Análisis de datos: Estimadores de diversidad genética y estructura poblacional en R

Cristian B. Canales-Aguirre & C. Eliza Claure

Centro i~mar, Universidad de Los Lagos, Puerto Montt, Chile

Martes 22 de noviembre de 2022

Objetivos

El curso tiene como objetivos familiarizar a los participantes con el cálculo de parámetros de diversidad genética y estimar indicadores de estructura poblacional. Esta actividad corresponde a un taller práctico de una tarde enfocado para estudiantes e investigadores que están iniciando en análisis de datos genéticos (i.e. SNPs).

Requerimientos

- Conceptos básicos de genética de poblaciones
- Manejo básico de R o RStudio y Linux
- Laptop

Programa

1. Estimadores de diversidad genética
2. Estructura poblacional Fst
3. Estructura poblacional PCA y DAPC
4. Estructura poblacional Migración Relativa

Iniciemos!!

Información y datos

Esta actividad práctica utilizará datos publicados, en Muir et al. 2020. Se obtuvieron los datos brutos desde NCBI SRA Bioproject 645800 y se realizó un pipeline similar a los descritos en el articulo, esto con la finalidad de obtener un archivo de datos en formato vcf. La especie utilizada es *Sabellaria alveolata* (L.) (honeycomb worm) muestreada en 8 localidades en Europa. Especificamente, se utilizan marcadores SNPs obtenidos desde librerias RAD digeridos con las enzimas de restricción EcoRI y MseI. Parte del código de este práctico fue tomado y modificado de https://tomjenkins.netlify.app/2020/09/21/r-popgen-getting-started/

Cargar librerias

Para comenzar necesitamos cargar los paquetes que utlizaremos en el curso. Es recomendable tenerlos instalados previamente.

```
library(vcfR)
library(poppr)
library(adegenet)
library(hierfstat)
library(StAMPP)
library(dplyr)
library(ggplot2)
library(zvau)
library(diveRsity)
library(RColorBrewer)
```

Cargar VCF

```
vcf <- read.vcfR("populations.snps.vcf")
```

```
vcf
```

```
## ***** Object of Class vcfR *****
## 64 samples
## 286 CHROMs
## 457 variants
## Object size: 2.1 Mb
## 19.52 percent missing data
## *****        *****         *****
```

Cargar información poblacional

```
pop <- read.table("popmap.txt", sep="\t", stringsAsFactors = TRUE)
```

Transformar vcf a genind

```
genind <- vcfR2genind(vcf)
```

Definir mapa poblacional en genind

```
genind@pop <- pop$V2
```

Mostrar información básica de archivo genind

```
genind
```

```
## /// GENIND OBJECT /////////
##
##  // 64 individuals; 457 loci; 912 alleles; size: 478.7 Kb
##
##  // Basic content
##    @tab:  64 x 912 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 1-2)
##    @loc.fac: locus factor for the 912 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: adegenet::df2genind(X = t(x), sep = sep)
##
##  // Optional content
##    @pop: population of each individual (group size range: 4-12)
```

Estadísticos resumen de diversidad genética

Mostrar número de individuos por localidad

```
table(genind$pop)
```

```
##
## BIA IRE ITA MOR MSM POR SCO WAL
##  12   7   9   5  11   4   8   8
```

Mostrar número de alelos privados por localidad a lo largo de todos los loci

```
private_alleles(genind) %>% apply(MARGIN = 1, FUN = sum)
```

```
## BIA IRE ITA MOR MSM POR SCO WAL
##  32  19  23  16  41   8  10  42
```

Mostrar riqueza alelica por localidad a lo largo de todos los loci

```
allelic.richness(genind2hierfstat(genind))$Ar %>% apply(MARGIN = 2, FUN = mean) %>% round(digits = 3)
```

```
## [1] 1.146 1.115 1.122 1.165 1.137 1.140 1.137 1.162
```

Heterocigosidad observada promedio por localidad

```
Ho = basic.stats(genind, diploid = T)$Ho %>% apply(MARGIN=2, FUN=mean, na.rm=T) %>% round(digits = 3)
Ho
```

```
##   BIA   IRE   ITA   MOR   MSM   POR   SCO   WAL
## 0.174 0.143 0.153 0.201 0.160 0.171 0.168 0.181
```

Heterocigosidad esperada promedio por localidad

```
He = basic.stats(genind, diploid = T)$Hs %>% apply(MARGIN=2, FUN=mean, na.rm=T) %>% round(digits = 3)
He
```

```
##   BIA   IRE   ITA   MOR   MSM   POR   SCO   WAL
## 0.144 0.111 0.119 0.158 0.135 0.123 0.134 0.160
```

Fst por pares en StAMPP

El código abajo permite estimar Fst por pares de localidades utilizando el paquete StAMPP (Pembleton et al. 2013). La estimación del FST siguió el método de Wright (1949), pero corregido por el tamaño desigual de la población actualizado por Weir & Cockerham (1984) (véase Pembleton et al. 2013).

Convertir genind a stampp

```
stampp<-stamppConvert(genind, type = "genlight")
```

Calcular fst por pares

```
stampp.fst<-stamppFst(stampp, nboots = 10000, percent = 99, nclusters = 6)
```

Mostrar Fst por pares

```
stampp.fst$Fsts %>% round(digits = 3)
```

```
##        BIA    IRE   ITA    MOR   MSM   POR SCO WAL
## BIA     NA     NA    NA     NA    NA    NA  NA  NA
## IRE  0.010     NA    NA     NA    NA    NA  NA  NA
## ITA  0.045  0.067    NA     NA    NA    NA  NA  NA
## MOR -0.022  0.007 0.062     NA    NA    NA  NA  NA
## MSM -0.003  0.004 0.049  0.000    NA    NA  NA  NA
## POR  0.045  0.052 0.074  0.056 0.033    NA  NA  NA
## SCO  0.007 -0.007 0.079 -0.011 0.011 0.061  NA  NA
## WAL -0.006 -0.002 0.029 -0.017 0.001 0.016   0  NA
```

Mostrar p-values por pares

```
stampp.fst$Pvalues %>% round(digits = 3)
```

```
##       BIA   IRE ITA   MOR   MSM   POR   SCO WAL
## BIA    NA    NA  NA    NA    NA    NA    NA  NA
## IRE 0.054    NA  NA    NA    NA    NA    NA  NA
## ITA 0.000 0.000  NA    NA    NA    NA    NA  NA
## MOR 1.000 0.122   0    NA    NA    NA    NA  NA
## MSM 0.940 0.158   0 0.515    NA    NA    NA  NA
## POR 0.003 0.000   0 0.000 0.044    NA    NA  NA
## SCO 0.108 0.915   0 0.988 0.008 0.000    NA  NA
## WAL 0.951 0.646   0 1.000 0.346 0.161 0.486  NA
```

PCA: Análisis de Componentes Principales en Adegenet

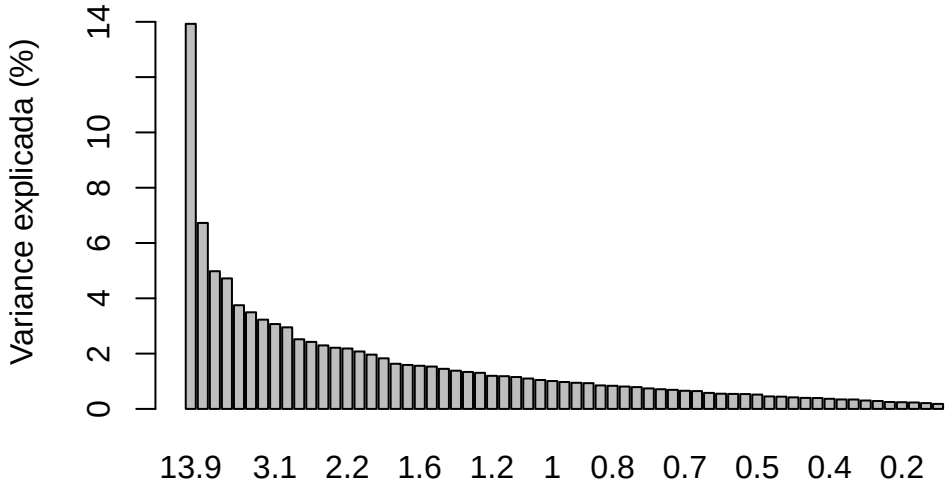
El código abajo permite realizar un Analisis de Componentes Principales (PCA) utilizando el paquete Adegenet (Jombart, 2008).

Cargar información poblacional e incluir al genind

```
ind=as.character(pop$V1) # ID por individuo
site = as.character(pop$V2) # ID por sitio
genind <- vcfR2genind(vcf, ploidy=2, ind.names = ind, pop = site)
```

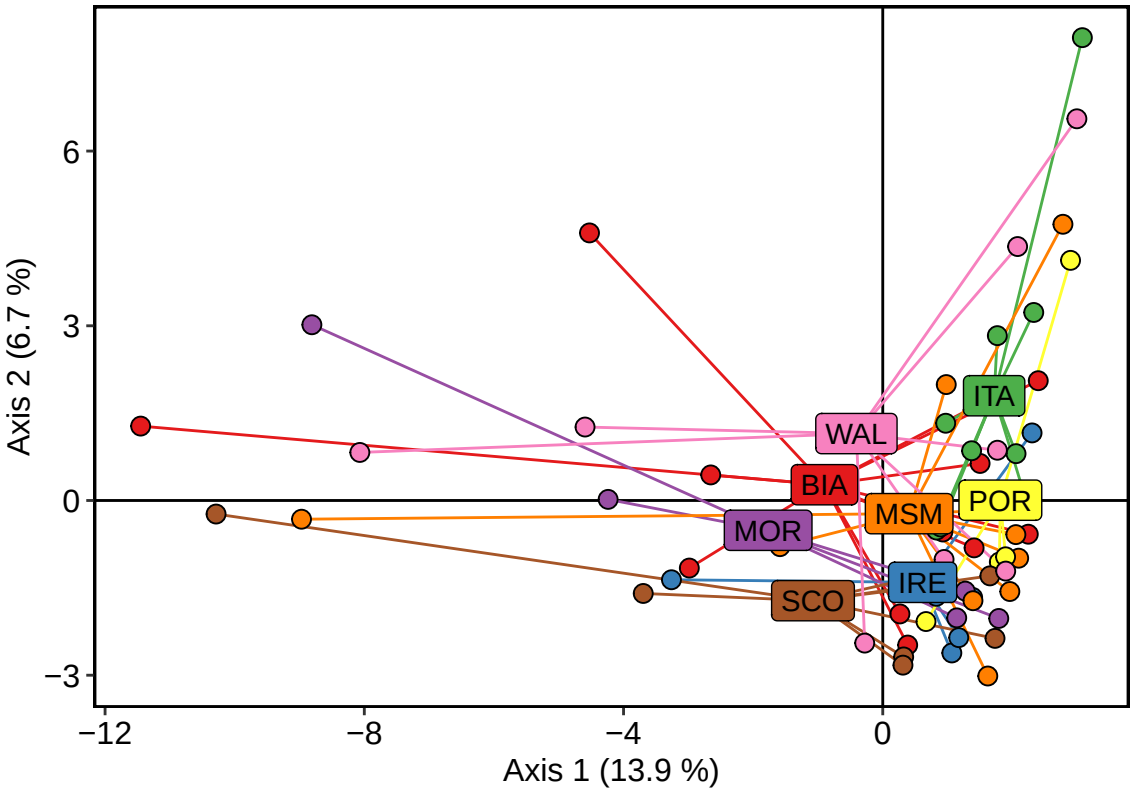
Realizar un análisis de componentes principales para el set de datos

```
# Sustituir los datos que faltan por las frecuencias alélicas medias
x = tab(genind, NA.method = "mean")

# Realizar el PCA
pca1 = dudi.pca(x, scannf = FALSE, scale = FALSE, nf = 3)

# Porcentaje de la varianza genética se explica por cada eje
percent = pca1$eig/sum(pca1$eig)*100
barplot(percent, ylab = "Variance explicada (%)", ylim = c(0,15),
        names.arg = round(percent, 1))
```



Visualisar resultados PCA

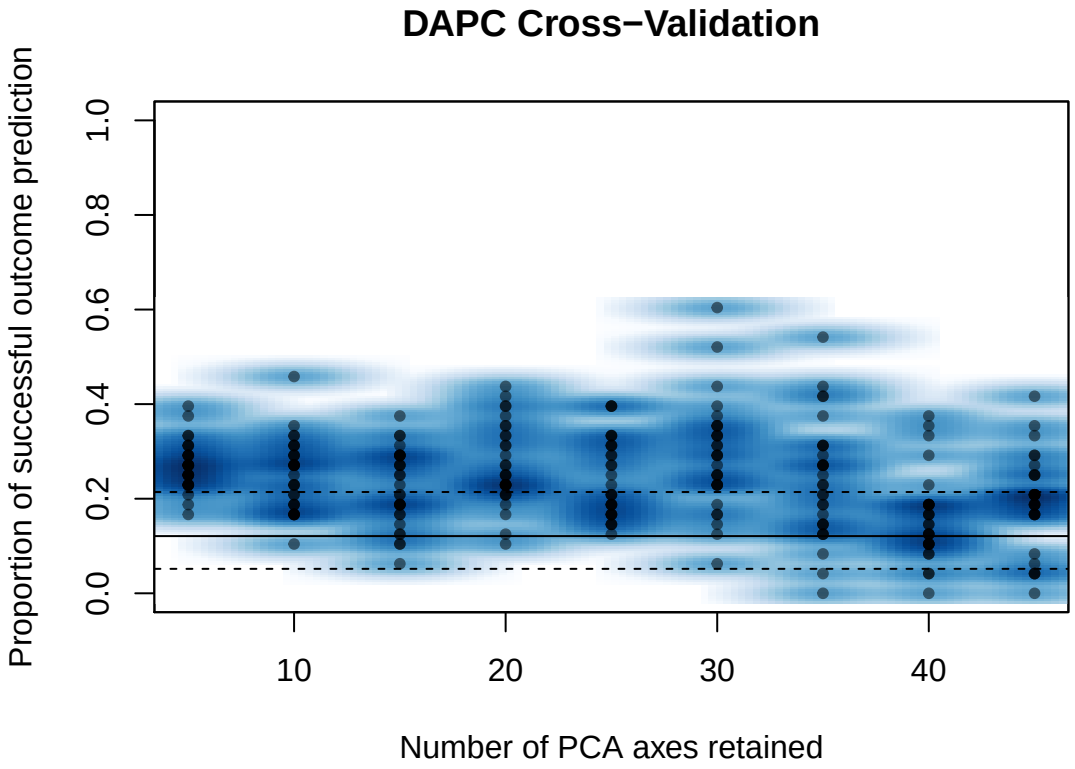
```
# Crear un data.frame con coordenadas individuales
ind_coords = as.data.frame(pca1$li)
# Renombrar las columnas del dataframe
colnames(ind_coords) = c("Axis1","Axis2","Axis3")
# Adherir una columna que contiene los individuos
ind_coords$Ind = indNames(genind)
# Adherir una columna que contiene los sitios
ind_coords$Site = genind$pop
# Calcular la posicion del centroide (promedio) por cada población
centroid = aggregate(cbind(Axis1, Axis2, Axis3) ~ Site, data = ind_coords, FUN = mean)
# Adherir las coordenadas del centroide al dataframe ind_coords
ind_coords = left_join(ind_coords, centroid, by = "Site", suffix = c("",".cen"))
cols = brewer.pal(nPop(genind), "Set1") # Define la paleta de colores
# Personalizar ejes x e y
xlab = paste("Axis 1 (", format(round(percent[1], 1), nsmall=1)," %)", sep="")
ylab = paste("Axis 2 (", format(round(percent[2], 1), nsmall=1)," %)", sep="")
# Personalizar el tema para ggplot2
ggtheme = theme(axis.text.y = element_text(colour="black", size=12),
                axis.text.x = element_text(colour="black", size=12),
                axis.title = element_text(colour="black", size=12),
                panel.border = element_rect(colour="black", fill=NA, size=1),
                panel.background = element_blank(),
                plot.title = element_text(hjust=0.5, size=15))
# Grafico de dispersion usando eje 1 vs. 2
ggplot(data = ind_coords, aes(x = Axis1, y = Axis2))+
  geom_hline(yintercept = 0)+
  geom_vline(xintercept = 0)+
  # spider segments
  geom_segment(aes(xend = Axis1.cen, yend = Axis2.cen, colour = Site), show.legend = F)+
  # Puntos
  geom_point(aes(fill = Site), shape = 21, size = 3, show.legend = F)+
  # Centroides
  geom_label(data = centroid, aes(label = Site, fill = Site), size = 4, show.legend = F)+
  # Colorear
  scale_fill_manual(values = cols)+
  scale_colour_manual(values = cols)+
  # Persoanlizar ejes
  labs(x = xlab, y = ylab)+
  ggtitle("")+
  # custom theme
  ggtheme
```


6
3
0
−3
Axis 2 (6.7 %)
−12
−8
−4
0
Axis 1 (13.9 %)
WAL
ITA
BIA
MSM
POR
MOR
IRE
SCO

DAPC: análisis discriminante de componentes principales

El código abajo permite realizar un análisis discriminante de componentes principales (DPCA) utilizando el paquete Adegenet (Jombart, 2008).

```
# Realizar una validación cruzada para encontrar el número óptimo de PCs a retener en el DAPC
set.seed(123)
x = tab(genind, NA.method = "mean")
crossval = xvalDapc(x, genind$pop, result = "groupMean", xval.plot = TRUE)
```



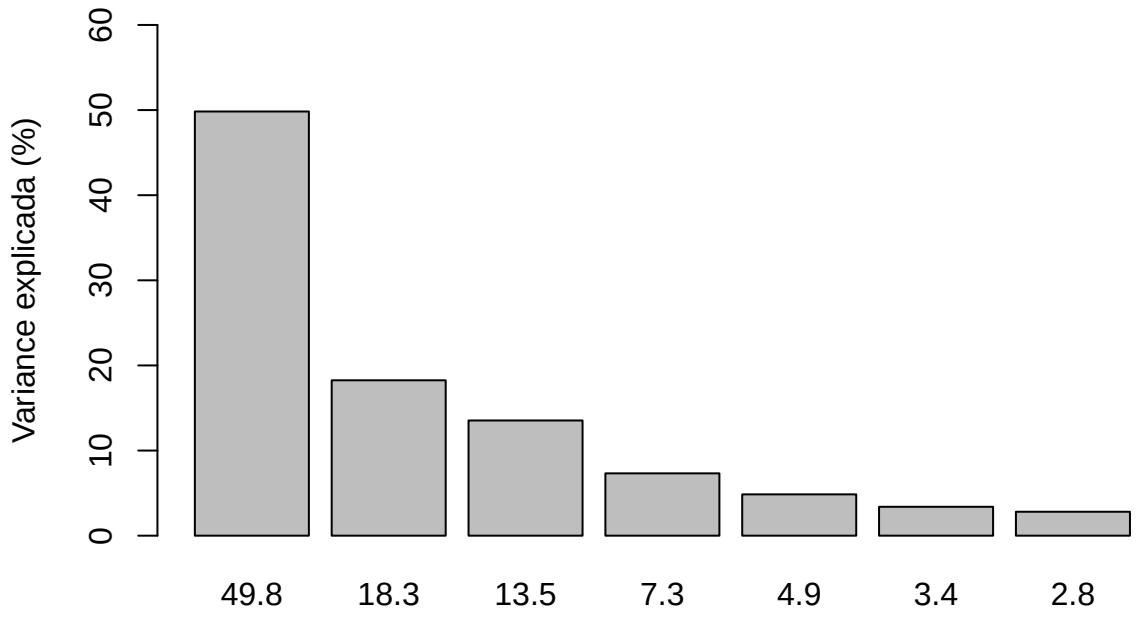

```
# Número de PC con las mejores estadísticas (menor puntuación = mejor)
crossval$`Root Mean Squared Error by Number of PCs of PCA` %>% round(digits = 3)
```

```
##     5    10    15    20    25    30    35    40    45
## 0.731 0.754 0.780 0.732 0.760 0.719 0.773 0.845 0.810
```

```
numPCs = as.numeric(crossval$`Number of PCs Achieving Lowest MSE`)
numPCs
```

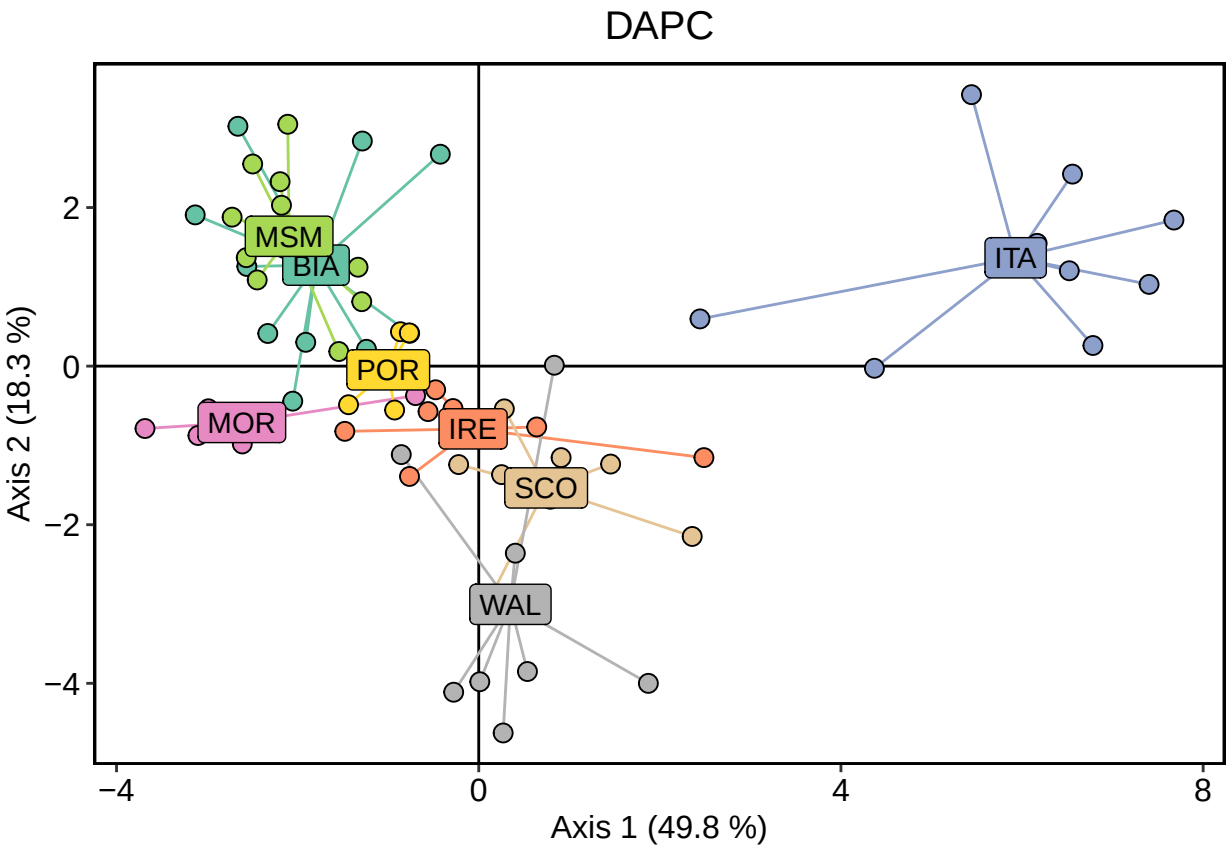
```
## [1] 30
```

```
# Ejecutar un DAPC utilizando los ID de los sitios como priois
dapc1 = dapc(genind, genind$pop, n.pca = numPCs, n.da = 3)

# Analizar qué porcentaje de varianza genética explica cada eje
percent = dapc1$eig/sum(dapc1$eig)*100
barplot(percent, ylab = "Variance explicada (%)", ylim = c(0,60),
        names.arg = round(percent, 1))
```



Visualise DAPC results.

```
# Crear un data.frame con coordenadas individuales
ind_coords = as.data.frame(pca1$li)
ind_coords = as.data.frame(dapc1$ind.coord)
# Renombrar las columnas del dataframe
colnames(ind_coords) = c("Axis1","Axis2","Axis3")
# Adherir una columna que contiene los individuos
ind_coords$Ind = indNames(genind)
# Adherir una columna que contiene los sitios
ind_coords$Site = genind$pop
# Calcular la posicion del centroide (promedio) por cada población
centroid = aggregate(cbind(Axis1, Axis2, Axis3) ~ Site, data = ind_coords, FUN = mean)
# Adherir las coordenadas del centroide al dataframe ind_coords
ind_coords = left_join(ind_coords, centroid, by = "Site", suffix = c("",".cen"))
# Definir paleta de colores
cols = brewer.pal(nPop(genind), "Set2")
# Personalizar ejes x e y
xlab = paste("Axis 1 (", format(round(percent[1], 1), nsmall=1)," %)", sep="")
ylab = paste("Axis 2 (", format(round(percent[2], 1), nsmall=1)," %)", sep="")

# Grafico de dispersion usando eje 1 vs. 2
ggplot(data = ind_coords, aes(x = Axis1, y = Axis2))+
  geom_hline(yintercept = 0)+
  geom_vline(xintercept = 0)+
  # spider segments
  geom_segment(aes(xend = Axis1.cen, yend = Axis2.cen, colour = Site), show.legend = F)+
  # Puntos
  geom_point(aes(fill = Site), shape = 21, size = 3, show.legend = F)+
  # Centroides
  geom_label(data = centroid, aes(label = Site, fill = Site), size = 4, show.legend = F)+
  # Colorear
  scale_fill_manual(values = cols)+
  scale_colour_manual(values = cols)+
  # Perosnalizar ejes
  labs(x = xlab, y = ylab)+
  ggtitle("DAPC")+
  # custom theme
  ggtheme
```


DAPC
MSM
BIA
POR
MOR
IRE
SCO
WAL
ITA
Axis 2 (18.3 %)
Axis 1 (49.8 %)
2
0
−2
−4
−4
0
4
8

Estimación de Migración Relativa

El código abajo permite estimar la migracion relativa entre localidades asi como su magnitud utilizando la función divMigrate incluido en el paquete diveRsity (Keenan 2017). divMigrate es una función experimental que permite investigar y trazar el patrón de migración en datos de alelos de microsatélites o SNPs. Para más detalles, incluir en el codigo ?divMigrate. El artículo del que se deriva el método es Sundqvist et al., (2016).

Transformar genind a genepop

```
gp_file<-writeGenPop(genind, file.name = "populations.snps.gen", comment = "#SNPsdata")
```

Estimar y graficar migración relativa usando parámetro "d"

```
d_div_res <- divMigrate("populations.snps.gen", plot_network = T, stat = "d", para = 6)
```