External validation - DK data

Erik Bulow / slightly modified by Ute Hahn

2021-03-25

Contents

Start

Attach packages

```
pkgs <- c("tidyverse", "doParallel", "pROC", "rms", "givitiR")

purrr::walk(pkgs, ~suppressPackageStartupMessages(library(., character.only = TRUE)))
```

Load the exported model to validate (the R object previsously sent, probably with another path).

```
# Load the exported model object
load("export_90d.RData")
```

Set random seed for reproducibility:

```
set.seed(123)
```

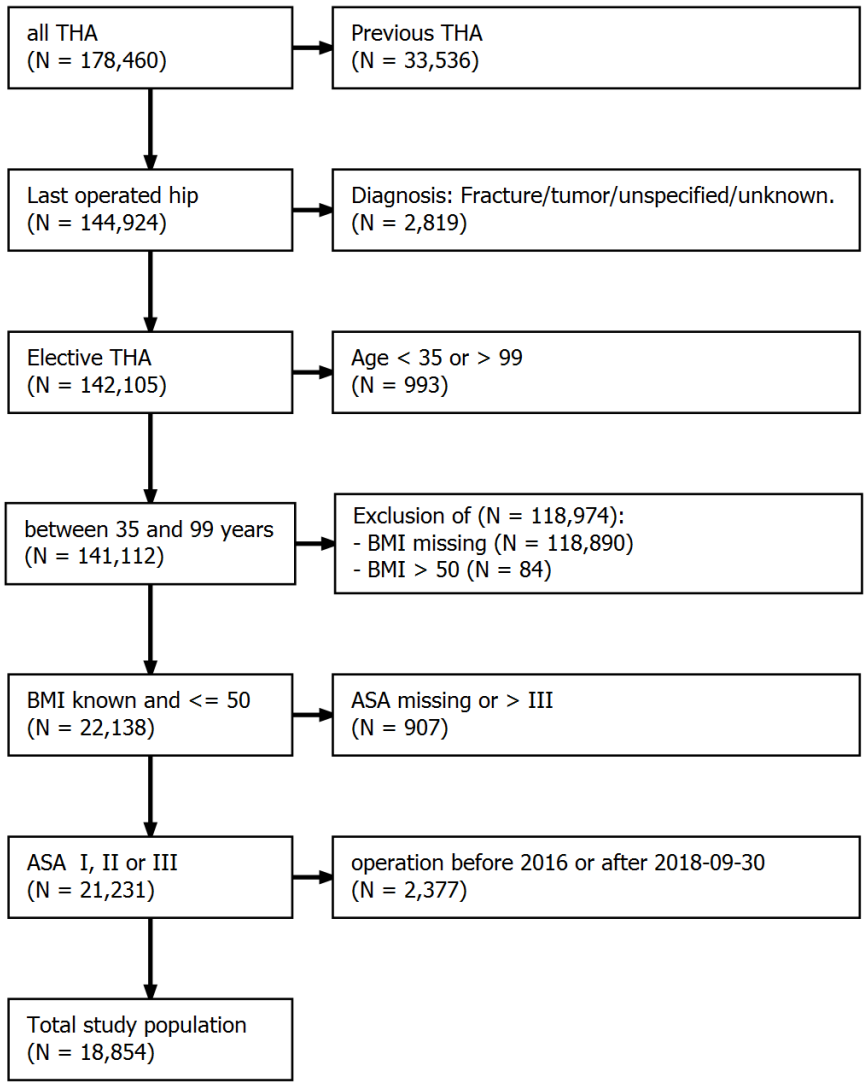
Prepare data

Inclusion/exlusion

Those were the inclusions/exklusions from Sweden. It might not be necessary to filter out on BMI, hospital and education however. Those variables are not used in the model.

Additional filter to ages 35-99 years to match the Swedish cohort.

```
knitr::include_graphics("../graphs/flowchartdk_detailed.png")
```



Variables

Outcome

The outcome variable is boolean (or numeric) indicating wether the patient died of any cause within 90 days after THA (`TRUE`/1) or not (`FALSE`/0). We did not have any censoring in Sweden. I guess we can simply drop cases where status is unknown?

The DK data set:

```
load("dkdata.RData")
```

Predictors

The DK data look like this:

```
# head(dkdata) - data not shown here because they include age and gender
```

Baseline variables

- **P_Gender**: Kvinna/Man = Female/Male
- **P_ASA**: level 1-3
- **P_Age**: 35 - 99

Comorbidities

Comorbidities have been handled in SAS, by Aurelie and Ina.

Danish data into variables

The outcome needs to be a numeric variable, not logical. Otherwise we get an error `Fejl i givitiCheckArgs(o, e, devel, thres, maxDeg) : The vector 'o' must be a numeric vectors with 0/1 values..`

```
y <- as.numeric(dkdata$outcome)

X <- dkdata
```

Eriks fitted model

```
load("../cache/model_reduced_lean.RData")
```

Validation of model as is

```
obspred <-
  tibble(
    obs  = y,
```

```
    pred = predict(model_reduced_lean, X, type = "response")
  )
```

```
# Tibble with observed and predicted outcome
obspred <-
  tibble(
    obs  = y,
    pred = predict(model_reduced_lean, X, type = "response")
  )

# ROC curve
ROC <- pROC::roc(obspred, "obs", "pred", direction = "<")

# Estimate CI for AUC based on bootstrapping
# Use parallel processing to speed up the process
doParallel::registerDoParallel()
AUCci <-
  pROC::ci.auc(
    ROC,
    method          = "bootstrap",
   #boot.n = 100,
    boot.stratified = FALSE,
    parallel        = TRUE
  )

# Check calibration. Note that devel should actually be "internal" for this example but I use
# "external", since that's what you will use for the DK data.
calibration <-
  givitiR::givitiCalibrationBelt(
    obspred$obs,
    obspred$pred,
    devel = "external"
  )
```

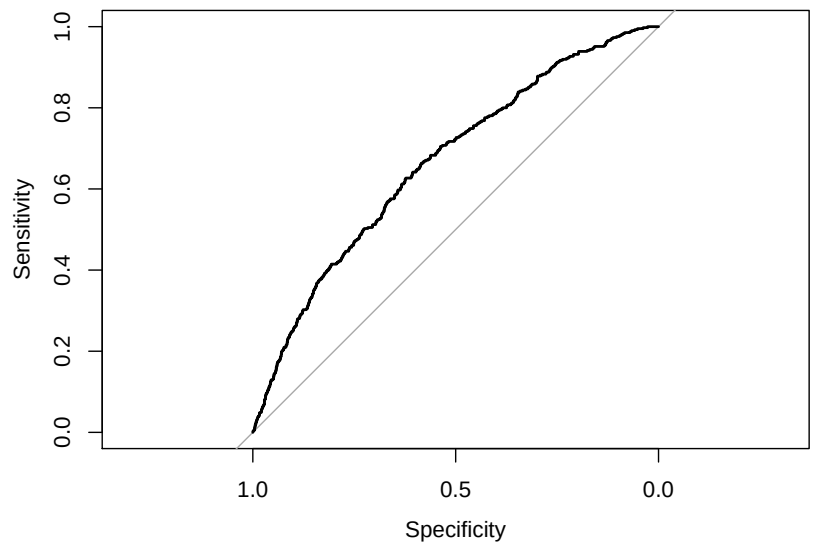
Results

For this example we had AUC:

```
AUCci
```

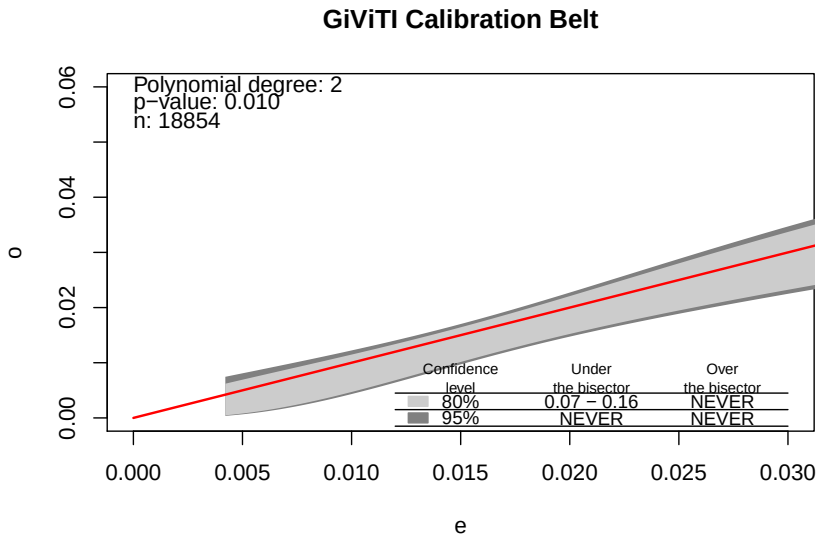
```
## 95% CI: 0.6373-0.6906 (2000 non-stratified bootstrap replicates)
```

```
plot(ROC)
```



A calibration belt plot might be illustrated as:

```
plot(calibration, xlim = c(0, 0.03), ylim = c(0, 0.06))
```




```
## $m
## [1] 2
##
## $p.value
## [1] 0.01013039
```

Re-calibrated intercept

Method 2 from table 1 in Steyerberg 2004.

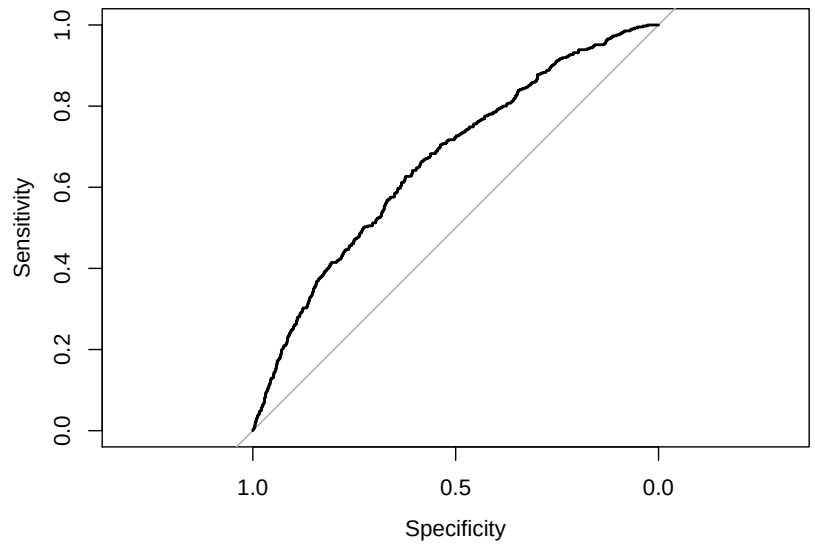
```r
Z <- predict(model_reduced_lean, X, type = "response")

# Refit the intercept using Z = a + Xb from above as offset
fit2 <- glm(y ~ 1, offset = Z)

# Same calibration and validation as above
obspred2     <- tibble(obs  = y, pred = predict(fit2, type = "response"))
ROC2         <- pROC::roc(obspred2, "obs", "pred", direction = "<")
AUCci2       <- pROC::ci.auc(ROC2, method = "bootstrap",
                             # boot.n = 100,
                              boot.stratified = FALSE, parallel = TRUE)
calibration2 <- givitiR::givitiCalibrationBelt(obspred2$obs, obspred2$pred, devel = "external")
```

Results

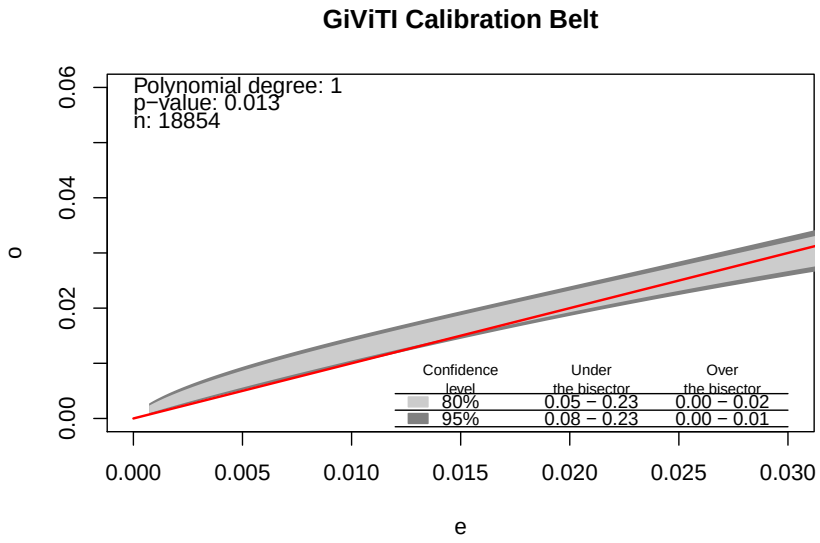
```r
AUCci2
```

```
## 95% CI: 0.6387-0.6912 (2000 non-stratified bootstrap replicates)
```

```r
plot(ROC2)
```




```r
plot(calibration2, xlim = c(0, 0.03), ylim = c(0, 0.06))
```




```
## $m
## [1] 1
##
## $p.value
## [1] 0.01328765
```

Re-calibration of intercept and calibration slope

Method 3 from table 1 in Steyerberg 2004.

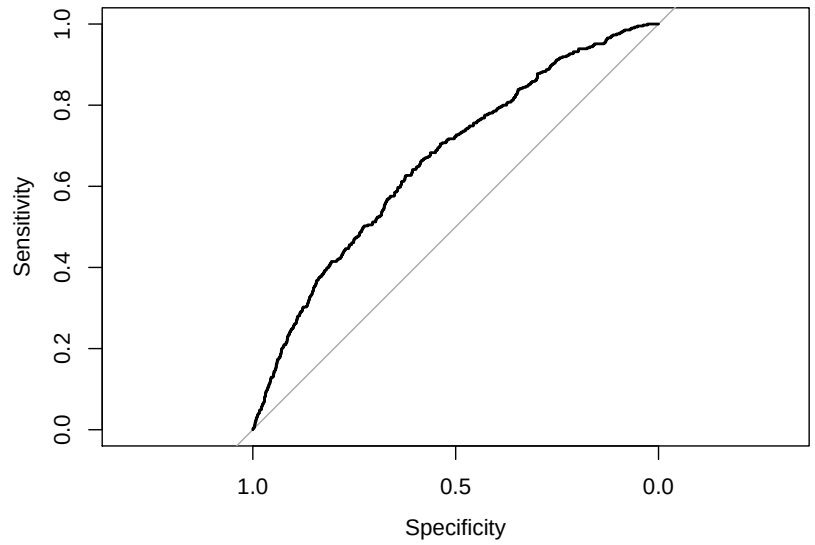
```
fit3         <- glm(y ~ 1 + Z)
obspred3     <- tibble(obs  = y, pred = predict(fit3, type = "response"))
ROC3         <- pROC::roc(obspred3, "obs", "pred", direction = "<")
AUCci3       <- pROC::ci.auc(ROC3, method = "bootstrap",
                             #boot.n = 100,
                             boot.stratified = FALSE, parallel = TRUE)
calibration3 <- givitiR::givitiCalibrationBelt(obspred3$obs, obspred3$pred, devel = "external")
```

Results

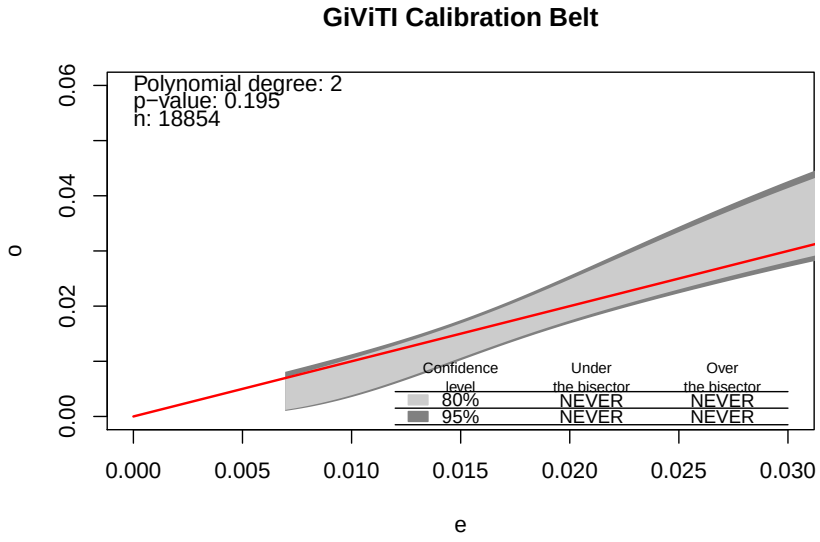
```
AUCci3
```

```
## 95% CI: 0.6389-0.6907 (2000 non-stratified bootstrap replicates)
```

```r
plot(ROC3)
```




```r
plot(calibration3, xlim = c(0, 0.03), ylim = c(0, 0.06))
```




```
## $m
## [1] 2
##
## $p.value
## [1] 0.1953893
```

Export data to Sweden

Store data here:

```
exportdir <- "../output_R/"
```

ROC

If it would be OK to export coordinates for ROC-plots I would recommend this code to extract only the minimal data needed:

```
roc_plot_coords <-
  data.frame(
    specificities = ROC$specificities,
    sensitivities = ROC$sensitivities
  )

roc3_plot_coords <-
  data.frame(
    specificities = ROC3$specificities,
    sensitivities = ROC3$sensitivities
  )
```

AUC with CI

The text output from `AUCci` and `AUCci3` should be enough. Hence, the same character string that gets printed above (but now stored in an object).

```
AUCci_print  <- capture.output(AUCci)
AUCci3_print <- capture.output(AUCci3)
```

Export objects

Save objects above to file `export.RData` (in the current working directory).

```
save(
  roc_plot_coords,
  roc3_plot_coords,
  AUCci_print,
  AUCci3_print,
  file = paste0(exportdir,"export.RData")
)
```

Calibration plots

Help function

This is a simple help function to make a clean calibration belt plot and save it as TIFF (in the curent working directory):

```
makeplot <- function(x, file_name = deparse(substitute(x))) {
  tiff(
    paste0(exportdir, file_name, ".tiff"),
    1024, 1024, pointsize = 36,
    compression = "lzw"
```

```
  )

  tcks <- seq(.0, .1, .01)

  plot(
    x,
    xlim             = c(0, .06),
    ylim             = c(0, .08),
    xlab             = "Predicted probabilities [%]",
    ylab             = "Observed  probabilities [%]",
    main             = NULL,
    table            = FALSE,
    polynomialString = FALSE,
    pvalueString     = FALSE,
    nString          = FALSE,
    mar              = c(5, 4, 0, 0) + 0.1,
    xaxt             = "n",
    yaxt             = "n"
  )
  abline(v = .03, lty = "dashed", col = "darkgreen", lwd = 3)
  axis(1, at = tcks, lab = sprintf("%.0f", tcks * 100), las = TRUE)
  axis(2, at = tcks, lab = sprintf("%.0f", tcks * 100), las = TRUE)

  dev.off()
}
```

Make and save figures

```
makeplot(calibration)
```

```
## pdf
##   2
```

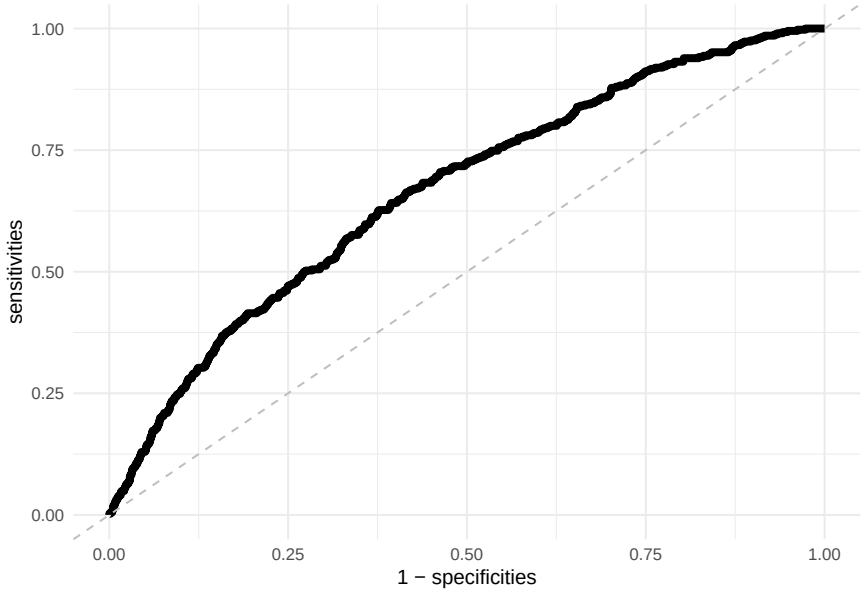
```
makeplot(calibration3)
```

```
## pdf
##   2
```

ROC plots

If it not possible to export data for ROC-plot, here is some code to make a figure and save it as TIFF (in the current working directory) instead:

```
roc_plot_coords %>%
  ggplot(aes(1 - specificities, sensitivities)) +
  geom_path(size = 2) +
  geom_abline(intercept = 0, slope = 1, color = "grey", linetype = 2) +
  theme_minimal() +
  theme(
    legend.position = c(1, 0),
    legend.justification = c(1, 0),
    legend.title = element_blank()
  )
```




```
ggsave(
  paste0(exportdir, "roc.tiff"),
  height = 10,
  width = 10,
  unit = "cm",
  dpi = 900,
  compression = "lzw"
)
```