Dataset and analysis pipeline included in the Publication:
Divergent pattern between phenotypic and genetic variation in Scots pine
Journal: Plant Communications
Authors: David Hall[1], Jenny Olsson[1], Wei Zhao[1,2], Johan Kroon[3], Ulfstand Wennström[3] and Xiao-Ru Wang[1,2]

Institutions:
[1]Department of Ecology and Environmental Science, Umeå Plant Science Center, Umeå University, Sweden.
[2]Advanced Innovation Center for Tree Breeding by Molecular Design; College of Biological Sciences and Technology, Beijing Forestry University, China.
[3]The Forestry Research Institute of Sweden (Skogforsk), Sweden

Abstract:
In boreal forests, autumn frost tolerance in seedlings is a critical fitness component because it determines survival rates during regeneration. To understand the forces that drive local adaptation in this trait, we conducted freezing tests in a common garden setting for 54 Pinus sylvestris (Scots pine) populations (>5000 seedlings) collected across Scandinavia into Western Russia, and genotyped 24 of these populations (>900 seedlings) at >10,000 SNPs. Variation in cold hardiness among populations, as measured by QST, was above 80% and followed a distinct cline along latitude and longitude, demonstrating significant adaptation to climate at origin. In contrast, the genetic differentiation was very weak (mean FST 0.37%). Despite even allele frequency distribution in the vast majority of SNPs among all populations, a few rare alleles appeared at very high or at fixation in marginal populations restricted to Northwestern Fennoscandia. Genotype-environment associations showed that climate variables explained 2.9% of the genetic differentiation, while genotype-phenotype associations revealed a high marker-estimated heritability of frost hardiness 0.56, but identified no major loci. Very extensive gene flow, strong local adaptation and signals of complex demographic history across markers are interesting topics of forthcoming studies on this species to better clarify signatures of selection and demography.

Data summary
The data consists of:

1. One file that contain the results from the freezing test of the > 5000 seedlings “freeze_res.txt”.
2. Two Variant Call Format files which contain genotype variants for pine seedlings which were a part of the freeze test, mapping reference Pinus taeda assembly NCBI GCA_000404065.2: “All_individuals_935.vcf.gz” and “Unrelated_individuals_746.vcf.gz”. The latter is a subset of the former. However the former includes slightly more loci (11039 vs 10925 SNPs respectively). Raw sequencing library data can be found at NCBI bioproject PRJNA687348 with accession numbers: SRX9752756 to SRX9752759
3. One file that assign each genotyped seedling to a specific population and it’s normalized damage score from the freeze test “pheno_pop.txt”.
4. One population specific file that give coordinates for each population the populations’ environmental variables “pop_env.txt”.

Details:
VCF-files:
The Variant Call Format, or VCF, was developed for the 1000 Genomes Project as a standardized format for storing large quantities of sequence variation data (SNPs, indels, larger structural variants, etc.) and any accompanying genotype data and annotation. A VCF file contains a header section and a data table section. VCF files are compressed (using bgzip), and easily accessed. See Danecek, et. al. 2011 (https://10.1093/bioinformatics/btr330) for a concise overview of VCF, and the GitHub site maintained by the The Global Alliance for Genomics and Health (GA4GH) for a more detailed description of the version of the VCF format used for this data (http://samtools.github.io/hts-specs/VCFv4.2.pdf). Briefly, the VCF format has two parts, a header section and a data section. The header contains an arbitrary number of meta-information lines, each starting with characters ‘##’, and a TAB delimited field definition line, starting with a single ‘#’ character. The meta-information header lines provide a standardized description of tags and

annotations used in the data section. The field definition line names eight mandatory columns, corresponding to data columns representing the chromosome (CHROM), a 1-based position of the start of the variant (POS), unique identifiers of the variant (ID), the reference allele (REF), a comma separated list of alternate non-reference alleles (ALT), a phred-scaled quality score (QUAL), site filtering information (FILTER) and a semicolon separated list of additional, user extensible annotation (INFO). In addition, if samples are present in the file, the mandatory header columns are followed by a FORMAT column and an arbitrary number of sample IDs that define the samples included in the VCF file. The FORMAT column is used to define the information contained within each subsequent genotype column, which consists of a colon separated list of comma separated fields.

Basic analysis of VCF-files can be done by VCFtools https://vcftools.github.io/index.html and other software in HTSlib.
HTSlib is an implementation of a unified C library for accessing common file formats, such as SAM, CRAM and VCF, used for high-throughput sequencing data, and is the core library used by samtools and bcftools. HTSlib only depends on zlib. It is known to be compatible with gcc, g++ and clang.
https://github.com/samtools/htslib

VCF-header of the two VCF-files:

```
##fileformat=VCFv4.2 (file format of the data)
##fileDate=20201222  (Date this version of the file was created)
##handle=XRWLAB      (lab name)
##batch=JOGBS        (internal name of dataset)
##bioproject_id=PRJNA687348    (NCBI project number for raw data)
##reference=GCA_000404065.2    (NCBI genome mapping reference)
```

Freeze test results
"freeze_res.txt"

`Repl:`	Replication 1 to 18
`Box:`	Seedling box number. Each box is 11 x 7 cells
`Row:`	Row number in box 1 to 11
`Plant:`	Plant number in box 1 to 7
`Seedlot:`	Seedlot number 1 to 55
`Day:`	The date of the freezing for this seedling
`Inv:`	Person that did the scoring
`Temp:`	Temperature of the freezer in Celsius
`EdgeF:`	How exposed or close to the edge the seedling was
`Round:`	Freezing round 1 to 12, could include one or more Replications.
`Damage:`	Damage score of 0 to 6
`lat:`	Latitude of origin for the sample
`long:`	Longitude of origin for the sample
`ddi:`	The number of days since the seedlings were exposed to increasing night length, or stimuli to start winter dormancy, Days since Dormancy Initiation
`Sddi, Slong and Slat:`	Standardisation of the three latter measurements by centering (subtracting the mean) and then scaled by dividing their standard deviations.
`DamageN:`	The damage scores were normalized over replication and freezing rounds with the quantile normalization procedure.

The naming of the samples in the file comes from a combination of box, replication seedlot, and plant number.

Reading in data files in R
See https://cran.r-project.org/ the following script can be imported in to R and data files loaded accordingly
Please install required libraries as they are introduced

For relatedness analysis see https://doi.org/10.5061/dryad.h44j0zpg5

```
# Read VCF-files
# There are several R-packages that deal with VCF-files with many tutorials and vignettes
# e.g vcfR               https://cran.r-project.org/web/packages/vcfR/
#     VariantAnnotation  https://www.bioconductor.org/packages/release/bioc/html/VariantAnnotation.html
# or  radiator           https://thierrygosselin.github.io/radiator/index.html

# Because the reference sequence these genotypes are mapped to, is not at the chromosome level
# our chromosome field in the vcf-file contain thousands of scaffolds which many software unfortunately
# can not handle, so the analysis has to be done manually for this case

infile <- "Unrelated_individuals_746.vcf.gz"

# infile <- "All_individuals_935.vcf.gz"
# The dtemp3 object generated by this infile is required to recreate the dataset for GEMMA association
# mapping analysis


testcon <- gzfile(infile)
out<-readLines(testcon)
dstart<-grep("CHROM", out) # At what line does the VCF data table start
dtemp<-scan(testcon, what=character(), sep="\t", skip={dstart-1}) # reads only the data table and the column
names
# cleaning up
close(testcon)
rm("out")
rm("testcon")

ncols <- grep("PASS", dtemp)[1:2] # To find out how many columns there are in the data table section
dtemp2<-matrix(dtemp, byrow=TRUE, ncol={ncols[2]-ncols[1]}) # converting to matrix format
dtemp3<-dtemp2[-1,] # removes the line of column names
colnames(dtemp3)<-dtemp2[1,] # labels the columns
rownames(dtemp3)<-paste(dtemp3[,1], ":", dtemp3[,2], sep="") # convert scaffold name and position into one
SNP name
# cleaning up
rm(dtemp)
rm(dtemp2)
rm(dstart)
rm(ncols)

# Because the data is diploid we also consider the third character i.e (1/0) in the VCF-file
# of each genotypes GT #FORMAT field (first field of the FORMAT field for each seedling)

##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=PL,Number=G,Type=Integer,Description="List of Phred-scaled genotype likelihoods">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Number of high-quality bases">
##FORMAT=<ID=AD,Number=R,Type=Integer,Description="Allelic depths">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Phred-scaled Genotype Quality">

# Example: Individual A-10-5-4 is heterozygous 0/1 for SNP C32249586:4051 and
# homozygous for the alternate allele 1/1 (different than reference sequence) for SNP scaffold723130:145170
dtemp3[c(16,716) ,c(9,55)]

# Conversion function of the VCF-data table to a more convenient format e.g 0 for 0/0, 1 for 0/1 and 2 for
# 1/1 genotype while ./. means missing data and get a NA.
binary_gen<-function(x) {
  allele1<-substring(x, 1, 1) # first allele is the first character
  allele2<-substring(x, 3, 3) # second allele is the second character
  return(ifelse(allele1=="1", 2, ifelse(allele2=="0", 0, ifelse(!allele1==allele2, 1, NA))))
}

# Conversion of the VCF-table to individual in rows and loci in columns as genotype scores (0,1 or 2).
# Skipping the first nine columns and applying the function over samples/columns (2) and then transposing
unrelpine<-t(apply(dtemp3[,-c(1:9)], 2, binary_gen))

# if the infile is "All_individuals_935.vcf.gz"
# allpine<-t(apply(dtemp3[,-c(1:9)], 2, binary_gen))

dim(unrelpine) # number of individuals and number of loci

# read in the genotypes' population description
# "All_individuals_935.vcf.gz"
pops <- read.table("pheno_pop.txt", head=TRUE, sep="\t", stringsAsFactors = FALSE)

# "Unrelated_individuals_746.vcf.gz"
pops_red <- pops[match(rownames(unrelpine), pops$ID),]

# Now we can get a summary of the number of samples for each population. Populations are ordered west to east
1 to 56.
# Only 24 of the 56 phenotyped populations have been genotyped.
tapply(rownames(unrelpine), pops_red$pop, length)

# Load coordinates and environmental variables for the populations
EnvPop <- read.table("pop_env.txt", head=TRUE, sep="\t", stringsAsFactors = FALSE)
# order according to population number
EnvPop <- EnvPop[order(EnvPop$pop), ]
```

Plot the populations on a map

```
# Population 32 and 42 had no germination and are excluded
library(maps)
plot(EnvPop[-c(32,42),c("Long", "Lat")],type="n",  xlab="", ylab = "Latitude", xlim=c(3,55), ylim = c(56,
71), las=1, yaxt="n", bg=NA)
axis(2, at=seq(58,70,2), labels = seq(58,70,2), las=1)
mtext("Longitude", side = 1, line=3.5, cex=1)
map(add = T, interior = F, mar = c(4.5, 4.5,1.2,1), mgp = c(3, 1, 0), mfrow = c(1,1))
points(EnvPop[-c(32,42),c("Long", "Lat")], pch = 19, cex=2.5)
text(EnvPop[-c(32,42), "Long"], EnvPop[-c(32,42), "Lat"], labels=EnvPop[-c(32,42),"pop"], col="white",
adj=c(0.5, 0.4), cex=0.7)
```

Conversion to BayeScan

```
# Conversion of genotype scores to population allele count so they can be used for BayeScan
# This produces data input for BayeScan see http://cmpg.unibe.ch/software/BayeScan/download.html
# unfortunately population and loci are labelled as sequential numbers, so this has to be remembered
sort(unique(pops_red$pop)) # becomes 1:24
filnam <- "Bscan_gen.txt"
write.table(matrix(c(paste0("[loci]=", ncol(unrelpine)), "",
                     paste0("[populations]=", length(unique(pops_red$pop)))), ncol=1), file=filnam,
            quote=FALSE, sep="\t", col.names = FALSE, row.names = FALSE)

for(i in 1:length(unique(pops_red$pop))){
Temp <- t(apply(unrelpine[pops_red$pop==sort(unique(pops_red$pop))[i], ], 2, function(x) {
  Tot <- sum(!is.na(x))*2
  P <- sum(!is.na(x) & x==0)*2 + sum(!is.na(x) & x==1)
  Q <- Tot-P
  X <- c(Tot, P, Q)
  return(X)}))
rownames(Temp) <- 1:nrow(Temp)
write.table(matrix(c("", paste0("[pop]=",i)), ncol=1), file=filnam, quote=FALSE, sep="\t", col.names = FALSE,
row.names = FALSE, append=TRUE)
write.table(Temp, file=filnam, quote=FALSE, sep="\t", col.names = FALSE, row.names = TRUE, append=TRUE)
}

# The BayeScan analysis was then run on HPC2N cluster at Umeå University similar to the following:
# bayescan Bscan_gen.txt -o bayescanLongRunOD100 -threads 7 -n 5000 -thin 10 -nbp 20 -pilot 5000
# -burn 50000 -pr_odds 100
# Which takes just above 9 hours using seven cores
```

Conversion to GEMMA

```
# Conversion of VCF-file so it can be used for association mapping of GEMMA,
# see https://github.com/genetics-statistics/GEMMA
# This produces data in the BIMBAM format see http://stephenslab.uchicago.edu/software.html
# FORMAT fields in this VCF-file "GT:PL:DP:AD:GQ"
average_gen <- function(x) {
  # second allele count/depth
  AD <- as.numeric(unlist(lapply(lapply(strsplit(x, ":"), "[[", 4), function(y) strsplit(y, ",")[[1]][2])))
  # total depth
  DP <- as.numeric(unlist(lapply(strsplit(x, ":"), "[[", 3)))
  AG <- round((AD/DP)*2, 2) # Average genotype between 0 and 2 rounded to second decimal
  AG[is.nan(AG)]<-NA
  return(AG)
}

# In this case the infile has to have been "All_individuals_935.vcf.gz" to get the same data set that was
# used in the manuscript

# This writes a table in BIMBAM format for average genotypes with SNP name (a combination of scaffold and
# position) with row.names =TRUE, the two alleles dtemp3[,4:5] and then the average genotype
# (second allele depth average) of the individuals for that locus e.g for the first two loci
# and 7 individuals
# C32056486:7356, A, G, 0, 0, 0, 0, 0, 2, 0,
# C32056486:7462, T, C, 0, 0, 0, 0, 0, 2, 0.37,

write.table(data.frame(dtemp3[,4:5],  apply(dtemp3[, -c(1:9)], 2, average_gen)),
file="avg_gen.txt", quote=FALSE, sep=", ", row.names=TRUE, col.names=FALSE)
write.table(phenotypes$phen, "pheno.txt", col.names=FALSE, row.names=FALSE, quote=FALSE, sep="\t")

#remove NW pops for comparison
idx_red<-which(!pops$pop %in% c(10,16,26))

write.table(data.frame(dtemp3[,4:5],  apply(dtemp3[, {idx_red+9}], 2, average_gen)),
file="avg_gen_red.txt", quote=FALSE, sep=", ", row.names=TRUE, col.names=FALSE)
write.table(phenotypes$phen[idx_red], "pheno_red.txt", col.names=FALSE, row.names=FALSE,
quote=FALSE, sep="\t")

# GEMMA example run syntax on the computer cluster HPC2N at Umeå University,
# this analysis would take approximately 10 hours.
# gemma-0.98-linux-static -g avg_gen_red.txt -p pheno_red.txt -miss 0.8 -maf 0.01 -hwe 0.00001 -notsnp -bslmm
# 1 -s 10000000 -smax 500 -o red_BSLMM
```

Population structure and outliers TESS3

```
# TESS3 analysis
# https://bcm-uga.github.io/TESS3_encho_sen/articles/main-vignette.html

# install.packages("Rcpp")
# install.packages("devtools")
# devtools::install_github("bcm-uga/TESS3_encho_sen")
library(Rcpp)
library(tess3r)
```

```
library(qvalue) # see https://www.bioconductor.org/packages/release/bioc/html/qvalue.html
library(sp)

my.colors <- c("tomato", "orange", "blue")
my.palette <- CreatePalette(my.colors, 9)
?tess3

# Create coordinates for the populations the genotypes belong to
Coords <- EnvPop[match(pops_red$pop, EnvPop$pop), c("Long", "Lat")]

# We want the populations in the west to east order
Ord<-order(pops_red$pop)
# run tess3 analysis
tess3.obj <- tess3(X = unrelpine[Ord,], coord = as.matrix(Coords[Ord,]), K = 1:5, rep=20,
                   method = "projected.ls", mask = 0.1, ploidy = 2, openMP.core.num = 4)

p.values <- pvalue(tess3.obj, K = 3)
hist(p.values, col = "lightblue")


# Manhatan plot
plot(p.values, main = "Manhattan plot", xlab = "Locus id", ylab = "-log10(P-values)", cex = .3, col = "grey")

# Cross-validation plot
plot(tess3.obj, pch = 19, col = "blue", xlab = "Number of anc. pop. (K)",
     ylab = "CV score\n", ylim=c(0.326, 0.3325), xlim=c(0.6, 5.4),las=1, cex.lab=0.9)

q.matrix <- qmatrix(tess3.obj, K = 3)

# Structure-like barplot
barplot(q.matrix, sort.by.Q = FALSE, border = NA, space = 0,
        main = "Ancestry matrix",
        xlab = "Population", ylab = "Ancestry proportions",
        col.palette = my.palette) -> bp

# draws a line to separate each population
abline(v=c(0, cumsum(tapply(rownames(unrelpine), pops_red$pop, length)), 1))

# Puts population name under each population
text(cumsum(tapply(rownames(unrelpine), pops_red$pop, length))-
               (tapply(rownames(unrelpine), pops_red$pop, length)/2),
               c(rep(-0.03, 12),-0.06, -0.03, -0.06, -0.03, -0.03, -0.06, -0.03),
               labels=names(tapply(rownames(unrelpine), pops_red$pop, length)),las=1, xpd=TRUE)

# Creating pie diagram for each population
Q1<-tapply(q.matrix[,1], pops_red$pop[Ord], mean)
Q2<-tapply(q.matrix[,2], pops_red$pop[Ord], mean)
Q3<-tapply(q.matrix[,3], pops_red$pop[Ord], mean)

AncComp<-data.frame(pop=names(Q1), Q1,Q2,Q3)
AncComp1<-data.frame(EnvPop[match(AncComp$pop, EnvPop$pop),c("Long", "Lat")], AncComp)

library(mapplots) #draw.pie function
par(mar=c(4,4,1,1))
plot(EnvPop[-c(32,42),c("Long", "Lat")],type="n",  xlab="", ylab = "Latitude",
               xlim=c(3,55), ylim = c(56, 71), las=1, yaxt="n", bg=NA)
axis(2, at=seq(58,70,2), labels = seq(58,70,2), las=1)
mtext("Longitude", side = 1, line=3, cex=1)

map(add = T, interior = T, mar = c(4.5, 4.5,1.2,1), mgp = c(3, 1, 0), mfrow = c(1,1),
               fill = TRUE, col="lightgrey")
draw.pie(AncComp1$Long, AncComp1$Lat, as.matrix(AncComp1[,c("Q1", "Q2","Q3")]), radius = 0.5,
               col=c("orange", "grey", "lightblue"))

# population labels for the pie diagrams with manually adjusted x and y positions
# text(x=LabPie$X, y=LabPie$Y, labels=LabPie$NR)

legend(x=45.9, y=58.9,border="#FFFFFF70", legend=c("cluster 1", "cluster 2", "cluster 3"),
               box.col="#FFFFFF70", pt.bg=c("grey", "orange", "lightblue" ), pch =22, pt.cex=2, cex=1.2)


# Phenotype analysis
# All phenotype data from the freeze test. Sample names are a combination of box number (A to E, P1, P2 and
P3 are trial boxes),
# Replication 1 to 18, Seedlot 1 to 55 and plant number in the row of a box.
freez <- read.table("freeze_res.txt", head=TRUE, sep=" ", stringsAsFactors = FALSE)
head(freez)
table(freez$Box)
BOX <- LETTERS[suppressWarnings(as.numeric(freez$Box))]
SamplesPhen <- paste(ifelse(!is.na(BOX), BOX, freez$Box), freez$Repl, freez$Seedlot, freez$Plant, sep="-")

# matching samples from the freezing results to those that have been genotyped with the following
head(freez[match(rownames(allpine), SamplesPhen),])
# for comparison
head(pops)

# quantile normalization
# Only using the first 15 of 18 replications, the last three are for the trial boxes and does not include all
populations

# First we need to create a matrix of damage scores where each column is a replication (wide format)
Mat1<-NULL
for(k in 1:15){
  Mat1<-cbind(Mat1, freez[freez$Repl==k,"Damage"])
}
```

```r
# then we get the index number for each of the values so we can return the normalized values as a vector that
# matches their counterpart
idxMat<-NULL
for(k in 1:15){
  idxMat<-cbind(idxMat, which(freez$Repl==k))
}

# https://bioconductor.org/packages/release/bioc/html/preprocessCore.html
# BiocManager::install("preprocessCore")
library(preprocessCore)
# rank quantile normalization across replicates (columns) to make all replicates comparable
Mat2<-normalize.quantiles(Mat1)

# Create an empty vector of the same length as the number of rows in the dataset
FrysN<-array(NA, nrow(freez))

FrysN[as.vector(matrix(idxMat, 1))] <- as.vector(matrix(Mat2, 1))

# for comparison
plot(freez$DamageN, FrysN)

# phenotype analysis
library(lme4)
library(emmeans)

# Final model used in the manuscript
modF <- glmer(Damage ~ Sddi*Slong + Sddi*Slat + Slong*Slat + EdgeF + (1|Plant),
              data =  freez[freez$Repl<16,], family=poisson, control = glmerControl(optimizer="Nelder_Mead",
              optCtrl = list(maxfun=2e6)))
summary(modF)
# However this is the more correct model, increases the effect size slightly for latitude while slightly
# decreasing the effect size for longitude
mod2 <- glmer(Damage ~ Sddi*Slong + Sddi*Slat + Slong*Slat + EdgeF + (Sddi|Seedlot),
              data =  freez[freez$Repl<16,], family=poisson, control = glmerControl(optimizer="Nelder_Mead",
              optCtrl = list(maxfun=2e6)))
summary(mod2)

# create 95% confidence interval
cc <- confint(modF, parm="beta_") # quite slow, approximately 10 minutes
EffSize <- summary(modF)$coefficients[-1, 1:2]

cc2 <- confint(mod2, parm="beta_")
EffSize2<-summary(mod2)$coefficients[-1, 1:2]

par(mfrow=c(1,2))
par(mar=c(4,9,1,1))
plot(EffSize[,1], 1:7, type="n", xlim=c(-0.8, 0.2), ylim=c(0.5, 7.5),
     yaxt="n",
     ylab="", xlab="Effect size  (95% CI)")
abline(v=0, lwd=2, col="grey")
segments(rev(cc[-1,1]), 1:7, rev(cc[-1,2]), 1:7)
points(rev(EffSize[,1]), 1:7, pch=21, bg="#ffffff", cex=1.1)
axis(2, at=1:7, labels=rev(rownames(EffSize)), las=1)
mtext("Fixed effects", 2, xpd=TRUE, line = 7)

plot(EffSize2[,1], 1:7, type="n", xlim=c(-0.8, 0.2), ylim=c(0.5, 7.5),
     yaxt="n",
     ylab="", xlab="Effect size  (95% CI)")
abline(v=0, lwd=2, col="grey")
segments(rev(cc2[-1,1]), 1:7, rev(cc2[-1,2]), 1:7)
points(rev(EffSize2[,1]), 1:7, pch=21, bg="#ffffff", cex=1.1)
axis(2, at=1:7, labels=rev(rownames(EffSize2)), las=1)
mtext("Fixed effects", 2, xpd=TRUE, line = 7)

# Fit a new model without latitude and longitude and seedlot as fixed factor
modF2<-glmer(Damage ~ Sddi + EdgeF + factor(Seedlot) + (1|Plant), data =  freez[freez$Repl<16,],
              family=poisson, control=glmerControl(optimizer="Nelder_Mead",optCtrl=list(maxfun=2e6)))
mod22<-glmer(Damage ~ Sddi + EdgeF + factor(Seedlot) + (Sddi|Seedlot), data =  freez[freez$Repl<16,],
              family=poisson, control=glmerControl(optimizer="Nelder_Mead",optCtrl=list(maxfun=2e6)))

plot(summary(emmeans(mod22, "Seedlot", type = "response"))$rate, summary(emmeans(modF2, "Seedlot", type =
"response"))$rate)
lsm_modF<-summary(emmeans(modF2, "Seedlot", type = "response"))[,-4]
lsm_modF$pop<-EnvPop[match(lsm_modF$Seedlot  , EnvPop$Seedlot), "pop"]
lsm_modF$lat<-EnvPop[match(lsm_modF$Seedlot  , EnvPop$Seedlot), "Lat"]
lsm_modF$long<-EnvPop[match(lsm_modF$Seedlot  , EnvPop$Seedlot), "Long"]
lsm_modF$gdd5<-EnvPop[match(lsm_modF$Seedlot  , EnvPop$Seedlot), "growingDegDays5"]
lsm_modF$gdd0<-EnvPop[match(lsm_modF$Seedlot  , EnvPop$Seedlot), "growingDegDays0"]
lsm_modF$bio1<-EnvPop[match(lsm_modF$Seedlot  , EnvPop$Seedlot), "bio01"]

head(lsm_modF)
lsm_mod2<-summary(emmeans(mod22, "Seedlot", type = "response"))[,-4]

plot(lsm_modF$gdd5, lsm_modF$rate)
plot(ifelse(lsm_modF$gdd5>4000 & lsm_modF$gdd5<7000, lsm_modF$gdd5/2,
            ifelse(lsm_modF$gdd5>7000 & lsm_modF$gdd5<11000, lsm_modF$gdd5/3,
                   ifelse(lsm_modF$gdd5>11000, lsm_modF$gdd5/4, lsm_modF$gdd5))), lsm_modF$rate)


# QST estimates
# https://cran.r-project.org/web/packages/MCMCglmm/vignettes/CourseNotes.pdf
# https://github.com/tmalsburg/MCMCglmm-intro
# Using MCMC
library(MCMCglmm)
modMCMC <- MCMCglmm(Damage~  Sddi * Slong + EdgeF + Sddi * Slat, ~Seedlot + Plant, family = "poisson",
                    data = freez[freez$Repl<16 & !is.na(freez$Damage),], # prior = prior,
                    nitt=50000, thin=10, burnin=10000, verbose = FALSE, pl = TRUE)
```

```r
QST<- modMCMC$VCV[,"Seedlot"]/(modMCMC$VCV[,"Seedlot"]+2*modMCMC$VCV[,"Plant"])
plot(density(QST))
str(modMCMC)

plot(modMCMC)

# Other MCMC package in R that follow the lme4 notation
# see https://cran.r-project.org/web/packages/rstanarm/vignettes/rstanarm.html and
# https://mc-stan.org/rstanarm/articles/index.html
library(rstanarm)
modStan<-stan_glmer(Damage ~ Sddi * Slong + EdgeF + Sddi * Slat + (1|Seedlot) + (1|Plant),
                    data =  freez[freez$Repl<16 & !is.na(freez$Damage),], family=poisson)
plot(modStan)

Qbw<-unlist(lapply(modStan$stanfit@sim$samples, "[[", "theta_L[1]"))
Qwi<-unlist(lapply(modStan$stanfit@sim$samples, "[[", "theta_L[2]"))

plot(density(Qbw/(Qbw+2*Qwi)))


#  RDA
# The idea behind the RDA is to identify which environmental variables can explain the variation in allele
# frequencies across the sampled distribution range while controlling
# See https://popgen.nescent.org/2018-03-27_RDA_GEA.html for additional tutorial


# populations from pops_red$pop above is hardcoded in the function below
popfreq<-function(y, pop=pops_red$pop) {
  tapply(y, pop, function(x) {
    N<-sum(!is.na(x))*2
    A<-sum(x[!is.na(x)])
    Freq<-A/N
    return(Freq)
  }
  )
}

# The reason for using apply and tapply for the calculations of allele frequencies for average values per
# population is that they use the same intrinsic sorting function and the results can be compared directly
# without sorting

PopSNPfreq<-apply(unrelpine, 2, popfreq)
PopSNPfreq[,1:5]

# unfortunately some SNPs are missing from population 8 and has to be removed before RDA
which(is.na(PopSNPfreq), arr.ind = TRUE)
PopFreq <- PopSNPfreq[,-which(is.na(PopSNPfreq), arr.ind = TRUE)[,2]]

# calculate PCNMs using Longitude and Latitude with geoXY from SoDA package
library(SoDA)

# match genotyped populations with those 24 in the environmental data
geo.data <- geoXY(EnvPop[match(rownames(PopFreq), EnvPop$pop), "Lat"], EnvPop[match(rownames(PopFreq),
                  EnvPop$pop),"Long"], unit = 1000)
geo.data<-dist(geo.data)

library(vegan)
# See https://mb3is.megx.net/gustame/spatial-analysis/pcnm
geo.pcnm<-pcnm(geo.data)

str(geo.pcnm) # should contain 24 populations only and thus 24 PCNMs

dist.rda <- rda(PopFreq~., data.frame(scores(geo.pcnm)), scale= FALSE) #  screeplot(env.rda)
stp.dist = ordistep(rda(PopFreq~1, data.frame(scores(geo.pcnm))), scope=formula(dist.rda), scale= FALSE,
                    direction="forward", pstep = 1000)
(selected.dist = attributes(stp.dist$terms)$term.labels)

# make a data frame with only relevant populations and environmental variable only and scale it
env_mat<-as.data.frame(apply(EnvPop[match(rownames(PopFreq), EnvPop$pop), 3:70], 2, scale))
env.rda <- rda(PopFreq~., data.frame(env_mat), scale= FALSE) #  screeplot(env.rda)


# Function ordistep performs step-wise selection of environmental variables based two criteria:
# if their inclusion into the model leads to significant increase of explained variance
# (the same as in forward.sel and ordiR2step), and if the AIC of the new model is lower than AIC of the more
# simple model. In contrary to previous functions, the function does not consider as criteria whether the
# adjusted R2 of the model exceeds the adjusted R2 of the global model.
# The use of ordistep function has the same logic as ordiR2step.

stp.env = ordistep(rda(PopFreq~1, data.frame(env_mat)), scope=formula(env.rda), scale= FALSE,
direction="forward", pstep = 1000) # Very slow
(selected.env = attributes(stp.env$terms)$term.labels)

env.rda2 <- rda(PopFreq~ ., as.data.frame(env_mat[,selected.env]), scale= FALSE)
vif.cca(env.rda2)
RsquareAdj(env.rda2)
anova(env.rda2)

# how many significant axis
anova.cca(env.rda2, by='axis', step=1000)

library(robust)
library(qvalue)

# Significance tests followed the method in (Capblancq et al., 2018), however the minimum number of K is 2
# otherwise no covariance matrix
# https://github.com/Capblancq/RDA-genome-scan/blob/master/Script_RDA.R
rdadapt<-function(rda,K)
```

```
{
  loadings<-rda$CCA$v[,1:as.numeric(K)]
  if(is.null(dim(loadings))) loadings <- matrix(loadings, ncol=1)
  resscale <- apply(loadings, 2, scale)
  resmaha <- covRob(resscale, distance = TRUE, na.action= na.omit, estim="pairwiseGK")$dist
  lambda <- median(resmaha)/qchisq(0.5,df=K)
  reschi2test <- pchisq(resmaha/lambda,K,lower.tail=FALSE)
  qval <- qvalue(reschi2test)
  q.values_rdadapt<-qval$qvalues
  return(data.frame(p.values=reschi2test, q.values=q.values_rdadapt))
}

res_rdadapt <- rdadapt(env.rda2, 2)
which(res_rdadapt$q.values <= 0.05)

# Plot the data on the first two RDA-axis
plot(env.rda2$CCA$v[,1], env.rda2$CCA$v[,2], pch=19, xlim=c(-0.1,0.075), ylim=c(-0.08, 0.06), col="gray83",
las=1, xlab="RDA 1", ylab="RDA 2")
points(env.rda2$CCA$v[which(res_rdadapt[,2] < 0.05),1], y=env.rda2$CCA$v[which(res_rdadapt[,2] < 0.05),2],
pch=19, col="midnightblue")
arrows(x0=0, y0=0, x1=env.rda2$CCA$biplot[,1]/10, y1=env.rda2$CCA$biplot[,2]/10, length = 0.1, col="blue",
lwd=2)
text(env.rda2$CCA$biplot[,1]/10, env.rda2$CCA$biplot[,2]/10, labels = rownames(env.rda2$CCA$biplot),
pos=c(3,4,1,4), cex=0.7)

# Two explanatory data frames -- Hellinger-transform Y
mod <- varpart(PopFreq, env_mat[,selected.env], scores(geo.pcnm)[, selected.dist], transfo="hel")
mod
plot(mod, bg=2:3, )


# partial RDA controlling for distance removes all significance
rda_nodist <- rda(PopFreq~as.matrix(env_mat[,selected.env]) + Condition(as.matrix(scores(geo.pcnm)[,
selected.dist])), scale= FALSE)
RsquareAdj(rda_nodist)
anova(rda_nodist, step=1000)
anova.cca(rda_nodist, by='axis', step=1000)
```