Age and Performance in Masters Swimming and Running

A Simplified Example

R. De Veaux, A. Plantinga, and E. Upton

2024-06-10

Introduction and data

This document illustrates the bootstrap modeling process. For this simplified walk through, we focus on one event. Note, the data for the entire analysis are available from the websites cited in our manuscript. For this working example, to ease reproducibility, we include the file `swim_1650.RDS` (data from the 1650Y swim).

```
## Load data
swimming <- readRDS("swim_1650.RDS")

swimming<- swimming %>%
  filter( Age %% 5 == 0, 29 < Age, Age < 81) %>%
  mutate(Female = ifelse(Sex == "M", 0, 1))

## Explore data
g <- ggplot(swimming, aes(x = Age, y = TimeSec, color = Sex, shape = Sex)) +
  geom_smooth(se = TRUE, method = 'loess', formula = 'y ~ x') +
  geom_point(alpha = .1)
```

Define the response (ratio to age 35):

```
#Calculating ratio (for response)
swimming_current_m <- swimming %>%
  group_by(Sex)%>%
  filter(Age == 35) %>%
  summarize(mean(TimeSec))

swimming_current_m <- as.matrix(swimming_current_m)

swimming <- swimming  %>%
  mutate(age35 = ifelse(Sex == 0,
  as.numeric(swimming_current_m[1,2]),
  as.numeric(swimming_current_m[2,2]))) %>%
  mutate(Ratio = TimeSec / age35)
```

Define the stacked model:

```
do.one <- function(seed, swim,  nhidden, nepoch, ndeg, spdeg) {

  mAge <- mean(swim$Age)
  sAge <- sd(swim$Age)
  mRatio <- mean(swim$Ratio)
  sRatio <- sd(swim$Ratio)
```

```r
#standardize
this_swim <- swim %>%
  mutate(zAge = (Age - mAge)/sAge,
         zAgeF = zAge * Female,
         zRatio = (Ratio - mRatio)/sRatio)

set.seed(seed)
swim_split <- initial_split(this_swim, prop = 0.8)
swim_train <- training(swim_split)
swim_test <- testing(swim_split)

#metric to evaluate
metric <- metric_set(rmse)
ctrl_res <- control_stack_resamples()

## Must be done AFTER you standardize
folds <- vfold_cv(swim_train, v = 5)

# Define models
#general recipe
swim_rec <-
    recipe(zRatio ~ zAge + Female + zAgeF, data = swim_train)

#neural net
#model definition
nn_spec <- mlp(
  hidden_units = nhidden,
  penalty = 0,
  epochs = nepoch) %>%
  set_engine("nnet") %>%
  set_mode("regression")

#recipe extension (no extension necessary)
nn_rec <-
  swim_rec

#workflow
nn_wflow <-
  workflow() %>%
  add_model(nn_spec) %>%
  add_recipe(nn_rec)

# fit to the 5-fold cv
nn_res <-
  fit_resamples(
    nn_wflow,
    resamples = folds,
    metrics = metric,
    control = ctrl_res
  )

nn.res <- fit(nn_wflow, data = swim_train)
```

```r
#polynomial
#create model definition
lin_reg_spec <-
  linear_reg(penalty = 0.0) %>%
  set_engine("glmnet")

#extend recipe
lin_reg_rec <-
  swim_rec %>%
  step_poly(zAge, degree = ndeg)  %>%
  step_poly(zAgeF, degree = ndeg)

# add both to a workflow
lin_reg_wflow <-
  workflow() %>%
  add_model(lin_reg_spec) %>%
  add_recipe(lin_reg_rec)

# fit to the 5-fold cv
lin_reg_res <-
  fit_resamples(
    lin_reg_wflow,
    resamples = folds,
    metrics = metric,
    control = ctrl_res
  )

lin.res <- fit(lin_reg_wflow, data = swim_train)

## Spline
spline_spec <-
  linear_reg() %>%
  set_engine(engine = 'lm') %>%
  set_mode('regression')

spline_rec <- swim_rec %>%
  step_ns(zAge, deg_free = spdeg) %>%
  step_ns(zAgeF, deg_free = spdeg)

spline_wflow <- workflow() %>%
  add_model(spline_spec) %>%
  add_recipe(spline_rec)

spline_res <- fit_resamples(
  spline_wflow,
  resamples = folds,
  metrics = metric,
  control = ctrl_res
)

spline.res <- fit(spline_wflow, data = swim_train)

## Stacked Model
```

```
  swim_st <-
    stacks() %>%
    add_candidates(lin_reg_res) %>%
    add_candidates(nn_res) %>%
    add_candidates(spline_res)

  st.res <-
    swim_st %>%
    blend_predictions() %>%
    fit_members()

  #keeping track of weights
  lin <- collect_parameters(st.res, "lin_reg_res")
  nn <- collect_parameters(st.res, "nn_res")
  spl <- collect_parameters(st.res, "spline_res")
  weights <- rbind(lin, nn, spl)

  preds <- data.frame(
    NN = predict(nn.res, new_data = swim_test)$`.pred`,
    PL = predict(lin.res, new_data = swim_test)$`.pred`,
    SP = predict(spline.res, new_data = swim_test)$`.pred`,
    ST = predict(st.res, new_data = swim_test)$`.pred`)
  preds$Avg <- apply(preds[,c("NN","PL", "SP")], 1, mean)
    preds$ObsTime = swim_test$zRatio

  ## MSE on scaled data
  pred.rmse <- apply(preds, 2, FUN = function(y) {
    sqrt(sum((preds$ObsTime - y)^2/nrow(preds)))
  })

  ## Set up plotting data
  plot.dat <- expand.grid(Age = 35:84,
                                      Female = 0:1) %>%
    mutate(zAge = (Age - mAge) / sAge, zAgeF = Female*zAge)

  ## Get standard predictions
  plot.preds <- data.frame(
    SP = predict(spline.res, new_data = plot.dat)$`.pred`,
    NN = predict(nn.res, new_data = plot.dat)$`.pred`,
    PL = predict(lin.res, new_data = plot.dat)$`.pred`,
    ST = predict(st.res, new_data = plot.dat)$`.pred`)
    plot.preds <- plot.preds * sRatio + mRatio

  plot.preds$Avg <- apply(plot.preds[, c("SP", "NN", "PL")], 1, mean)
  plot.out <- cbind(plot.dat, plot.preds)

  return(list(RMSE = pred.rmse,
              PlotData = plot.out,
              Weights = weights))
}

#one time through the function
#test <- do.one(seed = 1, swim = swimming, nhidden = 5, nepoch = 50,  ndeg = 6, spdeg = 5)
```

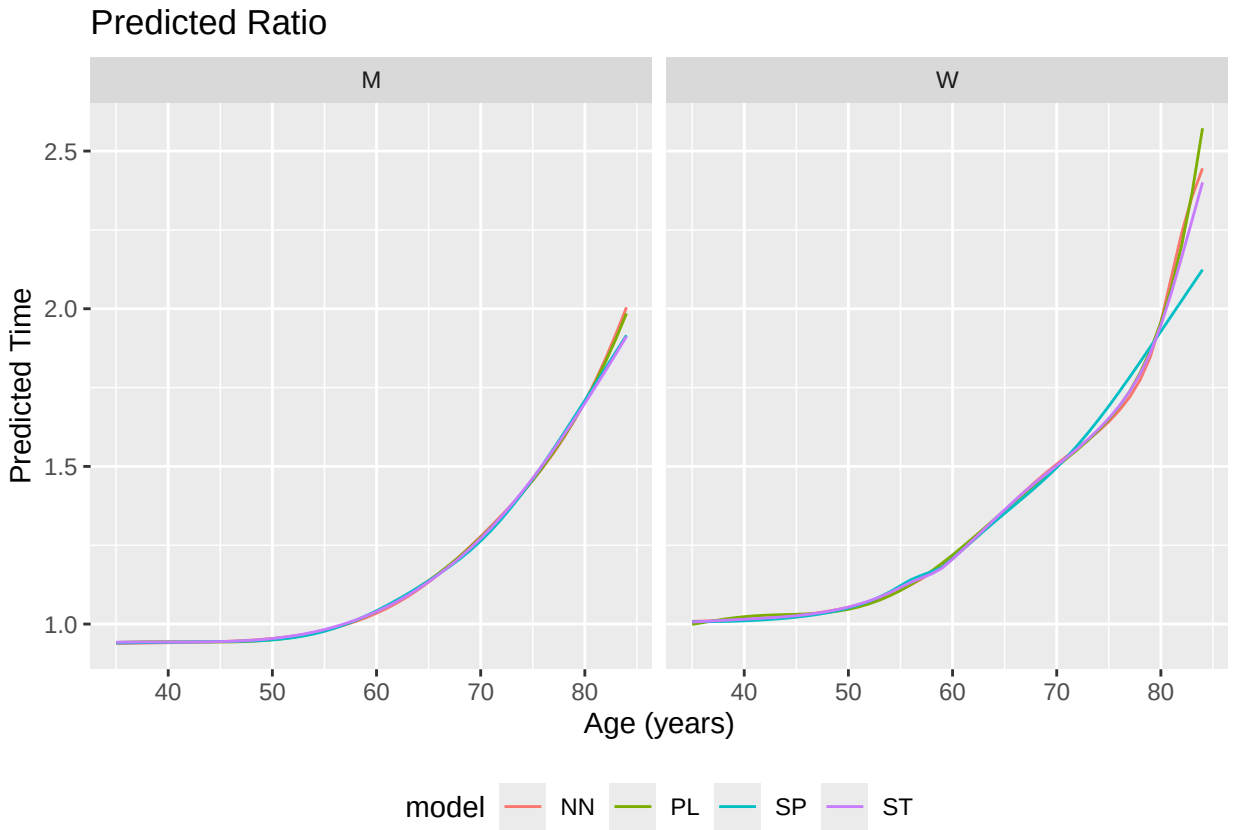
Run multiple iterations:

```
do.one.boot <- function(seed, swim, nhidden, nepoch, ndeg, spdeg) {
  set.seed(seed)
  this.swim <- swim[sample(1:nrow(swim), replace = TRUE), ]
  res <- do.one(seed, this.swim, nhidden = nhidden,
                nepoch = nepoch, ndeg = ndeg, spdeg = spdeg)
  return(res)
}

startseed = 1
bigB = 10

bigres.dividedres <- mclapply(1:bigB, FUN = function(i) {
  do.one.boot(startseed + i - 1, swimming,
              nhidden = 5, nepoch = 50,  ndeg = 6, spdeg = 5)
})
```

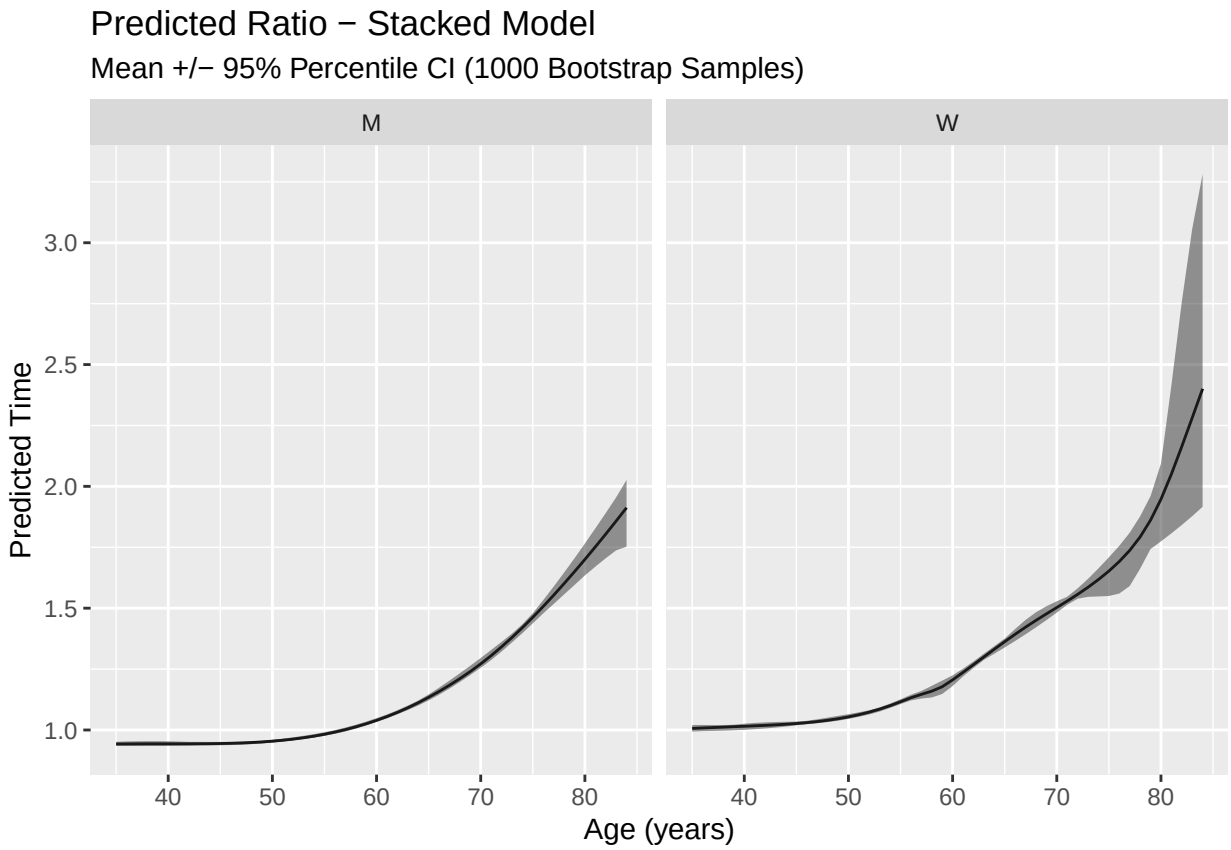
Plotting the results:

```
res <- bigres.dividedres

new_list<- list()
for (i in 1:bigB){
  new_list[[i]] <- res[[i]]$PlotData
}


##########
wide.preds <- do.call(rbind, new_list) %>%
  group_by(Age, Female) %>%
  summarize(
    NN.mean = mean(NN),
    NN.lo = quantile(NN, 0.025),
    NN.hi = quantile(NN, 0.975),
    PL.mean = mean(PL),
    PL.lo = quantile(PL, 0.025),
    PL.hi = quantile(PL, 0.975),
    SP.mean = mean(SP),
    SP.lo = quantile(SP, 0.025),
    SP.hi = quantile(SP, 0.975),
    ST.mean = mean(ST),
    ST.lo = quantile(ST, 0.025),
    ST.hi = quantile(ST, 0.975)) %>%
  mutate(Sex = ifelse(Female == 0, "M", "F"))

long.preds <- wide.preds %>%
  pivot_longer(., cols = NN.mean:ST.hi,
               names_pattern = "(.*)\\.(.*)",
               names_to = c("model", "outval"),
               values_to = "PredictedScore")

plot.preds <- long.preds %>%
  pivot_wider(names_from = outval, values_from = PredictedScore)
```

```
plot.preds <- plot.preds %>%
  mutate(Gender = ifelse(Female == "0", "M", "W"))

g <- plot.preds %>%
  ggplot() +
  geom_line(aes(x=Age, y=mean, group=model, col=model)) +
  facet_wrap(vars(Gender)) +
  ggtitle("Predicted Ratio") +
  xlab("Age (years)") + ylab("Predicted Time") +
  theme(legend.position = "bottom")

g
```




```
g2 <- plot.preds %>%
  filter(model == "ST") %>%
  ggplot() +
  geom_line(aes(x=Age, y=mean), size=.5) +
  geom_ribbon(aes(x=Age, ymin=lo, ymax=hi),
              alpha=0.5, linetype=2)  +
  facet_wrap(vars(Gender)) +
  ggtitle("Predicted Ratio - Stacked Model",
          sub = "Mean +/- 95% Percentile CI (1000 Bootstrap Samples)") +
  xlab("Age (years)") + ylab("Predicted Time") +
  theme(legend.position = "bottom")

g2
```


Predicted Ratio − Stacked Model
Mean +/− 95% Percentile CI (1000 Bootstrap Samples)
M
W
Predicted Time
3.0
2.5
2.0
1.5
1.0
40
50
60
70
80
Age (years)