Sim\_ABCRF

LEJEUNE Thomas

2025-04-29

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  
## ✔ ggplot2 3.5.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.4   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(abcrf)  
library(ggplot2)  
library(tinytex)

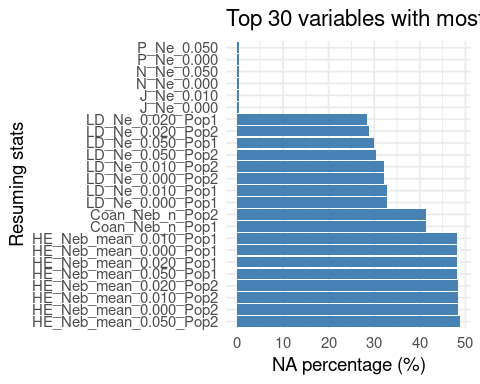
# Desktop detection  
home\_path <- Sys.getenv("HOME")  
  
desktop\_path <- if (dir.exists(file.path(home\_path, "Desktop"))) {  
 file.path(home\_path, "Desktop")  
} else if (dir.exists(file.path(home\_path, "Bureau"))) {  
 file.path(home\_path, "Bureau")  
} else {  
 stop  
}  
  
# Complete path  
file\_path <- file.path(desktop\_path, "simulations", "Ref\_table", "summary\_table.csv")  
  
# Reading file  
data <- read\_csv(file\_path)

## Rows: 13000 Columns: 99  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (19): simulation\_id, Coan\_f1\_Pop1, Coan\_f1\_Pop2, HE\_weighted\_D\_mean\_0.05...  
## dbl (76): pop\_size, num\_loci, sample1\_generation, sample2\_generation, sample...  
## num (4): HE\_weighted\_D\_mean\_0.050\_Pop1, HE\_weighted\_D\_mean\_0.020\_Pop1, HE\_w...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Function that will search every column and count the "NA", then remove the columns having too much missing values according to the threshold  
remove\_high\_na\_cols <- function(data, threshold = 0.05) {  
 total\_rows <- nrow(data)  
 na\_summary <- data %>%  
 summarise(across(everything(), ~ sum(is.na(.)))) %>%  
 pivot\_longer(cols = everything(), names\_to = "Variable", values\_to = "NA\_count") %>%  
 mutate(  
 Total\_rows = total\_rows,  
 NA\_percentage = round((NA\_count / Total\_rows) \* 100, 2)  
 )  
 cols\_to\_remove <- na\_summary %>%  
 filter(NA\_percentage > threshold \* 100) %>%  
 pull(Variable)  
 data\_clean <- data %>% select(-any\_of(cols\_to\_remove))  
 list(data\_clean = data\_clean, removed = cols\_to\_remove, na\_summary = na\_summary)  
}  
result <- remove\_high\_na\_cols(data, threshold = 0.05)  
data\_clean <- result$data   
result$na\_summary # List of removed col

## # A tibble: 99 × 4  
## Variable NA\_count Total\_rows NA\_percentage  
## <chr> <int> <int> <dbl>  
## 1 simulation\_id 0 13000 0  
## 2 pop\_size 0 13000 0  
## 3 num\_loci 0 13000 0  
## 4 sample1\_generation 0 13000 0  
## 5 sample2\_generation 0 13000 0  
## 6 sample1\_size\_Ne 0 13000 0  
## 7 sample2\_size\_Ne 0 13000 0  
## 8 sample1\_size\_CMR 0 13000 0  
## 9 sample2\_size\_CMR 0 13000 0  
## 10 low\_repeats 0 13000 0  
## # ℹ 89 more rows

top\_n\_display <- 20  
  
ggplot(result$na\_summary %>%   
 filter(NA\_percentage > 0) %>%   
 slice\_max(NA\_percentage, n = top\_n\_display),  
 aes(x = reorder(Variable, -NA\_percentage), y = NA\_percentage)) +  
 geom\_col(fill = "steelblue") +  
 coord\_flip() +  
 labs(  
 title = "Top 30 variables with most NA",  
 x = "Resuming stats",  
 y = "NA percentage (%)"  
 ) +  
 theme\_minimal(base\_size = 14)



data\_clean <- data\_clean %>%  
 mutate(census\_N = ifelse(census\_N == 0, NA, census\_N))  
  
data\_clean <- data\_clean %>%  
 drop\_na()

# Search every column for NA then calculate the median of the existing values and add it to the missing values  
data\_imputed <- data %>% mutate(across(everything(), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))

## Warning: There were 5 warnings in `mutate()`.  
## The first warning was:  
## ℹ In argument: `across(...)`.  
## Caused by warning in `mean.default()`:  
## ! l'argument n'est ni numérique, ni logique : renvoi de NA  
## ℹ Run `dplyr::last\_dplyr\_warnings()` to see the 4 remaining warnings.

# Hard version of the last method, choose the columns to delete to avoid suppressing too many lines while removing the NAs  
to\_remove <- c(  
 "HE\_Neb\_mean\_0.050\_Pop2", "HE\_Neb\_mean\_0.020\_Pop2", "HE\_Neb\_mean\_0.010\_Pop2", "HE\_Neb\_mean\_0.000\_Pop2",  
 "HE\_Neb\_mean\_0.050\_Pop1", "HE\_Neb\_mean\_0.020\_Pop1", "HE\_Neb\_mean\_0.010\_Pop1", "HE\_Neb\_mean\_0.000\_Pop1",  
 "Coan\_Neb\_n\_Pop1", "Coan\_Neb\_n\_Pop2",  
 "LD\_Ne\_0.050\_Pop1", "LD\_Ne\_0.020\_Pop1", "LD\_Ne\_0.010\_Pop1", "LD\_Ne\_0.000\_Pop1",  
 "LD\_Ne\_0.050\_Pop2", "LD\_Ne\_0.020\_Pop2", "LD\_Ne\_0.010\_Pop2", "LD\_Ne\_0.000\_Pop2"  
)  
  
# Remove the lines that contain at least one "NA"  
data <- data %>% select(-any\_of(to\_remove)) %>% na.omit()

# List of parameters  
param\_cols <- c("simulation\_id", "pop\_size", "num\_loci", "sample1\_size\_Ne", "sample2\_size\_Ne",  
 "sample1\_size\_CMR", "sample2\_size\_CMR", "mutation\_rate", "recap\_Ne")  
params <- data\_clean %>% select(all\_of(param\_cols))  
  
# List of resuming statistics  
stat\_keywords <- c("id", "LD", "HE", "Coan", "het", "alleles", "P\_", "N\_", "J\_")  
stat\_cols <- names(data\_clean)[sapply(names(data\_clean), function(col) any(str\_detect(col, stat\_keywords)))]  
stats\_table <- data\_clean %>% select(all\_of(stat\_cols))

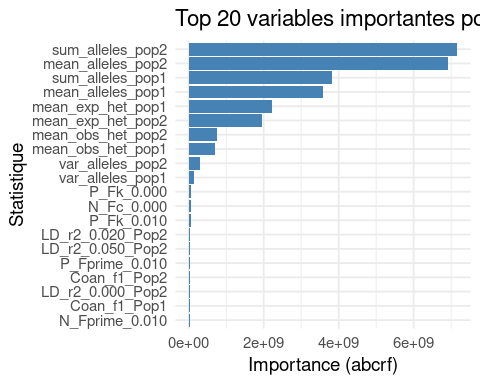
# Choose the parameter that will be predicted and create a new dataframe with it in the first column and the resuming stats  
target\_param <- "pop\_size"  
learning\_data <- bind\_cols(y = params[[target\_param]], stats\_table) %>% rename(!!target\_param := y)  
  
# Regression ABCRF that will predict the values of a parameters from the data base formed by the simulations  
model\_rf <- regAbcrf(as.formula(paste(target\_param, "~ .")), data = learning\_data, ntree = 500)

## Growing trees.. Progress: 42%. Estimated remaining time: 42 seconds.  
## Growing trees.. Progress: 84%. Estimated remaining time: 11 seconds.

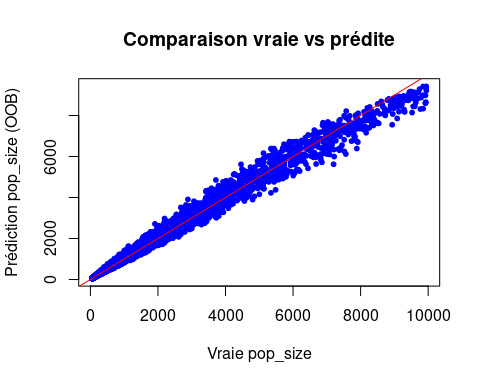
summary(model\_rf)

## Length Class Mode  
## call 4 -none- call  
## formula 3 formula call  
## model.rf 19 ranger list

# Determines the order of importance of every resuming stat in the dataframe concerning the prediction of the parameter  
importance\_df <- model\_rf$model.rf$variable.importance %>%  
 sort(decreasing = TRUE) %>%  
 enframe(name = "Statistic", value = "Importance")  
top\_n <- 20  
importance\_top <- importance\_df %>%  
 slice\_head(n = top\_n) %>%  
 mutate(Statistic = fct\_reorder(Statistic, Importance))  
  
# Plotting the barplot showing the order of importance  
ggplot(importance\_top, aes(x = Statistic, y = Importance)) +  
 geom\_col(fill = "steelblue") +  
 coord\_flip() +  
 labs(  
 title = paste("Top", top\_n, "variables importantes pour l'estimation de", target\_param),  
 x = "Statistique",  
 y = "Importance (abcrf)"  
 ) +  
 theme\_minimal(base\_size = 14)



# Use the model to make the predictions according to the "observed data" from the Out-Of-Bag "OOB"  
predictions <- predict(model\_rf, training = learning\_data, obs = learning\_data)$expectation  
  
# Plotting the curve demonstrating the differences of the predictions of the dataframe studied with the target data from the Out-Of-Bag  
plot(learning\_data[[target\_param]], predictions,  
 xlab = "Vraie pop\_size", ylab = "Prédiction pop\_size (OOB)",  
 main = "Comparaison vraie vs prédite",  
 pch = 20, col = "blue")  
abline(0, 1, col = "red")



# ---- Courbe POD vs tolérance ----  
  
# Vecteur de tolérances (exprimées en %)  
tolerance\_vec <- seq(0.01, 0.2, by = 0.01) # de 1% à 20%  
  
# Calcul de la POD pour chaque tolérance  
pod\_df <- data.frame(  
 Tolerance = tolerance\_vec,  
 POD = sapply(tolerance\_vec, function(tol) {  
 mean(abs(predictions - learning\_data$pop\_size) / learning\_data$pop\_size < tol)  
 })  
)  
  
# Affichage du tableau si besoin  
print(pod\_df)

## Tolerance POD  
## 1 0.01 0.0960322  
## 2 0.02 0.1864098  
## 3 0.03 0.2759249  
## 4 0.04 0.3610312  
## 5 0.05 0.4479586  
## 6 0.06 0.5239601  
## 7 0.07 0.5983324  
## 8 0.08 0.6587119  
## 9 0.09 0.7100824  
## 10 0.10 0.7559900  
## 11 0.11 0.7975848  
## 12 0.12 0.8295956  
## 13 0.13 0.8591144  
## 14 0.14 0.8819245  
## 15 0.15 0.9028177  
## 16 0.16 0.9200690  
## 17 0.17 0.9347326  
## 18 0.18 0.9458501  
## 19 0.19 0.9547633  
## 20 0.20 0.9623347

# ---- Visualisation avec ggplot2 ----  
library(ggplot2)  
  
ggplot(pod\_df, aes(x = Tolerance \* 100, y = POD \* 100)) +  
 geom\_line(color = "steelblue", size = 1.2) +  
 geom\_point(color = "darkblue", size = 2) +  
 labs(  
 title = "Probabilty of Detection (POD)",  
 x = "Tolerance (%)",  
 y = "POD (%)"  
 ) +  
 theme\_minimal(base\_size = 14)

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

