SV calling from whole genome resequencing data

Today we have explored the importance of structural variants in Ecology and Evolution, and you have tested different approaches to investigate haploblocks identified through reduced representation approaches such as RADseq. However, explicitely testing for the presence of SVs associated with haploblocks is not really possible with RADseq data (but see Yann's paper on detecting CNVs from RADseq data; Dorant et al. 2020, Molecular Ecology), so we'll now use whole genome resequencing data (Illumina short reads) from a few capelin samples. From these data, we can call both SNPs (sequence variation - done Monday) and SVs (structural variation) and test whether the two provide concordant patterns of variation and differentiation, or not.

Table of contents

1. SVs calling

Calling structural variants (i.e. inversions, deletions, insertions, duplications, translocations) requires analyzing different types of information from the sequence data, such as read overlap, orientation and splitting. Therefore, we need software specifically designed to extract this kind of information. One commonly used of these programs is Delly2, which takes into account all of these 3 types of information. This below is the general approach to call SVs from Delly:

```
# germline SV calling
# SV calling is done by sample for high-coverage genomes or in small
batches for low-coverage genomes
delly call -g hg19.fa -o s1.bcf -x hg19.excl sample1.bam

# merge SV sites into a unified site list
delly merge -o sites.bcf s1.bcf s2.bcf ... sN.bcf

# genotype this merged SV site list across all samples. This can be run in
parallel for each sample.
delly call -g hg19.fa -v sites.bcf -o s1.geno.bcf -x hg19.excl s1.bam

delly call -g hg19.fa -v sites.bcf -o sN.geno.bcf -x hg19.excl sN.bam

# merge all genotyped samples to get a single VCF/BCF using bcftools merge
bcftools merge -m id -O b -o merged.bcf s1.geno.bcf s2.geno.bcf ...
sN.geno.bcf

# apply the germline SV filter which requires at least 20 unrelated
samples
delly filter -f germline -o germline.bcf merged.bcf
```

However, in our case, we have low-coverage whole genome resequencing data from only 12 individuals, so we can call SVs directly from all the samples combined and forego the filtering step. Keep in mind that this is just a toy dataset, and the quality of these SV calls may not be very high. You can run this code, but it will take > 1h. If you prefer, you can copy the VCF file directly with the second block of commands.

```
cd
cd wgr
mkdir svs_delly

ln -s /home/ubuntu/Share/WGS_bam/* .

# run delly to call SVs on all samples combined
delly call -g ~/Share/ressources/genome_mallotus_dummy.fasta -o
svs_delly/capelin_sv.bcf BELB9.bam BELD3.bam BLA13.bam BLA15.bam BLA16.bam
BLA17.bam BLA22.bam BLA24.bam BSO17.bam BSO23.bam BSO28.bam POR19.bam

# convert file from .bcf to .vcf
cd svs_delly
bcftools convert -O v -o capelin_sv.vcf capelin_sv.bcf
```

To copy the VCF file containing all the SVs into ~/wgr/svs_delly

```
# make the wgr directory and the svs_delly directory if you haven't yet
mkdir wgr
cd wgr
mkdir svs_delly
cd svs_delly
# copy the vcf of SVs that was produced for all of us with the code above
cp ~/Share/WGS_bam/svs_delly/capelin_sv.vcf ~/wgr/svs_delly/.

# copy also the popmap file
cp ~/Share/WGS_bam/svs_delly/popmap_capelin_wgr_delly.txt
~/wgr/svs_delly/.
```

2. VCF filtering and splitting

Though we don't have enough samples to run `delly filter` properly, we can do some filtering using the same approach we use for SNP filtering but in this case we'll filter just by missing data

```
vcftools --vcf capelin_sv.vcf \
    --max-missing 0.7 \
    --recode \
    --stdout > capelin_sv_filtered.vcf
```

Then, we can split the vcf file by SV type (or extract one particular SV type of interest with this script

```
grep "#" capelin_sv_filtered.vcf > capelin_sv_ins.vcf && grep "SVTYPE=INS"
capelin_sv.vcf >> capelin_sv_ins.vcf ###insertions
grep "#" capelin_sv_filtered.vcf > capelin_sv_del.vcf && grep "SVTYPE=DEL"
capelin_sv.vcf >> capelin_sv_del.vcf ###deletions
grep "#" capelin_sv_filtered.vcf > capelin_sv_inv.vcf && grep "SVTYPE=INV"
capelin_sv.vcf >> capelin_sv_inv.vcf ###inversions
grep "#" capelin_sv_filtered.vcf > capelin_sv_dup.vcf && grep "SVTYPE=DUP"
capelin_sv.vcf >> capelin_sv_dup.vcf ###duplications
grep "#" capelin_sv_filtered.vcf > capelin_sv_bnd.vcf && grep "SVTYPE=BND"
capelin_sv.vcf >> capelin_sv_bnd.vcf ###breakends
```

and count the number of SV identified, overall or for each type with

```
grep -v "#" capelin_sv_ins.vcf | wc -l
```

What is the most abundant SV type?

Next, we'll convert the genotypes in 012 format for PCA of the whole dataset and each specific

```
# recode genotypes for PCA
vcftools --vcf capelin_sv_filtered.vcf --012 --out capelin_sv

# and for each SV type
vcftools --vcf capelin_sv_ins.vcf --012 --out capelin_sv_ins
vcftools --vcf capelin_sv_del.vcf --012 --out capelin_sv_del
vcftools --vcf capelin_sv_inv.vcf --012 --out capelin_sv_inv
vcftools --vcf capelin_sv_dup.vcf --012 --out capelin_sv_dup
vcftools --vcf capelin_sv_bnd.vcf --012 --out capelin_sv_bnd
```

and download the resulting files on your computer.

3. PCA to compare patterns from different types of genetic variation

On your computer in R, perform one PCA and plot results based on each type of structural variant. The one below is the code for the PCA based on all SVs. Once you've done this, modify the script to perform the analysis on each type of SV and on the SNPs.

```
# load the population map with population assignment for each individual
library(dplyr)
popmap_delly <- read.table("popmap_capelin_wgr_delly.txt", header = FALSE)
colnames(popmap_delly) <- c("Sample", "Pop")

# load geno data for SVs
geno_capelin_sv <- read.table("capelin_sv.012")[, -1] #load genotype
matrix
geno_capelin_sv.pos <- read.table("capelin_sv.012.pos") %>% #load SNPs
```

```
info
  mutate(., locus = paste(V1, V2, sep = "_")) #create a new column for SNP
info name (CHR + position)
geno_capelin_sv.indv <- read.table("capelin_sv.012.indv") #load
individuals info

# set rownames and colnames to the geno matrix
dimnames(geno_capelin_sv) <- list(geno_capelin_sv.indv$V1,
geno_capelin_sv.pos$locus)
# check the geno matrix
geno_capelin_sv[1:12,1:10]

# impute missing data
geno_capelin_sv[geno_capelin_sv == -1] <- NA
geno_capelin_sv.imp <- apply(geno_capelin_sv, 2, function(x){
  replace(x, is.na(x), as.numeric(names(which.max(table(x))))) })

geno_capelin_sv.imp[1:12,1:9]

# run and visualized PCA
pca.geno_capelin_sv <- prcomp(geno_capelin_sv.imp)
screeplot(pca.geno_capelin_sv)

# get stats info from the pca
sum.pca <- summary(pca.geno_capelin_sv)
# print stats info
sum.pca$importance[, 1:5]

#prepare dataset to plot PCAs
pca.geno_capelin_sv.sub <- pca.geno_capelin_sv$x[, 1:4] %>% # retain the
first four PCs
  as.data.frame(.) %>% # transform to dataframe object
  tibble::rownames_to_column(., var = "Sample") %>% # set rownames to a
new column for samples ids
  dplyr::left_join(., popmap_delly, by = "Sample")

# here we use the left_join() function
# from dplyr to wrap the population vector
# of our samples
ggplot(pca.geno_capelin_sv.sub) + aes(x = PC1, y = PC2, col = Pop) +
  ggtitle("PCA with all SVs from Delly") +
  geom_point() +
  coord_fixed() +
  theme_bw()
```

To compare SVs with SNPs, we propose to do the same PCA on the SNPs called for those 12 samples sequenced with WGS. You can find the vcf file generated by bcftools and filtered, as well as the genotype matrix (.012 format) (see Day 1 step 4) in the share folder. You may copy it to your own directory with the following command.

```
cd ~/wgr/snps_bcftools
cp ~/Share/WGS_bam/snps_bcftools/capelin_wgs_filtered.vcf
~/wgr/snps_bcftools/.
cp ~/Share/WGS_bam/snps_bcftools/capelin_wgs_filtered.012
~/wgr/snps_bcftools/.
cp ~/Share/WGS_bam/snps_bcftools/capelin_wgs_filtered.012*
~/wgr/snps_bcftools/.
```

Now you can do the PCA as above. Do you see the same patterns from each type of variant? Let's discuss.