp3.3a libraries_110517 proj python_for_barcodes R RAXML-FluxCompiled reads stacks structure-FluxCompiled SVDquartets
-bash-4.1$
-bash-4.1$
-bash-4.1$ cd /truba_scratch/sarkaya/structure/
-bash-4.1$ ls
bos_array1.slurm slurm-5439276_2.out slurm-5439288_10.out slurm-5439298_8.out slurm-5439298_6.out slurm-5439339_4.out slurm-5439352_2.out slurm_str
bos_array2.slurm slurm-5439276_3.out slurm-5439288_1.out slurm-5439298_9.out slurm-5439298_7.out slurm-5439339_5.out slurm-5439352_3.out structure
bos_array3.slurm slurm-5439276_4.out slurm-5439288_2.out slurm-5439298_10.out slurm-5439298_8.out slurm-5439339_6.out slurm-5439352_4.out
bos_array4.slurm slurm-5439276_5.out slurm-5439288_3.out slurm-5439298_1.out slurm-5439298_9.out slurm-5439339_7.out slurm-5439352_5.out
bos_array5.slurm slurm-5439276_6.out slurm-5439288_4.out slurm-5439298_2.out slurm-5439339_10.out slurm-5439339_8.out slurm-5439352_6.out
pop_bos slurm-5439276_7.out slurm-5439288_5.out slurm-5439298_3.out slurm-5439339_1.out slurm-5439339_9.out slurm-5439352_7.out
slurm-5439276_10.out slurm-5439276_8.out slurm-5439288_6.out slurm-5439298_4.out slurm-5439339_2.out slurm-5439352_10.out slurm-5439352_8.out
slurm-5439276_1.out slurm-5439276_9.out slurm-5439288_7.out slurm-5439298_5.out slurm-5439339_3.out slurm-5439352_1.out slurm-5439352_9.out
-bash-4.1$ cd pop_bos/
-bash-4.1$ ls
Mel10_4_extraparams.txt Mel10_4_K1_rep5.log Mel10_4_K3_rep4.txt_f Mel10_4_K5_rep4.log Mel10_4_K7_rep3.txt_f Mel10_4_K9_rep3.log
Mel10_4_K10_rep1.log Mel10_4_K1_rep5.txt_f Mel10_4_K3_rep5.log Mel10_4_K5_rep4.txt_f Mel10_4_K7_rep4.log Mel10_4_K9_rep3.txt_f
Mel10_4_K10_rep2.log Mel10_4_K2_rep1.txt_f Mel10_4_K4_rep1.log_f Mel10_4_K6_rep5.txt_f Mel10_4_K7_rep5.log Mel10_4_K9_rep4.log
Mel10_4_K10_rep2.txt_f Mel10_4_K2_rep2.log Mel10_4_K4_rep1.txt_f Mel10_4_K6_rep1.log Mel10_4_K7_rep5.txt_f Mel10_4_K9_rep5.log
Mel10_4_K10_rep3.log Mel10_4_K2_rep2.txt_f Mel10_4_K4_rep2.log Mel10_4_K6_rep1.txt_f Mel10_4_K8_rep1.log Mel10_4_K9_rep5.txt_f
Mel10_4_K10_rep3.txt_f Mel10_4_K2_rep3.log Mel10_4_K4_rep2.txt_f Mel10_4_K6_rep2.log Mel10_4_K8_rep1.txt_f Mel10_4_mainparams10.txt
Mel10_4_K10_rep4.log Mel10_4_K2_rep3.txt_f Mel10_4_K4_rep3.log Mel10_4_K6_rep2.txt_f Mel10_4_K8_rep2.log Mel10_4_mainparams1.txt
Mel10_4_K10_rep4.txt_f Mel10_4_K2_rep4.log Mel10_4_K4_rep3.txt_f Mel10_4_K6_rep3.log Mel10_4_K8_rep2.txt_f Mel10_4_mainparams2.txt
Mel10_4_K10_rep5.log Mel10_4_K2_rep4.txt_f Mel10_4_K4_rep4.log Mel10_4_K6_rep3.txt_f Mel10_4_K8_rep3.log Mel10_4_mainparams3.txt
Mel10_4_K10_rep5.txt_f Mel10_4_K2_rep5.log Mel10_4_K4_rep4.txt_f Mel10_4_K6_rep4.log Mel10_4_K8_rep3.txt_f Mel10_4_mainparams4.txt
Mel10_4_K1_rep1.log Mel10_4_K3_rep1.txt_f Mel10_4_K4_rep5.log Mel10_4_K6_rep4.txt_f Mel10_4_K8_rep4.log Mel10_4_mainparams5.txt
Mel10_4_K1_rep1.txt_f Mel10_4_K3_rep1.log Mel10_4_K4_rep5.txt_f Mel10_4_K6_rep5.log Mel10_4_K8_rep4.txt_f Mel10_4_mainparams6.txt
Mel10_4_K1_rep2.log Mel10_4_K3_rep1.txt_f Mel10_4_K5_rep1.log Mel10_4_K6_rep5.txt_f Mel10_4_K8_rep5.log Mel10_4_mainparams7.txt
Mel10_4_K1_rep2.txt_f Mel10_4_K3_rep2.log Mel10_4_K5_rep1.txt_f Mel10_4_K7_rep1.log Mel10_4_K8_rep5.txt_f Mel10_4_mainparams8.txt
Mel10_4_K1_rep3.log Mel10_4_K3_rep2.txt_f Mel10_4_K5_rep2.log Mel10_4_K7_rep1.txt_f Mel10_4_K9_rep1.log Mel10_4_mainparams9.txt
Mel10_4_K1_rep3.txt_f Mel10_4_K3_rep3.log Mel10_4_K5_rep2.txt_f Mel10_4_K7_rep2.log Mel10_4_K9_rep1.txt_f Mel10_4_str
Mel10_4_K1_rep4.log Mel10_4_K3_rep3.txt_f Mel10_4_K5_rep3.log Mel10_4_K7_rep2.txt_f Mel10_4_K9_rep2.log seed.txt
Mel10_4_K1_rep4.txt_f Mel10_4_K3_rep4.log Mel10_4_K5_rep3.txt_f Mel10_4_K7_rep3.log Mel10_4_K9_rep2.txt_f
-bash-4.1$
```

Analiz bittikten sonra analiz dosyalarını kümeden masa üstüne indir ve K parametresini hesaplamak için <http://taylor0.biology.ucla.edu/structureHarvester/> sayfasına sonuçları yükle. Bu sayfaya sadece .txt_f uzantılı

dosyaları yükleyeceksin. Bunun için tüm analizlerin .txt_f dosyalarını bir klasöre kopyala ve zipleyip (rar değil zip ile) sayfaya yükledim.

The Evanno table output is also available as a tab-delimited text file (for use with Excel) [here](#).

K	Reps	Mean LnP(K)	Stdev LnP(K)	Ln[K]	Ln ² [K]	Delta K
1	5	-55083.960000	0.634023	—	—	—
2	5	-47873.360000	1.197080	17210.600000	10813.780000	9033.466423
3	5	-41476.540000	0.907193	6396.620000	1166.680000	1286.032178
4	5	-36246.400000	1.787178641	5230.140000	2549.940000	1.426796
5	5	-33566.200000	454.117738	2680.200000	136.260000	0.300054
6	5	-31022.260000	471.549979	2543.940000	722.700000	1.532695
7	5	-29291.020000	500.344089	1821.240000	127.440000	0.254795
8	5	-27597.220000	952.332724	1693.800000	156207.640000	164.026328
9	5	-162021.060000	343368.484719	-154513.840000	286817.580000	0.835305
10	5	-49717.320000	33678.516717	132303.740000	—	—

Analiz en olası K sayısını 2 olarak hesaplandı. Sonuçlar str_harvested_5x_310817 klasörü içerisinde. Elde edilen her bir K parametresine göre plotları elde etmek için CLUMPAK server'ı kullandım.

<http://clumpak.tau.ac.il/index.html>



bu server sana aynı anda tüm k kümelerine göre kümeleri verir ve şekli manupule etmenide sağlar. Bunun için structureHarvester server'ına yüklediğin zip filea ve bitanede popların/türlerin isimlerinin bulunduğu txt dosyasına ihtiyacın var.

Sayfada bulunan ilk/üstteki dosya seç kısmına zipli dosyayı ikinci/alttaki dosya seç kısmına türlerin numaralarının bulunduğu yada yeni isimlerinin yazılı olduğu txt dosyasını yükle.

please refer to the [Help](#) section.

Run Main Pipeline (click for [Instructions](#)):

Upload zip file containing STRUCTURE/ADMIXTURE runs: Dosya seçilmedi

Please indicate the format of the uploaded result files:

☒ STRUCTURE [example](#)

☐ ADMIXTURE [example](#)

Upload labels file for DISTRUCT (optional): Dosya seçilmedi

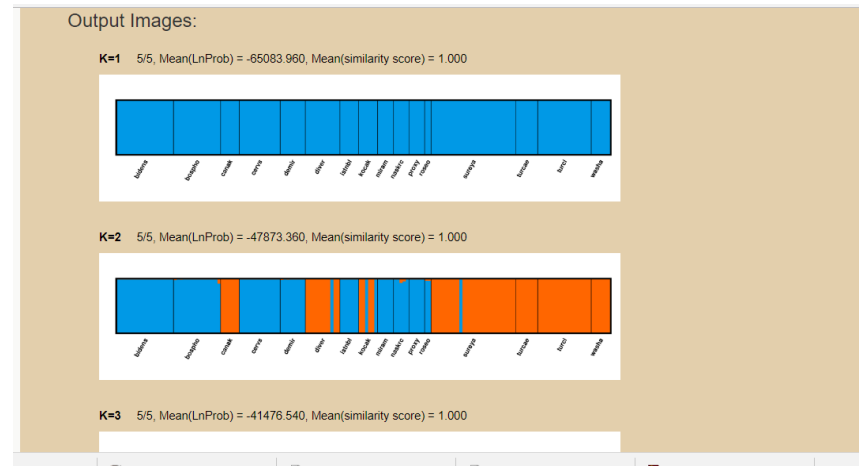
Str dosyasını oluştururken her bir türe numara vermiştim burada bunları değiştirebilir türlerin sırasını oluşturacağın txt dosyasındaki isim sırasına göre organize edebilirsin. Bunun için isim dosyasını aşağıdaki gibi oluşturdum. Numara ile isim arasındaki boşluk space olmalı

12 roseo
2 bos
7 istnbl
1 bidens
9 miram
11 proxy
4 cervus
5 demirsy
10 naskrck
8 kocak

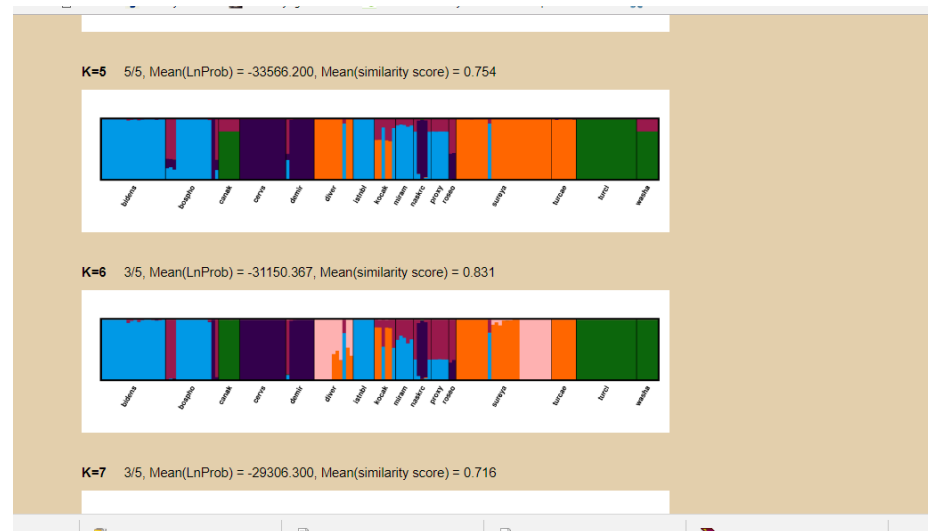
13 sureyins
6 diversus
14 turciae
15 turcicus
3 canak
16 washa

Bunları yükledikten sonra alltaki kısma email adresini ve gir ve submit et

Daha sonra server hesaplamaya ve grafikleri oluşturmaya başlar bunun için biraz zaman gerekli. Email adresinde email gönderiri başladığına ve bittiğine yönelik. İş yüklendiğinde sana bir job numarası verir. tüm .txt_f dosyalarındaki K parametrelerindeki tekrüre göre plotları teker teker oluştur.



Burada örneğin K=2 için 5/5 'in anlamı 5 denmenin 5 ide bu durmu desteklemiş. denen



K=6 için 3/5 de denen 5 tekrarın 3'ü bu durmu desteklemiş. Diğer plotlar sayfada aşağı inilerek görülebilir.

Analizi 5 tekrarlı yapmıştın 10 tekrarlı yapmak iş yüklemeye betiğini aşağıdaki gibi düzenle, sarı olan yerleri

değiştirdik

```
#!/bin/bash
#SBATCH -p mercan
#SBATCH -A sarkaya
#SBATCH -J 10.bos_str
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --time=14-05:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=kaya_sarp@hotmail.com
#SBATCH --array=1-10
#SBATCH --output=slurm-%A_%a.out

export OMP_NUM_THREADS=24

echo "SLURM_NODELIST $SLURM_NODELIST"
echo "NUMBER OF CORES $SLURM_NTASKS"

echo "SLURM_JOBID: " $SLURM_JOBID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID

cd /truba_scratch/sarkaya/structure/pop_bos

structure -m Mel10_4_mainparams"${SLURM_ARRAY_TASK_ID}".txt -e Mel10_4_extraparams.txt -K
"${SLURM_ARRAY_TASK_ID}" -D 21315"${SLURM_ARRAY_TASK_ID}" -o
Mel10_4_K"${SLURM_ARRAY_TASK_ID}"_rep10.txt > Mel10_4_K"${SLURM_ARRAY_TASK_ID}"_rep10.log

exit
```