SIG Project Real-Time Imputation

Steven Nijman, Thomas Debray, Maarten van Smeden, Gerko Vink, Hanne Oberman

Aim

This document contain the set-up of our SIG project titled "An evaluation of 'real-time' missing data handling in machine learning and prevailing statistical models". The aim is to compare different strategies for developing prediction models that can handle the presence of missing values real time in a single patient.

Data Generating Mechanism

Discussion topics: DGM

- Alle modellen alleen op complete data fitten (RF, log reg, Pattern mixture submodel)
- Dataset voor ontwikkelen predictiemodel regressie, imputatie, en rf
- En een dataset om te imputeren, waarin missing values moeten komen
- Onder hetzelfde model genereren we nieuwe data voor het valideren
- Idealiter zouden de regressie en rf modellen op de complete data dezelfde c-index/prestatie hebben
- En dan trekken we steeds een validatieset op basis van hetzelfde mechanisme en daar schieten we gaten in volgens de missingness mechanisms
- Calibratieslope varieert enorm: onder en overfitting wisselt elkaar af, dus je kunt pech of geluk hebben. Bij een dev set van 1 miljoen dan heeft ie perfecte calibratie. In de gemiddelde performance verdwijnt het, maar wat bij rf steeds misgaat is de variantie in de prestatie. Als je een model ontwikkelt, zit je soms heel ver van een goed model af.
- Conclusie: zowel dev als val set trekken we meermaals. Voorafgaand aan de analyse pipeline.

Complete Data

We will use a prediction model with:

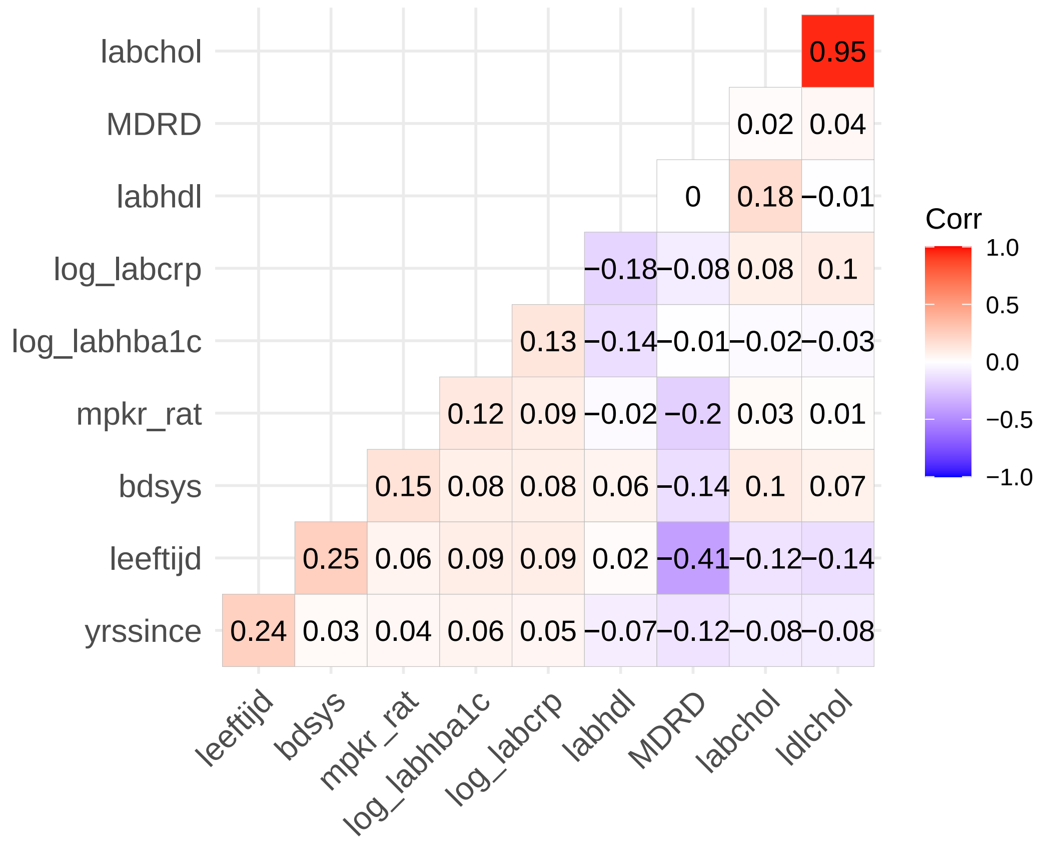
- 10 continuous predictors (X_1, X_2, $\ldots$, X_{10}), generated from a multivariate normal distribution;
- 1 binary outcome (Y), calculated from the 10 predictors.

Our sample will include at least these 11 variables, with a sample size of 10.000 cases.

Initially, we were going to use the SMART data as the basis for our predictor space. But this would result in the same limitations as described in Nijman et al. (2021; i.e., low correlations between predictor variables).

These correlations between variables from the SMART dataset are visualized below.

```r
# we'll use the SMART data to define the relations between predictors
# the variance-covariance matrix is stored in the object varcov
ggcorrplot::ggcorrplot(cov2cor(varcov), hc.order = TRUE, type = "lower", lab = TRUE)
```



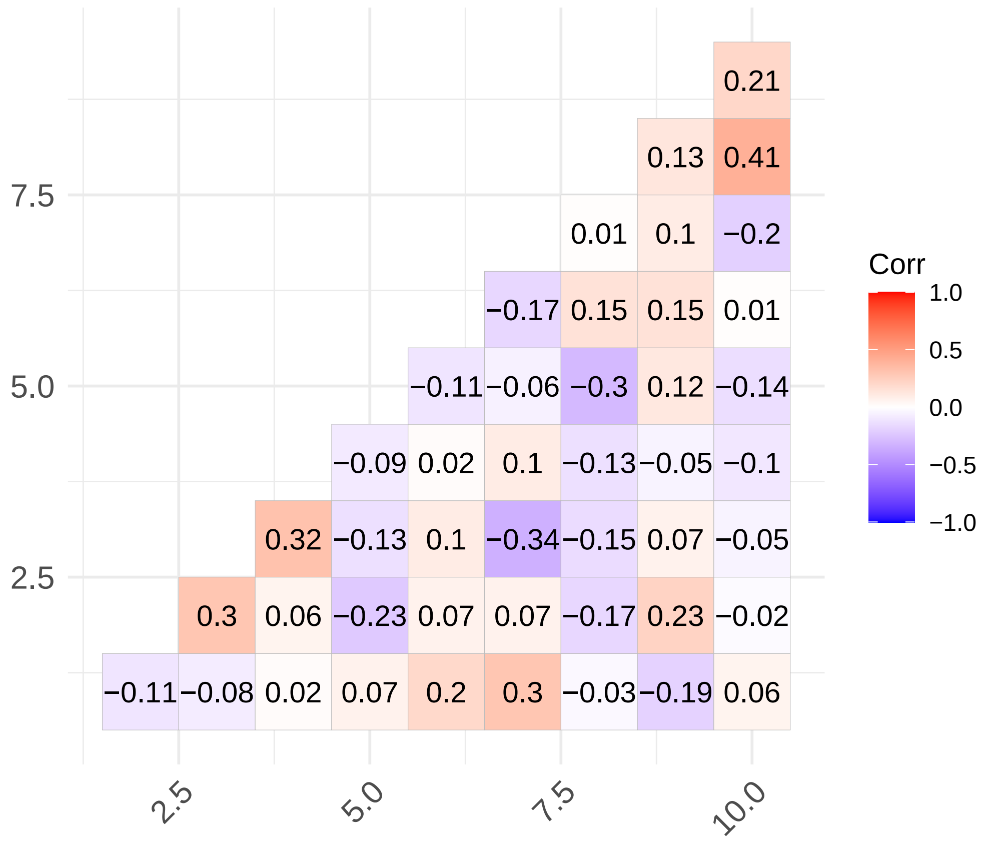
So instead, we'll create one[1] ourselves.

```
# create a variance-covariance matrix with p predictors
set.seed(123)
p <- 10
betas <- rnorm(p*(p-1)/2, 0, 0.1)
sigma <- diag(p)
sigma[upper.tri(sigma)] <- betas
sigma[lower.tri(sigma)]  <- t(sigma)[lower.tri(sigma)]
Sigma <- t(sigma) %*% sigma
isSymmetric(Sigma)
```

```
## [1] TRUE
```

```
ggcorrplot::ggcorrplot(cov2cor(Sigma), type = "lower", lab = TRUE)
```

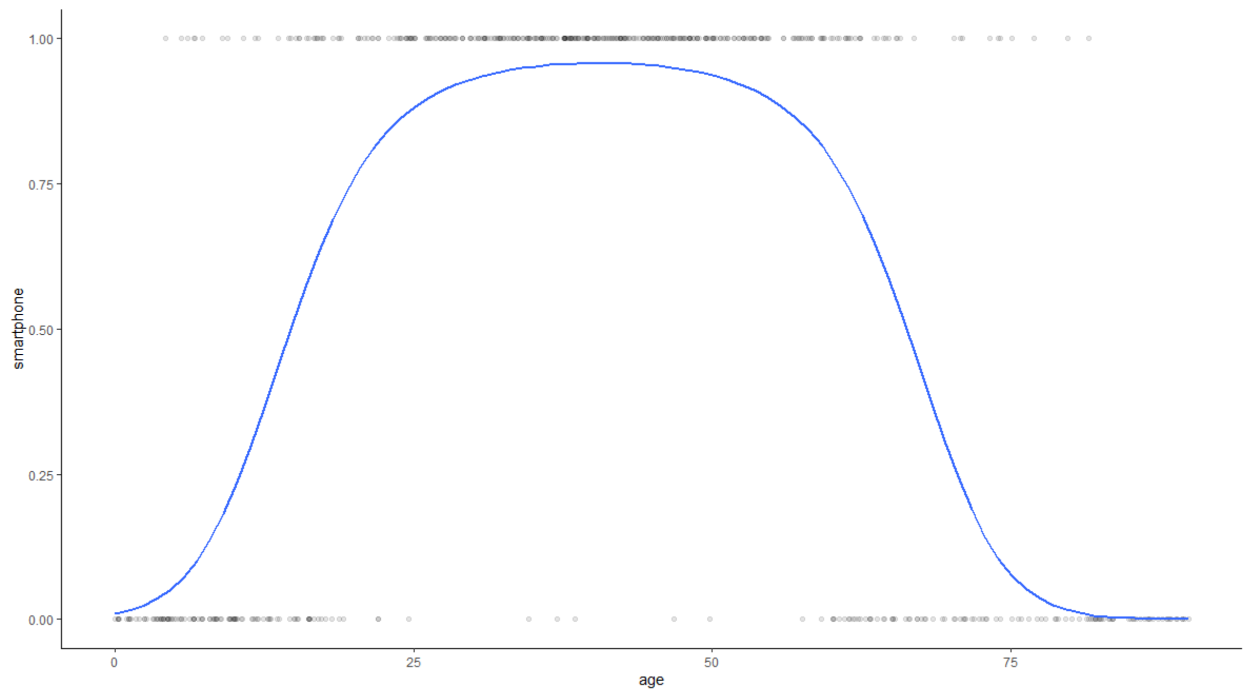
[1]Note: maybe add more variance-covariance matrices for sensitivity analyses later.



We use these 10 predictors to calculate the dichotomous outcome. The current DGM model includes one cubic term and 9 interactions. We also added one log-transformation (natural logarithm of the absolute value of the second predictor), and will add one (restricted cubic) spline.

Would the qubic term be sufficient instead of a spline, since this results in a 'U shaped' logistic curve?

That is, a quadratic relation between a predictor and a binary outcome looks like the figure below. You can imagine to fit a spline with 4 knots through this logistic curve. Is an explicitly simulated spline then still necessary?



For now, the outcome is calculated using one log-transformation, one cubic term, and 9 interactions.

```
# let's generate some data
n <- 10000
set.seed(11)
dat <- generate_sample(n, Sigma, interaction = TRUE)

# what do the data look like?
glimpse(dat)
```

```
## Rows: 10,000
## Columns: 11
## $ Y   <int> 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, ~
## $ X1  <dbl> -0.45477207, -0.85491670, -0.71086194, -1.07582917, -2.14555597, 0~
## $ X2  <dbl> -0.374718960, -0.571452611, -0.003440048, -0.905723246, 0.00508102~
## $ X3  <dbl> -2.1279703, -1.5639255, -0.2602677, 0.2749396, 1.0980441, -0.71557~
## $ X4  <dbl> -1.596110637, -0.507058683, 0.347096157, -1.515271218, -1.01265959~
## $ X5  <dbl> 1.262381131, -1.174347914, 0.244571781, -1.087862294, 0.813273507,~
## $ X6  <dbl> -1.219028700, -0.042354662, -0.137939052, 0.476734455, -0.20062500~
## $ X7  <dbl> 1.598916998, -0.002754757, -0.108245672, -0.667327888, -2.67499317~
## $ X8  <dbl> 0.35310919, 1.07824202, -0.98214962, 1.78197050, 0.78825071, 0.605~
## $ X9  <dbl> 0.05435534, -0.73785034, -0.35442131, -0.51400533, 0.93342584, 1.4~
## $ X10 <dbl> -1.03869527, -0.41379092, -1.19544113, -0.02858801, 0.63535024, 0.~
```

```
# save data
saveRDS(dat, file = "Data/dataset.RDS")
```

```
# check prevalence and auc
logistic <- fit_logistic(dat)
saveRDS(logistic, file = "Data/logistic_model.RDS")

# # fit random forest on the complete data
# rf <- fit_rf(dat)
# saveRDS(rf, file = "Data/rf_model.RDS")
```

The prevalence of the outcome is 0.4856. The C index/AUC of the logistic model (without interactions) is 0.75.

Incomplete Data

The next step is to ampute the complete set using several missing data patterns. **Note that this part of the simulation pipeline is still very much under construction.**

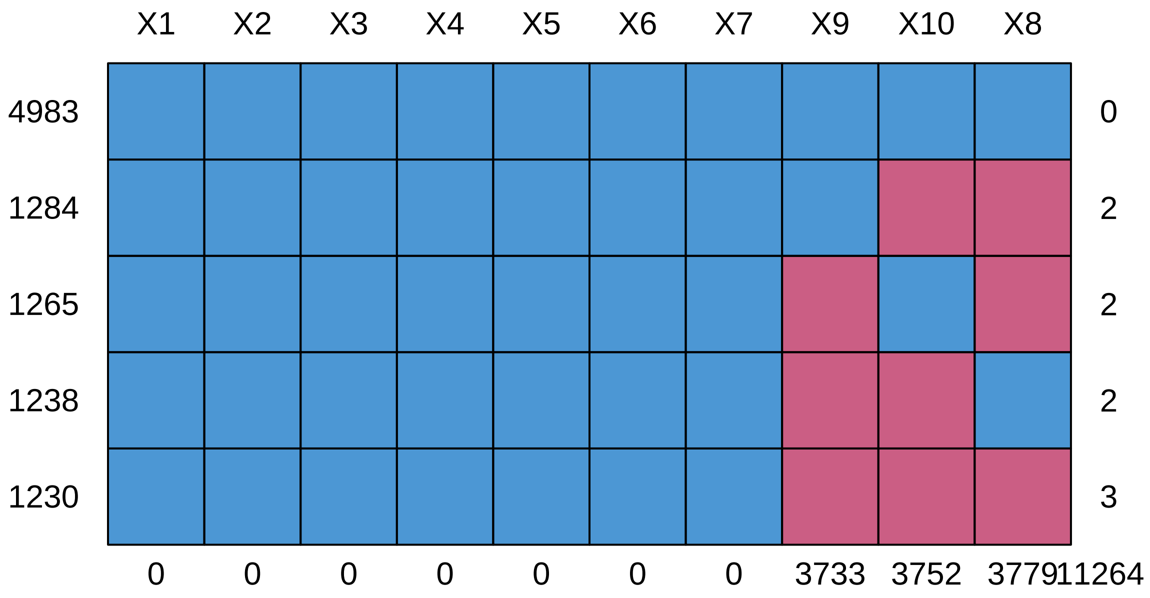
We will define one or more missing data patterns, which will be used to create incomplete datasets according to each of the three missingness mechanisms (MCAR, MAR, and MNAR).

We'll start with a missing data pattern where joint missingness occurs in 3 variables. 50-75% of the rows has at least one of the sets of 3 variables missing.

```
# create missing data pattern
pat <-  matrix(1, 4, 10) %>%
  data.frame()
# three var missing
pat[1,8:10] <- 0
# two var missing
pat[2,c(8, 9)] <- 0
pat[3,c(8, 10)] <- 0
pat[4,c(9, 10)] <- 0
# Q: add univariate missingness as well?
## Yes! Maar voor 10 patienten 1 missing

# 4, 6 of 8 pred ontbrekend per respondent

# create MCAR missingness --> MAR is het relevantste, nu nog wachten met MNAR
# MCAR is totaal niet realistisch, dus alleen als niks werkt onder MAR
MCAR <- dat[,-1] %>%
  mice::ampute(mech = "MCAR", prop = 0.5, patterns = pat)
MCARmd <- mice::md.pattern(MCAR$amp)
```



```
# # now redundant: what is the strongest predictor?
# dat %>%
#   glm(Y ~ ., family = "binomial", data = .) %>%
#   broom::tidy() %>%
#   dplyr::arrange(desc(abs(estimate)))
```

Discussion topics: Missingness conditions

- Missingness mechanisms: de val set incompleet maken adhv MAR (want MCAR is niet realistisch, en MNAR is te ingewikkeld zonder de MAR performance te weten) –> mixture van verschillende MAR patronen –> mixen over kolommen: een kolom is altijd MAR right
- Proportion of incomplete cases: voor alle individuen in de praktijk ontbreekt er iets.
- Number of predictors with missingness: 4, 6 of 8 van de 10 missend binnen 1 set
- Missingness in 'simple' vs 'complex' predictors

Does it matter if we develop the logistic regression prediction model on the complete dataset or the imputed incomplete set?

What we do know:

- The training data comes from the same population as the hold out set.
- We compare the methods at the prediction level.

What we don't know:

- Would we need 4 datasets then?

- Does that mean that we fit 4*3 prediction models?
- Should we create 2 missing data patterns (i.e., just missingness in the ‘regular’ predictors vs. in the non-linear predictors)?

Estimands

First, we calculate the absolute outcome risk according to different strategies for dealing with missing values.

Then we evaluate the predicted outcome risk against the original outcome risk (i.e. the absolute outcome risk that would be obtained if we have no missing values). Calculate:

- Root mean square prediction error;
- Brier score (predicted risk vs observed outcome).

Finally, across all hold-out patients:

- Calibration of (predicted Y | one or more missing predictor values) versus (predicted Y | original predictor values);
- Discrimination of (predicted Y | one or more missing predictor values) versus (predicted Y | original predictor values);
- Visual inspection of calibration plot (see plot Gary Collins in Stat med paper min 100 events).

Prediction Models

Methods for prediction model development:

- The prediction model is a flexible regression model (including non-linear effects using RCS (restricted cubic splines) with 4 knots and interaction terms). We also store the means and covariance of all predictor variables (which can be used to generate imputations).
- The prediction model is a “box” of submodels: a flexible regression model is developed for each possible combination of available predictors. If we have 10 predictor variables, this means that we would have to fit 1 + 10 + 45 + 120 + 210 + 252 + 252 + 210 + 120 + 45 + 10 + 1 = 1276 regression models. However, there is no need to estimate all these models. We can first look in the hold-out sample what variable is missing, and then estimate the necessary “submodel”.
- The prediction model is a random forest, as implemented by cforest() in the R package party. We generate a certain number of surrogate splits for each node. These splits attempt to mimic the primary split, and thus to achieve similar separation using another (observed) variable. By default set 4 surrogate splits (since we set max 3 missings).

Methods for generating absolute risk predictions:

- (Only for strategy 1 and 3): Missing values are imputed by their conditional mean => This strategy should work just fine.
- (Only for strategy 1 and 3): Missing values are imputed by a random draw from their conditional multivariate distribution
- (Only for strategy 1 and 3): Missing values are imputed 50 times by a random draw from their distribution. The resulting 50 absolute risk predictions are then averaged to obtain the final prediction.
- (Only for strategy 2): The appropriate pattern submodel is selected for calculating an absolute risk
- (Only for strategy 3): Missing values are handled using the surrogate splits