TUG Predictive Analytics Report

Chong H. Kim

07 September, 2018

Introduction

This is the markdown file that explains the steps for the time to up and go predictive analysis.

Steps are as follows:

1. Data Import and Check
2. Use Brokenstick model to predict 90 day timed-up-and-go (TUG)
3. Predict 90 day TUG based om preoperative markers (i.e. age, gender, and etc) using predictive mean matching (i.e. linear model) in order to match patients that have closest prediction. These 90 day TUG is obtained only for the purposes of selecting patients for the GAMLSS regression for "close-to-me" matched regression.
4. GAMLSS regression based on n-matches selected. The n will be selected based on cross validation. Additionally, the GAMLSS hyperparameters will be selected based on cross validation.

Next steps will be adjusted according to our wednesday meetings.

Methods

Data Import and Check

We need a few libraries to be loaded. Specifically `gamlss` and `brokenstick` are the libraries that will be used for predicting the TUG and `dplyr` and `data.table` for data import.

Import Data

```
rm(list=ls())
library(gamlss)
library(brokenstick)
library(dplyr)
library(data.table)
library(readxl)
library(memoise)
full  <- readxl::read_xlsx("../data/TUG_070118.xlsx")
```

```
head(full)
```

```
## # A tibble: 6 x 9
##   patient_id train_test dataset_id  time   bmi gender   age   tug b_tug
##        <dbl>      <dbl>      <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1          1          1          1   -11  22.1      2    65  7.59  7.59
## 2          1          1          1    22  22.1      2    65 11.1   7.59
## 3          1          1          1    52  22.1      2    65  8.37  7.59
## 4          1          1          1   100  22.1      2    65  7.86  7.59
## 5          1          1          1   196  22.1      2    65  7.62  7.59
## 6          1          1          1   378  22.1      2    65  8.09  7.59
```

```
summary(full)
```

```
##    patient_id      train_test      dataset_id         time
##  Min.   :  1.0   Min.   :1.000   Min.   :1.000   Min.   :-339.00
##  1st Qu.:128.0   1st Qu.:1.000   1st Qu.:3.000   1st Qu.:  15.00
##  Median :264.5   Median :1.000   Median :3.000   Median :  45.00
##  Mean   :275.3   Mean   :1.282   Mean   :3.357   Mean   :  83.48
##  3rd Qu.:423.0   3rd Qu.:2.000   3rd Qu.:4.000   3rd Qu.:  90.00
##  Max.   :605.0   Max.   :2.000   Max.   :6.000   Max.   :1182.00
##       bmi            gender            age             tug
##  Min.   :17.85   Min.   :1.000   Min.   :18.26   Min.   :  1.80
##  1st Qu.:27.15   1st Qu.:1.000   1st Qu.:59.00   1st Qu.:  7.06
##  Median :30.70   Median :2.000   Median :65.00   Median :  8.62
##  Mean   :31.36   Mean   :1.542   Mean   :64.71   Mean   : 10.78
##  3rd Qu.:35.10   3rd Qu.:2.000   3rd Qu.:70.00   3rd Qu.: 10.94
##  Max.   :56.24   Max.   :2.000   Max.   :88.88   Max.   :452.00
##      b_tug
##  Min.   : 4.190
##  1st Qu.: 7.303
##  Median : 8.970
##  Mean   :10.127
##  3rd Qu.:11.342
##  Max.   :59.100
```

Clean Data

Here we want to remove rows where key measures are missing. note: it would be nice to automate this in the future where the user specifies the outcomes of interest and the timeframe

```
# remove na TUG
# make 3 datasts, traininig (pre and post), test
train <- full %>% filter(train_test == 1)
test <- full %>% filter(train_test == 2)
```

```
# quick data check
# make sure there aren't any missing data and etc
summary(full) ;  sapply(full, function(x) {
                          table(is.na(x))})
```

```
##    patient_id      train_test      dataset_id         time
##  Min.   :  1.0   Min.   :1.000   Min.   :1.000   Min.   :-339.00
##  1st Qu.:128.0   1st Qu.:1.000   1st Qu.:3.000   1st Qu.:  15.00
##  Median :264.5   Median :1.000   Median :3.000   Median :  45.00
##  Mean   :275.3   Mean   :1.282   Mean   :3.357   Mean   :  83.48
##  3rd Qu.:423.0   3rd Qu.:2.000   3rd Qu.:4.000   3rd Qu.:  90.00
##  Max.   :605.0   Max.   :2.000   Max.   :6.000   Max.   :1182.00
##       bmi            gender            age             tug
##  Min.   :17.85   Min.   :1.000   Min.   :18.26   Min.   :  1.80
##  1st Qu.:27.15   1st Qu.:1.000   1st Qu.:59.00   1st Qu.:  7.06
##  Median :30.70   Median :2.000   Median :65.00   Median :  8.62
##  Mean   :31.36   Mean   :1.542   Mean   :64.71   Mean   : 10.78
##  3rd Qu.:35.10   3rd Qu.:2.000   3rd Qu.:70.00   3rd Qu.: 10.94
##  Max.   :56.24   Max.   :2.000   Max.   :88.88   Max.   :452.00
##      b_tug
##  Min.   : 4.190
##  1st Qu.: 7.303
##  Median : 8.970
##  Mean   :10.127
##  3rd Qu.:11.342
##  Max.   :59.100
```

```
## patient_id.FALSE train_test.FALSE dataset_id.FALSE       time.FALSE
##             2930             2930             2930             2930
##        bmi.FALSE     gender.FALSE        age.FALSE        tug.FALSE
##             2930             2930             2930             2930
##      b_tug.FALSE
##             2930
```

```
# train pre data
train_pre <- train %>%
    filter(time < 0)
# train post data
train_post  <- train %>%
    filter(time > 0)
```

Brokenstick model

Here we'll set up a few knots where we want to estimate the y day post `tug` based on `time`

```
knots <- c(0, 20, 55, 90)

fit <- brokenstick(y = train_post$tug,
                   x = train_post$time,
                   subjid = train_post$patient_id,
                   knots = knots)

est1<-predict(fit, at="knots")
head(est1)
```

```
##   subjid    x  y      yhat knot
## 1      1    0 NA 75.135074 TRUE
## 2      1   20 NA 11.327329 TRUE
## 3      1   55 NA  8.255861 TRUE
## 4      1   90 NA  7.769774 TRUE
## 5      1 1182 NA  7.312630 TRUE
## 6      2    0 NA 11.039865 TRUE
```

```
#extract fitted outcome Y90
train_all <- left_join(
                    left_join(
                              # 90 day covariates
                              est1[est1$x==90,] %>%
                                  rename(y90 = yhat,
                                         patient_id = subjid) %>%
                              mutate(patient_id  = as.numeric(patient_id))
                              ,
                              # early covariates
                              est1[est1$x==20,] %>%
                                  rename(y20 = yhat,
                                         patient_id = subjid) %>%
                              mutate(patient_id  = as.numeric(patient_id))
                          ,
                          by="patient_id"
                              )  %>%
                    # select only the relevant variables
                    dplyr::select(patient_id, y90, y20) %>%
                    mutate(patient_id = as.numeric(patient_id))
                ,
                # preoperative data for baseline info like age bmi, kaf
                train_pre
                , by = "patient_id"
                ) %>%
# select those without missing variables and age above 20
filter(!is.na(tug), !is.na(y90), !is.na(age), age > 20) %>%
mutate(gender = as.factor(gender)) %>%
dplyr::select(patient_id, age, bmi, gender, tug, time,
       y20, y90)
head(train_all); dim(train_all) # 1 patient lost from the filter ... too bad
```

```
##   patient_id      age      bmi gender   tug time       y20       y90
## 1          1 65.00000 22.06783      2  7.59  -11 11.327329  7.769774
## 2        112 56.00000 36.11000      2  7.00   -1 24.087250 10.238241
## 3        223 69.00000 31.87000      1 11.81   -4  9.723917  8.476251
## 4        334 68.67945 42.06708      2 22.51  -35 10.744596  6.056926
## 5        349 60.69041 56.24185      2 17.04   -4 17.766901 10.953105
## 6        360 70.99726 32.95222      2  9.67  -36 15.966988 10.518881
```

```
## [1] 402   8
```

Predictive Mean Matching and GAMLSS

```
pmm<-lm(y90~tug + age , data=train_all)
pmmg<-gamlss(y90~tug + age, family = NO , data=train_all)
```

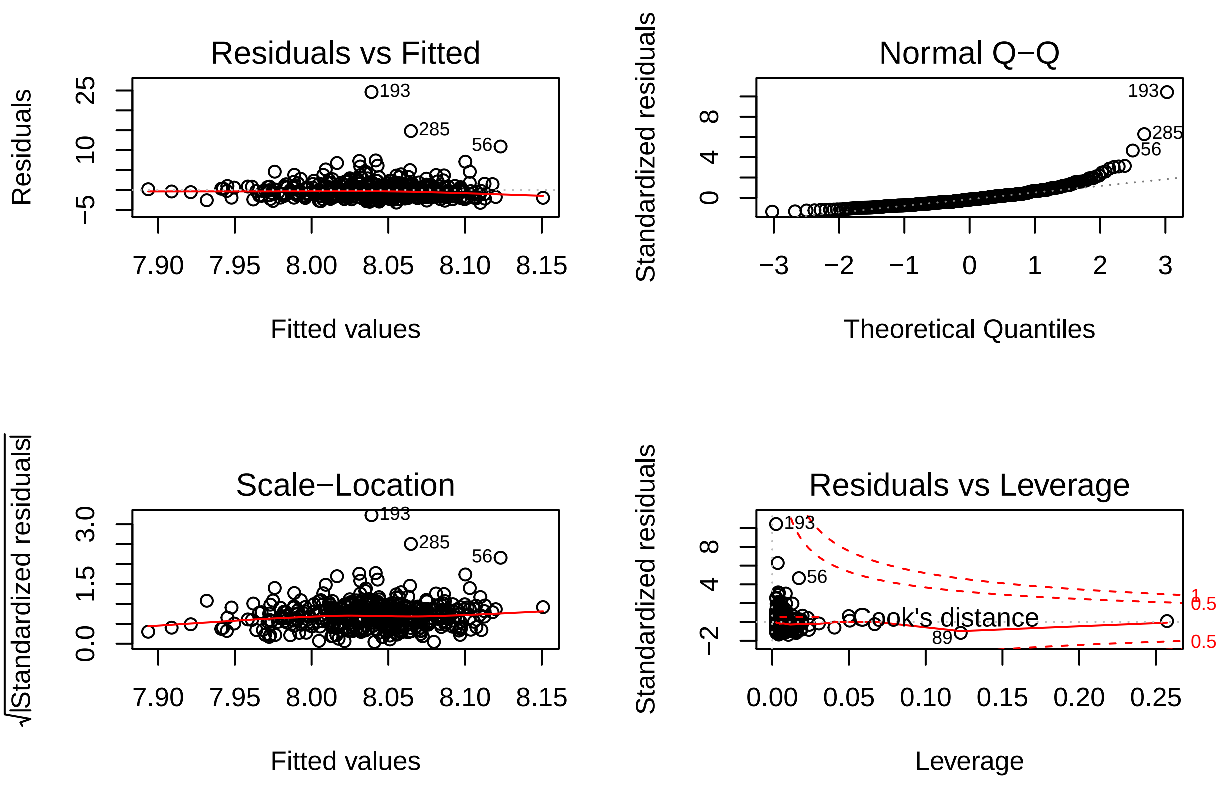
```
## GAMLSS-RS iteration 1: Global Deviance = 1829.609
## GAMLSS-RS iteration 2: Global Deviance = 1829.609
```

```
summary(pmm)
```

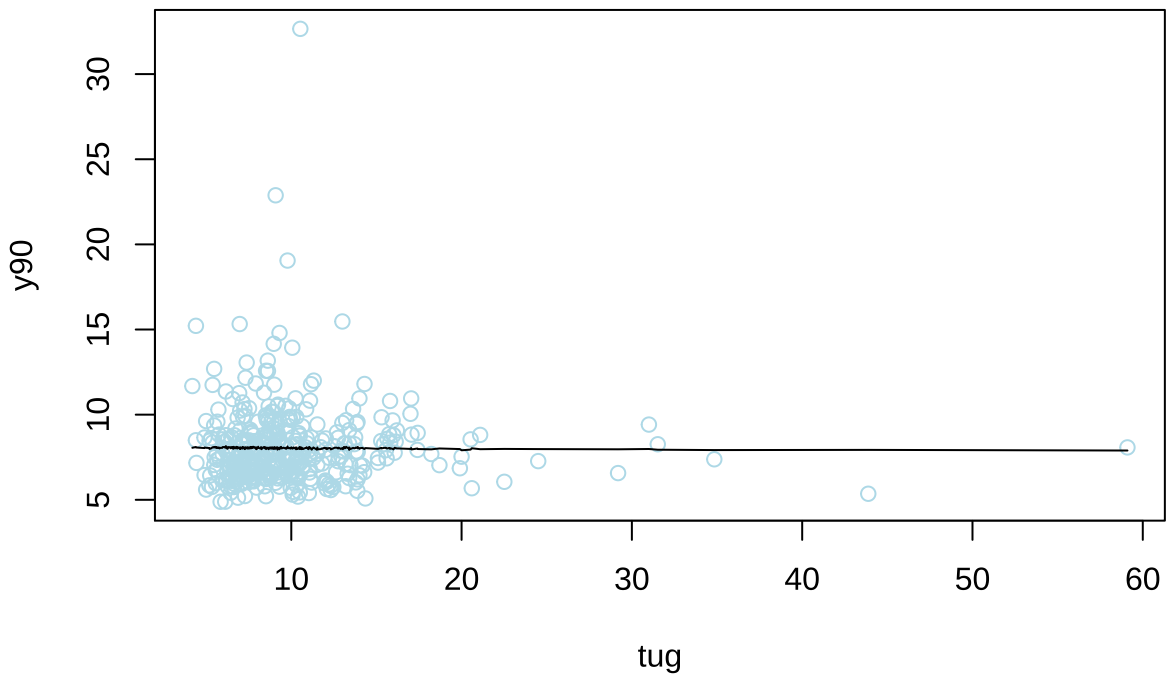
```
##
## Call:
## lm(formula = y90 ~ tug + age, data = train_all)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2217 -1.3731 -0.4048  0.7138 24.6177
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.338764   0.939550   8.875   <2e-16 ***
## tug         -0.002554   0.024559  -0.104    0.917
## age         -0.004263   0.014846  -0.287    0.774
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.364 on 399 degrees of freedom
## Multiple R-squared:  0.0002786,  Adjusted R-squared:  -0.004733
## F-statistic: 0.05559 on 2 and 399 DF,  p-value: 0.9459
```

Although the `R-sq` isn't perfect, this `pmm` model is used only to get at selecting patients with similar predicted outcomes. We can try a stepwise AIC variable selection procedure… the model that fits best is a model with only an intercept…

```r
# diagnostics
par(mfrow=c(2,2))
plot(pmm)
```



```r
par(mfrow=c(1,1))
plot(y90~tug, col = "lightblue", data=train_all)
lines(fitted(pmmg)[order(train_all$tug)]~train_all$tug[order(train_all$tug)])
```



What to do?!?

Now add the fitted y90 valuse to our `train_all` dataset:

```
# extract fitted values for the linear model (PREOP only here)
train_all <- train_all %>%
        select(patient_id, y20, y90, tug, age) %>%
            cbind(pmm$fitted.values) %>%
                rename(id = patient_id, y20rom = y20,
                                  Fitted = `pmm$fitted.values`) %>%
    #dataset ordered by fitted values
    arrange(Fitted)
```

Let's check the reference model. We'll try using various distributions with GAIC as our model selection tool.

```
ref1<-gamlss(tug~cs(time^0.5, df=2),
             sigma.formula = ~cs(time^0.5, df=1),
             data=train_post, family=NO)
ref2<-gamlss(tug~pb(time),
             sigma.formula = ~pb(time),
             data=train_post, family=NO)
ref3<-gamlss(tug~pb(time),
             sigma.formula = ~pb(time),
             data=train_post, family=GA)
#ref4<-gamlss(tug~pb(time),
             #sigma.formula = ~pb(time),
             ## no nu
             #data=train_post, family=BCCG)
```

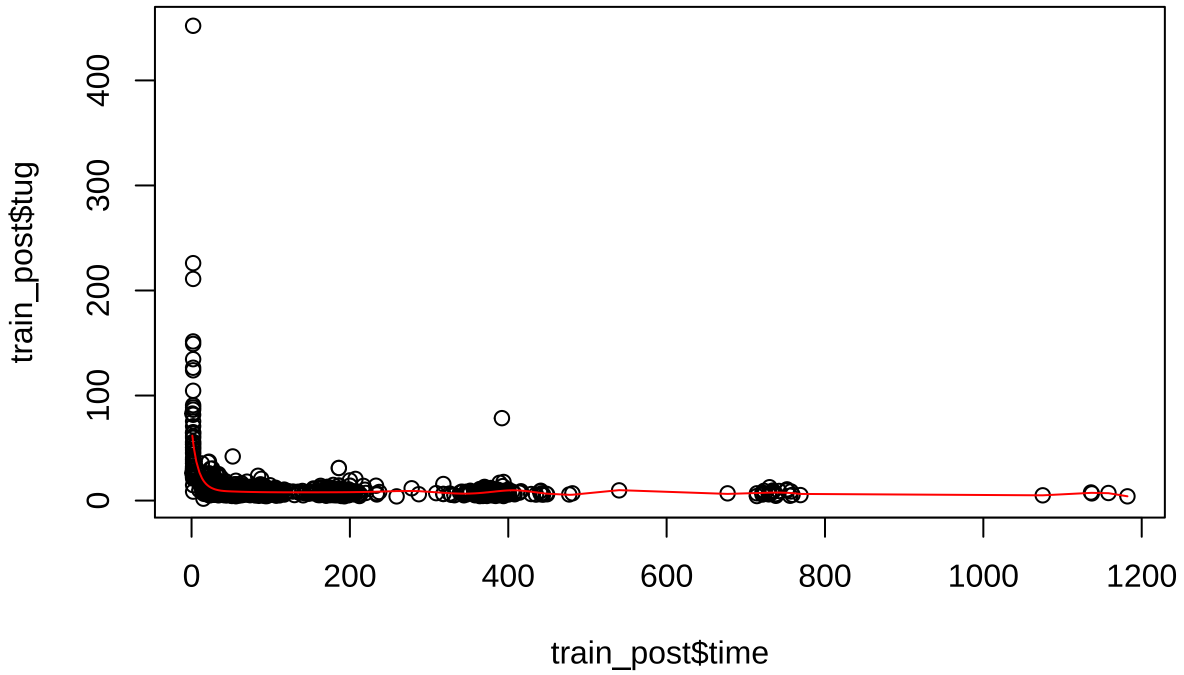
```r
#ref5<-gamlss(tug~pb(time),
             #sigma.formula = ~pb(time),
             ## yes nu
             #nu.formula = ~pb(time),
             #data=train_post, family=BCCG)
```

```r
#GAIC(ref1, ref2, ref3, ref4, ref5, k = log(length(train_post$patient_id)))
GAIC(ref1, ref2,ref3, k = log(length(train_post$patient_id)))
```

```
##              df       AIC
## ref3 25.003319  8560.779
## ref2 21.705223  9525.156
## ref1  7.000906 10145.563
```

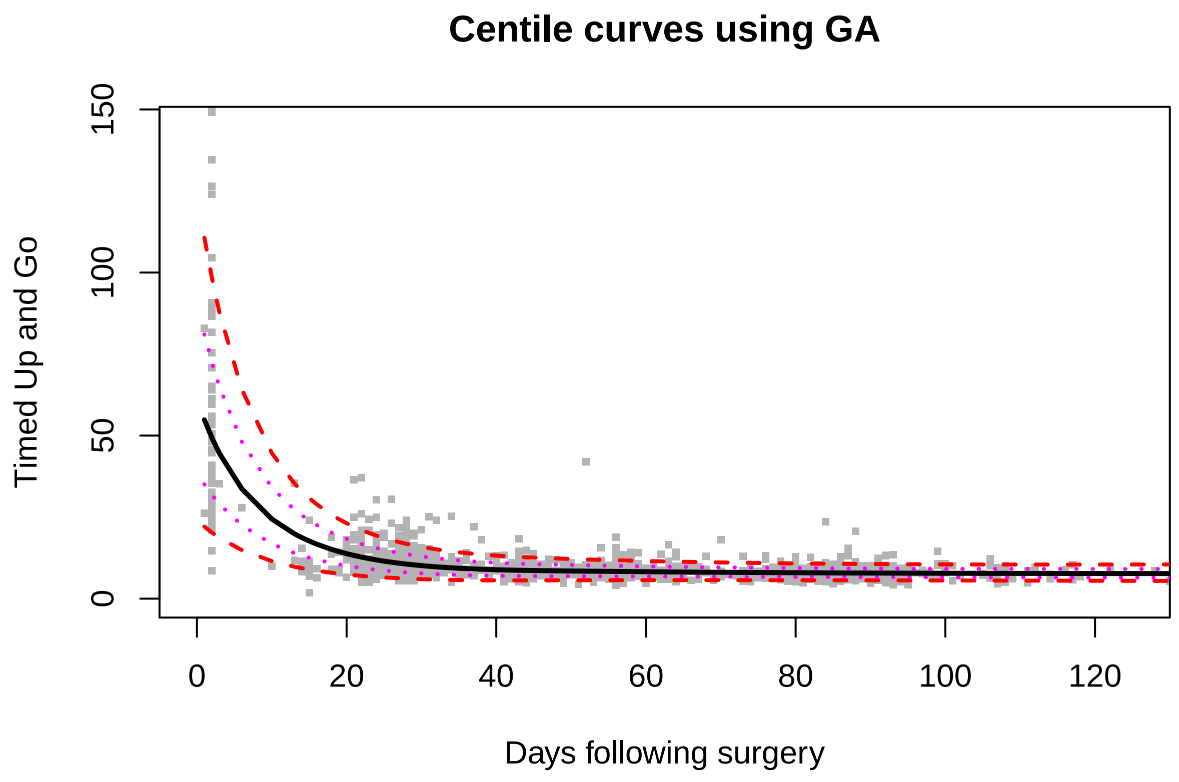
It seems that based on the fitting of both shape and scale smoothing functions, the `GA` distribution seems to fit the data the best. quick plot:

```r
plot(train_post$tug~train_post$time)
lines(fitted(ref3)[order(train_post$time)]~train_post$time[order(train_post$time)], col='red')
```



GIven that the data will probably be updating continuously, this process will also have to be configured to get the optimal fitting model.

```r
par(mfrow=c(1,1))
centiles(ref3, xvar=train_post$time, cent=c(10,25,50,75,90),
         ylab= "Timed Up and Go", xlab="Days following surgery",
         xleg=150, yleg=90, col.centiles = c(2,6,1,6,2), xlim=range(0,125),
         ylim=range(0,145),
         lty.centiles = c(2,3,1,3,2),
         lwd.centiles =c(2,2,2.5,2,2), points=T)
```

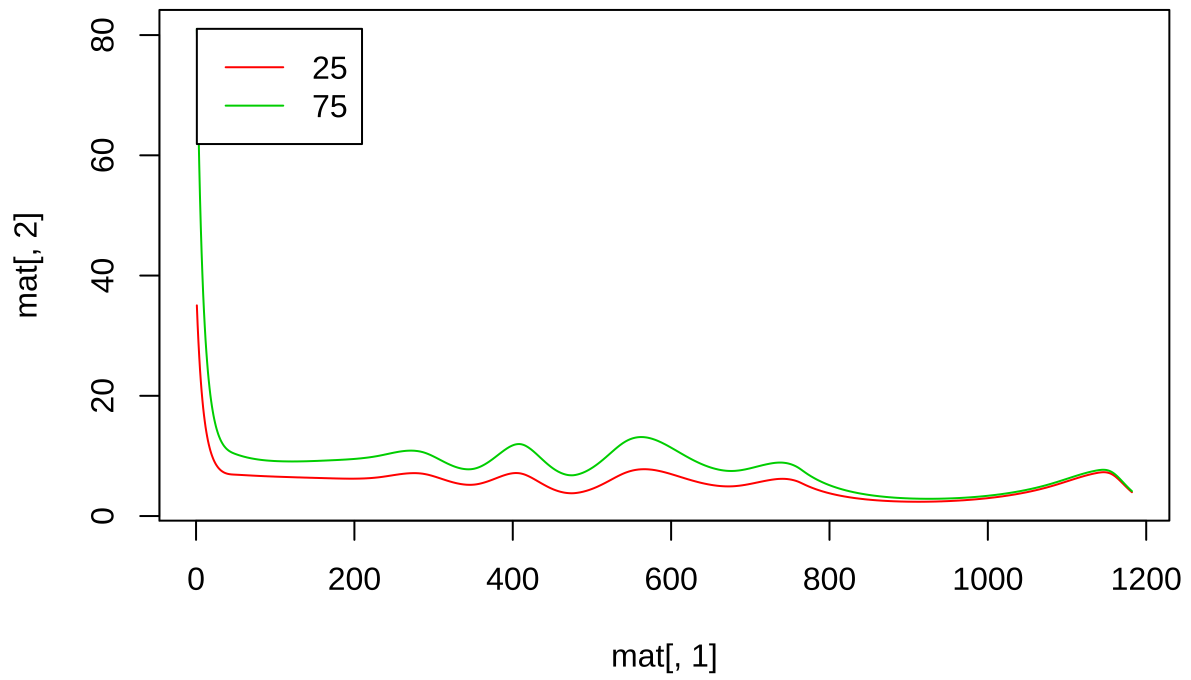


```
## % of cases below  10 centile is  5.768099
## % of cases below  25 centile is  24.1907
## % of cases below  50 centile is  54.38493
## % of cases below  75 centile is  80.28252
## % of cases below  90 centile is  92.23072
```

Here are the centiles for $10\%, 25\%, 50\%, 75\%, and 90\%$. Below is the IQR's.

```
mint<-min(train_post$time)
maxt<-max(train_post$time)
iqrfull<-centiles.pred(ref3, type="centiles",  xname = "time", xvalues=c(mint:maxt),
                   cent=c(25,75), plot=TRUE)
```

```
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
```



```
iqrfull$iqr<-iqrfull$C75-iqrfull$C25
mean(iqrfull$iqr)
```

```
## [1] 2.744069
```

Cross Validation

Now the goal is the fit a GAMLSS model to the training post operative data to:

1. Find the optimal number of matched patients (patient-like me) and
2. Find the optimal GAMLSS hyperparameters.

```
# Here is the function
```