NGS Relate and PC AngSD Pipeline Documents for NYU

Production Run 5-23

Michael Dhar

Contents:

- Nextflow script
- Wrapper Slurm script
- Nextflow configuration script
- R scripts for NGSRelate and PC AngSD output
- R plots for NGSRelate and PC AngSD output

//NEXTFLOW SCRIPT

```
/* Prod run script, 5-28-23
 * M. Dhar < mid224@nyu.edu >
 *  Usage: nextflow run /path/to/main_index.nf
 */

// Setting some defaults here,
// can be overridden in config or via command line
//params.out = "${params.outdir}/out"
params.out = "${params.outdir}"
script_dir = params.scriptdir


// Define modules here
R = 'r/gcc/4.2.0'
ANGSD = 'angsd/intel/0.933'
NGSRELATE = 'ngsrelate/intel/20210103'
BEDTOOLS = 'bedtools/intel/2.29.2'

// Print paths
println "bam list: $params.bam_list"
println "ref: $params.ref"
println "output: $params.out"

// Set up the reference file and other inputs
ref = file(params.ref)
//Set up clones list
clones_list = params.clones_list
//Set up minInd level
minInd = params.minInd
//Set up out_angsdngsrel
out_angsdngsrel = params.out_angsdngsrel
//Set up bed_file
bed_file = params.bed_file
//Set up out_ngsrel
out_ngsrel = params.out_ngsrel
//Set up out_angsd
out_angsd = params.out_angsd
//Set up out_pca
out_pca = params.out_pca

/* Read in files
*/
//Channel
```

```
bam_list_ch = Channel
   .fromPath( params.bam_list )
   .ifEmpty { error "Cannot find BAM list"  }
   .map { file -> tuple(file.baseName, file) }

process input_parse {
    publishDir "${params.out}/$bam_list_ID/input_parse", mode:'copy'

   input:
   set bam_list_ID, file(bam_list) from bam_list_ch

   output:
    set bam_list_ID, file("bam_path_col.txt"), \
      file("bam_ID_col.txt"), \
      file ("concat_list.txt"), \
          file ("sample_no.txt")
    set bam_list_ID, file("bam_path_col.txt"), \
          file("bam_ID_col.txt"), \
          file("concat_list.txt"), \
          file(bam_list), \
          env(n) \
                  into(ngsrel_input_ch, angsd_input_ch)
   script:

   """

   #concat step
   tail -n +2 $clones_list >> concat_list.txt
   tail -n +2 $bam_list >> concat_list.txt

   #processing step
   cut -f 6 concat_list.txt > bam_path_col.txt
   #Get column 2 for ID: ID_label
   cut -f 2 concat_list.txt > bam_ID_col.txt
   n=\$(wc -l concat_list.txt | awk '{print \$1}')  # save nb of accessions in 'n'

   #Print number of samples
   m=\$(tail -n +2 $bam_list | wc -l | awk '{print \$1}')  # save nb of test samples in 'm'
   echo "There are \$m samples in the test file." >> sample_no.txt
   echo "There are \$n samples in the concatenated clone and test sample list." >>
sample_no.txt

   """
```

```
}

process ngsrel {
   echo true
    publishDir "${params.out}/$bam_list_ID/ngsrel", mode:'copy'

   input:
    set bam_list_ID, file(bam_path_col), file(bam_ID_col), file(concat_list), file(bam_list), val(n)
from ngsrel_input_ch

   output:
         set bam_list_ID, file("${out_angsdngsrel}.arg"), \
                  file("${out_angsdngsrel}.glf.gz"), \
                  file("${out_angsdngsrel}.glf.pos.gz"), \
                  file("${out_angsdngsrel}.mafs.gz"), \
                  file("${out_angsdngsrel}.freq"), \
                  file("NGSrelate10.res")
         set bam_list_ID, file("NGSrelate10.res"), file(bam_list) \
           into (ngsrel_muriel_rchan)

   script:

   """
   module purge
   module load $ANGSD
   module load $NGSRELATE

   ### First we generate a file with allele frequencies (angsdput.mafs.gz) and a file with
genotype likelihoods (angsdput.glf.gz)
   angsd -b $bam_path_col -ref $ref -out $out_angsdngsrel \
         -GL 2 -doGlf 3 -doMajorMinor 1 -doMaf 1 -doCounts 1 \
         -rf $bed_file \
         -uniqueOnly 1 -remove_bads 1 -only_proper_pairs 1 -C 50 -baq 1 -skipTriallelic 1 \
     -minMapQ 20 -minQ 20 -minInd $minInd -snp_pval 1e-6 \
         -setMinDepthInd 5

   # Extract the frequency column from the allele frequency file and remove the header (to
make it in the format NgsRelate needs)
   zcat ${out_angsdngsrel}.mafs.gz | cut -f6 | sed 1d > ${out_angsdngsrel}.freq

   # run NGSrelate, last version (download on 15 Jan 2021)
   # Run NGSrelate
   ngsRelate -g ${out_angsdngsrel}.glf.gz -n $n -f ${out_angsdngsrel}.freq -z $bam_ID_col -O
$out_ngsrel
```

```
    """

}

process pcangsd {
    echo true
     publishDir "${params.out}/$bam_list_ID/pcangsd", mode:'copy'

    input:
    set bam_list_ID, file(bam_path_col), file(bam_ID_col), file(concat_list), file (bam_list), val(n)
from angsd_input_ch

    output:
        set bam_list_ID, file("${out_angsd}.arg"), \
            file("${out_angsd}.beagle.gz"), \
            file("${out_angsd}.mafs.gz"), \
                    file("${out_pca}.args"), \
            file("${out_pca}.cov"), \
            file("nb_sites_beagle")
            set bam_list_ID, file("${out_pca}.cov"), file(concat_list) \
              into (pcangsd_muriel_rchan)

    script:

    """
    module purge
    module load $ANGSD
    module load $BEDTOOLS

    angsd -b $bam_path_col -ref $ref -out $out_angsd \
        -rf $bed_file \
        -GL 2 -doGlf 2 -doMajorMinor 1 -doMaf 2 -doCounts 1 \
        -uniqueOnly 1 -remove_bads 1 -only_proper_pairs 1 -C 50 -baq 1 -skipTriallelic 1 \
        -minMapQ 20 -minQ 20 -minInd $minInd -snp_pval 1e-6 \
        -setMinDepthInd 5

    # Note that doCounts is only necessary for being able to filter on depth

    zcat ${out_angsd}.beagle.gz | wc -l > nb_sites_beagle #

    # Run PCAngsd
    /scratch/work/public/singularity/run-pacangsd.bash python /ext3/pcangsd/pcangsd.py -
beagle ${out_angsd}.beagle.gz -out $out_pca
```

```
    """

}

process r_script_muriel_ngsrel {
    echo true
     publishDir "${params.out}/$bam_list_ID/r_script_muriel_ngsrel", mode:'copy'

    input:
    set bam_list_ID, file (ngsrel_output), file(bam_list) from ngsrel_muriel_rchan

    output:
    set file("summary_table_KING.txt"), file("heatmap.pdf")

    script:

    """

    module load $R

    #Rscript --vanilla ${script_dir}/Muriel_NGSRelate_RScript_Integ.R $bam_list $clones_list
$ngsrel_output
    Rscript --vanilla ${script_dir}/Muriel_NGSRelate_RScript_Integ_ColIDFix.R $bam_list
$clones_list $ngsrel_output

    """

}

process r_script_muriel_pcangsd {
    echo true
     publishDir "${params.out}/$bam_list_ID/r_script_muriel_pcangsd", mode:'copy'

    input:
    set bam_list_ID, file (pcangsd_output), file(concat_list) from pcangsd_muriel_rchan

    output:
    set file("pca1.pdf"), file("pca2.pdf")

    script:

    """
```

```
    module load $R

    Rscript --vanilla ${script_dir}/Muriel_PCAngSD_RScript_Integ_LegendColShp_3-26-23.R
$pcangsd_output $concat_list

    """

}
```

#SLURM SBATCH WRAPPER SCRIPT FOR NGSREL AND PC ANGSD NEXTFLOW PIPELINE

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=168:00:00
#SBATCH --mem=2GB
#SBATCH --job-name=prod_run_first3lists_5-28-23
#SBATCH --mail-type=END
#SBATCH --mail-user=mid224@nyu.edu
#SBATCH --output=slurm_%j.out

module purge
module load nextflow/21.10.5

#Run local script with resume and report
nextflow prod_5-28-23.nf -with-report report_slurm_%j.html -resume
```

//CONFIG FILE FOR NGSRELATE AND PC ANGSD NEXTFLOW PIPELINE

```
// Required Parameters
//script dir is current, working dir, where all scripts being used, including R scripts, are located
params.scriptdir = "/scratch/work/alula/analysis_divintra"

//out directory is the out folder for the current run
params.outdir = "$params.scriptdir/pipeline_out/first_3_out"

// Set the Nextflow Working Directory
workDir = params.scriptdir + '/nextflow_work_dir_first_3'

//clones list
//PRODUCTION RUN 5-28-23
params.clones_list = "/scratch/work/alula/analysis_divintra/bamlists/clones_2023.05.10.txt"

//bam list
params.bam_list = "/scratch/work/alula/analysis_divintra/bamlists/First_3_5-28-23/*.txt"

//reference genome
params.ref = "$SCRATCH/data/genome_assemblies/Pdac_Barhee_chr_unan_cp_180126.fasta"

//minInd level
params.minInd = 2
//out_angsd base
params.out_angsd="barni1_angsdpca"
//Set up out_angsdngsrel
params.out_angsdngsrel = "out_angsd10"
//bed_file
params.bed_file = "$SCRATCH/alUla_bam_pipeline/Data_Lists/intervals.bed"
//Set up out_ngsrel
params.out_ngsrel = "NGSrelate10.res"
//out_pca base
params.out_pca = "barni1_PCAngsd"


// slurm and resource allocation
process {
   executor = 'slurm'
   cpus = 1
   memory = { 1.GB * task.attempt }
   time = { 10.min * task.attempt }
   withName: ngsrel { memory = '25.GB' }
   withName: ngsrel { time = '168.h' }
```

```
    withName: pcangsd { memory = '20.GB' }
    withName: pcangsd { time = '168.h' }
    withName: r_script_muriel_ngsrel { memory = '5.GB' }
    withName: r_script_muriel_ngsrel { time = '2.h' }
    withName: r_script_muriel_pcangsd { memory = '5.GB' }
    withName: r_script_muriel_pcangsd { time = '2.h' }
}
```

//R SCRIPT NGSRELATE

```
#!/usr/bin/env Rscript
args = commandArgs(trailingOnly=TRUE)

#print test command
print("R Muriel NGS script running.")

# Plot the results of NGSrelate for the al-ula project
# Load libraries
library(dplyr); library(igraph); library(network); library(extrafont); library(tidyverse);
library(gplots); library(viridis); library(circlize); library(GGally); library(igraph); library(gridExtra);
library(MetBrewer)#; library(Hsm)
# Read the sample info file.
test.samples <- read.table(args[1], h=T)
clones <- read.table(args[2], h=T)
mysamples <- rbind(clones, test.samples)
# Read the output of NGSrelate (.res file).
ngsrelate <- read.table(args[3], h=T)
# Concatenate output files and sample info file and keep only important columns
x <- merge(ngsrelate, mysamples, by.x="ida", by.y="ID_label")
y <- merge(x, mysamples, by.x="idb", by.y="ID_label")
#fixed to include cols a (idb) and 2 (ida) from BAM lists (via y)
res <- y[,c(2,1,36,37,38,42,43,44,5,29,33)]
# Calculate some stats for four groups:
#    - the asela clones
#    - the barni clones
#    - the dakar clones
#    - the test variety
# Make those four groups
clone_barni <- res %>% filter(Clone.x=="clone_barni" & Clone.y=="clone_barni")
clone_asela <- res %>% filter(Clone.x=="clone_asela" & Clone.y=="clone_asela")
clone_dakar <- res %>% filter(Clone.x=="clone_dakar" & Clone.y=="clone_dakar")
test <- res %>% filter(is.na(Clone.x) & is.na(Clone.y))
# Make the summary and save the table
summary.clone_barni <- summarize(clone_barni, mean=mean(KING), sd=sd(KING),
min=min(KING), max=max(KING))
summary.clone_asela <- summarize(clone_asela, mean=mean(KING), sd=sd(KING),
min=min(KING), max=max(KING))
summary.clone_dakar <- summarize(clone_dakar, mean=mean(KING), sd=sd(KING),
min=min(KING), max=max(KING))
summary.test <- summarize(test, mean=mean(KING), sd=sd(KING), min=min(KING),
max=max(KING))
```

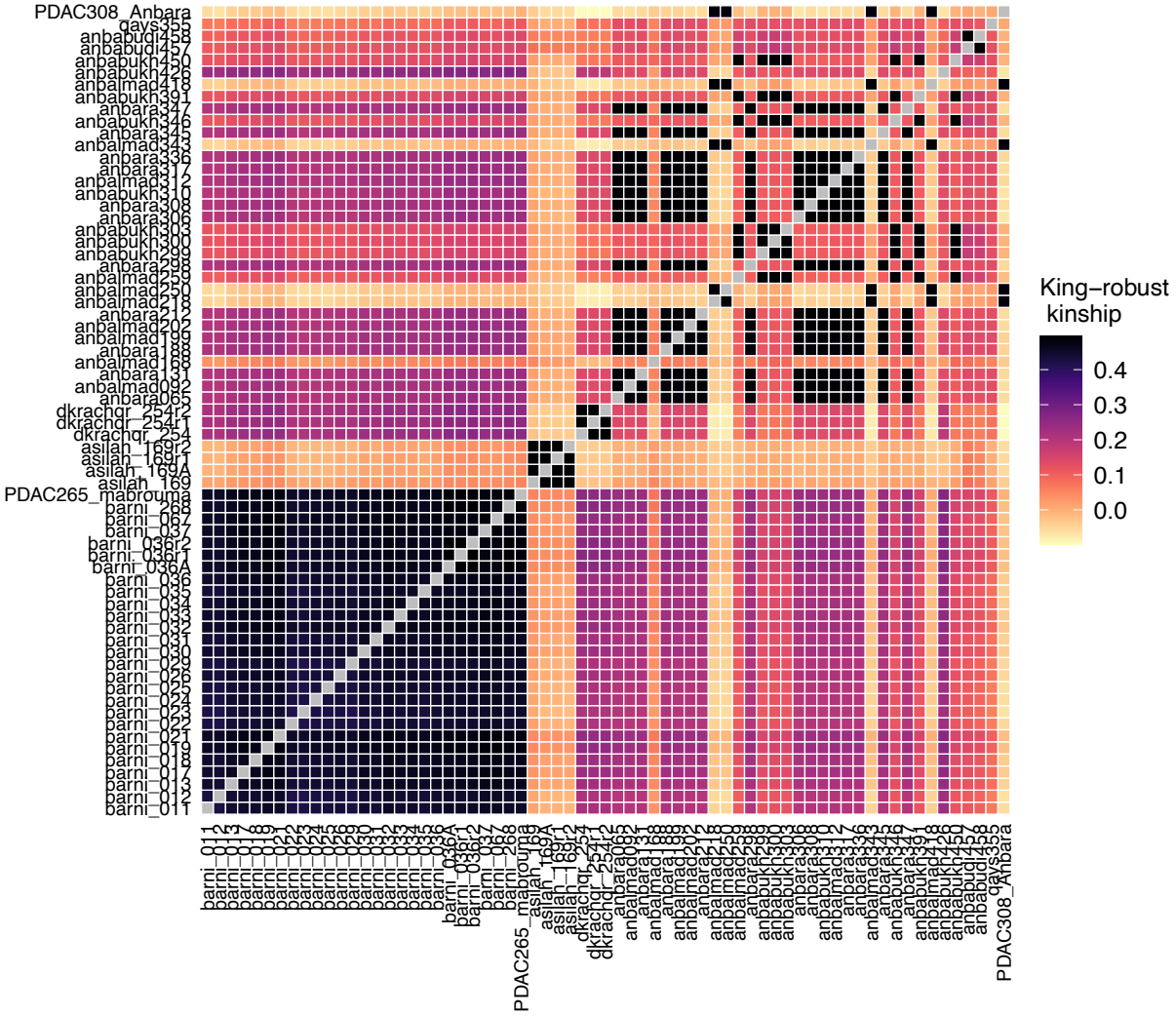
```
summary.table <- rbind(summary.clone_barni, summary.clone_asela, summary.clone_dakar,
summary.test)
row.names(summary.table) <- c("clone_barni", "clone_asela", "clone_dakar", "test")
write.table(summary.table, "summary_table_KING.txt", row.names = T, col.names = T,
quote=F)
# Create network for the heatmap
#changed above to refer to ID_label columns 3-18-23
links <- data.frame('from'=res$ida, 'to'=res$idb, 'r'=res$KING)
#fixed to refer to proper columns
nodes <- mysamples[,c(2,4)]
net <- graph.data.frame(d = links, vertices = nodes, directed=F)
netm <- get.adjacency(net, attr="r", sparse=F, type="both")
# Plot the heatmap
diag(netm) <- NA
# Make a three-cols df out of the symm matrix
my.df <- as.data.frame(as.table(netm))
colnames(my.df) <- c("ida", "idb", "King")
ggplot(my.df, aes(ida, idb, fill=King)) +
  geom_tile(colour="white", size=0.25) +
  scale_fill_viridis(discrete=F, name="King-robust\n kinship", direction = -1, na.value="grey",
option="magma") +
  labs(x="", y="") +
  theme_bw(base_size=8) +
  theme(panel.border=element_blank(),
      #legend.position = "right",
      legend.text=element_text(colour="black", size=10),
      legend.title=element_text(size=10),
      axis.text.x=element_text(size=8, vjust=0.5, hjust = 1, angle=90, colour="black"),
      axis.text.y=element_text(size=8, colour="black"),
      axis.ticks=element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.background = element_blank(),
      aspect.ratio = 1)
dev.off()
```

//R SCRIPT FOR PC ANGSD

```
#!/usr/bin/env Rscript
args = commandArgs(trailingOnly=TRUE)

#print test command
print("R Muriel PCAngSD script running.")

# PCAngsd barni #1 - Dec 2021
library(plotly); library(esquisse); library(gridExtra); library(grid); library(ggplot2); library(lattice);
library(ggrepel); library(RColorBrewer)
# Read the cov matrix (output of PCAngsd)
cov.mat <- as.matrix(read.table(args[1]))
# Prepare sample file
mysamples <- read.table(args[2]) #concat_list does not include header
mysamples <- mysamples[,c(1:4)]
# very importantly here, we need to read a sample file but it has to be in the same order as the
bamlist used for running pcangsd
# Save the eigen values
eig <- as.data.frame(eigen(cov.mat)$values)
# Save the coord and make ready for plotting
coord <- as.data.frame(eigen(cov.mat)$vectors)
coord2 <- cbind(mysamples, coord)
#coord3 for full BAM lists
coord3 <- coord2[,c(1:10)]; colnames(coord3) <- c("ID", "label", "Cultivar", "Clone", "PC1",
"PC2", "PC3", "PC4", "PC5", "PC6")#; coord3
#coord3 for short lists
#coord3 <- coord2[,c(1:8)]; colnames(coord3) <- c("ID", "label", "Cultivar", "Clone", "PC1",
"PC2", "PC3", "PC4")#; coord3
#Sub NA clone values for test samples
coord3_fill <- coord3
coord3_fill$Clone[is.na(coord3_fill$Clone)] <- "test_sample"

##### Plot

# Have a glimpse with labels
pdf("pca1.pdf")
par(pty="s"); plot(coord3$PC1, coord3$PC2, col="white")
text(coord3$PC1, coord3$PC2, coord3$label)
dev.off()

#set group colors
group.colors <- c(clone_barni = "#CC6666", clone_dakar = "#9999CC", clone_asela = "#66CC99",
test_sample = "#000000")
```

```
#set group shapes and fills
group.shapes <- c(clone_barni = 2, clone_dakar = 3, clone_asela = 0, test_sample = 1)
group.fills <- c(clone_barni = "#CC6666", clone_dakar = "#9999CC", clone_asela = "#66CC99",
test_sample = "#00000044")

# Make a plot with dots
pdf("pca2.pdf")
#ggplot(coord3) +
  ggplot(coord3_fill) +
  aes(x = PC1, y = PC2, col = Clone, shape = Clone, fill = Clone) +
  geom_point(size=5) +
  #set plotting of shapes and colors to above group settings
  scale_shape_manual(values=group.shapes) +
  scale_color_manual(values=group.colors) +
  scale_fill_manual(values=group.fills) +
  xlab(paste("PC1 (",format(round(eig[1,1], 2), nsmall = 2),"%)")) +
  ylab(paste("PC2 (",format(round(eig[2,1], 2), nsmall = 2),"%)")) +
  theme_bw() +
  theme(aspect.ratio=1,
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x = element_text(size=10, colour="black", margin=margin(5,0,0,0)),
        axis.text.y= element_text(size=10, colour="black", margin=margin(0,5,0,0)))
dev.off()
```

//EXAMPLE R PLOT (HEATMAP) FOR NGS RELATE



//EXAMPLE R PLOT FOR PC ANGSD