Reference manual for PAC

The PAC reference manual provides with detailed information on PAC software. The manual starts with description how to obtain and run PAC. Following this, a description of each process within the PAC, starting from first to last process, is described.

Table of Contents

1.1 Software setup

PAC requires Nextflow, Java v8+, and a docker or singularity (depending on the profile the user selects).

To download PAC, download it from the GitHub with the following command:

```
git clone https://github.com/anna-saukkonen/PAC.git
```

To download Nextlow, run following command:

```
curl -fsSL get.nextflow.io | bash
```

1.2 PAC processes

This sections describes each process within PAC, as the software is written. However, once a process has available input files available from previous processes, it will start running to speed up the run time by parallelisation. See thesis section 3.3 for more information.

1.2.1 setting parameters

```
/*

 * Defines some parameters in order to specify the refence
genomes

 * and read pairs by using the command line options

 */


params.genome         = params.genomes[ params.genome_version
]?.genome

params.annot          = params.genomes[ params.genome_version
]?.annot

params.gencode_bed    = params.genomes[ params.genome_version
]?.gencode_bed



// Check if genome exists in the config file

if (params.genomes && params.genome_version &&
!params.genomes.containsKey(params.genome_version)) {

    exit 1, "The provided genome '${params.genome_version}' is
not available. Currently the available genomes are
${params.genomes.keySet().join(", ")}. Please check your
spelling."

}
```

```
if (!params.variants) exit 1, "Path to phased variants has to
be specified!"

if (!params.reads) exit 1, "Path to reads has to be
specified!"

if (!params.id) exit 1, "Sample ID not supplied, needs to be
same as in the VCF"



Channel

  .fromFilePairs(params.reads)

  .ifEmpty { exit 1, "Cannot find any reads matching:
${reads}\nNB: Path needs to be enclosed in quotes!\n"}

  .into {reads_ch; reads_ch1; reads_ch2; reads_ch3}
```

The first step, although not a process, checks that all essential parameters are specified when executing PAC. The essential parameters are the genome version, path to RNA-seq reads, path to variants VCF file, and sample ID. If any of these are missing, PAC stops the run and gives an error message stating which parameter is missing. This section also places RNA-seq reads into multiple channels as multiple processes take them as inputs.

1.2.2 process read_length

```
process read_length {


  input:

    set val(id), file(reads) from reads_ch


  output:

    file "readLength_file.txt" into readlen_file_ch


  shell:

  '''

  gunzip -c *_1.{fq,fastq}.gz | sed '2q;d' | wc -m | awk
'{print $1-1}' >> readLength_file.txt

  '''

}


readlen_file_ch.map { it.text.trim().toInteger() }.into {
read_len_ch1; read_len_ch2; read_len_ch3; read_len_ch4;
read_len_ch5; read_len_ch6 }
```

Input: This process takes in RNA-seq read files as input file.

Process: Custom bash script calculates the read length.

Output: The output is a file with read length value that is used in the downstream processes throughout PAC.

Outside of the process the value from the output file is placed into different channels as multiple processes need this value.

1.2.3 process prepare_star_genome_index

```
process prepare_star_genome_index {

  input:
    path genome from params.genome
    path annot from params.annot
    val x from read_len_ch1
    val cpus from params.cpus

  output:
    path STARhaploid into genome_dir_ch

  script:
  """
  mkdir STARhaploid
  STAR --runMode genomeGenerate \
       --genomeDir STARhaploid \
       --genomeFastaFiles ${genome} \
       --sjdbGTFfile ${annot} \
       --sjdbOverhang ${x} \
       --runThreadN ${cpus}
  """
}
```

Input: This process takes in the reference genome specified in options, annotation file, read length information from the previous process, and number of cpus as an optional input.

Process: It then generates a genome index with STAR --runMode genomeGenerate.

Output: The genome indices in STARhaploid directory. This step is necessary for standard alignment in the next process.

1.2.4 process rnaseq_mapping_star

```
process rnaseq_mapping_star {


  input:

    path genome from params.genome

    path STARhaploid from genome_dir_ch

    set val(id), file(reads) from reads_ch1

    val x from read_len_ch2

    val id from params.id

    val cpus from params.cpus


  output:

    tuple \

      val(id), \

path("${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.b
am"), \


path("${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.b
am.bai") into aligned_bam_ch


  script:

  """

  # Align reads to genome

  STAR --genomeDir STARhaploid \

       --readFilesIn ${reads} \

       --readFilesCommand zcat \

       --runThreadN ${cpus} \

       --outSAMstrandField intronMotif \
```

```
        --outFilterMultimapNmax 30 \
        --alignIntronMax 1000000 \
        --alignMatesGapMax 1000000 \
        --outMultimapperOrder Random \
        --outSAMunmapped Within \
        --outSAMattrIHstart 0 \
        --outFilterIntronMotifs RemoveNoncanonicalUnannotated \
        --sjdbOverhang ${x} \
        --outFilterMismatchNmax ${(x-(x%13))/13} \
        --outSAMattributes NH nM NM MD HI \
        --outSAMattrRGline  ID:${id} PU:Illumina PL:Illumina
LB:${id}.SOFT.NOTRIM SM:${id}.SOFT.NOTRIM CN:Seq_centre \
        --outSAMtype BAM SortedByCoordinate \
        --twopassMode Basic \
        --outFileNamePrefix ${id}.SOFT.NOTRIM.STAR.pass2. \
        --outSAMprimaryFlag AllBestScore
  # Index the BAM file
  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.bam
  """
}
```

Input: This process takes in the reference genome, genome index generated from process prepare_star_genome_index, read length information from process read_length, the RNA-seq reads, sample ID and number of cpus as an optional input.

Process: The step aligns reads to the reference genome and indexes the BAM file with SAMtools index. This process provides the standard alignment that the user can use as a comparison for the PAC results. The output also feeds into the phaser_step.

Output: BAM and BAM.bai files of mapped RNA-seq reads.

1.2.5 process clean_up_reads

```
process clean_up_reads {


  input:

    tuple val(id), path(bam), path(index) from aligned_bam_ch

    path variants from params.variants

    val id from params.id

    val cpus from params.cpus


  output:

    path ("STAR_original/phaser_version.bam") into phaser_ch

    path ("STAR_original/phaser_version.bam.bai") into
phaser_bai_ch

    path
("STAR_original/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") into pp_um_ch


  script:

  """

  mkdir STAR_original

  #KEEP ONLY PROPERLY PAIRED READS

  samtools view -@ ${cpus} -f 0x0002 -b -o
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.bam

  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam

  #KEEP UNIQUELY MAPPED READS

  samtools view -h
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam
| grep -P "NH:i:1\t|^@" | samtools view -bS - >
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am
```

```
  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am

  #Create BAM compatible with PHASER:

  gunzip -c ${variants} | grep -q 'chr' || (samtools view -h
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am | sed -e 's/chr//' >> phaser_version.sam; samtools view -bh
phaser_version.sam >> phaser_version.bam; samtools index
phaser_version.bam; rm phaser_version.sam)

  gunzip -c  ${variants} | grep -q 'chr' && (samtools view -bh
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am >> phaser_version.bam; samtools index phaser_version.bam)

  mv phaser_version.bam STAR_original/phaser_version.bam

  mv phaser_version.bam.bai
STAR_original/phaser_version.bam.bai

  mv
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am
STAR_original/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam

  """

}
```

Input: The process takes in the BAM files generated from process rnaseq_mapping_star, variants VCF file, sample ID and number of cpus as an optional input.

Process: In this step the mapped RNA-seq reads are filtered. SAMtools is used to keep only properly paired (where the read orientation of read pairs is as expected and the gap between them is likely based on sequencing technology) and uniquely mapped (reads mapping to single location) reads. The BAM is then created that is compatible for downstream process phaser_step.

Output: Properly paired and uniquely mapped BAM file, phaser_step process compatible BAM and BAI files in separate channels.

1.2.6 process phaser_step

```
process phaser_step {


  input:

  path variants from params.variants

  path ("phaser_version.bam") from phaser_ch

  path ("phaser_version.bam.bai") from phaser_bai_ch

  val id from params.id

  val cpus from params.cpus


  output:

  path ("${id}_output_phaser.vcf") into (phaser_out_ch1,
phaser_out_ch2)


  script:

  """

  tabix -f -p vcf ${variants}

  python2 /phaser/phaser/phaser.py --vcf ${variants} --bam
phaser_version.bam --paired_end 1 --mapq 0 --baseq 10 --isize
0 --include_indels 1 --sample ${id} --id_separator + --
pass_only 0 --gw_phase_vcf 1 --threads ${cpus} --o
${id}_output_phaser

  gunzip ${id}_output_phaser.vcf.gz

  rm phaser_version.bam

  rm phaser_version.bam.bai

  """

}
```

Input: Variants VCF file, BAM and BAI files from process clean_up_reads, sample ID and number of cpus.

Process: This step uses phASER to phase variants incorporating aligned RNA-seq reads. phASER uses a read-aware mode for phasing. It selects RNA-seq reads where there are two variants, that can be split across larger genomic distances due to splicing, hence it can incorporate variants over longer distances and thereby improve phasing. This allows better phasing of rare variants and longer haplotypes.

Output: Phased variants VCF file.

1.2.7 process create_parental_genomes

```
process create_parental_genomes {


  input:

    path genome from params.genome

    path annot from params.annot

    path ("${id}_output_phaser.vcf") from phaser_out_ch1

    val id from params.id

    path gencode_bed from params.gencode_bed


  output:

    path ("STAR_2Gen_Ref/maternal.chain") into
maternal_chain_ch

    path ("STAR_2Gen_Ref/paternal.chain") into
paternal_chain_ch

    path ("STAR_2Gen_Ref/${id}_maternal.fa") into (mat_fa1,
mat_fa2)

    path ("STAR_2Gen_Ref/${id}_paternal.fa") into (pat_fa1,
pat_fa2)

    path ("STAR_2Gen_Ref/mat_annotation.gtf") into
(mat_annotation_ch1, mat_annotation_ch2)

    path ("STAR_2Gen_Ref/not_lifted_m.txt") into not_lift_m_ch

    path ("STAR_2Gen_Ref/pat_annotation.gtf") into
(pat_annotation_ch1, pat_annotation_ch2)

    path ("STAR_2Gen_Ref/not_lifted_p.txt") into not_lift_p_ch

    path ("STAR_2Gen_Ref/map_over.txt") into adjusted_ref_ch

    path ("STAR_2Gen_Ref/${id}_output_phaser.mother.vcf.gz")
into mothervcf_ch

    path ("STAR_2Gen_Ref/${id}_output_phaser.father.vcf.gz")
into fathervcf_ch

    path ("STAR_2Gen_Ref/mat.bed") into mat_bed_ch
```

```
    path ("STAR_2Gen_Ref/pat.bed") into pat_bed_ch


  script:

  """

  mkdir STAR_2Gen_Ref

  java -Xmx10000m -jar /vcf2diploid_v0.2.6a/vcf2diploid.jar -
id ${id} -chr ${genome} -vcf ${id}_output_phaser.vcf -outDir
STAR_2Gen_Ref > logfile.txt



  liftOver -gff ${annot} STAR_2Gen_Ref/maternal.chain
STAR_2Gen_Ref/mat_annotation.gtf
STAR_2Gen_Ref/not_lifted_m.txt

  liftOver -gff ${annot} STAR_2Gen_Ref/paternal.chain
STAR_2Gen_Ref/pat_annotation.gtf
STAR_2Gen_Ref/not_lifted_p.txt

  liftOver ${gencode_bed} STAR_2Gen_Ref/maternal.chain
STAR_2Gen_Ref/mat.bed STAR_2Gen_Ref/not_bed_lifted_m.txt

  liftOver ${gencode_bed} STAR_2Gen_Ref/paternal.chain
STAR_2Gen_Ref/pat.bed STAR_2Gen_Ref/not_bed_lifted_p.txt




  cat STAR_2Gen_Ref/chr1_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr2_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr3_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr4_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr5_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr6_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa
```

```
  cat STAR_2Gen_Ref/chr7_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr8_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr9_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr10_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr11_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr12_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr13_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr14_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr15_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr16_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr17_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr18_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr19_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr20_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr21_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chr22_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chrX_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa

  cat STAR_2Gen_Ref/chrY_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa
```

```
  cat STAR_2Gen_Ref/chrM_${id}_maternal.fa >>
STAR_2Gen_Ref/${id}_maternal.fa




  cat STAR_2Gen_Ref/chr1_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr2_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr3_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr4_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr5_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr6_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr7_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr8_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr9_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr10_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr11_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr12_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr13_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr14_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr15_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa
```

```
  cat STAR_2Gen_Ref/chr16_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr17_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr18_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr19_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr20_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr21_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chr22_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chrX_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chrY_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa

  cat STAR_2Gen_Ref/chrM_${id}_paternal.fa >>
STAR_2Gen_Ref/${id}_paternal.fa




  sed 's/\\*/N/g' STAR_2Gen_Ref/${id}_maternal.fa >
STAR_2Gen_Ref/${id}_maternal.hold.fa

  mv STAR_2Gen_Ref/${id}_maternal.hold.fa
STAR_2Gen_Ref/${id}_maternal.fa



  sed 's/\\*/N/g' STAR_2Gen_Ref/${id}_paternal.fa >
STAR_2Gen_Ref/${id}_paternal.hold.fa

  mv STAR_2Gen_Ref/${id}_paternal.hold.fa
STAR_2Gen_Ref/${id}_paternal.fa

  mv ${id}_output_phaser.vcf
STAR_2Gen_Ref/${id}_output_phaser.vcf

  cd STAR_2Gen_Ref/
```

```
  perl ${baseDir}/bin/adjust_reference.pl
${id}_output_phaser.vcf ${id}

  perl ${baseDir}/bin/adjust_reference_vcf.pl
${id}_output_phaser.vcf ${id}

  grep "^#" ${id}_output_phaser.mother.vcf >
${id}_output_phaser.mother.s.vcf

  grep -v "^#" ${id}_output_phaser.mother.vcf | sort -k1,1V -
k2,2g >> ${id}_output_phaser.mother.s.vcf

  grep "^#" ${id}_output_phaser.father.vcf >
${id}_output_phaser.father.s.vcf

  grep -v "^#" ${id}_output_phaser.father.vcf | sort -k1,1V -
k2,2g >> ${id}_output_phaser.father.s.vcf

  mv ${id}_output_phaser.mother.s.vcf
${id}_output_phaser.mother.vcf

  mv ${id}_output_phaser.father.s.vcf
${id}_output_phaser.father.vcf

  bcftools view ${id}_output_phaser.mother.vcf -Oz -o
${id}_output_phaser.mother.vcf.gz

  bcftools view ${id}_output_phaser.father.vcf -Oz -o
${id}_output_phaser.father.vcf.gz

  tabix ${id}_output_phaser.father.vcf.gz

  tabix ${id}_output_phaser.mother.vcf.gz

  """

}
```

Input: The reference genome, annotation file, phased variants VCF file from process phaser_step, sample ID and BED annotation file.

Process: This step creates personalised parental genomes. The phased variants are incorporated into the reference genome using vcf2diploid, generating maternal and paternal genomes. liftOver is then used to generate GTF and BED files with adjusted genomic coordinates for maternal and paternal genomes. This is because the coordinates

will be shifted due to indels present in the VCF file. The custom scripts generate maternal and paternal VCF files where the heterozygous site coordinates are shifted to the maternal and paternal genomes.

Output: Maternal and paternal genomes, chain files for both genomes that are needed for liftOver (not needed in the downsteam process but output ensures files can be found on users' system should they need them for their own analysis), maternal and paternal GTF and BED files, files containing regions not lifted for maternal and paternal genomes, maternal and paternal VCF files.

1.2.8 process STAR_reference_maternal_genomes

```
process STAR_reference_maternal_genomes {


  input:

    path ("STAR_2Gen_Ref/${id}_maternal.fa") from mat_fa1

    path ("STAR_2Gen_Ref/mat_annotation.gtf") from
mat_annotation_ch1

    val x from read_len_ch3

    val id from params.id

    val cpus from params.cpus


  output:

    path Maternal_STAR into Maternal_STAR_ch


  script:

  """

  mkdir Maternal_STAR

  STAR --runMode genomeGenerate --genomeDir Maternal_STAR --
genomeFastaFiles STAR_2Gen_Ref/${id}_maternal.fa --sjdbGTFfile
STAR_2Gen_Ref/mat_annotation.gtf --sjdbOverhang ${x} --
runThreadN ${cpus} --outTmpDir mat

  """


}
```

Input: Maternal genome and maternal GTF file from process create_parental_genomes, read length information from process read_length, sample ID and number of cpus.

Process: This step generates maternal genome index with STAR --runMode genomeGenerate. This step feeds into map_maternal_gen_filter, where the RNA-seq reads are mapped to the maternal genomes.

Output: Maternal genome indices in Maternal_STAR directory.

1.2.9 process STAR_reference_paternal_genomes

```
process STAR_reference_paternal_genomes {


  input:

    path ("STAR_2Gen_Ref/${id}_paternal.fa") from pat_fa1

    path ("STAR_2Gen_Ref/pat_annotation.gtf") from
pat_annotation_ch1

    val x from read_len_ch4

    val id from params.id

    val cpus from params.cpus


  output:

    path Paternal_STAR into Paternal_STAR_ch


  script:

  """

  mkdir Paternal_STAR

  STAR --runMode genomeGenerate --genomeDir Paternal_STAR --
genomeFastaFiles STAR_2Gen_Ref/${id}_paternal.fa --sjdbGTFfile
STAR_2Gen_Ref/pat_annotation.gtf --sjdbOverhang ${x} --
runThreadN ${cpus} --outTmpDir pat

  """


}
```

This process is identical to process STAR_reference_maternal_genomes above but it is performed on the paternal genome.

1.2.10 process map_paternal_gen_filter

```
process map_paternal_gen_filter {

  tag "$id"


  input:

    path Paternal_STAR from Paternal_STAR_ch

    set val(id), file(reads) from reads_ch2

    path ("STAR_2Gen_Ref/pat_annotation.gtf") from
pat_annotation_ch2

    path ("STAR_2Gen_Ref/${id}_paternal.fa") from pat_fa2

    val x from read_len_ch5

    val id from params.id

    val cpus from params.cpus


  output:

    path
("STAR_Paternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") into (paternal_mapgen_ch1,
paternal_mapgen_ch2)

    path ("STAR_Paternal/${id}.RSEM.TEST.genome.PP.SM.bam")
into pat_rsem_ch


  script:

  """

  STAR --genomeDir Paternal_STAR \

       --runThreadN ${cpus} \

       --quantMode TranscriptomeSAM \

       --readFilesIn $reads \

       --readFilesCommand zcat \
```

```
        --outSAMstrandField intronMotif \

        --outFilterMultimapNmax 30 \

        --alignIntronMax 1000000 \

        --alignMatesGapMax 1000000 \

        --outMultimapperOrder Random \

        --outSAMunmapped Within \

        --outSAMattrIHstart 0 \

        --outFilterIntronMotifs RemoveNoncanonicalUnannotated \

        --sjdbOverhang ${x} \

        --outFilterMismatchNmax ${(x-(x%13))/13} \

        --outSAMattributes NH nM NM MD HI \

        --outSAMattrRGline  ID:${id}.SOFT.NOTRIM PU:Illumina
PL:Illumina LB:${id}.SOFT.NOTRIM SM:${id}.SOFT.NOTRIM
CN:Seq_centre \

        --outSAMtype BAM SortedByCoordinate \

        --twopassMode Basic \

        --outFileNamePrefix ${id}.SOFT.NOTRIM.STAR.pass2. \

        --outSAMprimaryFlag AllBestScore



  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.bam

  #KEEP ONLY PROPERLY PAIRED READS

  samtools view -@ ${cpus} -f 0x0002 -b -o
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.bam

  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam

  #KEEP UNIQUELY MAPPED READS

  samtools view -h
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam
| grep -P "NH:i:1\t|^@" | samtools view -bS - >
```

```
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am

  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am

  mkdir STAR_Paternal

  mv
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am
STAR_Paternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam

  ##Create RSEM Files:

  mkdir RSEM_MAT_GEN

  /RSEM/rsem-prepare-reference -p ${cpus} --gtf
STAR_2Gen_Ref/pat_annotation.gtf
STAR_2Gen_Ref/${id}_paternal.fa RSEM_MAT_GEN/RSEM_MAT_GEN

  /RSEM/rsem-calculate-expression --bam --output-genome-bam --
sampling-for-bam -p ${cpus} --paired-end
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.toTranscriptome.out.bam
RSEM_MAT_GEN/RSEM_MAT_GEN ${id}.RSEM.TEST

  samtools view -@ ${cpus} -f 0x0002 -b -o
${id}.RSEM.TEST.genome.PP.bam ${id}.RSEM.TEST.genome.bam

  samtools sort -@ ${cpus} -o ${id}.RSEM.TEST.genome.PP.s.bam
${id}.RSEM.TEST.genome.PP.bam

  mv ${id}.RSEM.TEST.genome.PP.s.bam
${id}.RSEM.TEST.genome.PP.bam

  samtools index ${id}.RSEM.TEST.genome.PP.bam

  samtools view -h ${id}.RSEM.TEST.genome.PP.bam | grep -P
"ZW:f:1|^@" | samtools view -bS - >
${id}.RSEM.TEST.genome.PP.SM.bam

  samtools index ${id}.RSEM.TEST.genome.PP.SM.bam

  mv ${id}.RSEM.TEST.genome.PP.SM.bam
STAR_Paternal/${id}.RSEM.TEST.genome.PP.SM.bam

  """



}
```

Input: Paternal genome indices from process STAR_reference_paternal_genomes, RNA-seq reads, paternal genome and GTF file from process create_parental_genomes, read length information from process read_length, sample ID and number of cpus.

Process: In this step the RNA-seq reads are aligned to the paternal genome with STAR. The BAM file generated from this is indexed and filtered with SAMtools to keep only properly paired and uniquely mapped reads.

RSEM is used to index the paternal genome. Following this, RSEM is used with the STAR transcriptome.bam to map the same RNA-seq reads with RSEM instead. In this case, reads that would map to multiple locations are not discarded but are allocated one location. All uniquely mapped reads are used to calculate the expression of each of these loci, and then the multi-mapping reads are allocated a location based on these weights. The allocation is based on probabilities based on ratios of uniquely mapped reads from genomic loci where the multi-mapping read aligns to. The file is then filtered with SAMtools to keep only properly paired reads.

Output: BAM file of mapped reads to paternal genome and BAM file generated with RSEM.

1.2.11 process map_maternal_gen_filter

```
process map_maternal_gen_filter {

  tag "$id"

  input:

    path Maternal_STAR from Maternal_STAR_ch

    set val(id), file(reads) from reads_ch3

    path ("STAR_2Gen_Ref/mat_annotation.gtf") from
mat_annotation_ch2

    path ("STAR_2Gen_Ref/${id}_maternal.fa") from mat_fa2

    val x from read_len_ch6

    val id from params.id

    val cpus from params.cpus


  output:

    path
("STAR_Maternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") into (maternal_mapgen_ch1,
maternal_mapgen_ch2)

    path ("STAR_Maternal/${id}.RSEM.TEST.genome.PP.SM.bam")
into mat_rsem_ch


  script:

  """

  STAR --genomeDir Maternal_STAR \

       --runThreadN ${cpus} \

       --quantMode TranscriptomeSAM \

       --readFilesIn $reads \

       --readFilesCommand zcat \

       --outSAMstrandField intronMotif \
```

```
        --outFilterMultimapNmax 30 \

        --alignIntronMax 1000000 \

        --alignMatesGapMax 1000000 \

        --outMultimapperOrder Random \

        --outSAMunmapped Within \

        --outSAMattrIHstart 0 \

        --outFilterIntronMotifs RemoveNoncanonicalUnannotated \

        --sjdbOverhang ${x} \

        --outFilterMismatchNmax ${(x-(x%13))/13} \

        --outSAMattributes NH nM NM MD HI \

        --outSAMattrRGline  ID:${id}.SOFT.NOTRIM PU:Illumina
PL:Illumina LB:${id}.SOFT.NOTRIM SM:${id}.SOFT.NOTRIM
CN:Seq_centre \

        --outSAMtype BAM SortedByCoordinate \

        --twopassMode Basic \

        --outFileNamePrefix ${id}.SOFT.NOTRIM.STAR.pass2. \

        --outSAMprimaryFlag AllBestScore

  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.bam

  #KEEP ONLY PROPERLY PAIRED READS

  samtools view -@ ${cpus} -f 0x0002 -b -o
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.bam

  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam

  #KEEP UNIQUELY MAPPED READS

  samtools view -h
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.bam
| grep -P "NH:i:1\t|^@" | samtools view -bS - >
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am
```

```
  samtools index
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am

  mkdir STAR_Maternal

  mv
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoord.out.PP.UM.b
am
STAR_Maternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam

  ##Create RSEM Files:

  mkdir RSEM_MAT_GEN

  /RSEM/rsem-prepare-reference -p ${cpus} --gtf
STAR_2Gen_Ref/mat_annotation.gtf
STAR_2Gen_Ref/${id}_maternal.fa RSEM_MAT_GEN/RSEM_MAT_GEN

  /RSEM/rsem-calculate-expression --bam --output-genome-bam --
sampling-for-bam -p ${cpus} --paired-end
${id}.SOFT.NOTRIM.STAR.pass2.Aligned.toTranscriptome.out.bam
RSEM_MAT_GEN/RSEM_MAT_GEN ${id}.RSEM.TEST



  samtools view -@ ${cpus} -f 0x0002 -b -o
${id}.RSEM.TEST.genome.PP.bam ${id}.RSEM.TEST.genome.bam

  samtools sort -@ ${cpus} -o ${id}.RSEM.TEST.genome.PP.s.bam
${id}.RSEM.TEST.genome.PP.bam

  mv ${id}.RSEM.TEST.genome.PP.s.bam
${id}.RSEM.TEST.genome.PP.bam

  samtools view -h ${id}.RSEM.TEST.genome.PP.bam | grep -P
"ZW:f:1|^@" | samtools view -bS - >
${id}.RSEM.TEST.genome.PP.SM.bam

  samtools index ${id}.RSEM.TEST.genome.PP.SM.bam

  mv ${id}.RSEM.TEST.genome.PP.SM.bam
STAR_Maternal/${id}.RSEM.TEST.genome.PP.SM.bam

  """

}
```

This process is identical to process map_paternal_gen_filter but performed on the maternal genome.

1.2.12 process extra_reads_rsem

```
process extra_reads_rsem {


  input:

    path
("STAR_Maternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") from maternal_mapgen_ch1

    path ("STAR_Maternal/${id}.RSEM.TEST.genome.PP.SM.bam")
from mat_rsem_ch

    path
("STAR_Paternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") from paternal_mapgen_ch1

    path ("STAR_Paternal/${id}.RSEM.TEST.genome.PP.SM.bam")
from pat_rsem_ch

    val id from params.id



  output:

    path ("Maternal.RSEM.bam") into mat_rsembam

    path ("Paternal.RSEM.bam") into pat_rsembam



  script:

  """

  samtools view
STAR_Maternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam | cut -f1 | sort | uniq >>
maternal_tags_UM.txt

  samtools view STAR_Maternal/${id}.RSEM.TEST.genome.PP.SM.bam
| cut -f1 | sort | uniq > maternal_tags_UM.RSEM.txt

  perl ${baseDir}/bin/filter_rsem.pl maternal

  samtools view
STAR_Paternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam | cut -f1 | sort | uniq >>
paternal_tags_UM.txt
```

```
  samtools view STAR_Paternal/${id}.RSEM.TEST.genome.PP.SM.bam
| cut -f1 | sort | uniq > paternal_tags_UM.RSEM.txt

  perl ${baseDir}/bin/filter_rsem.pl paternal

  samtools view -H
STAR_Maternal/${id}.RSEM.TEST.genome.PP.SM.bam >
Maternal.RSEM.sam

  samtools view STAR_Maternal/${id}.RSEM.TEST.genome.PP.SM.bam
| grep -Fwf extra.rsem.maternal.txt | sed -e
's/339\tchr/83\tchr/' | sed -e 's/355\tchr/99\tchr/' | sed -e
's/403\tchr/147\tchr/' | sed -e 's/419\tchr/163\tchr/' >>
Maternal.RSEM.sam

  samtools view -bS Maternal.RSEM.sam -o Maternal.RSEM.bam

  samtools view -H
STAR_Paternal/${id}.RSEM.TEST.genome.PP.SM.bam >
Paternal.RSEM.sam

  samtools view STAR_Paternal/${id}.RSEM.TEST.genome.PP.SM.bam
| grep -Fwf extra.rsem.paternal.txt | sed -e
's/339\tchr/83\tchr/' | sed -e 's/355\tchr/99\tchr/' | sed -e
's/403\tchr/147\tchr/' | sed -e 's/419\tchr/163\tchr/' >>
Paternal.RSEM.sam

  samtools view -bS Paternal.RSEM.sam -o Paternal.RSEM.bam

  """

}
```

Input: Filtered BAM file from process map_maternal_gen_filter and map_paternal_gen_filter, RSEM sampled BAM files from map_maternal_gen_filter and map_paternal_gen_filter, and sample ID.

Process: Custom script gets the extra multi-mapping reads (which now only have one location allocated by weight in the previous step) that are aligned in RSEM, but not in STAR and creates a file extra.rsem.maternal/paternal.txt. Then a new RSEM BAM file is created containing only these extra reads.

Output: BAM file for maternal and paternal extra reads that originally aligned to multiple locations, now with a single location.

1.2.13 process add_rsemreads_bam

```
process add_rsemreads_bam {

  publishDir "$params.outdir/", mode: 'copy'


  input:

    path ("Maternal.RSEM.bam") from mat_rsembam

    path ("Paternal.RSEM.bam") from pat_rsembam

    path
("STAR_Paternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") from paternal_mapgen_ch2

    path
("STAR_Maternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") from maternal_mapgen_ch2

    path ("STAR_2Gen_Ref/map_over.txt") from adjusted_ref_ch

    path ("${id}_output_phaser.vcf") from phaser_out_ch2

    val id from params.id

    val cpus from params.cpus

    path
("STAR_original/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByC
oord.out.PP.UM.bam") from pp_um_ch

    path ("STAR_2Gen_Ref/${id}_output_phaser.mother.vcf.gz")
from mothervcf_ch

    path ("STAR_2Gen_Ref/${id}_output_phaser.father.vcf.gz")
from fathervcf_ch

    path ("STAR_2Gen_Ref/mat.bed") from mat_bed_ch

    path ("STAR_2Gen_Ref/pat.bed") from pat_bed_ch

    path gencode_bed from params.gencode_bed



  output:

    path ("results*.txt")
```

```
    path ("${id}_gene_level_ae.txt")


  script:

  """

  samtools merge Maternal.RSEM.STAR.bam
STAR_Maternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam Maternal.RSEM.bam

  samtools merge Paternal.RSEM.STAR.bam
STAR_Paternal/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam Paternal.RSEM.bam

  samtools view Maternal.RSEM.STAR.bam | cut -f1 | sort | uniq
>> maternal_tags.txt

  samtools view Paternal.RSEM.STAR.bam | cut -f1 | sort | uniq
>> paternal_tags.txt

  cat maternal_tags.txt paternal_tags.txt | sort | uniq -u >>
unique_tags.txt

  cat maternal_tags.txt paternal_tags.txt | sort | uniq -d >>
duplicate_tags.txt

  samtools view Maternal.RSEM.STAR.bam | grep -Fwf
duplicate_tags.txt >> tempout_mat.sam

  samtools view Paternal.RSEM.STAR.bam | grep -Fwf
duplicate_tags.txt >> tempout_pat.sam

  sort -k 1,1 tempout_mat.sam > tempout_mat.sort.sam

  sort -k 1,1 tempout_pat.sam > tempout_pat.sort.sam

  perl ${baseDir}/bin/filter_2genomes.pl tempout_mat.sort.sam
tempout_pat.sort.sam

  cat maternal_wins.txt unique_tags.txt >
maternal_wins_final.txt

  cat paternal_wins.txt unique_tags.txt >
paternal_wins_final.txt

  samtools view -H Maternal.RSEM.STAR.bam > final_mat.sam

  samtools view -H Paternal.RSEM.STAR.bam > final_pat.sam

  samtools view Maternal.RSEM.STAR.bam | grep -Fwf
maternal_wins_final.txt >> final_mat.sam
```

```
  samtools view Paternal.RSEM.STAR.bam | grep -Fwf
paternal_wins_final.txt >> final_pat.sam

  samtools view -bS final_mat.sam -o final_mat.bam

  samtools sort -@ ${cpus} -o final_mat.sorted.bam
final_mat.bam

  samtools index final_mat.sorted.bam

  samtools view -bS final_pat.sam -o final_pat.bam

  samtools sort -@ ${cpus} -o final_pat.sorted.bam
final_pat.bam

  samtools index final_pat.sorted.bam

  perl ${baseDir}/bin/compare_basic_map.pl
${id}_output_phaser.vcf
STAR_original/${id}.SOFT.NOTRIM.STAR.pass2.Aligned.sortedByCoo
rd.out.PP.UM.bam ${id}
results_1genome_${id}.SOFT.NOTRIM_baq.txt
results_1genome_${id}.SOFT.NOTRIM.txt

  perl ${baseDir}/bin/compare_2genomes.pl
STAR_2Gen_Ref/map_over.txt ${id}_output_phaser.vcf
final_mat.sorted.bam final_pat.sorted.bam ${id}
results_2genomes_${id}.RSEM.STAR.SOFT.NOTRIM_baq.txt
results_2genomes_${id}.RSEM.STAR.SOFT.NOTRIM.txt

  tabix STAR_2Gen_Ref/${id}_output_phaser.mother.vcf.gz

  tabix STAR_2Gen_Ref/${id}_output_phaser.father.vcf.gz

  python2 /phaser/phaser/phaser.py --vcf
STAR_2Gen_Ref/${id}_output_phaser.mother.vcf.gz --bam
final_mat.sorted.bam --paired_end 1 --mapq 0 --baseq 10 --
isize 0 --include_indels 1 --sample ${id} --id_separator + --
pass_only 0 --gw_phase_vcf 1 --threads ${cpus} --o
${id}_mat_output_phaser

  python2 /phaser/phaser/phaser.py --vcf
STAR_2Gen_Ref/${id}_output_phaser.father.vcf.gz --bam
final_pat.sorted.bam --paired_end 1 --mapq 0 --baseq 10 --
isize 0 --include_indels 1 --sample ${id} --id_separator + --
pass_only 0 --gw_phase_vcf 1 --threads ${cpus} --o
${id}_pat_output_phaser

  python2 /phaser/phaser_gene_ae/phaser_gene_ae.py --
haplotypic_counts
${id}_mat_output_phaser.haplotypic_counts.txt --features
```

```
STAR_2Gen_Ref/mat.bed --id_separator +  --o
${id}_maternal_phaser_gene_ae.txt

  python2 /phaser/phaser_gene_ae/phaser_gene_ae.py --
haplotypic_counts
${id}_pat_output_phaser.haplotypic_counts.txt --features
STAR_2Gen_Ref/pat.bed --id_separator +  --o
${id}_paternal_phaser_gene_ae.txt

  perl ${baseDir}/bin/merge_gene_level.pl ${gencode_bed}
${id}_maternal_phaser_gene_ae.txt
${id}_paternal_phaser_gene_ae.txt ${id}

  """



}
```

Input: Maternal and paternal extra reads from RSEM generated in process extra_reads_rsem; BAM file of reads mapped to maternal and paternal genomes from map_maternal_gen_filter and map_paternal_gen_filter; map_over, and and maternal and paternal bed files with adjusted coordinates, and maternal and paternal phased VCF file from process create_parental_genomes, phased VCF file from process phaser_step, sample ID, number of cpus, properly paired and uniquely mapped reads to the reference genome from process clean_up_reads; and GENCODE BED file.

Process: For each parental genome, the STAR and RSEM BAM files are merged. Then PAC finds reads only aligned in one parent and not the other. When the reads are aligned in both maternal and paternal genomes, a custom script (filter_2genomes.pl) selects the best alignment for each read from the two alignments (scoring reads by the number of matching nucleotides minus two times the number of indel positions, drawing at random when the two alignments have equal scores).

Then two custom scripts (compare_basic_map.pl and compare_2genomes.pl) are used to count the number of alleles at each heterozygous site. Initially, this is done with standard alignment. Then the same is performed for two genomes parental alignment using the liftOver variant files.

Then phASER is used to generate the gene-level calculations using the VCF files and GTF files from each parent (generated in process create_parental_genomes). PAC then produces allele counts at haplotypic level using phASER Gene AE.

Finally, the last custom script (merge_gene_level.pl) merges the gene level counts across the two parents.

Output: The results files: site and haplotype level allelic counts and single genome alignment for comparison.

1.3 Output

PAC generates 5 output files:

- haplotype level ASE calls:
    1. 'id'_gene_level_ae.txt

Haplotype level ASE results columns	Description
contig	chromosome
start	gene start position
stop	gene end position
name	gene name
aCount	haplotype a coverage
bCount	haplotype b coverage
totalCount	total coverage

Figure 1. Columns and their descriptions for haplotype level ASE results from PAC output. The 'id'_gene_level_ae.txt contains this file format.

- single nucleotide level ASE calls from PAC:
    2. results_2genomes_'id'.RSEM.STAR.SOFT.NOTRIM_baq.txt
    3. results_2genomes_'id'.RSEM.STAR.SOFT.NOTRIM.txt
- single nucleotide level ASE calls based on standard single genome mapping for comparison:
    4. results_1genome_'id'.SOFT.NOTRIM_baq.txt
    5. results_1genome_'id'.SOFT.NOTRIM.txt

Single nucleotide level ASE results columns	Description
Chr	chromosome
Pos	position along chromosome
RefAl	reference allele
AltAl	alternative allele
MapRef	reference allele coverage
MapAlt	alternative allele coverage
MapRatio	reference allele ratio
Mapcov	total coverage at the site

Figure 2. Columns and their descriptions for single nucleotide level ASE results from PAC output.
The results_2genomes_ID.RSEM.STAR.SOFT.NOTRIM_baq.txt, results_2genomes_ID.RSEM.STAR.SOFT.NOTRIM.txt, results_1genome_ID.SOFT.NOTRIM_baq.txt and results_1genome_ID.SOFT.NOTRIM.txt contain this file format.