Overview of changed code

This document will illustrate how much of the code is simply copied from your code, in green. Code from the standard CHAP framework in blue. The extra code to write will be in red. Note that some of the code in green has been slightly altered from your original code, for instance using helper functions from utils.R instead of some functions from libraries, but it should be equivalent, and could also have stayed as it was.

**train.R**

source("utils.R")

library(dplyr)

library(fable)

library(tsibble)

library(lubridate)

train\_chap <- function(csv\_fn, model\_fn) {

dataframe\_list <- get\_df\_per\_location(csv\_fn)

models <- list()

for (location in names(dataframe\_list)){

model <- train\_single\_region(dataframe\_list[[location]], location)

models[[location]] <- model

}

saveRDS(models, file=model\_fn)

}

train\_single\_region <- function(df, location){

df <- mutate(df, date = yearmonth(date)) |> #so tsibble understands it is monthly data, fails with exact date

create\_lagged\_feature("rain\_mm", 3, include\_all = FALSE) |>

create\_lagged\_feature("temp\_c", 3, include\_all = FALSE) |>

cut\_top\_rows(3)

df\_tsibble <- as\_tsibble(df, index = date)

if ("net\_time" %in% colnames(df)){

model <- df\_tsibble |>

model(

ARIMA(disease\_cases ~ rainfall\_3 + mean\_temperature\_3 + net\_time)

)

} else {

model <- df\_tsibble |>

model(

ARIMA(disease\_cases ~ rainfall\_3 + mean\_temperature\_3)

)

}

return(model)

}

args <- commandArgs(trailingOnly = TRUE)

if (length(args) == 2) {

csv\_fn <- args[1]

model\_fn <- args[2]

train\_chap(csv\_fn, model\_fn)

}# else {

# stop("Usage: Rscript train.R <csv\_fn> <model\_fn>")

#}

**predict.R**

source("utils.R")

library(dplyr)

library(fable)

library(tsibble)

library(lubridate)

library(distributional) #to extract info from dist objects

predict\_chap <- function(model\_fn, historic\_data\_fn, future\_climatedata\_fn, predictions\_fn) {

future\_per\_location <- get\_df\_per\_location(future\_climatedata\_fn)

historic\_per\_location <- get\_df\_per\_location(historic\_data\_fn)

models <- readRDS(model\_fn) # Assumes the model was saved using saveRDS

first\_location <- TRUE

for (location in names(future\_per\_location)){

df <- future\_per\_location[[location]]

historic\_df <- historic\_per\_location[[location]]

model <- models[[location]]

df$disease\_cases <- NA #so the dataframes have the same columns

tot\_tible <- rbind(historic\_df, df) |> #row-bind them together

mutate(time\_period = yearmonth(time\_period)) |>

create\_lagged\_feature("rainfall", 3, include\_all = FALSE) |>

create\_lagged\_feature("mean\_temperature", 3, include\_all = FALSE) |>

as\_tsibble(index=time\_period)

historic\_tible = tot\_tible[1:nrow(historic\_df),]

future\_tible <- tot\_tible[(nrow(historic\_df) + 1): nrow(tot\_tible),]

model = refit(model, historic\_tible)

predicted\_dists <- forecast(model, new\_data = future\_tible)

n\_samples <- 100

preds <- data.frame(matrix(ncol = n\_samples, nrow = nrow(future\_tible)))

colnames(preds) <- paste("sample", 0:(n\_samples-1), sep = "\_")

for(i in 1:nrow(future\_tible)){

dist <- predicted\_dists[i, "disease\_cases"]$disease\_cases

preds[i,] <- rnorm(n\_samples, mean = mean(dist), sd = sqrt(variance(dist)))

}

sample\_df <- cbind(df, preds)

if (first\_location){

full\_df <- sample\_df

first\_location <- FALSE

}

else {

full\_df <- rbind(full\_df, sample\_df)

}

#print(paste("Forecasted values:", paste(df[, "sample\_0", drop=TRUE], collapse = ", ")))

}

full\_df["time\_period"] <- df["time\_period"]

write.csv(full\_df, predictions\_fn, row.names = FALSE)

}

args <- commandArgs(trailingOnly = TRUE)

if (length(args) == 4) {

model\_fn <- args[1]

historic\_data\_fn <- args[2]

future\_climatedata\_fn <- args[3]

predictions\_fn <- args[4]

predict\_chap(model\_fn, historic\_data\_fn, future\_climatedata\_fn, predictions\_fn)

}

**isolated\_run.R**

source("train.R")

source("predict.R")

train\_chap("input/trainData.csv", "output/model.bin")

predict\_chap("output/model.bin", "input/trainData.csv", "input/futureClimateData.csv", "output/predictions.csv")

**dataPreperation.R**

This is individual for each model and situation, but when needed it is generally straight forward.

**utils.R**

This is just some standard helper functions. Use them if you want, and you can of course add additional functions as well. But this file can safely be ignored altogether.

**The Docker Image**

The Docker Image is created in the docker\_for-MadagascarArima repository. Here you only need to change the name in the build.yml file and add the necessary libraries in the Dockerfile. If all the packages are available with the standard *install.packages(“library”)* this is straight forward, and it is advised to build it on an existing docker image, like r-base or one we have made for a different model, as they already include many common libraries. Further information can be found in the repositories docker\_r\_template and docker\_for-MadagascarArima.

**MLproject**

name: Madagascar\_ARIMA

docker\_env:

  image: ghcr.io/dhis2-chap/docker\_for\_madagascararima:master

entry\_points:

  train:

    parameters:

      train\_data: path

      model: str

    command: "Rscript train.R {train\_data} {model}"

  predict:

    parameters:

      historic\_data: path

      future\_data: path

      model: str

      out\_file: path

    command: "Rscript predict.R {model} {historic\_data} {future\_data} {out\_file}"