1 Setup some data

Packages we use

```
library(riskRegression)
library(gam)
library(data.table)
library(randomForestSRC)
library(nnls)
```

Construct a small data sample to play with. The full data set should first be loaded as pph.

```
## if(Sys.info()[["user"]] == "amnudn") source("./sandbox/get-data.R") ##
    Replace to load data
pph[, intendedCS:=1*(SecondBirthIntended=="PlannedCS")]
pph[, intendedCS:=factor(intendedCS,levels=c("0","1"),labels=c("No","Yes")
    )]
treatment_var <- "intendedCS"
covariates_binary <- names(pph)[grepl("Prev", names(pph))][c(1,11:14)]
covariates_cont <- c("MAlder", "PrevYearOfDelivery")
covariates <- c(covariates_binary, covariates_cont)
outcome_var <- "PPHbin"

## Get subsample to play with
set.seed(341)
toy_data <- pph[sample(1:.N, size = 7000), c(covariates,treatment_var,
    outcome_var), with = FALSE]
for (nn in c(covariates_binary,outcome_var)) set(toy_data, j = nn, value =
     factor(toy_data[[nn]]))
toy_data[, subject_ID:=1:.N]
```

2 Fit some models

2.1 Manual cross validation

Fit some models, e.g., using glm, splines, random forest, etc. Evaluate the performance using the Brier score, i.e., the mean squared error between the predicted risk from the model and the observed outcome.

```
## Split data using tool from riskRegression package
splits <- getSplitMethod("cv5", N = toy_data[, .N], seed = 42)

## Fit some models and get predictions on test set.
form0 <- formula(paste(outcome_var, "~", paste(c(covariates_binary,
    treatment_var), collapse = "+")))
form <- update(form0, paste(".~.+", paste(covariates_cont, collapse = "+")
    ))
model_preds <- do.call(rbind, lapply(1:splits$k, function(fold.k){
  train_data = toy_data[splits$index[,1] != fold.k]
  test_data = toy_data[splits$index[,1] == fold.k]
  model_fits <- list(glm = glm(form,data=train_data, family="binomial"),
              spline = gam(update(form0, paste(".~.+", paste("s(", paste0(
    covariates_cont, ", 3)"), collapse = "+"))), data=train_data,family="
    binomial"),
```

```
            rf_big = rfsrc(form,mtry = 3, nodesize = 10,data=train_data),
            rf_small = rfsrc(form,mtry = 7, nodesize = 1,data=train_data)
    )
  do.call(rbind, lapply(seq_along(model_fits), function(mm){
    copy(test_data)[, ":="(model = names(model_fits)[mm], fold = fold.k,
    risk_pred = predictRisk(model_fits[[mm]], newdata = test_data))]
  }))
}))

## Calculate the Brier score
model_preds[, .(Brier_score = 100*mean((risk_pred - (get(outcome_var) == "
    Yes"))^2)), model]
```

2.2 Using the `riskRegression` package

Can also use the `Score` function from the `riskRegression` package. Bootstraps the data instead and validates the model on the out-of-bag data.

```
model_fits <- list(glm = glm(form,data=toy_data, family="binomial"),
           spline = gam(update(form0, paste(".~.+", paste("s(", paste0(
    covariates_cont, ", 3)"), collapse = "+"))), data=toy_data,family="
    binomial"),
           rf_small = rfsrc(form,mtry = 3, nodesize = 10,data=toy_data),
           rf_big = rfsrc(form,mtry = 7, nodesize = 1,data=toy_data))
model_scores <- Score(model_fits,formula = formula(paste0(outcome_var, "~1
    ")),data = toy_data,split.method = "bootcv",B = 10)
model_scores
```

3 Average treatment effect

With an estimate of the outcome model (the conditional expectation of the outcome given covariates and treatment) we can obtain an estimate of the average treatment effect.

```
dat0 = copy(toy_data)[, intendedCS := "No"]
dat1 = copy(toy_data)[, intendedCS := "Yes"]
ate_estimates <- do.call(rbind, lapply(seq_along(model_fits), function(mm)
    {
  pred1 = predictRisk(model_fits[[mm]], newdata = dat1)
  pred0 = predictRisk(model_fits[[mm]], newdata = dat0)
  data.table(model = names(model_fits)[mm],
         ate = mean(pred1-pred0))
}))
ate_estimates
```

4 Super Learner

To combine models, we first construct "level one" data with the outcome and prediction for all individuals and all models.

```
level1_data <- dcast(model_preds,
            formula(paste("subject_ID +", outcome_var, " ~ model")),
```

```
            value.var = "risk_pred")
head(level1_data)
```

Combine the prediction using a "meta learner". This can be done in several ways, two of which are shown below.

```
## Combine models using glm
meta_glm <- glm(formula(paste(outcome_var, "~", paste(names(model_fits),
    collapse = "+"))),
        data = level1_data,
        family = "binomial")

## Combine models by solving a non-negative least squares problem and
    normalize the weights
meta_nnls <- nnls(as.matrix(level1_data[, names(model_fits), with = FALSE
    ]), 1*(level1_data[[outcome_var]] == "Yes"))
nnls_weights <- coef(meta_nnls)/sum(coef(meta_nnls))
```

Finally, we apply all of the prediction models to all of the data, and then apply of the fitted meta learners to these prediction to obtain the final predictions.

```
## Fit the models on *all* of the data
full_pred <- data.table(do.call(cbind, lapply(model_fits, function(mm)
    predictRisk(mm, newdata = toy_data))))

## Combine the model predictions using the weights from the glm
full_pred[, meta_glm_pred := predict(meta_glm, newdata = full_pred, type =
     "response")]

## Combine the model predictions using the weights nnls
full_pred[, meta_nnls_pred := as.matrix(.SD) %*% nnls_weights, .SDcols =
    names(model_fits)]
```

Alternatively we can use the `SuperLearner` package, see for instance this tutorial: Hoffman's blog.

5 Setup simulation

It can be very useful to simulate data as we then know the true value of our parameter of interest, which makes it easy to evaluate the performance of the estimators we consider. The are many different ways to do this. To construct data that "looks like" the real data (at least superficially) we could do something like the following.

```
propensity_model <- glm(update(form, paste(treatment_var, "~ . -",
    treatment_var)), data = toy_data, family = binomial)
outcome_model <- glm(form, data = toy_data, family = binomial)

sim_data <- function(n = 200){
  sample0 = toy_data[sample(1:toy_data[, .N], size = n, replace = TRUE), c
    (covariates, treatment_var), with = FALSE]
  ## Construct counterfactual outcomes:
  sample0[, eval(treatment_var) := "No"] ## No treatment given
  sample0[, Y0_prob := predict(outcome_model, newdata = sample0, type = "
    response")]
  sample0[, eval(treatment_var) := "Yes"] ## Treatment given
```

```
  sample0[, Y1_prob := predict(outcome_model, newdata = sample0, type = "
    response")]
  sample0[, ":=" (Y0 = runif(n) < Y0_prob, Y1 = runif(n) < Y1_prob)]
  ## Construct the actually observed outcome:
  sample0[, treatment_prob := predict(propensity_model, newdata = .SD,
    type = "response")]
  sample0[, eval(treatment_var) := runif(n) < treatment_prob] ## The
    treatment actually given
  sample0[, eval(outcome_var) := (Y0*get(treatment_var) + Y1*(!(get(
    treatment_var)))) == 1]
  ## Clean up
  for (nn in c("Y0", "Y1", treatment_var, outcome_var))
    set(sample0, j = nn, value = factor(sample0[[nn]], levels = c(FALSE,
    TRUE), labels = c("No", "Yes")))
  return(sample0[])
}

sim_data()
```

6 Literature

- Overview of targeted inference, causality, and efficient estimation: Kennedy (2016) "Semiparametric Theory and Empirical Processes in Causal Inference"
- Some information about the data: Wikkelsø et al. 2014 (The Journal of Maternal-Fetal & Neonatal Medicine, 27(16):1661-1667, 2014)
- Super Learner: Hoffman's blog with tutorial and Chapter 3 in van der Laan & Rose (2011) "Targeted Learning".
- More details on Targeted Learning and Debiased ML: Kennedy (2022) "Semiparametric Doubly Robust Targeted Double Machine Learning: A Review" and the introduction of Chernozhukov et al. (2018) "Double/debiased machine learning for treatment and structural parameters"
- More about causality: Hernán and Robins (2020) "What If"
- Risk prediction and cross validation: Gerds & Kattan (2021) "Medical Risk Prediction Models"