Simulation of selected genotypes

José Tiago

2024-06-25

# Simulating two traits

library(FieldSimR)  
library(AlphaSimR)

## Loading required package: R6

library(ggplot2)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ lubridate 1.9.3 ✔ tibble 3.2.1  
## ✔ purrr 1.0.2 ✔ tidyr 1.3.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ dplyr::mutate() masks AlphaSimR::mutate()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(caret)

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(ggpubr)

##   
## Attaching package: 'ggpubr'  
##   
## The following object is masked from 'package:AlphaSimR':  
##   
## mutate

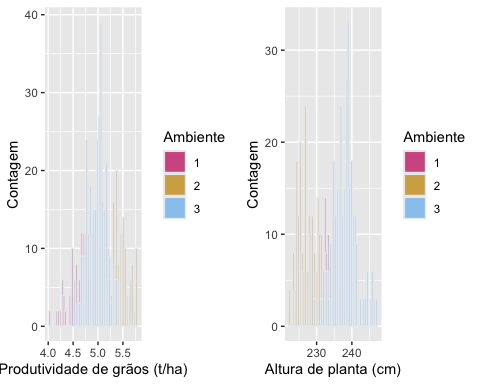
##Compound Simmetry

ntraits <- 2 # Number of traits.  
nenvs <- 3 # Number of environments.  
nreps <- c(2, 2, 3) # Number of replicates of each genotype in environments 1, 2, and 3.  
  
  
nind <- 20 # Number of founder genotypes in the population.  
nchr <- 10 # Number of chromosomes.  
nseg\_sites <- 200 # Number of QTN per chromosome.  
  
mean <- c(4.9, 5.4, 5.1, 235.2, 228.5, 239.1) # c(Yld:E1, Yld:E2, Yld:E3, Pht:E1, Pht:E2, Pht:E3)  
  
  
  
var <- c(0.08, 13) # c(grain yield, plant height)  
  
prop\_main <- c(0.4, 0.6) # c(grain yield, plant height)  
  
corA <- matrix( # Matrix of additive genetic correlations grain yield and plant height.  
 c(  
 1.0, 0.5,  
 0.5, 1.0  
 ),  
 ncol = 2  
)  
  
meanDD <- c(0.4, 0.4, 0.4, 0.1, 0.1, 0.1) # c(Yld:E1, Yld:E2, Yld:E3, Pht:E1, Pht:E2, Pht:E3)  
  
varDD <- c(0.2, 0.2) # c(grain yield, plant height)  
  
prop\_mainDD <- 0.4 # Same value set for traits 1 and 2.  
  
corDD <- diag(2)  
  
input\_asr <- compsym\_asr\_input(  
 ntraits = ntraits,  
 nenvs = nenvs,  
 mean = mean,  
 var = var,  
 prop.main = prop\_main,  
 corA = corA,  
 meanDD = meanDD,  
 varDD = varDD,  
 prop.mainDD = prop\_mainDD,  
 corDD = corDD  
)  
  
founders <- runMacs( # Simulation of founder genotypes using AlphaSimR's "MAIZE" presets  
 nInd = nind, # to mimic the species' evolutionary history.  
 nChr = nchr,  
 segSites = nseg\_sites,  
 species = "MAIZE",  
 nThreads = 2  
)  
  
SP <- SimParam$new(founders)  
  
SP$addTraitAD( # Additive + dominance trait simulation.  
 nQtlPerChr = nseg\_sites,  
 mean = input\_asr$mean,  
 var = input\_asr$var,  
 corA = input\_asr$corA,  
 meanDD = input\_asr$meanDD,  
 varDD = input\_asr$varDD,  
 corDD = input\_asr$corDD,  
 useVarA = FALSE  
)  
  
founders <- newPop(founders)  
  
pool\_A <- makeDH(founders[1:10], nDH = 1) # Pool A: 1 DH line from founders 1 to 10, respectively.  
pool\_B <- makeDH(founders[11:20], nDH = 1) # Pool B: 1 DH line from founders 11 to 20, respectively.  
  
dh\_lines <- mergePops(list(pool\_A, pool\_B))  
  
factorial\_plan <- as.matrix(expand.grid(A = pool\_A@id, B = pool\_B@id)) # Factorial crossing plan.  
  
hybrid\_pop <- makeCross(pop = dh\_lines, crossPlan = factorial\_plan, nProgeny = 1) # Hybrid genotypes.  
  
gv\_df <- compsym\_asr\_output(  
 pop = hybrid\_pop,  
 ntraits = ntraits,  
 nenvs = nenvs,  
 nreps = nreps  
)

library(ggplot2)  
gv\_df=readRDS("gv\_df.rds")  
P1cs=ggplot(gv\_df, aes(x = gv.Trait1, fill = factor(env))) +  
 geom\_histogram(color = "#e9ecef", alpha = 0.8, position = "identity", bins = 50) +  
 scale\_fill\_manual(values = c("violetred3", "goldenrod3", "skyblue2")) +  
 labs(x = "Produtividade de grãos (t/ha)", y = "Contagem", fill = "Ambiente")  
  
P2cs=ggplot(gv\_df, aes(x = gv.Trait2, fill = factor(env))) +  
 geom\_histogram(color = "#e9ecef", alpha = 0.8, position = "identity", bins = 50) +  
 scale\_fill\_manual(values = c("violetred3", "goldenrod3", "skyblue2")) +  
 labs(x = "Altura de planta (cm)", y = "Contagem", fill = "Ambiente")  
  
Plotscs=ggarrange(P1cs,P2cs)  
ggsave(plot = Plotscs, filename = "plotcsmodel.png",device = "png",dpi = "retina")

## Saving 5 x 4 in image

saveRDS(gv\_df, file = "gv\_df.rds")  
plot(Plotscs)



## errors

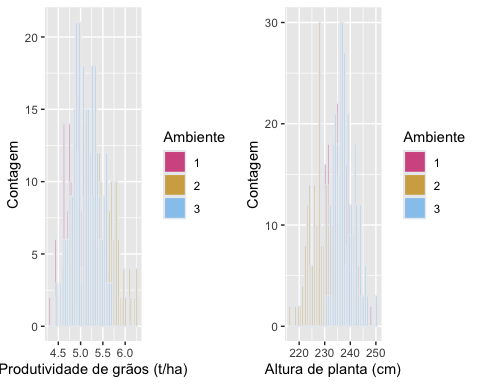
## Unstructured

ntraits <- 2 # Number of traits.  
nenvs <- 3 # Number of environments.  
nreps <- c(2, 2, 3) # Number of replicates tested within environments 1, 2 and 3.  
  
  
nind <- 20 # Number of founder genotypes in the population.  
nchr <- 10 # Number of chromosomes.  
nseg\_sites <- 200 # Number of QTN per chromosome.  
  
mean <- c(4.9, 5.4, 5.1, 235.2, 228.5, 239.1) # c(Yld:E1, Yld:E2, Yld:E3, Prt:E1, Prt:E2, Prt:E3)  
  
var <- c(0.085, 0.12, 0.06, 15.1, 8.5, 11.7) # c(Yld:E1, Yld:E2, Yld:E3, Pht:E1, Pht:E2, Pht:E3)  
  
meanDD <- c(0.4, 0.4, 0.4, 0.1, 0.1, 0.1) # c(Yld:E1, Yld:E2, Yld:E3, Pht:E1, Pht:E2, Pht:E3)  
varDD <- 0.2 # Same value set for all environment-within-trait combinations  
  
TcorA <- matrix( # Matrix of additive genetic correlations between the two traits.  
 c(  
 1.0, 0.6,  
 0.6, 1.0  
 ),  
 ncol = 2  
)  
  
EcorA <- matrix(  
 c( # Matrix of additive genetic correlations between the three environments.  
 1.0, 0.4, 0.6,  
 0.4, 1.0, 0.5,  
 0.6, 0.5, 1.0  
 ),  
 ncol = 3  
)  
  
corA <- rand\_cor\_mat( # Additive genetic correlation structure.  
 (ntraits \* nenvs), # Could be used instead of TcorA and EcorA.  
 min.cor = 0.1,  
 max.cor = 0.9,  
 pos.def = TRUE  
)  
  
round(corA, 2)  
  
corDD <- diag(6)  
  
input\_asr <- unstr\_asr\_input(  
 ntraits = ntraits,  
 nenvs = nenvs,  
 mean = mean,  
 var = var,  
 TcorA = TcorA,  
 EcorA = EcorA,  
 meanDD = meanDD,  
 varDD = varDD,  
 corDD = corDD  
)  
  
founders <- runMacs( # Simulation of founder genotypes using AlphaSimR's "MAIZE" presets  
 nInd = nind, # to mimic the species' evolutionary history.  
 nChr = nchr,  
 segSites = nseg\_sites,  
 inbred = FALSE,  
 species = "MAIZE",  
 nThreads = 2  
)  
  
SP <- SimParam$new(founders)  
  
SP$addTraitAD( # Additive + dominance trait simulation.  
 nQtlPerChr = nseg\_sites,  
 mean = input\_asr$mean,  
 var = input\_asr$var,  
 corA = input\_asr$corA,  
 meanDD = input\_asr$meanDD,  
 varDD = input\_asr$varDD,  
 corDD = input\_asr$corDD,  
 useVarA = FALSE  
)  
  
founders <- newPop(founders)  
  
pool\_A <- makeDH(founders[1:10], nDH = 1) # Pool A: 1 DH line from founders 1 to 10, respectively.  
pool\_B <- makeDH(founders[11:20], nDH = 1) # Pool B: 1 DH line from founders 11 to 20, respectively.  
  
dh\_lines <- mergePops(list(pool\_A, pool\_B))  
  
factorial\_plan <- as.matrix(expand.grid(A = pool\_A@id, B = pool\_B@id)) # Factorial crossing plan.  
  
hybrid\_pop <- makeCross(pop = dh\_lines, crossPlan = factorial\_plan, nProgeny = 1) # Hybrid genotypes.  
  
gv\_df\_us <- unstr\_asr\_output(  
 pop = hybrid\_pop,  
 ntraits = ntraits,  
 nenvs = nenvs,  
 nreps = nreps  
)

gv\_df\_us=readRDS("gv\_df\_us.rds")  
  
  
#  
P1us=ggplot(gv\_df\_us, aes(x = gv.Trait1, fill = factor(env))) +  
 geom\_histogram(color = "#e9ecef", alpha = 0.8, position = "identity", bins = 50) +  
 scale\_fill\_manual(values = c("violetred3", "goldenrod3", "skyblue2")) +  
 labs(x = "Produtividade de grãos (t/ha)", y = "Contagem", fill = "Ambiente")  
  
P2us=ggplot(gv\_df\_us, aes(x = gv.Trait2, fill = factor(env))) +  
 geom\_histogram(color = "#e9ecef", alpha = 0.8, position = "identity", bins = 50) +  
 scale\_fill\_manual(values = c("violetred3", "goldenrod3", "skyblue2")) +  
 labs(x = "Altura de planta (cm)", y = "Contagem", fill = "Ambiente")  
  
  
Plotsus=ggarrange(P1us,P2us)  
ggsave(plot = Plotsus, filename = "usmodel.png",device = "png",dpi = "retina")

## Saving 5 x 4 in image

plot(Plotsus)



saveRDS(gv\_df\_us, file = "gv\_df\_us.rds")

# Applying Selection Index

## prediction

# select genotypes CS

#gv\_df=readRDS("gv\_df.rds")  
geno = gv\_df |> group\_by(env, rep,id) |> mutate(rank.gy =ifelse(gv.Trait1>mean(gv\_df[which(gv\_df$env==cur\_group()$env),"gv.Trait1"]), 1,0)) |> mutate(rank.ph =ifelse(gv.Trait2>mean(gv\_df[which(gv\_df$env==cur\_group()$env),"gv.Trait2"]), 1,0)) |> ungroup()   
  
  
  
ind.sel =geno |> group\_by(id,rep) |> mutate(avg=sum(rank.gy,rank.ph)) |> ungroup()  
#gv\_df |> filter(id=="41") |> View()  
genos = ind.sel |> pivot\_wider(values\_from = c(gv.Trait1,gv.Trait2), names\_from = c(env) ) |> as.data.frame()

## classification

## CS model

library(xgboost)

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

genos = genos[,-c(1:4)]  
  
  
#geno$id = as.numeric(geno$id)  
  
genos$avg = as.factor(genos$avg)  
  
  
  
# Convert the Species factor to an integer class starting at 0  
# This is picky, but it's a requirement for XGBoost  
avg = genos$avg  
  
label = as.integer(genos$avg)-1  
genos$avg = NULL  
  
n = nrow(genos)  
train.index = sample(n,floor(0.75\*n))  
train.data = as.matrix(genos[train.index,])  
train.label = label[train.index]  
test.data = as.matrix(genos[-train.index,])  
test.label = label[-train.index]  
parallel::detectCores()  
#nthread <- (parallel::detectCores())-1  
# Transform the two data sets into xgb.Matrix  
xgb.train = xgb.DMatrix(data=train.data,label=train.label)  
xgb.test = xgb.DMatrix(data=test.data,label=test.label)  
xgb.test  
# Define the parameters for multinomial classification  
  
num\_class = length(levels(avg))  
params = list(  
 booster="gbtree",  
 eta=0.001,  
 max\_depth=5,  
 gamma=3,  
 subsample=0.75,  
 colsample\_bytree=1,  
 objective="multi:softprob",  
 eval\_metric="mlogloss",  
 num\_class=num\_class  
)  
  
# Train the XGBoost classifer  
xgb.fit=xgb.train(  
 params=params,  
 data=xgb.train,  
 nrounds=10000,  
 nthreads=1,  
 early\_stopping\_rounds=10,  
 watchlist=list(val1=xgb.train,val2=xgb.test),  
 verbose=0  
)  
  
  
# Review the final model and results  
xgb.fit  
  
# Predict outcomes with the test data  
xgb.pred = predict(xgb.fit,test.data,reshape=T)  
xgb.pred = as.data.frame(xgb.pred)  
colnames(xgb.pred) = levels(avg)  
  
# Use the predicted label with the highest probability  
xgb.pred$prediction = apply(xgb.pred,1,function(x) colnames(xgb.pred)[which.max(x)])  
xgb.pred$label = levels(avg)[test.label+1]

# Calculate the final accuracy

result = sum(xgb.pred$prediction==xgb.pred$label)/nrow(xgb.pred)  
print(paste("Final Accuracy =",sprintf("%1.2f%%", 100\*result)))

## [1] "Final Accuracy = 53.97%"

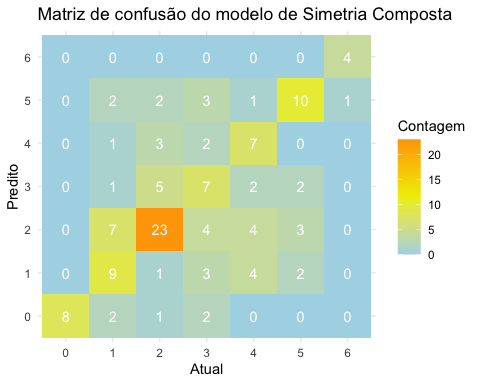
pred=factor(xgb.pred$prediction, levels = c("0","1","2","3","4","5","6") )  
test=factor(xgb.pred$label, levels = c("0","1","2","3","4","5","6"))  
  
  
xtab=table(pred,test)   
str(xtab)

## 'table' int [1:7, 1:7] 8 0 0 0 0 0 0 2 9 7 ...  
## - attr(\*, "dimnames")=List of 2  
## ..$ pred: chr [1:7] "0" "1" "2" "3" ...  
## ..$ test: chr [1:7] "0" "1" "2" "3" ...

# Convert the confusion matrix to a long format  
xtab\_melt <- as.data.frame(as.table(xtab))  
  
# Rename columns for better understanding  
colnames(xtab\_melt) <- c("Predicted", "Actual", "Count")  
csmat=ggplot(xtab\_melt, aes(x = Actual, y = Predicted, fill = Count)) +  
 geom\_tile() +  
 geom\_text(aes(label = Count), color = "white") +  
 scale\_fill\_gradientn(colors = c("lightblue", "yellow2", "orange")) +  
 theme\_minimal() +  
 labs(title = "Matriz de confusão do modelo de Simetria Composta", x = "Atual", y = "Predito", fill="Contagem")  
  
  
cscm=confusionMatrix(xtab)   
print(cscm)

## Confusion Matrix and Statistics  
##   
## test  
## pred 0 1 2 3 4 5 6  
## 0 8 2 1 2 0 0 0  
## 1 0 9 1 3 4 2 0  
## 2 0 7 23 4 4 3 0  
## 3 0 1 5 7 2 2 0  
## 4 0 1 3 2 7 0 0  
## 5 0 2 2 3 1 10 1  
## 6 0 0 0 0 0 0 4  
##   
## Overall Statistics  
##   
## Accuracy : 0.5397   
## 95% CI : (0.4486, 0.6288)  
## No Information Rate : 0.2778   
## P-Value [Acc > NIR] : 5.63e-10   
##   
## Kappa : 0.4372   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 1.00000 0.40909 0.6571 0.33333 0.38889 0.58824  
## Specificity 0.95763 0.90385 0.8022 0.90476 0.94444 0.91743  
## Pos Pred Value 0.61538 0.47368 0.5610 0.41176 0.53846 0.52632  
## Neg Pred Value 1.00000 0.87850 0.8588 0.87156 0.90265 0.93458  
## Prevalence 0.06349 0.17460 0.2778 0.16667 0.14286 0.13492  
## Detection Rate 0.06349 0.07143 0.1825 0.05556 0.05556 0.07937  
## Detection Prevalence 0.10317 0.15079 0.3254 0.13492 0.10317 0.15079  
## Balanced Accuracy 0.97881 0.65647 0.7297 0.61905 0.66667 0.75283  
## Class: 6  
## Sensitivity 0.80000  
## Specificity 1.00000  
## Pos Pred Value 1.00000  
## Neg Pred Value 0.99180  
## Prevalence 0.03968  
## Detection Rate 0.03175  
## Detection Prevalence 0.03175  
## Balanced Accuracy 0.90000

plot(csmat)



ggsave(plot = csmat, filename = "confusionmatrix\_modelcs.png",device = "png",dpi = "retina")

## Saving 5 x 4 in image

# select genotypes US

gv\_df\_us=readRDS("gv\_df\_us.rds")  
  
geno\_us = gv\_df\_us |> group\_by(env, rep,id) |> mutate(rank.gy =ifelse(gv.Trait1>mean(gv\_df\_us[which(gv\_df\_us$env==cur\_group()$env),"gv.Trait1"]), 1,0)) |> mutate(rank.ph =ifelse(gv.Trait2>mean(gv\_df\_us[which(gv\_df\_us$env==cur\_group()$env),"gv.Trait2"]), 1,0)) |> ungroup()   
  
ind.sel =geno\_us |> group\_by(id,rep) |> mutate(avg=sum(rank.gy,rank.ph)) |> ungroup()  
  
genos\_us = ind.sel |> pivot\_wider(values\_from = c(gv.Trait1,gv.Trait2), names\_from = c(env) ) |> as.data.frame()

# classification

## US model

##classification example US  
  
genos = genos\_us[,-c(1:4)]  
  
#geno$id = as.numeric(geno$id)  
  
genos$avg = as.factor(genos$avg)  
  
# Convert the Species factor to an integer class starting at 0  
# This is picky, but it's a requirement for XGBoost  
avg = genos$avg  
  
label = as.integer(genos$avg)-1  
genos$avg = NULL  
  
n = nrow(genos)  
train.index = sample(n,floor(0.75\*n))  
train.data = as.matrix(genos[train.index,])  
train.label = label[train.index]  
test.data = as.matrix(genos[-train.index,])  
test.label = label[-train.index]  
  
# Transform the two data sets into xgb.Matrix  
xgb.train = xgb.DMatrix(data=train.data,label=train.label)  
xgb.test = xgb.DMatrix(data=test.data,label=test.label)  
xgb.test  
# Define the parameters for multinomial classification  
num\_class = length(levels(avg))  
params = list(  
 booster="gbtree",  
 eta=0.001,  
 max\_depth=5,  
 gamma=3,  
 subsample=0.75,  
 colsample\_bytree=1,  
 objective="multi:softprob",  
 eval\_metric="mlogloss",  
 num\_class=num\_class  
)  
  
# Train the XGBoost classifer  
xgb.fit=xgb.train(  
 params=params,  
 data=xgb.train,  
 nrounds=10000,  
 nthreads=1,  
 early\_stopping\_rounds=10,  
 watchlist=list(val1=xgb.train,val2=xgb.test),  
 verbose=0  
)  
  
# Review the final model and results  
xgb.fit  
  
# Predict outcomes with the test data  
xgb.pred = predict(xgb.fit,test.data,reshape=T)  
xgb.pred = as.data.frame(xgb.pred)  
colnames(xgb.pred) = levels(avg)  
  
# Use the predicted label with the highest probability  
xgb.pred$prediction = apply(xgb.pred,1,function(x) colnames(xgb.pred)[which.max(x)])  
xgb.pred$label = levels(avg)[test.label+1]

# Calculate the final accuracy

result\_us = sum(xgb.pred$prediction==xgb.pred$label)/nrow(xgb.pred)  
print(paste("Final Accuracy =",sprintf("%1.2f%%", 100\*result\_us)))

## [1] "Final Accuracy = 62.71%"

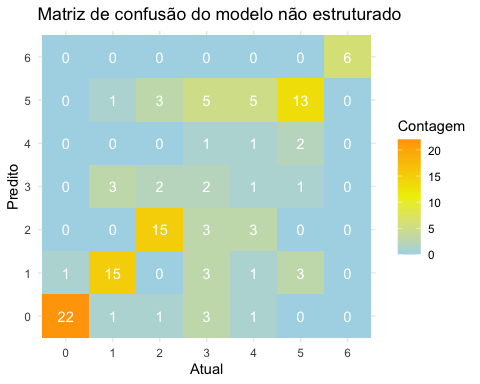
pred=factor(xgb.pred$prediction, levels = c("0","1","2","3","4","5","6") )  
test=factor(xgb.pred$label, levels = c("0","1","2","3","4","5","6"))  
  
  
xtab=table(pred,test)  
  
uscm=confusionMatrix(xtab)  
  
# Convert the confusion matrix to a long format  
xtab\_melt <- as.data.frame(as.table(xtab))  
  
# Rename columns for better understanding  
colnames(xtab\_melt) <- c("Predicted", "Actual", "Count")  
  
usmat=ggplot(xtab\_melt, aes(x = Actual, y = Predicted, fill = Count)) +  
 geom\_tile() +  
 geom\_text(aes(label = Count), color = "white") +  
 scale\_fill\_gradientn(colors = c("lightblue", "yellow2", "orange")) +  
 theme\_minimal() +  
 labs(title = "Matriz de confusão do modelo não estruturado", x = "Atual", y = "Predito", fill="Contagem")  
  
ggsave(plot = usmat, filename = "confusionmat\_usmodel.png",device = "png",dpi = "retina")

## Saving 5 x 4 in image

print(uscm)

## Confusion Matrix and Statistics  
##   
## test  
## pred 0 1 2 3 4 5 6  
## 0 22 1 1 3 1 0 0  
## 1 1 15 0 3 1 3 0  
## 2 0 0 15 3 3 0 0  
## 3 0 3 2 2 1 1 0  
## 4 0 0 0 1 1 2 0  
## 5 0 1 3 5 5 13 0  
## 6 0 0 0 0 0 0 6  
##   
## Overall Statistics  
##   
## Accuracy : 0.6271   
## 95% CI : (0.5333, 0.7144)  
## No Information Rate : 0.1949   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5535   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.9565 0.7500 0.7143 0.11765 0.083333 0.6842  
## Specificity 0.9368 0.9184 0.9381 0.93069 0.971698 0.8586  
## Pos Pred Value 0.7857 0.6522 0.7143 0.22222 0.250000 0.4815  
## Neg Pred Value 0.9889 0.9474 0.9381 0.86239 0.903509 0.9341  
## Prevalence 0.1949 0.1695 0.1780 0.14407 0.101695 0.1610  
## Detection Rate 0.1864 0.1271 0.1271 0.01695 0.008475 0.1102  
## Detection Prevalence 0.2373 0.1949 0.1780 0.07627 0.033898 0.2288  
## Balanced Accuracy 0.9467 0.8342 0.8262 0.52417 0.527516 0.7714  
## Class: 6  
## Sensitivity 1.00000  
## Specificity 1.00000  
## Pos Pred Value 1.00000  
## Neg Pred Value 1.00000  
## Prevalence 0.05085  
## Detection Rate 0.05085  
## Detection Prevalence 0.05085  
## Balanced Accuracy 1.00000

plot(usmat)



#plots

head(gv\_df)

## env rep id gv.Trait1 gv.Trait2  
## 1 1 1 41 4.674634 232.1747  
## 2 1 1 42 4.694956 235.4380  
## 3 1 1 43 4.846733 236.3927  
## 4 1 1 44 4.558753 235.2459  
## 5 1 1 45 4.751661 240.3850  
## 6 1 1 46 4.820861 233.1440

head(gv\_df\_us)

## env rep id gv.Trait1 gv.Trait2  
## 1 1 1 41 5.142452 247.8281  
## 2 1 1 42 5.005393 241.9710  
## 3 1 1 43 5.046226 242.3506  
## 4 1 1 44 5.343467 243.7550  
## 5 1 1 45 5.382726 242.9141  
## 6 1 1 46 5.143707 237.8012

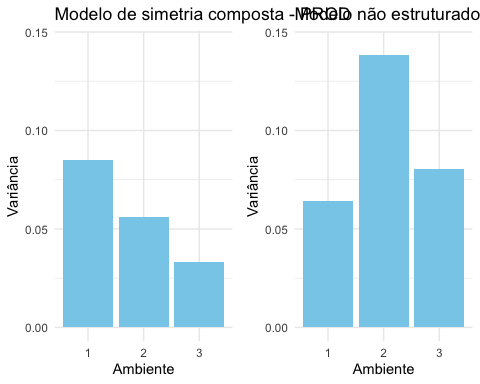
var(gv\_df\_us$gv.Trait1)

## [1] 0.1333394

var(gv\_df$gv.Trait1)

## [1] 0.1003251

# Summarize data to calculate variance for each group  
gvdf=gv\_df |> group\_by(env) |> summarise(variance =var(gv.Trait1)) |> ggplot() + geom\_col(aes(x=env, y=variance),fill="skyblue") + theme\_minimal() +  
 labs(title = "Modelo de simetria composta - PROD", x = "Ambiente", y = "Variância") +  
 ylim(0, var(gv\_df\_us$gv.Trait1)+0.01)  
  
gvdfus=gv\_df\_us |> group\_by(env) |> summarise(variance =var(gv.Trait1)) |> ggplot() + geom\_col(aes(x=env, y=variance),fill="skyblue") + theme\_minimal() +  
 labs(title = "Modelo não estruturado - PROD", x = "Ambiente", y = "Variância") +  
 ylim(0, var(gv\_df\_us$gv.Trait1)+0.01)  
  
plot1=ggarrange(gvdf,gvdfus)  
plot(plot1)



ggsave(plot = plot1, filename = "plotvartrais1.png",device = "png",dpi = "retina")

## Saving 5 x 4 in image

# Summarize data to calculate variance for each group  
gvdf2=gv\_df |> group\_by(env) |> summarise(variance =var(gv.Trait2)) |> ggplot() + geom\_col(aes(x=env, y=variance),fill="skyblue") + theme\_minimal() +  
 labs(title = "Modelo de simetria composta - AP", x = "Ambiente", y = "Variância") +  
 ylim(0, var(gv\_df\_us$gv.Trait2)-19)  
  
gvdfus2=gv\_df\_us |> group\_by(env) |> summarise(variance =var(gv.Trait2)) |> ggplot() + geom\_col(aes(x=env, y=variance),fill="skyblue") + theme\_minimal() +  
 labs(title = "Modelo não estruturado - AP", x = "Ambiente", y = "Variância") +  
 ylim(0, var(gv\_df\_us$gv.Trait2)-19)  
  
plot2=ggarrange(gvdf2,gvdfus2)  
  
ggsave(plot = plot2, filename = "plotvartrais.png",device = "png",dpi = "retina")

## Saving 5 x 4 in image

plot(plot2)

