Practical: RISK PREDICTION

Advanced Statistical Analysis

Research question

In this session, we will explore the dataset of 2000 participants we met in the lecture, and fit a risk prediction model for death within 5 years, based on some simple patient characteristics.

Objectives

By the end of this practical, you should be able to:

1. Fit a logistic model to create risk predictions.
2. Assess model discrimination by calculating the Area Under the Curve.
3. Assess model calibration by graphing observed and predicted risks.

Dataset and analysis

For this practical we will use a (simulated) dataset called `data_predict`. This contains data for 2,000 patients, with information on six variables.

Variable	Description
id	Unique patient ID
age	Age (years)
sbp	Systolic Blood Pressure
bmi	Body Max Index kg/m^2
sex	Female / Male
dead	Alive / Dead

```r
# Install Libraries
if (!require(pacman)) install.packages("pacman")
#> Loading required package: pacman
pacman::p_load(tidyr, dplyr, ggplot2, broom)

# Load data
load("../data/data_predict.rda")
```

Data exploration

Have a look at the data, then try answering the following participants: * How many participants died at the end of the follow-up? * What is the proportion of female participants? * What is the mean, standard deviation, minimum, and maximum age of these participants?

```r
# Base R
summary(data_predict)
#>        id              age             sbp             bmi             sex
#>  Min.   :   1.0   Min.   :40.00   Min.   : 76.6   Min.   :15.50   Female: 978
#>  1st Qu.: 500.8   1st Qu.:51.00   1st Qu.:113.7   1st Qu.:23.20   Male  :1022
#>  Median :1000.5   Median :60.00   Median :120.6   Median :25.10
#>  Mean   :1000.5   Mean   :60.45   Mean   :120.3   Mean   :25.19
#>  3rd Qu.:1500.2   3rd Qu.:70.00   3rd Qu.:127.2   3rd Qu.:27.20
#>  Max.   :2000.0   Max.   :80.00   Max.   :152.2   Max.   :35.60
#>     dead
#>  Alive:1491
#>  Dead : 509
#>
#>
#>
#>

# Tidyverse (More verbose but more control)
data_predict %>%
  group_by(dead) %>%
  tally %>%
  mutate(percent = n/sum(n)*100)
#> # A tibble: 2 x 3
#>   dead      n percent
#>   <fct> <int>   <dbl>
#> 1 Alive  1491    74.6
#> 2 Dead    509    25.4

data_predict %>%
  group_by(sex) %>%
  tally %>%
  mutate(percent = n/sum(n)*100)
#> # A tibble: 2 x 3
#>   sex        n percent
#>   <fct>  <int>   <dbl>
#> 1 Female   978    48.9
#> 2 Male    1022    51.1

data_predict %>%
  # filter out missing observation at any variable
  filter_all(all_vars(!is.na(.))) %>%
  summarise(n = n(),
            mean = mean(age),
            sd = sd(age),
            min = min(age),
            max = max(age))
#> # A tibble: 1 x 5
#>       n  mean    sd   min   max
#>   <int> <dbl> <dbl> <dbl> <dbl>
#> 1  2000  60.5  11.5    40    80
```

About 25% of the 2,000 individuals die – this is a high-risk population. There is roughly 50% males and 50% females, aged 40-80.

Randomly split data into training and test sets

```
set.seed(777)

n_total <- nrow(data_predict)
proportion_train <- 0.5
n_train <- floor(proportion_train * n_total)

sample_train <- sample(1:n_total, n_train) %>% sort
sample_test <- which(!(1:n_total %in% sample_train))

data_train <- data_predict[sample_train,]
data_test <- data_predict[sample_test,]
```

Fit model in training data

```
# Fit logistic regression
model <- glm(formula = dead ~ age + sex + sbp + bmi,
             family = "binomial",
             data = data_train)

# View model output summary
summary(model)
#>
#> Call:
#> glm(formula = dead ~ age + sex + sbp + bmi, family = "binomial",
#>     data = data_train)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -1.9187  -0.7526  -0.4669   0.6399   2.7754
#>
#> Coefficients:
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -13.297927   1.440507  -9.231  < 2e-16 ***
#> age           0.086203   0.007921  10.883  < 2e-16 ***
#> sexMale       0.155126   0.160836   0.965    0.335
#> sbp           0.052679   0.008465   6.223 4.87e-10 ***
#> bmi           0.013306   0.027782   0.479    0.632
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 1126.86  on 999  degrees of freedom
#> Residual deviance:  950.39  on 995  degrees of freedom
#> AIC: 960.39
#>
#> Number of Fisher Scoring iterations: 5

# Get odds ratio and 95% CI
cbind(OR = coef(model), confint(model)) %>% exp
#> Waiting for profiling to be done...
#>                       OR        2.5 %       97.5 %
#> (Intercept) 1.677968e-06 9.341761e-08 2.662586e-05
#> age         1.090028e+00 1.073596e+00 1.107490e+00
#> sexMale     1.167805e+00 8.524485e-01 1.602203e+00
#> sbp         1.054091e+00 1.037001e+00 1.072023e+00
#> bmi         1.013395e+00 9.596049e-01 1.070147e+00
```

Predict risk based on trained model

```
# update data_train and data_test with predicted probabilites
data_train$prob_dead <-
  predict(model, type = "response")

# compare predicted risk
data_train %>%
  group_by(dead) %>%
  summarise(n = n (),
            mean = mean(prob_dead),
            sd = sd(prob_dead),
            min = min(prob_dead),
            max = max(prob_dead)) %>%
  pivot_longer(-dead) %>%
  pivot_wider(id_cols = name, names_from = dead)
#> # A tibble: 5 x 3
#>   name     Alive     Dead
#>   <chr>    <dbl>    <dbl>
#> 1 n      749      251
#> 2 mean     0.207    0.383
#> 3 sd       0.155    0.189
#> 4 min      0.0134   0.0213
#> 5 max      0.841    0.820
```

On training dataset

```
data_test$prob_dead <-
  predict(model, type = "response", newdata = data_test)

data_test %>%
  group_by(dead) %>%
  summarise(n = n (),
            mean = mean(prob_dead),
            sd = sd(prob_dead),
            min = min(prob_dead),
            max = max(prob_dead)) %>%
  pivot_longer(-dead) %>%
  pivot_wider(id_cols = name, names_from = dead)
#> # A tibble: 5 x 3
#>   name     Alive     Dead
#>   <chr>    <dbl>    <dbl>
#> 1 n      742      258
#> 2 mean     0.212    0.406
#> 3 sd       0.160    0.192
#> 4 min      0.0149   0.0256
#> 5 max      0.850    0.844
```

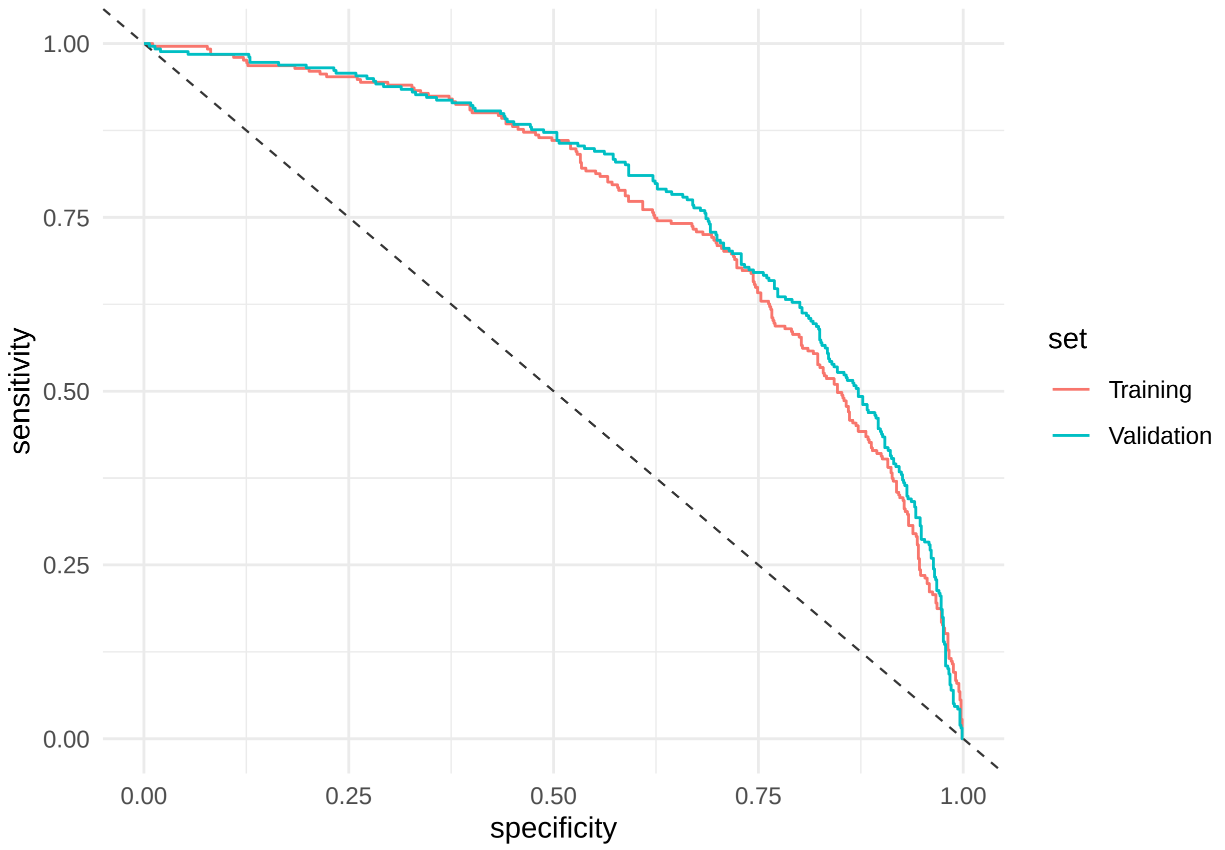
```
# Alternative solution (with broom::augment() )
alt_data_train <- model %>%
  # augment creates new columns with some useful information from the model
  # .fitted = predicted values
  augment(type.predict = "response")

alt_data_test <- model %>%
  augment(type.predict = "response", newdata = data_test)
```

On test dataset

Validation

ROC In the training dataset, the ROC is 79%. This means that a person who did die has a 79% probability of having a higher predicted risk (of dying) than someone who did not. This shows the model has fairly good discrimination (ability to separate those who did and did not experience the event of interest).

```
pacman::p_load(yardstick)

# set the second level of factor variable as the event (i.e. dead)
options(yardstick.event_first = FALSE)



# combine dataset
data_grouped <- bind_rows(data_train %>% mutate(set = "Training"),
                          data_test %>% mutate(set = "Validation")) %>%
  group_by(set)

# Calculate ROC
data_grouped %>%
  roc_auc(truth = dead, prob_dead)
#> Warning: The `yardstick.event_first` option has been deprecated as of yardstick 0.0.7 and will be co
#> Instead, set the following argument directly in the metric function:
#> `options(yardstick.event_first = TRUE)`  -> `event_level = 'first'` (the default)
#> `options(yardstick.event_first = FALSE)` -> `event_level = 'second'`
#> This warning is displayed once per session.
#> # A tibble: 2 x 4
#>   set        .metric .estimator .estimate
#>   <chr>      <chr>   <chr>          <dbl>
#> 1 Training   roc_auc binary         0.767
#> 2 Validation roc_auc binary         0.783

# Visualise ROC
data_roc <- data_grouped %>%
  roc_curve(truth = dead, prob_dead)

ggplot(data_roc, aes(x = specificity, y = sensitivity, color = set)) +
  geom_abline(slope = -1, intercept = 1,
              size = 0.4, color = "grey21", linetype = "dashed") +
  geom_line() +
  theme_minimal()
```



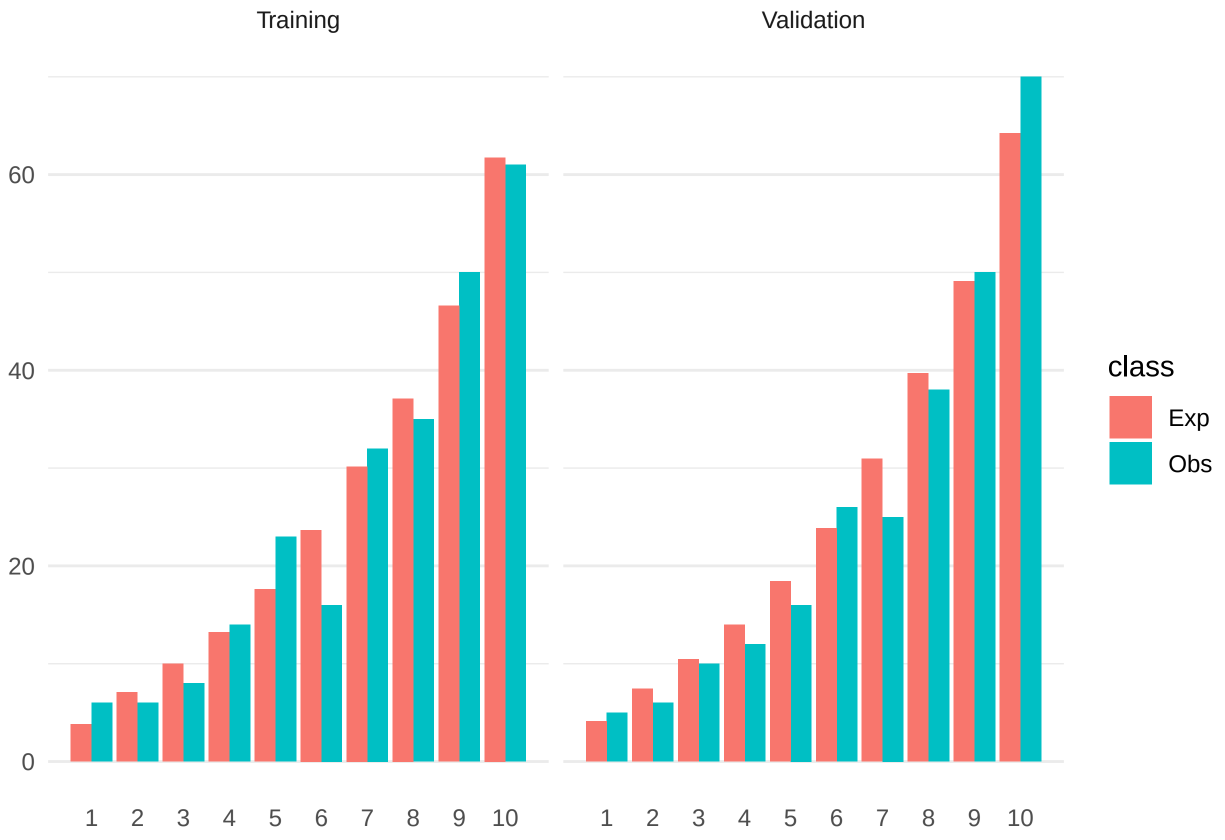
Hosmer-Lemeshow The Hosmer-Lemeshow goodness of fit table for the two (S=0 and the S=1) datasets were very similar. Both showed evidence of a well calibrated model.

```
pacman::p_load(generalhoslem)

gof <- data_grouped %>%
  group_map( ~ logitgof(
    obs = .$dead,
    exp = .$prob_dead,
    g = 10
  ))

# Fix name
names(gof) <- group_keys(data_grouped)[[group_vars(data_grouped)]]

# output Goodness of Fit metrics
gof
#> $Training
#>
#>  Hosmer and Lemeshow test (binary model)
#>
#> data:  .$dead, .$prob_dead
#> X-squared = 8.105, df = 8, p-value = 0.4233
#>
#>
#> $Validation
#>
```

```
#>  Hosmer and Lemeshow test (binary model)
#>
#> data:  .$dead, .$prob_dead
#> X-squared = 4.7631, df = 8, p-value = 0.7826

# create GoF table
gof_table <- lapply(gof, function(x){
  cbind(x$observed, x$expected) %>%
    as_tibble(rownames = "threshold") %>%
    mutate(group = 1:nrow(.))
})

gof_table$Training
#> # A tibble: 10 x 6
#>    threshold           y0    y1 yhat0 yhat1 group
#>    <chr>            <dbl> <dbl> <dbl> <dbl> <int>
#>  1 [0.0134,0.0541]     94     6  96.2  3.79     1
#>  2 (0.0541,0.085]      94     6  92.9  7.09     2
#>  3 (0.085,0.115]       92     8  90.0 10.0      3
#>  4 (0.115,0.153]       86    14  86.8 13.2      4
#>  5 (0.153,0.204]       77    23  82.4 17.6      5
#>  6 (0.204,0.267]       84    16  76.3 23.7      6
#>  7 (0.267,0.337]       68    32  69.8 30.2      7
#>  8 (0.337,0.415]       65    35  62.9 37.1      8
#>  9 (0.415,0.522]       50    50  53.4 46.6      9
#> 10 (0.522,0.841]       39    61  38.2 61.8     10
gof_table$Validation
#> # A tibble: 10 x 6
#>    threshold           y0    y1 yhat0 yhat1 group
#>    <chr>            <dbl> <dbl> <dbl> <dbl> <int>
#>  1 [0.0149,0.0582]     95     5  95.9  4.12     1
#>  2 (0.0582,0.0897]     94     6  92.6  7.43     2
#>  3 (0.0897,0.121]      90    10  89.5 10.5      3
#>  4 (0.121,0.161]       88    12  86.0 14.0      4
#>  5 (0.161,0.208]       84    16  81.6 18.4      5
#>  6 (0.208,0.274]       74    26  76.2 23.8      6
#>  7 (0.274,0.35]        75    25  69.0 31.0      7
#>  8 (0.35,0.437]        62    38  60.3 39.7      8
#>  9 (0.437,0.547]       50    50  50.9 49.1      9
#> 10 (0.547,0.85]        30    70  35.8 64.2     10
```

Bar graphs comparing the predicted and observed risks in the S=0 and S=1 datasets also show good calibration (in both the training and validation data).

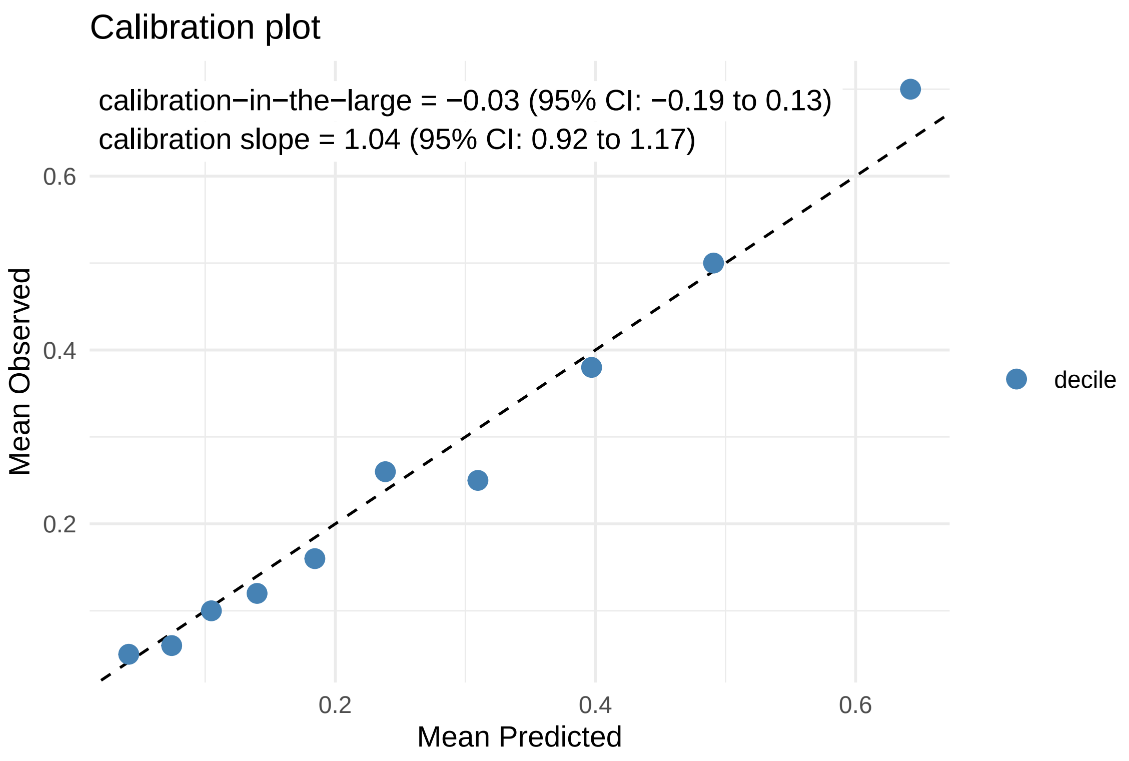
```
# tidy up gof_table for plotting
tidy_gof <- bind_rows(gof_table, .id = "set") %>%
  dplyr::select(set, group, Obs = y1, Exp = yhat1) %>%
  pivot_longer(
    cols = c(Obs, Exp),
    names_to = "class"
  )

# plotting
tidy_gof %>%
  ggplot(aes(x = group, y = value, fill = class)) +
  facet_grid(cols = vars(set)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  scale_x_continuous(breaks = 1:10) +
  theme_minimal() +
  theme(panel.grid.minor.x = element_blank(),
        panel.grid.major.x = element_blank(),
        axis.title = element_blank())
```



Calibration curve

```
df_model <- model %>%
  # augment creates new columns with some useful information from the model
  # .fitted = predicted values
  augment(type.predict = "link", newdata = data_test) %>%
  mutate(dead_count = as.numeric(dead) - 1) %>%
  arrange(.fitted)

# calculate alpha (calibration at large) & beta (calibration slope)
# alpha is calculated by constraining beta to 1

df_alpha <- glm(dead_count ~ offset(.fitted),
               data = df_model,
               family = binomial) %>%
  tidy(conf.int = TRUE)
#> Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
#> Using compatibility `.name_repair`.
```

```r
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.

alpha_text <- paste0(round(df_alpha$estimate[1], 2),
                     " (95% CI: ", round(df_alpha$conf.low[1], 2),
                     " to ", round(df_alpha$conf.high[1], 2),
                     ")")

# beta is calculated by constraining alpha to 0
df_beta <- glm(dead_count ~ .fitted - 1,
               data = df_model,
               family = binomial) %>%
  tidy(conf.int = TRUE)

beta_text <- paste0(round(df_beta$estimate[1], 2),
                    " (95% CI: ", round(df_beta$conf.low[1], 2),
                    " to ", round(df_beta$conf.high[1], 2),
                    ")")


# calculate mean predicted results per decile
df_sum_decile <- model %>%
  # augment creates new columns with some useful information from the model
  # .fitted = predicted values
  augment(type.predict = "response", newdata = data_test) %>%
  mutate(dead_count = as.numeric(dead) - 1) %>%
  arrange(.fitted) %>%
  mutate(decile = ntile(.fitted, 10)) %>%
  group_by(decile) %>%
  summarize(mean_prediction = mean(.fitted),
            sd_prediction = sd(.fitted),
            prop_observed = mean(dead_count)) %>%
  mutate(dummy = "decile")

# plotting
ggplot(df_sum_decile, aes(x = mean_prediction, y = prop_observed)) +
  theme_minimal() +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  # geom_abline(slope = 1, intercept = df_alpha$estimate[1], color = "#1b5299") +
  geom_point(aes(color = dummy), size = 3) +
  scale_color_manual(values = "steelblue") +
  labs(x = "Mean Predicted", y = "Mean Observed", title = "Calibration plot") +
  annotate("label", x = -Inf, y = Inf, hjust = 0, vjust = 1.5, label.size = 0,
           label = paste("calibration-in-the-large =",
                         alpha_text)) +
  annotate("label", x = -Inf, y = Inf, hjust = 0, vjust = 2.5, label.size = 0,
           label = paste("calibration slope =",
                         beta_text)) +
  theme(legend.title = element_blank())
```



Since calibration-in-the-large is close to 0, and calibration slope is close to 1, we can say that the model is well calibrated