Code used for the Development of the Xgboost algorithm

“AUC_target” = the name of the column of the explained variable (Y)
“data” = the name of the dataset
“C_X” = the concentration measured at time X in mg/L
“MODN” = the categorical covariate for simulated population (from 1 to 6)
“SCR” = serum creatinine in µM
“C_diff”= the measured difference between the peak and trough in mg/L,
“TBW” = the body weight in Kg
“AGE” = age in years

R code

```
### Data splitting
set.seed(123)
data_split = initial_split(data, strata = AUC_target)
data_train = training(data_split)
data_test = testing(data_split)

### Preprocessing
data_recipe  <- recipe(AUC_target ~ C_X + MODN +TBW +SEX+AGE+SCR + C_diff,  data =
data_train)  %>%
  update_role(MODN, new_role = “simulated population”)

data_prep <-  prep(data_recipe)
 data_train1 <-juice(data_prep)
 data_test1 <-bake(data_prep, new_data = data_test)

### Definition of used model and hyperparameters to tune
xgb_spec <- boost_tree(mode = "regression",
                       mtry = tune(),
                       trees = 1000
                       min_n = tune(),
                       tree_depth = tune(),
                       learn_rate = tune()) %>% set_engine("xgboost")

## Creation of workflow, add of recipe and model
xgb_wf <- workflow() %>%
   add_recipe(data_recipe) %>%
   add_model(xgb_spec)
```

```
### Cross validation hyperparameters
set.seed(2345)
 data_folds <- vfold_cv(data_train)

###Tuning of hyperparameters
set.seed(345)
 tune_xgb<- tune_grid(
 xgb_wf,
 resamples = data_folds,
 finalize(mtry(), data_train1),
 grid = 30 )

### Visualisation of results plot as function of rmse
autoplot (tune_xgb, metric = "rmse",  scientific = FALSE) +
  ggtitle("tuning hyperparameter")

###Choice of best model
best_rmse_xgb <- select_best(tune_xgb, "rmse")

### Finalization of model with the best hyperparameters
final_xgb<- finalize_model(xgb_spec,
   best_rmse_xgb)

### Creation of Variables Importance Plot
final_xgb %>%
   set_engine("xgboost") %>%
   fit(AUC_target ~ .,
       data = juice(data_train)
   ) %>%
   vip::vip(geom = "col")


### Finalization of workflow
final_wf_xgb <- workflow() %>%
   add_recipe(data_recipe) %>%
  add_model(final_xgb)
final_wf_xgb

### Resamples
set.seed( 456 )
folds <- vfold_cv(data_train) #par défaut 10 fois
```

```
### Fit with resampled training dataset
xgb_rs <- fit_resamples(object = final_wf_xgb,
                        resamples = folds,
                        control = control_resamples(verbose=T, save_pred = T))

### Obtain the predicted outcomes
pred_obs_xgb <- xgb_rs %>% collect_predictions() %>%
mutate (bias_rel = (.pred - AUC_target)/AUC_target,
bias_rel_square = bias_rel * bias_rel)
pred_obs_xgb_gen <- pred_obs_xgb %>% summarise(biais_rel = mean(bias_rel),
relative_rmse = sqrt(mean(bias_rel_square)))

### Fit workflow
fit_workflow <- fit(final_wf_xgb, data_train)

### Save fitworkflow
saveRDS(fit_workflow, file = "fit_workflow_07.12.rds")

### Validation in test set
final_fit <- fit_workflow %>%
last_fit(data_split)
final_fit %>% collect_metrics()
final_fit_gen <- final_fit %>% collect_predictions() %>%
mutate (bias_rel = (.pred - AUC_target)/AUC_target,
bias_rel_square = bias_rel * bias_rel)
final_fit_gen %>% summarise(biais_rel = mean(bias_rel), relative_rmse =
sqrt(mean(bias_rel_square)))
```