# ddhazard Diagnostics

*Benjamin Christoffersen*

*2019-10-09*

## Introduction

This vignette will show examples of how the `residuals` and `hatvalues` functions can be used for an object returned by `ddhazard`. See `vignette("ddhazard", "dynamichazard")` for details of the computations. Firstly, we will present all the tools that turned out to be useful for data sets. This is in the section called "First round of checks". Then we return to the data sets using the other methods. The latter part is included to illustrate how the function in this package work. This is in the section called "Second round of checks".

## First round of checks

### Data set 1: Prisoner recidivism

The first data set we will look at is an experimental study of recidivism of 432 male prisoners a year after being released from prison. The details of the data set are in Rossi, Berk, & Lenihan (1980). The study involved randomly giving financial aid to the prisoners when they where released to see if this affected recidivism. The variables we will look at are:

- `fin`: 1 if the prisoner got aid and zero otherwise.
- `age`: age at time of release.
- `prio`: number of prior convictions.
- `employed.cumsum`: Cumulative number of weeks employed from the date of release. This will vary through time.
- `event`: 1 if the prisoner is rearrested.

A `.pdf` file called `Appendix-Cox-Regression.pdf` was previously on CRAN where they analyze this data set with the Cox regression model. They found:

- No very influential observation.
- No sign that the proportional-hazards assumption is violated. That is, no sign that the coefficients vary through time.
- Minor sign of non-linear effects.

#### Loading the data set

We load the data set with the next line. The details of how the `.RData` file is made is on the github site in the `vignettes/Diagnostics/` folder.

```
load("Diagnostics/Rossi.RData")

# We only keep some of the columns
Rossi <- Rossi[
  , c("id","start","stop","event", "fin", "age", "prio", "employed.cumsum")]
```

The data is in the typical start-stop form for Survival analysis. We print the number of individuals and show an example of how the data looks for one of the individuals:

```
# Number of unique individauls
length(unique(Rossi$id))
```

```
## [1] 432
```

```
# Show example for a given individual
Rossi[Rossi$id == 2, ]
```

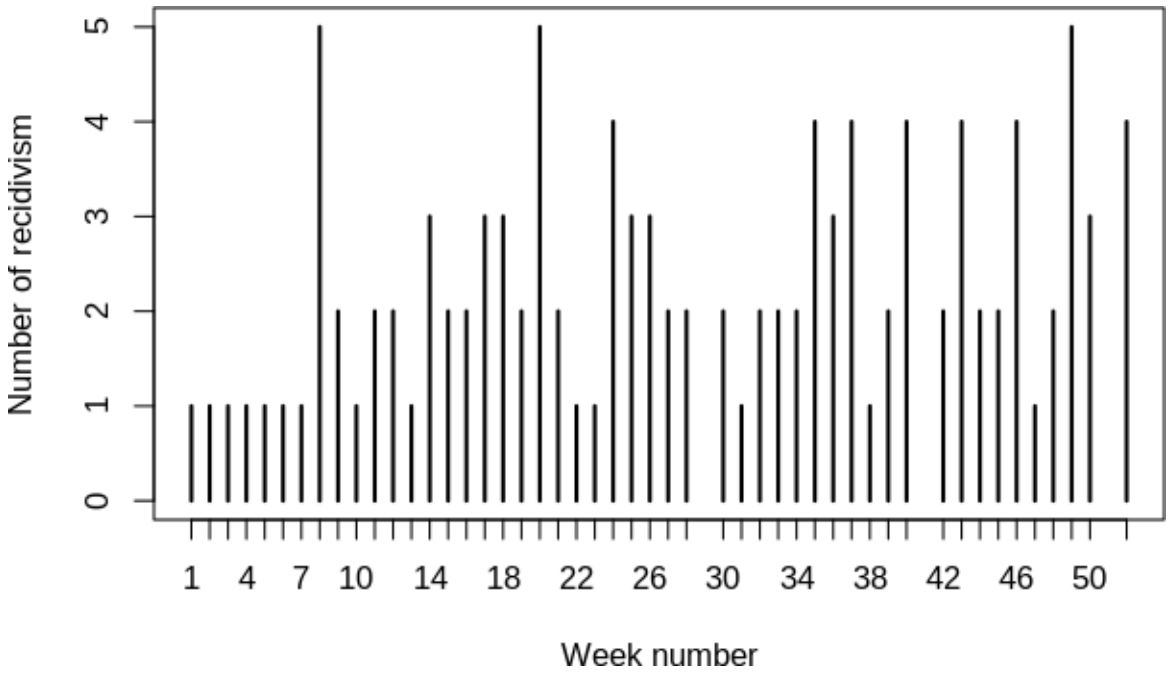
```
##    id start stop event fin age prio employed.cumsum
## 21  2     0    1     0   0  18    8                0
## 22  2     1    2     0   0  18    8                0
## 23  2     2    3     0   0  18    8                0
## 24  2     3    4     0   0  18    8                0
## 25  2     4    5     0   0  18    8                0
## 26  2     5    6     0   0  18    8                0
## 27  2     6    7     0   0  18    8                0
## 28  2     7    8     0   0  18    8                0
## 29  2     8    9     0   0  18    8                0
## 30  2     9   10     0   0  18    8                1
## 31  2    10   11     0   0  18    8                2
## 32  2    11   12     0   0  18    8                3
## 33  2    12   13     0   0  18    8                4
## 34  2    13   14     0   0  18    8                5
## 35  2    14   15     0   0  18    8                5
## 36  2    15   16     0   0  18    8                5
## 37  2    16   17     1   0  18    8                5
```

Next, we illustrate which of the variables are and which are not time-varying:

```
# See the varying and non-varying covariates
# The output shows the mean number of unique values for each individual
tmp <-
  by(Rossi[, ], Rossi$id, function(dat)
    apply(dat, 2, function(x) sum(!duplicated(x))))
colMeans(do.call(rbind, as.list(tmp)))
```

```
##             id          start           stop          event
##           1.00          45.85          45.85           1.26
##            fin            age           prio employed.cumsum
##           1.00           1.00           1.00          22.34
```

The events happens more less evenly across time after the first 8 weeks as the next plot shows. Hence, we may see an increasing intercept later since all individuals start at time zero. Thus, the risk sets only get smaller as time progress while roughly the same number of people gets rearrested:



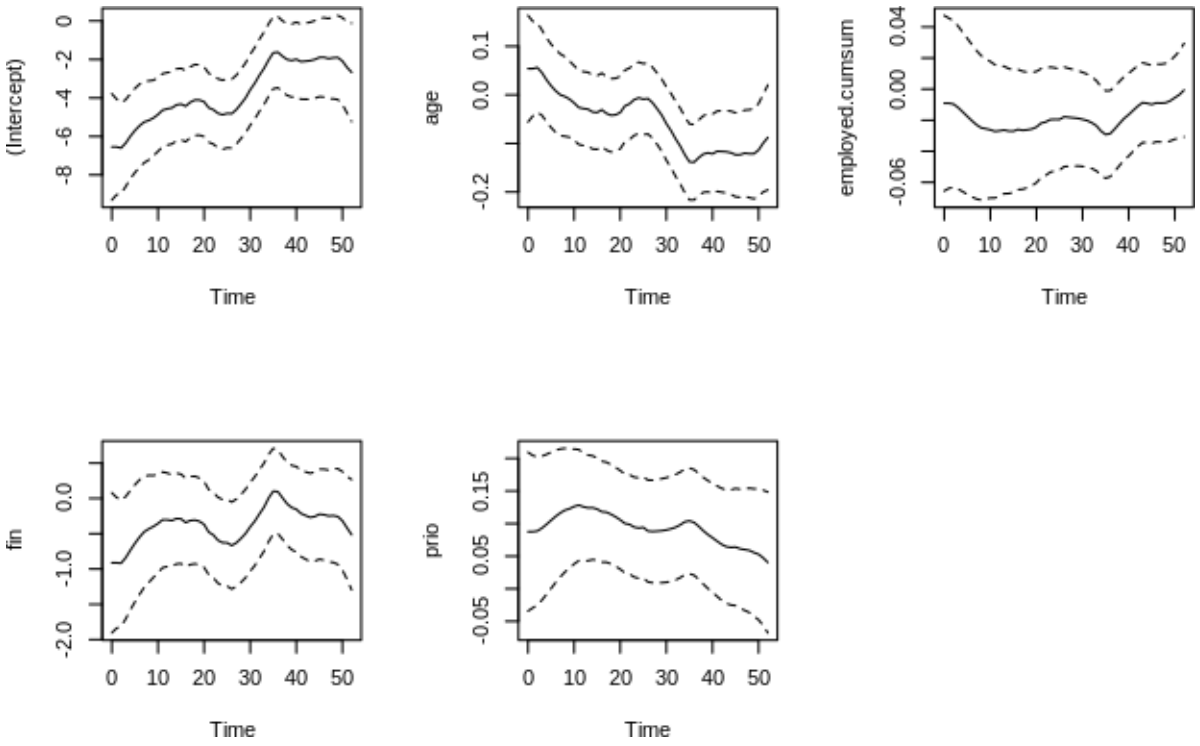

**Estimation**

We estimate the model with 4 terms and a intercept as follows:

```
library(dynamichazard)
dd_rossi <- ddhazard(
  Surv(start, stop, event) ~ fin + age + prio + employed.cumsum,
  data = Rossi, id = Rossi$id, by = 1, max_T = 52,
  Q_0 = diag(10000, 5), Q = diag(.01, 5),
  control = ddhazard_control(eps = .001, n_max = 250))
```

Then we plot the predicted coefficients:

```
plot(dd_rossi)
```




The dashed lines are 95% confidence bounds from the smoothed covariance matrices. Both the intercept and the age seems to have time-varying coefficients.

### Hat values

We start by looking at the "hat-like" values which are suggested in the ddhazard vignette. These are computed by calling the `hatvalues` function as follows:

```
hats <- hatvalues(dd_rossi)
```

The returned object is list with a matrix for each interval. Each matrix has a column for the hat values, the row number in the original data set and the id of the individual the hat values belongs to:

```
str(hats[1:3]) # str of first three elements
```

```
## List of 3
##  $ : num [1:432, 1:3] 0.000565 0.001602 0.004939 0.000512 0.000663 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "hat_value" "row_num" "id"
##  $ : num [1:431, 1:3] 0.000537 0.001491 0.004671 0.000477 0.000615 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "hat_value" "row_num" "id"
##  $ : num [1:430, 1:3] 0.000504 0.001555 0.004919 0.000495 0.000623 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "hat_value" "row_num" "id"
```

```
head(hats[[1]], 10) # Print the head of first matrix
```

| hat_value | row_num | id |
|---|---|---|

| hat_value | row_num | id |
|---|---|---|
| 0.001 | 1 | 1 |
| 0.002 | 21 | 2 |
| 0.005 | 38 | 3 |
| 0.001 | 63 | 4 |
| 0.001 | 115 | 5 |
| 0.000 | 167 | 6 |
| 0.001 | 219 | 7 |
| 0.001 | 242 | 8 |
| 0.001 | 294 | 9 |
| 0.001 | 346 | 10 |

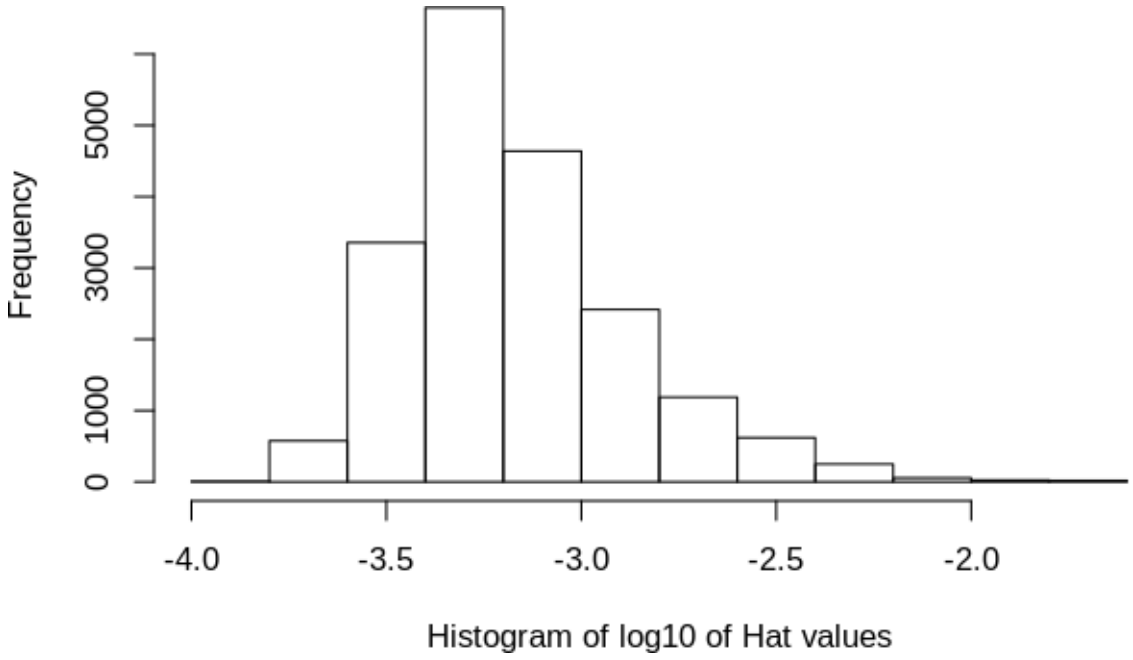
We have defined a function to stack the hat values for each individual called `stack_hats` such that we get a single matrix instead of list of matrices. Further, the function also adds an extra column for the interval number. The definition of the function is printed at the end of this vignette. The stacked values looks as follows:

```
hats_stacked <- stack_hats(hats)

head(hats_stacked)
```

```
##      hat_value row_num id interval_n
## 1     0.000565       1  1          1
## 433   0.000537       2  1          2
## 864   0.000504       3  1          3
## 1294  0.000479       4  1          4
## 1723  0.000464       5  1          5
## 2151  0.000455       6  1          6
```

The hat values are skewed with a few large values as shown in the histogram below:

```
# Draw histogram of hat values
hist(log10(hats_stacked$hat_value), main = "",
     xlab = "Histogram of log10 of Hat values")
```



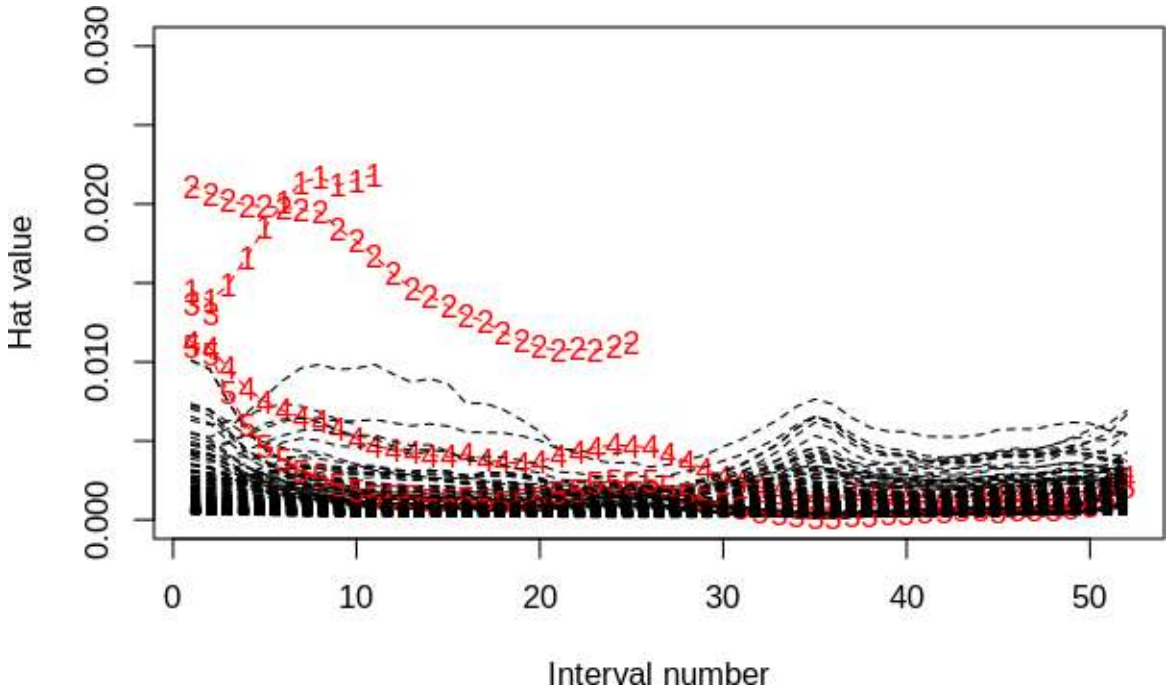

We find the maximum hat value for each individual and print the largest of them:

```
# Print the largest values
max_hat <- tapply(hats_stacked$hat_value, hats_stacked$id, max)
head(sort(max_hat, decreasing = T), 5)
```

```
##    317    242    101     64    376
## 0.0219 0.0211 0.0137 0.0113 0.0110
```

The names of the returned vector is the id of the individual. One seems to stand out. Next, we plot the hat values against time and highlight the individuals with the largest maximum hat value by giving them a red color and adding a number for the rank of their maximum hat value:

```
# We will highlight the individuals with the highest hatvalues
is_large <-
  names(head(sort(max_hat, decreasing = T), 5))

# Plot hat values
plot(range(hats_stacked$interval_n), c(0, 0.03), type = "n",
     xlab = "Interval number", ylab = "Hat value")

invisible(
  by(hats_stacked, hats_stacked$id, function(rows){
    has_large <- rows$id[1] %in% is_large
    col <- if(has_large) "Red" else "Black"
    lines(rows$interval_n, rows$hat_value, lty = 2,
          col = col)

    if(has_large){
      pch <- as.character(which(rows$id[1] == is_large))
      points(rows$interval_n, rows$hat_value, pch = pch, col = col)
    }
  }))
```




We print the last record for each of the above shown in red to get an idea of why their hat values are large:

```
# These are the individuals id
is_large
```

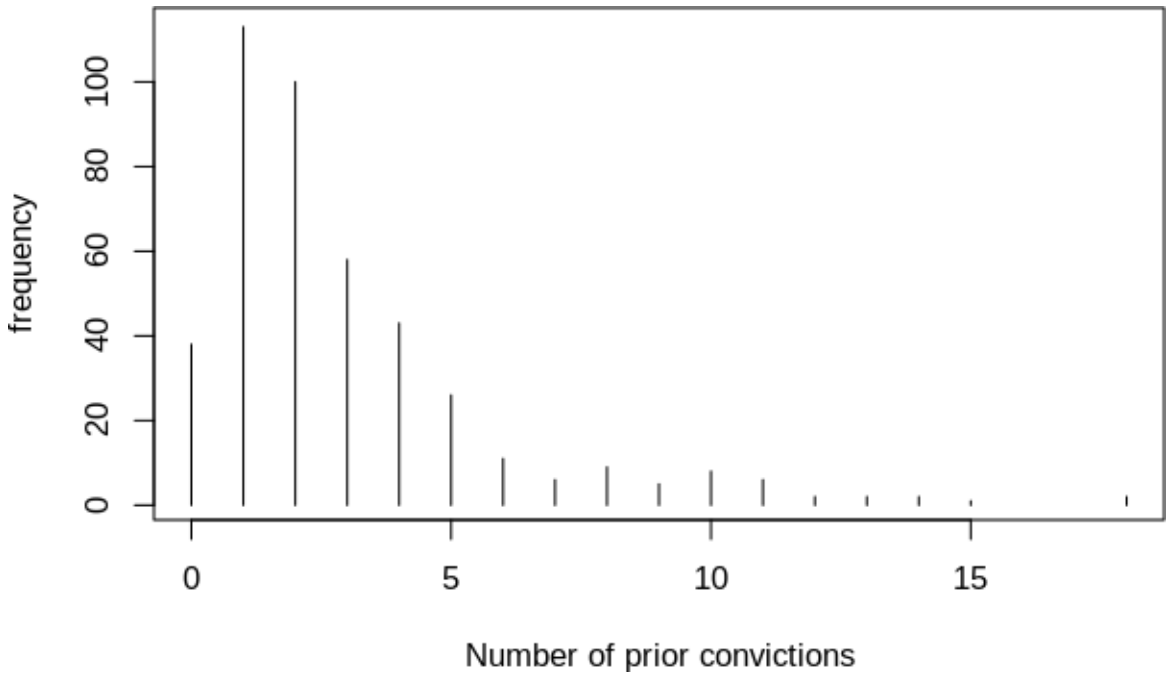
```
## [1] "317" "242" "101" "64"  "376"
```

```
# We print the last record each of these
Rossi_subset <- Rossi[
  unlist(sapply(is_large, function(x) which(x == Rossi$id))), ]
Rossi_subset <- Rossi_subset[nrow(Rossi_subset):1, ]
Rossi_subset[!duplicated(Rossi_subset$id), ]
```

```
##        id start stop event fin age prio employed.cumsum
## 17241 376    51   52     0   0  43    1              27
## 2981   64    51   52     0   0  36   11               4
## 4679  101     1    2     1   0  44    2               0
## 11204 242    24   25     1   0  28   18               0
## 14550 317    10   11     1   0  19   18               0
```

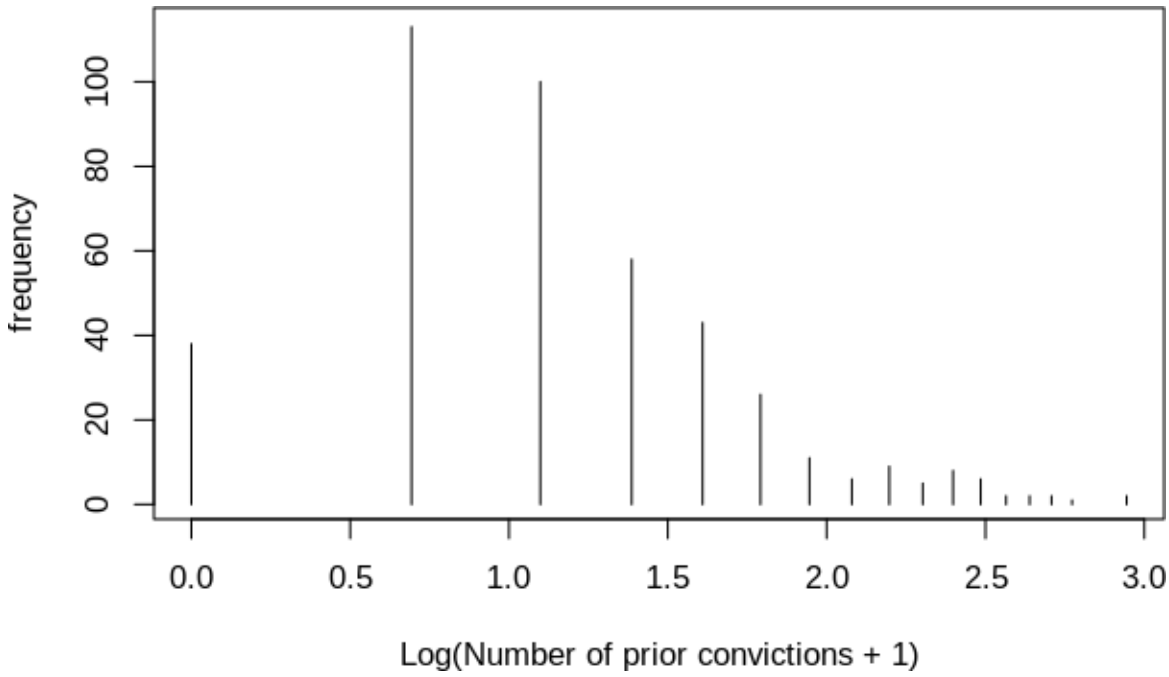
Some have a large number of prior convictions as shown in the next plot. Moreover, we can see by the next histogram that the number of prior convictions is skewed:

```
tmp <- xtabs(~Rossi$prio[!duplicated(Rossi$id)])
plot(as.numeric(names(tmp)), c(tmp), ylab = "frequency", type = "h",
     xlab = "Number of prior convictions")
```




This could suggest a transformation of the variables. Thus, we try with the logarithm of the value plus one with one added to avoid log(0) (with one arbitrarily chosen):

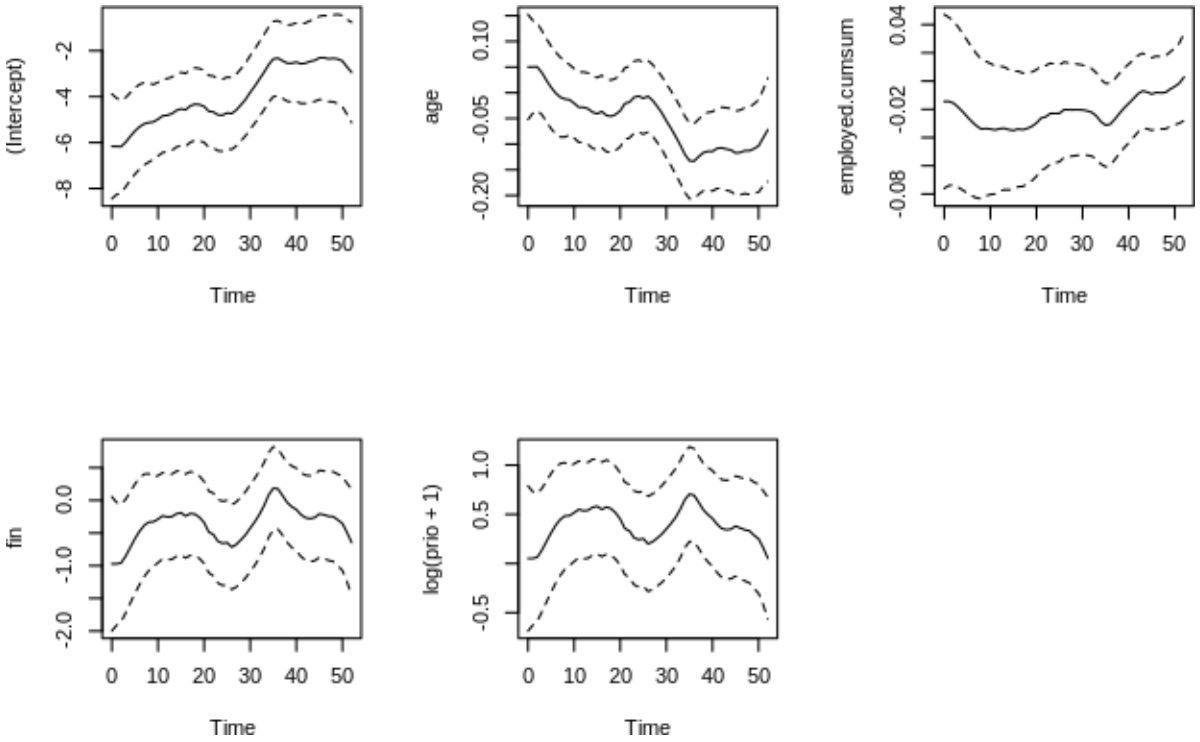
```
tmp <- xtabs(~log(Rossi$prio[!duplicated(Rossi$id)] + 1))
plot(as.numeric(names(tmp)), c(tmp), ylab = "frequency", type = "h",
     xlab = "Log(Number of prior convictions + 1)")
```



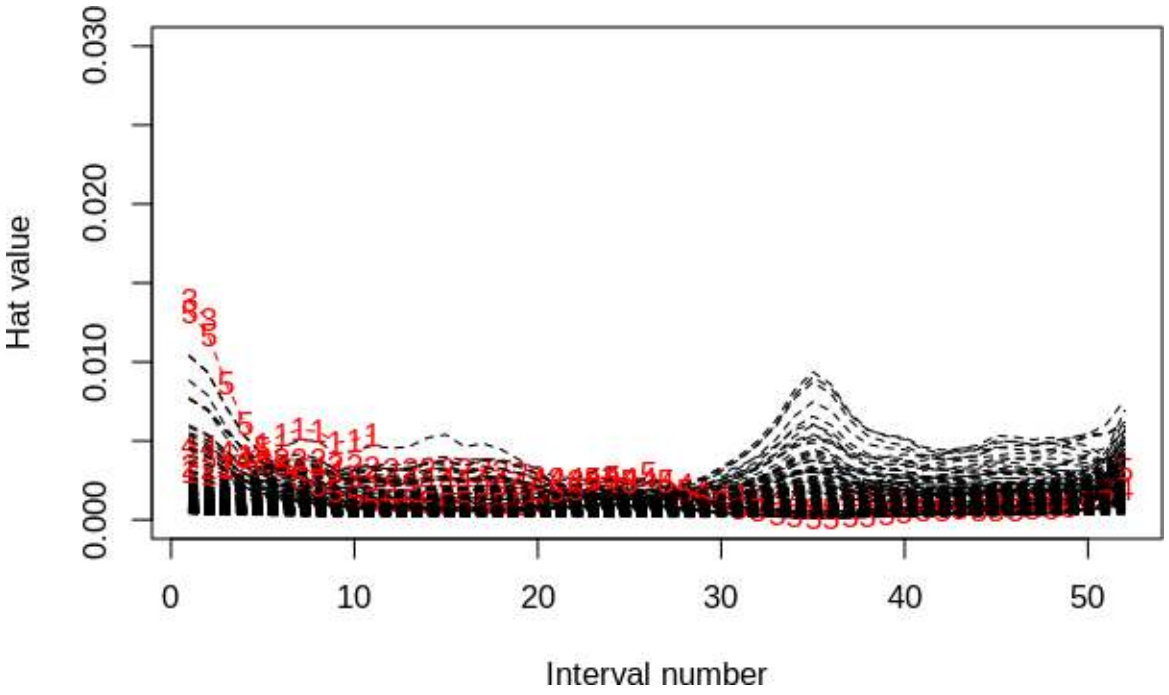

Below, we make a fit where we use this transformation of prior convictions instead:

```
dd_rossi_trans <- ddhazard(
  Surv(start, stop, event) ~ fin + age + log(prio + 1) + employed.cumsum,
  data = Rossi, id = Rossi$id, by = 1, max_T = 52,
```

```
  Q_0 = diag(10000, 5), Q = diag(.01, 5),
  control = ddhazard_control(eps = .001, n_max = 250))

plot(dd_rossi_trans)
```




computing the hat values and making a similar plot to the one before shows that the individuals are much less influential.




A question is whether the new model fits better. Thus, we compare the mean logistic loss of the two models

in-sample:

```
# Fit the two models
f1 <- ddhazard(
  Surv(start, stop, event) ~ fin + age + prio + employed.cumsum,
  data = Rossi, id = Rossi$id, by = 1, max_T = 52,
  Q_0 = diag(10000, 5), Q = diag(.01, 5),
  control = ddhazard_control(eps = .001, n_max = 250))

f2 <- ddhazard(
  Surv(start, stop, event) ~ fin + age + log(prio + 1) + employed.cumsum ,
  data = Rossi, id = Rossi$id, by = 1, max_T = 52,
  Q_0 = diag(10000, 5), Q = diag(.01, 5),
  control = ddhazard_control(eps = .001, n_max = 250))

# Compute residuals
res1 <- residuals(f1, type = "pearson")
res2 <- residuals(f2, type = "pearson")

# Compute logistic loss
log_error1 <- unlist(
  lapply(res1$residuals, function(x)
    ifelse(x[, "Y"] == 1, log(x[, "p_est"]), log(1 - x[, "p_est"]))))
log_error2 <- unlist(
  lapply(res2$residuals, function(x)
    ifelse(x[, "Y"] == 1, log(x[, "p_est"]), log(1 - x[, "p_est"]))))

# Compare mean
print(c(res1 = mean(log_error1), res2 = mean(log_error2)),
      digits = 8)
```

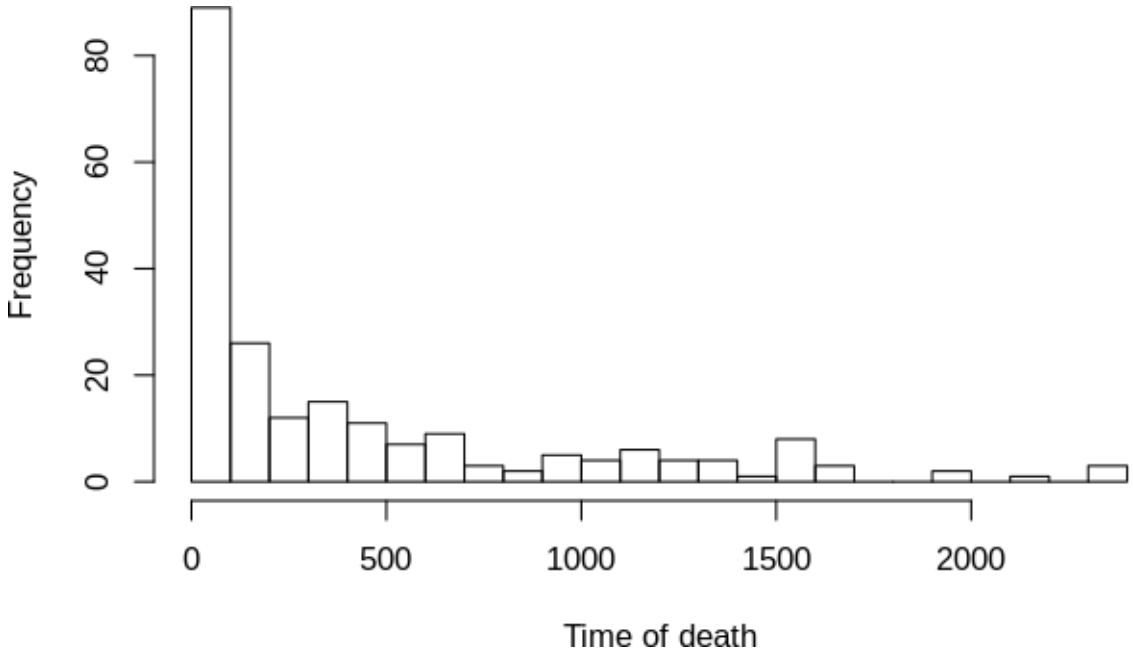
```
##         res1         res2
## -0.033830524 -0.033795251
```

The difference is very small.

## Data set 2: Worcester Heart Attack Study

Next, We will look at the Worcester Heart Attack Study. The dataset contains individuals who had a heart attack and is then followed up to check if they die within the following days. Below, we load the the data and plot the date of deaths:

```
load("Diagnostics/whas500.RData")

hist(whas500$lenfol[whas500$fstat == 1], breaks = 20,
     xlab = "Time of death", main = "")
```




The large peak at the start is due to a lot of individuals who dies doing the hospital stay shortly after their first heart attack. All covariates in the dataset are time-invariant records from the day that the individual had the first heart attack. We will look at gender, age, BMI, binary for whether the individual has a history of cardiovascular disease (`cvd`) and heart rate (`hr`). The first entries and summary stats are printed below:

```
# We only keep some of the columns
whas500 <- whas500[
  , c("id", "lenfol", "fstat", "gender",  "age", "bmi", "hr", "cvd")]

# First rows
head(whas500, 10)
```

```
##    id lenfol fstat gender age  bmi  hr cvd
## 1   1   2178     0      0  83 25.5  89   1
## 2   2   2172     0      0  49 24.0  84   1
## 3   3   2190     0      1  70 22.1  83   0
## 4   4    297     1      0  70 26.6  65   1
## 5   5   2131     0      0  70 24.4  63   1
## 6   6      1     1      0  70 23.2  76   1
## 7   7   2122     0      0  57 39.5  73   1
## 8   8   1496     1      0  55 27.1  91   1
## 9   9    920     1      1  88 27.4  63   1
## 10 10   2175     0      0  54 25.5 104   1
```

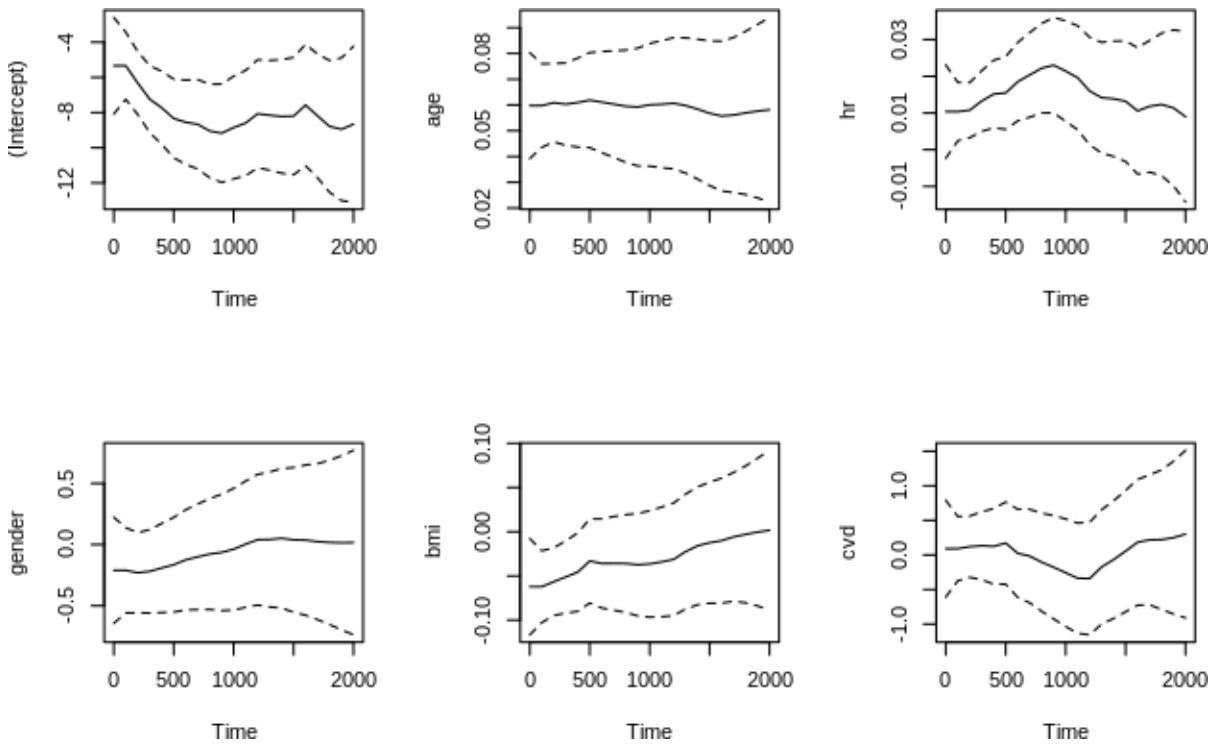
```
# Summary stats
summary(whas500[, c("age", "bmi", "hr", "gender",  "cvd")])
```

```
##       age             bmi             hr          gender         cvd
##  Min.   : 30.0   Min.   :13.0   Min.   : 35   Min.   :0.0   Min.   :0.00
##  1st Qu.: 59.0   1st Qu.:23.2   1st Qu.: 69   1st Qu.:0.0   1st Qu.:0.75
##  Median : 72.0   Median :25.9   Median : 85   Median :0.0   Median :1.00
##  Mean   : 69.8   Mean   :26.6   Mean   : 87   Mean   :0.4   Mean   :0.75
##  3rd Qu.: 82.0   3rd Qu.:29.4   3rd Qu.:100   3rd Qu.:1.0   3rd Qu.:1.00
##  Max.   :104.0   Max.   :44.8   Max.   :186   Max.   :1.0   Max.   :1.00
```

### Estimation

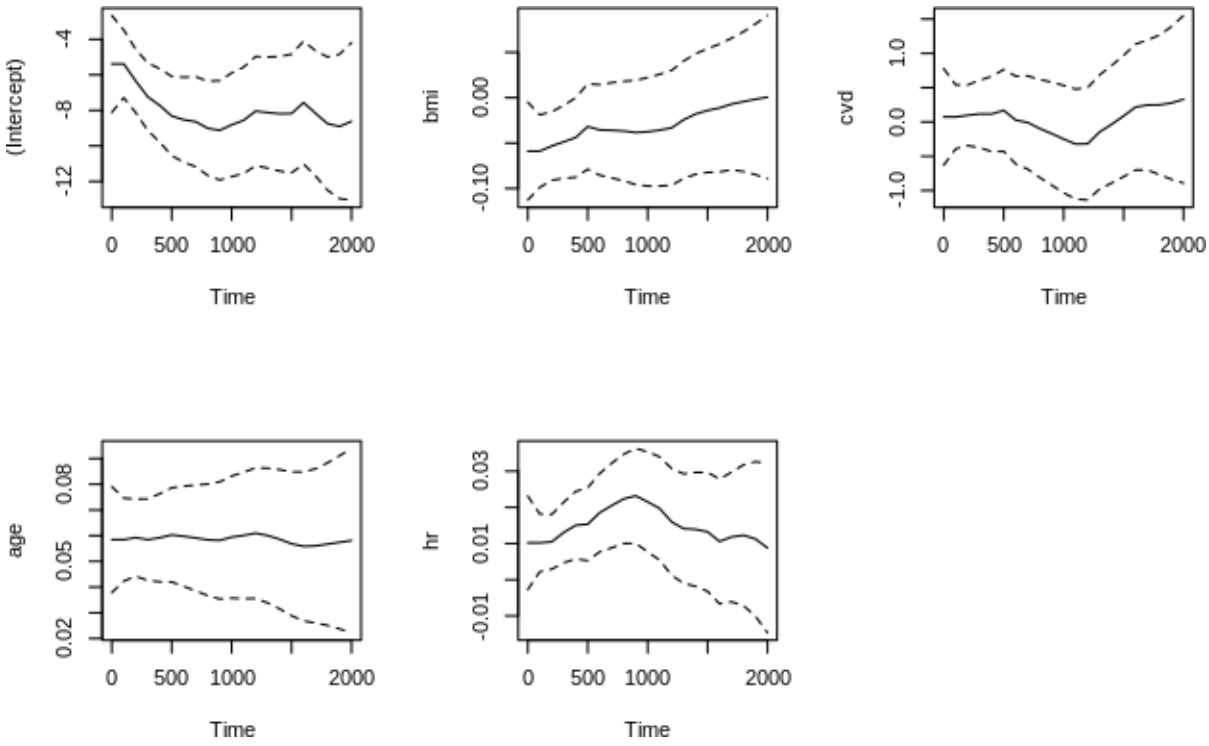
We estimate the model as follows:

```
dd_whas <- ddhazard(
  Surv(lenfol, fstat) ~ gender + age + bmi + hr + cvd,
  data = whas500, by = 100, max_T = 2000,
  Q_0 = diag(10000, 6), Q = diag(.1, 6),
  control = ddhazard_control(eps = .001))

plot(dd_whas)
```




The intercept drops in the first period which possibly is due to the initial high number of deaths right after the first heart attack. We further simplify the model the model by removing `gender` variable which seems to be zero:

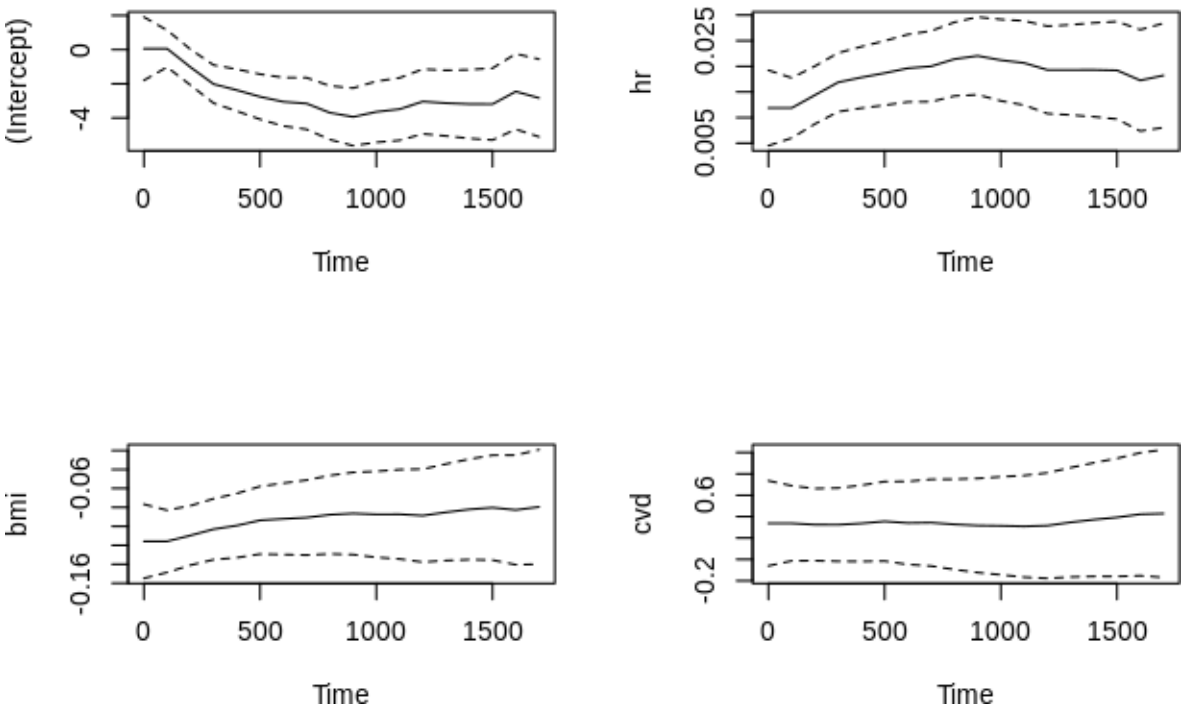
```
dd_whas <- ddhazard(
  Surv(lenfol, fstat) ~ age + bmi + hr + cvd,
  data = whas500, by = 100, max_T = 2000,
  Q_0 = diag(10000, 5), Q = diag(.1, 5),
  control = ddhazard_control(eps = .001))

plot(dd_whas)
```




### Leaving out a covaraite

Suppose that we had not included the age to start with:

```
dd_whas_no_age <- ddhazard(
  Surv(lenfol, fstat) ~ bmi + hr + cvd,    # No age
  data = whas500, by = 100, max_T = 1700,
  Q_0 = diag(10000, 4), Q = diag(.1, 4),
  control = ddhazard_control(eps = .001))

plot(dd_whas_no_age)
```




Then we could think that we would be able to see that the covariate was left out using the Pearson residuals as in a regular logistic regression. We compute the Pearson residuals to see if this would work:

```
obs_res <- residuals(dd_whas_no_age, type = "pearson")
```

The returned object is a list with two elements. One element denotes the type of the residuals and another contains the residuals. The latter is a list with a matrix for each interval. Each matrix has four columns for the residuals, the computed likelihood of an event, the outcome and the row number in the initial data matrix for those that were at risk in the interval. This is illustrated in the next lines:

```
# We have matrix for each interval
length(obs_res$residuals)
```

```
## [1] 17
```

```
# Shows the structure of the matrices. We only print take the first 5 matrices
str(obs_res$residuals[1:5])
```

```
## List of 5
##  $ : num [1:500, 1:4] -0.47 -0.498 -0.467 -0.383 -0.43 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "residuals" "p_est" "Y" "row_num"
##  $ : num [1:411, 1:4] -0.329 -0.345 -0.323 -0.261 -0.291 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "residuals" "p_est" "Y" "row_num"
##  $ : num [1:385, 1:4] -0.244 -0.252 -0.235 5.318 -0.207 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "residuals" "p_est" "Y" "row_num"
##  $ : num [1:360, 1:4] -0.2251 -0.2321 -0.2134 -0.1888 -0.0977 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
```

```
##   .. ..$ : chr [1:4] "residuals" "p_est" "Y" "row_num"
##  $ : num [1:305, 1:4] -0.2091 -0.2142 -0.1939 -0.1729 -0.0937 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "residuals" "p_est" "Y" "row_num"
```

```
# Print the first entries of the first interval
head(obs_res$residuals[[1]])
```

| residuals | p_est | Y | row_num |
|---|---|---|---|
| -0.470 | 0.181 | 0 | 1 |
| -0.498 | 0.199 | 0 | 2 |
| -0.467 | 0.179 | 0 | 3 |
| -0.383 | 0.128 | 0 | 4 |
| -0.430 | 0.156 | 0 | 5 |
| 2.011 | 0.198 | 1 | 6 |

The list of matrices is un-handy so we have defined a function called `stack_residuals` to stack the matrices, add the interval number and the id of that the residuals belong to in. The definition of the function is given at the end of this vignette.

```
resids_stacked <- stack_residuals(fit = dd_whas_no_age, resids = obs_res)

# print the first entries
head(resids_stacked, 10)
```

```
##      residuals  p_est Y row_num interval_n id
## 1       -0.470 0.1811 0       1          1  1
## 501     -0.329 0.0978 0       1          2  1
## 912     -0.244 0.0560 0       1          3  1
## 1297    -0.225 0.0482 0       1          4  1
## 1657    -0.209 0.0419 0       1          5  1
## 1962    -0.190 0.0349 0       1          6  1
## 2225    -0.187 0.0339 0       1          7  1
## 2474    -0.157 0.0240 0       1          8  1
## 2713    -0.144 0.0204 0       1          9  1
## 2949    -0.159 0.0248 0       1         10  1
```

Next, we add the age variable to the stacked residuals, stratify the age variable, compute cumulated mean across each stratum in each interval and plot against time:

```
# Add age variable
resids_stacked$age <-
  whas500$age[resids_stacked$row_num]

# Stratify
age_levels <- quantile(whas500$age, seq(0, 1, by = .2))
age_levels
```

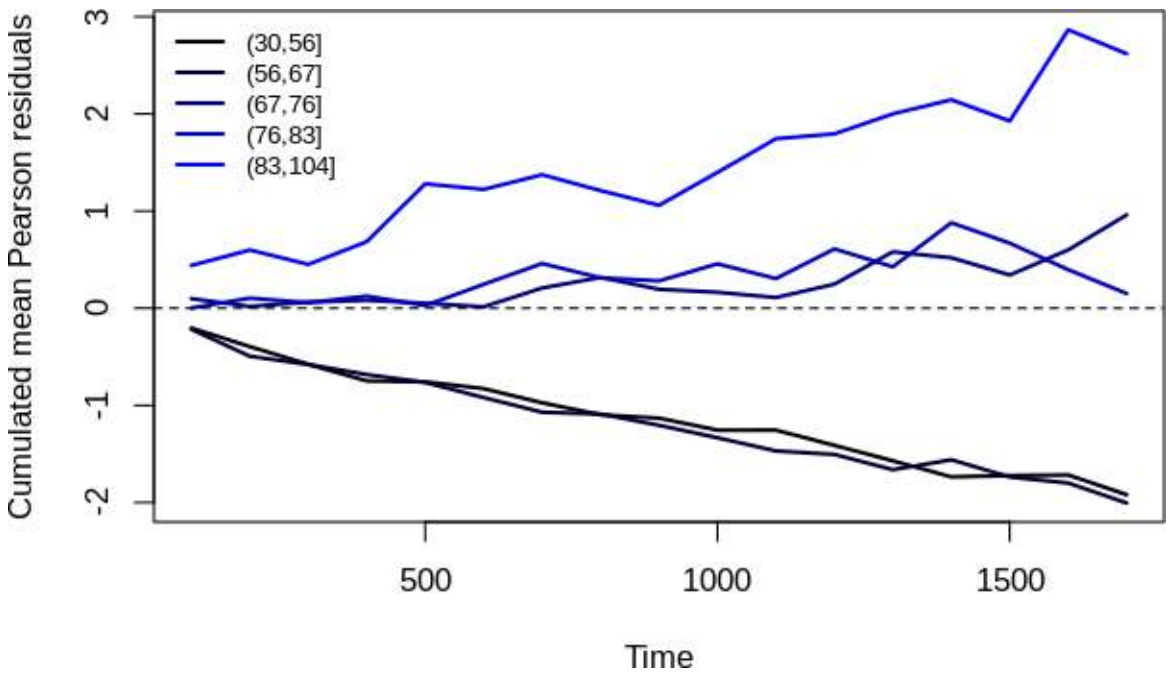
```
##   0%  20%  40%  60%  80% 100%
##   30   56   67   76   83  104
```

```
resids_stacked$age_cut <- cut(resids_stacked$age, age_levels)

# Compute the means
cut_means <-
```

```
  tapply(resids_stacked$residuals,
         list(resids_stacked$interval_n, resids_stacked$age_cut),
         mean)

head(cut_means)
```

| (30,56] | (56,67] | (67,76] | (76,83] | (83,104] |
|---|---|---|---|---|
| -0.205 | -0.217 | 0.099 | 0.002 | 0.440 |
| -0.190 | -0.277 | -0.081 | 0.102 | 0.159 |
| -0.183 | -0.085 | 0.045 | -0.047 | -0.148 |
| -0.170 | -0.103 | 0.019 | 0.068 | 0.234 |
| -0.010 | -0.082 | -0.031 | -0.098 | 0.595 |
| -0.070 | -0.156 | -0.037 | 0.219 | -0.057 |

```
# Plot against time
colfunc <- colorRampPalette(c("Black", "Blue"))
cols <- colfunc(ncol(cut_means))

matplot(dd_whas_no_age$times[-1], apply(cut_means, 2, cumsum),
        type = "l", col = cols, xlab = "Time", lwd = 2,
        lty = 1, ylab = "Cumulated mean Pearson residuals")
abline(h = 0, lty = 2)

legend("topleft", bty = "n",
       lty = rep(1, ncol(cut_means)),
       legend = colnames(cut_means),
       col = cols, lwd = 2,
       cex = par()$cex * .8)
```




We see that the older and youngest strata stand out and deviates from zero. Hence, suggesting that the age variable should have been in the model.

## Second round of checks

We will illustrate some other uses of the `residuals` function in this section. This part is included more to show how the function works as they do not "discover" anything new about the data sets.

### Residuals from state space vector

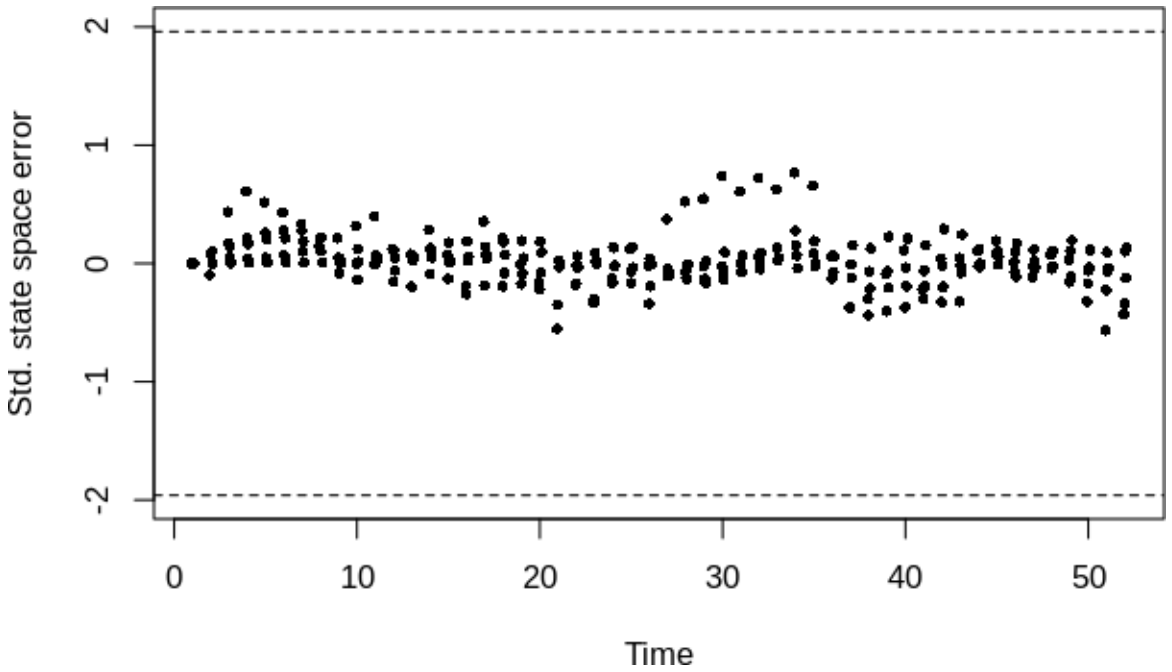
We start by looking at the standardized state space errors as explained in the ddhazard vignette. We may expect these to be standard iid normal distributed. First, we compute the values with the `residuals` by passing `type = "std_space_error"` with the first fit we made with the Rossi data set:

```
stat_res <- residuals(dd_rossi, type = "std_space_error")

str(stat_res)
```

```
## List of 3
##  $ residuals  : num [1:52, 1:5] -3.78e-07 -9.84e-02 4.34e-01 6.10e-01 5.17e-01 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:5] "(Intercept)" "fin" "age" "prio" ...
##  $ standardize: logi TRUE
##  $ Covariances: num [1:5, 1:5, 1:53] 0.36602 0.08669 -0.015 0.00237 -0.00154 ...
##   ..- attr(*, "dimnames")=List of 3
##   .. ..$ : chr [1:5] "(Intercept)" "fin" "age" "prio" ...
##   .. ..$ : chr [1:5] "(Intercept)" "fin" "age" "prio" ...
##   .. ..$ : NULL
##  - attr(*, "class")= chr "ddhazard_space_errors"
```

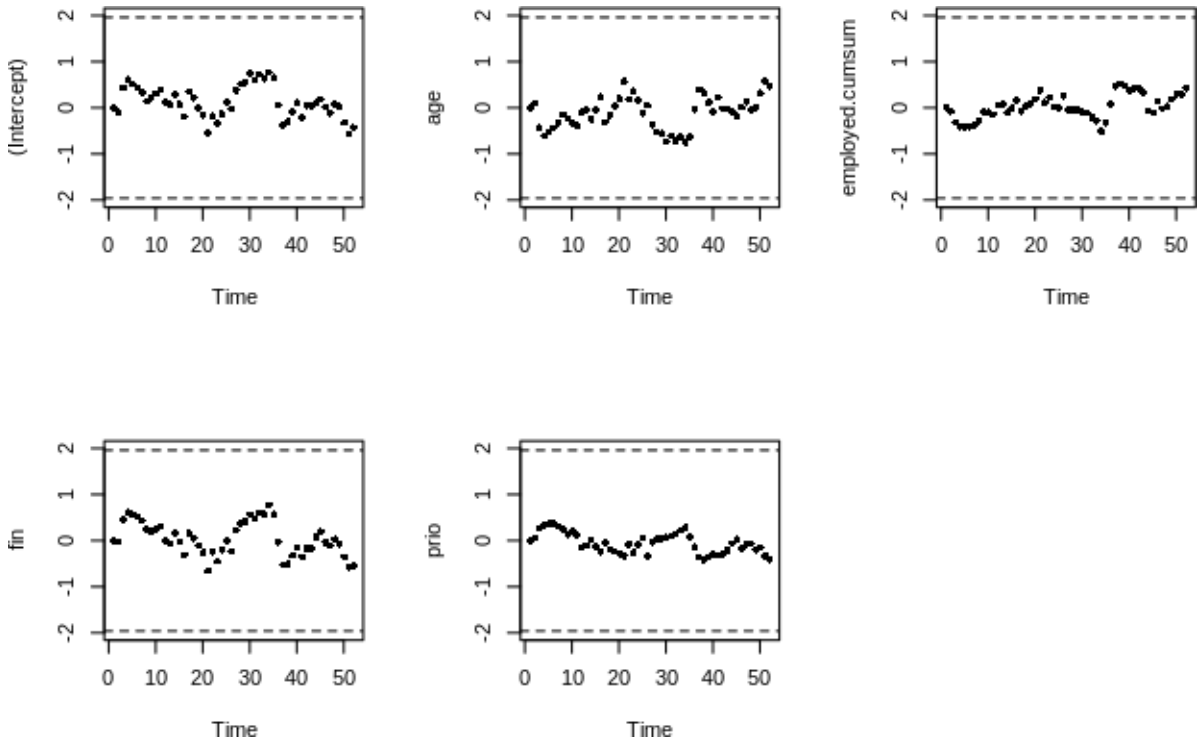
The output is a list with the residuals, smoothed covariance matrices for the errors and a binary variable to indicate whether or not the residuals are standardized. Next, we can plot the residuals as follows:

```
plot(stat_res, mod = dd_rossi, p_cex = .75, ylim = c(-2, 2))
```



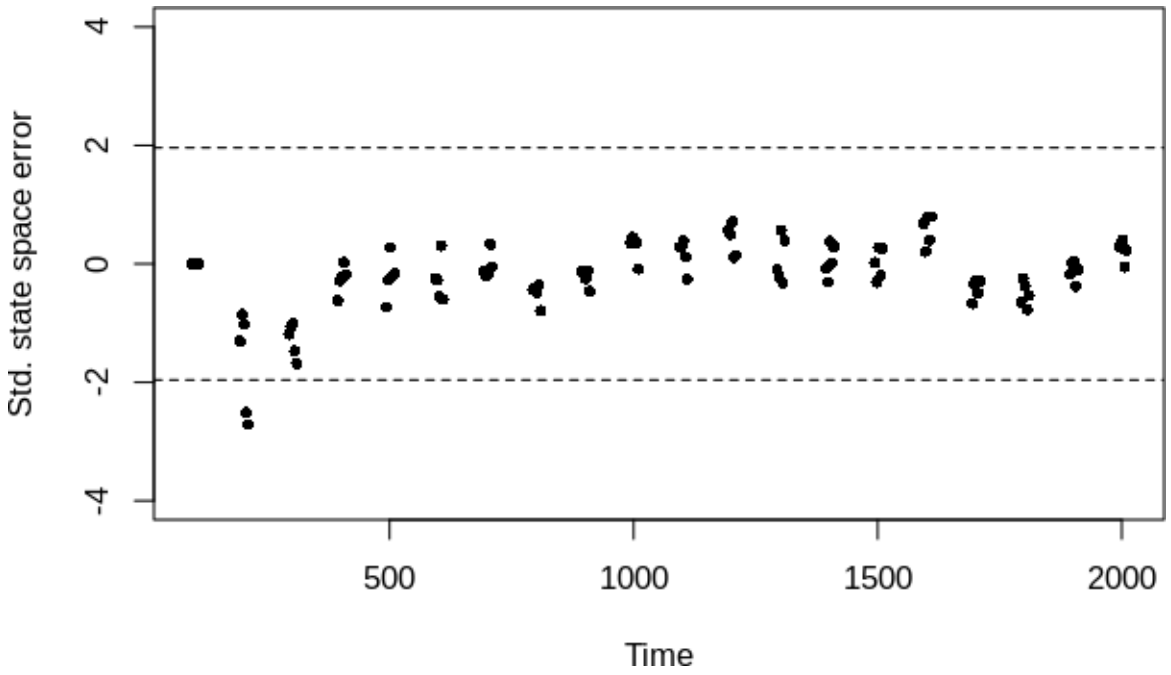

The variables appears to be nothing like standard normal (we have $52 \cdot 6$ residuals with no one outside $\pm 1.96$). Another idea is only marginally standardize (that is, not rotate the residuals). This can be done as follows:

```
# Get non-standardized residuals
stat_res <- residuals(dd_rossi, type = "space_error")

# Standardize marginally
for(i in 1:nrow(stat_res$residuals))
  stat_res$residuals[i, ] <- stat_res$residuals[i, ] /
    sqrt(diag(stat_res$Covariances[, , i]))

# Plot
par(mfcol = c(2, 3))
for(i in 1:ncol(stat_res$residuals))
  plot(stat_res, mod = dd_rossi, p_cex = .75, cov_index = i,
       ylab = colnames(stat_res$residuals)[i],
       ylim = c(-2, 2))
```




which I again find hard to draw any conclusion from. It seems like there is structure in the errors for the intercept and `fin`. However, this might be what we expected. For instance, we may expect the intercept to increase through time (i.e. not be random as assumed by the model). Further, assuming that the covariance estimate are conservative then there might be an error for `prio` in interval 20-25 and an error in `age` in interval 10-12 that seem extreme. We can do the same thing for the first fit with the Rossi data set:

```
stat_res <- residuals(dd_whas, type = "std_space_error")

plot(stat_res, mod = dd_whas, ylim = c(-4, 4), p_cex = .8)
```




Again, the errors seems to have to low variance to be standard normal apart from one which is more than three standard deviations away. I am not sure what to conclude from this. A question is whether we would see the same for a simulated data set where the true coefficients follows a random walk. We check this in the next paragraph.

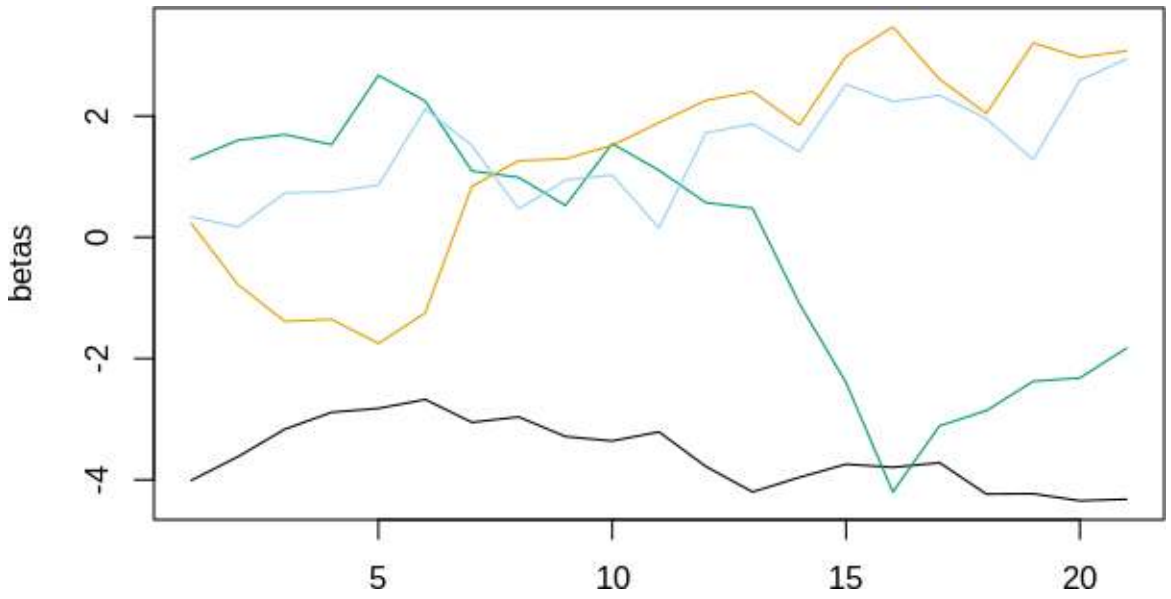
### Simmulation

We start by getting a the definition of the `test_sim_func_logit` function in the `dynamichazard` package which is not exported. We will use it to simulate the individuals series:

```
sim_func <- with(environment(ddhazard), test_sim_func_logit)
```

Then we simulate the coefficients and plot them:

```
# Simulate the coefficients
set.seed(556189)
betas <- matrix(rnorm(21 * 4), ncol = 4)
betas[, 1] <- betas[, 1] * 0.25  # reduce the variance of the intercept
betas <- apply(betas, 2, cumsum) # accumulate the innovations
betas[, 1] <- betas[, 1] - 4     # we reduce the intercept

# Plot the simulated coefficients
matplot(betas, col = cols, lty = 1, type = "l")
```




We reduce the variance of the intercept and decrease the intercept to yield a lower baseline risk of an event. The individuals and their outcomes are simulated as follows:

- Each individual start at time 0.
- The individuals covariates are simulated from Unif$(-1, 1)$.
- We update the individuals covariate with intervals times drawn from a Exp$(1/10)$.

This is done in the following call:

```
# Simulate series
sim_dat <- sim_func(
  n_series = 500,       # number of individuals
  t_max = 20,           # the last stop time
  x_range = 2,          # what is the uniform range to draw from
  x_mean = 0,           # the mean of the uniform covariates
  n_vars = 3,           # 4 - 1 for the intercept
  lambda = 1/10,        # lambda in the exponential distribution for time
                        # between updates of covariate vectors
  betas = betas)
```

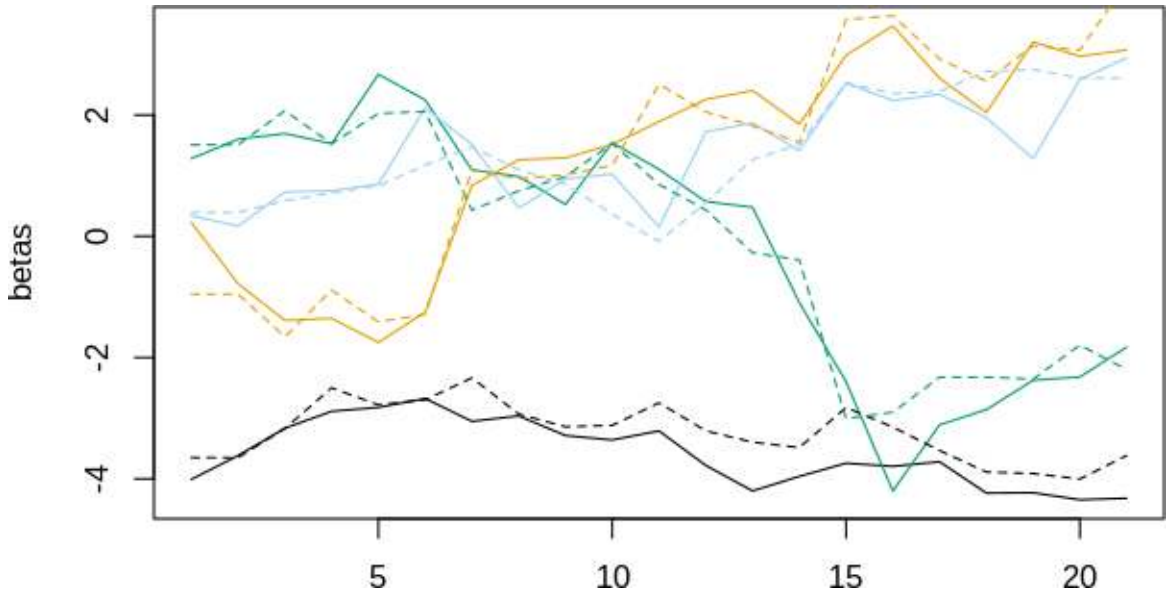
The first rows of the simulation looks as follows

```
head(sim_dat$res, 10)
```

```
##    id tstart tstop event      x1     x2      x3
## 1   1   0.00  9.00     1 -0.0764  0.275  0.1445
## 2   2   0.00 13.00     1 -0.8339  0.357  0.8905
## 3   3   0.00 16.79     0 -0.6834 -0.535 -0.6035
## 4   3  16.79 20.00     0 -0.4489 -0.600  0.0503
## 5   4   0.00 10.00     1  0.2756  0.825  0.2564
## 6   5   0.00  1.64     0 -0.1247  0.309 -0.5942
## 7   5   1.64  6.00     1 -0.5084  0.920  0.7204
## 8   6   0.00  5.00     1  0.2816  0.255  0.2460
## 9   7   0.00  2.36     0 -0.0337 -0.255  0.3031
## 10  7   2.36  5.00     1  0.5968 -0.283  0.0303
```

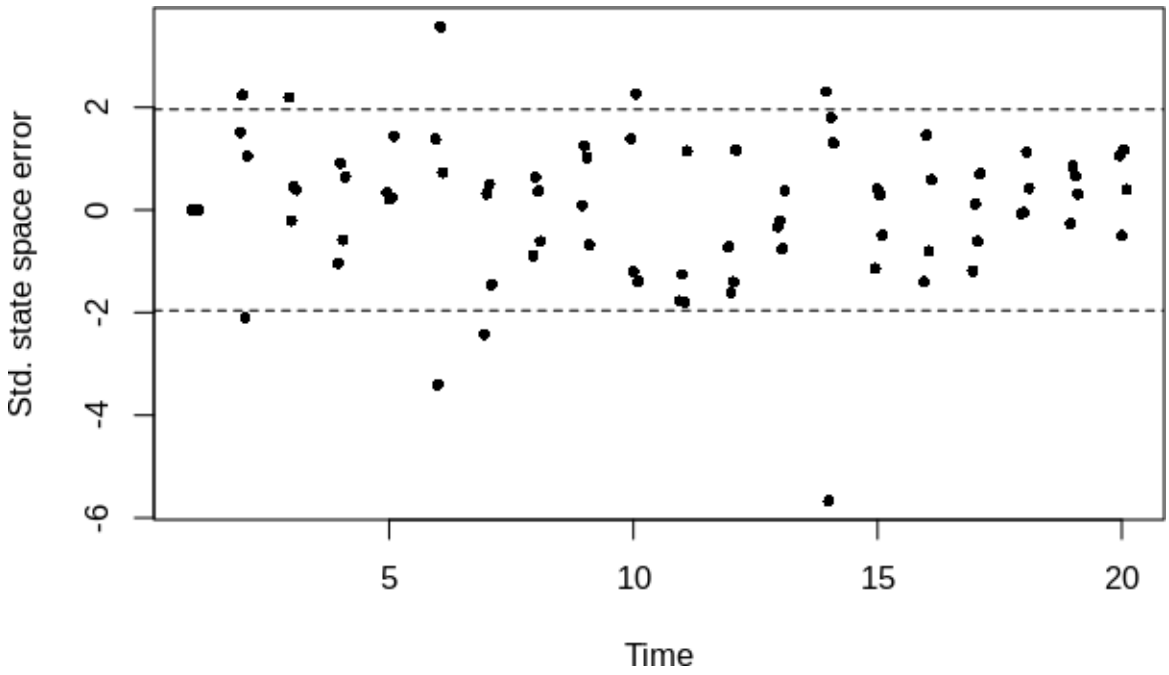
Next, we estimate the model and compare the estimated coefficients with the fit:

```
f1 <- ddhazard(
  Surv(tstart, tstop, event) ~ x1 + x2 + x3,
  sim_dat$res, by = 1, max_T = 20, id = sim_dat$res$id,
  Q_0 = diag(10000, 4), Q = diag(.1, 4),
  control = ddhazard_control(eps = .001))

matplot(betas, col = cols, lty = 1, type = "l")
matplot(f1$state_vecs, col = cols, lty = 2, type = "l", add = T)
```



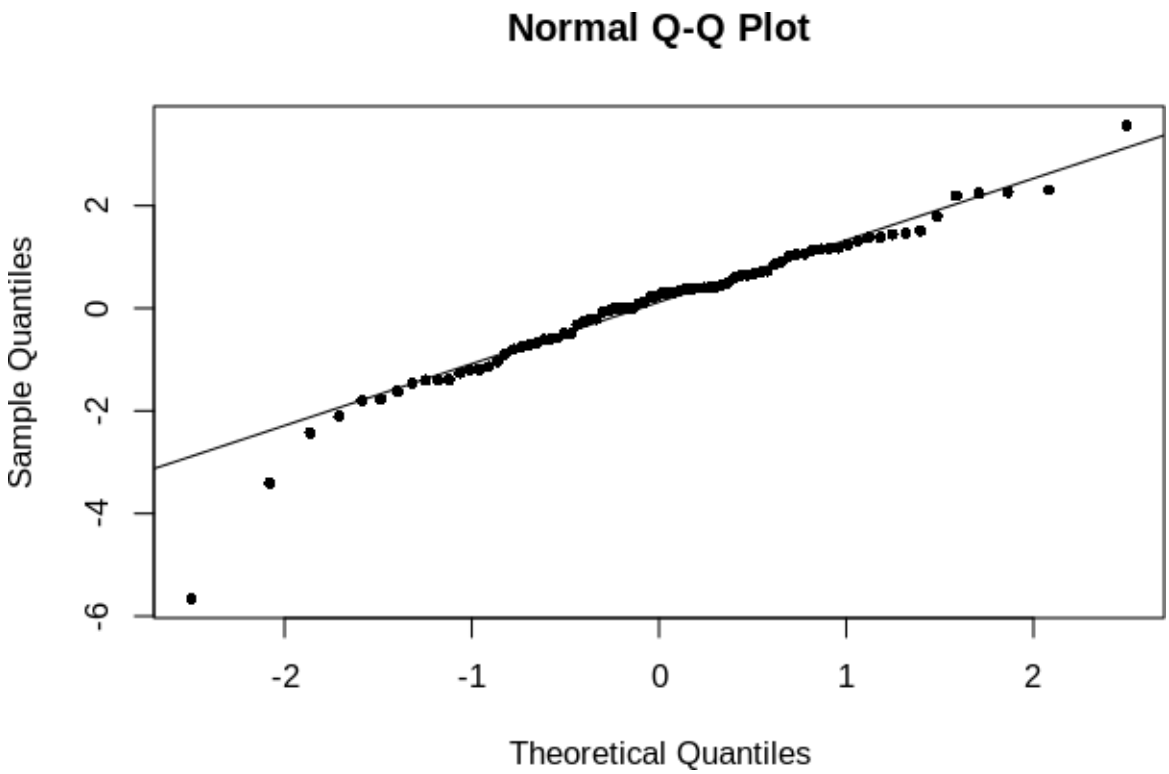

The full lines are the true coefficients and the dashed lines are the estimates. The question is then how the standardized state space errors look. We plot these below

```
stat_res <- residuals(f1, type = "std_space_error")

plot(stat_res, mod = f1, p_cex = .8)
```

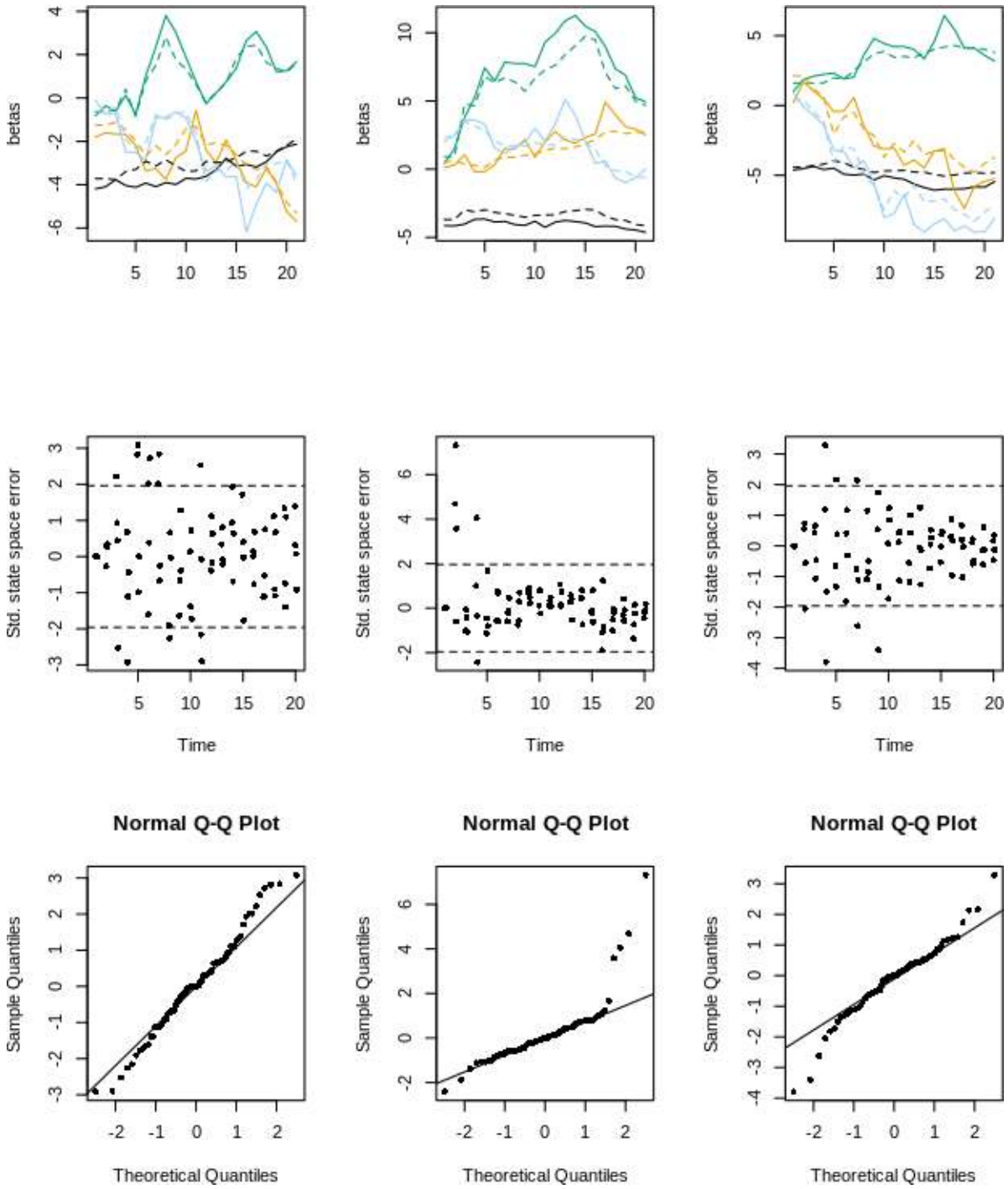



The errors seems more variable than before. We make a QQ-plot below. Apart from one error it seems quite close to a normal distribution.

```
qqnorm(c(stat_res$residuals), pch = 16, cex = .8)
qqline(c(stat_res$residuals))
```




Re-running the above three times gives the following plots:




It is worth stressing that neither the initial seed nor the parameters have been selected here to yield a particular result. The take away for me is that the previous finding with the Rossi data set and WHAS data set is an artifact of the data set and model specification and not something we would see if we have an actual random walk model it seems.

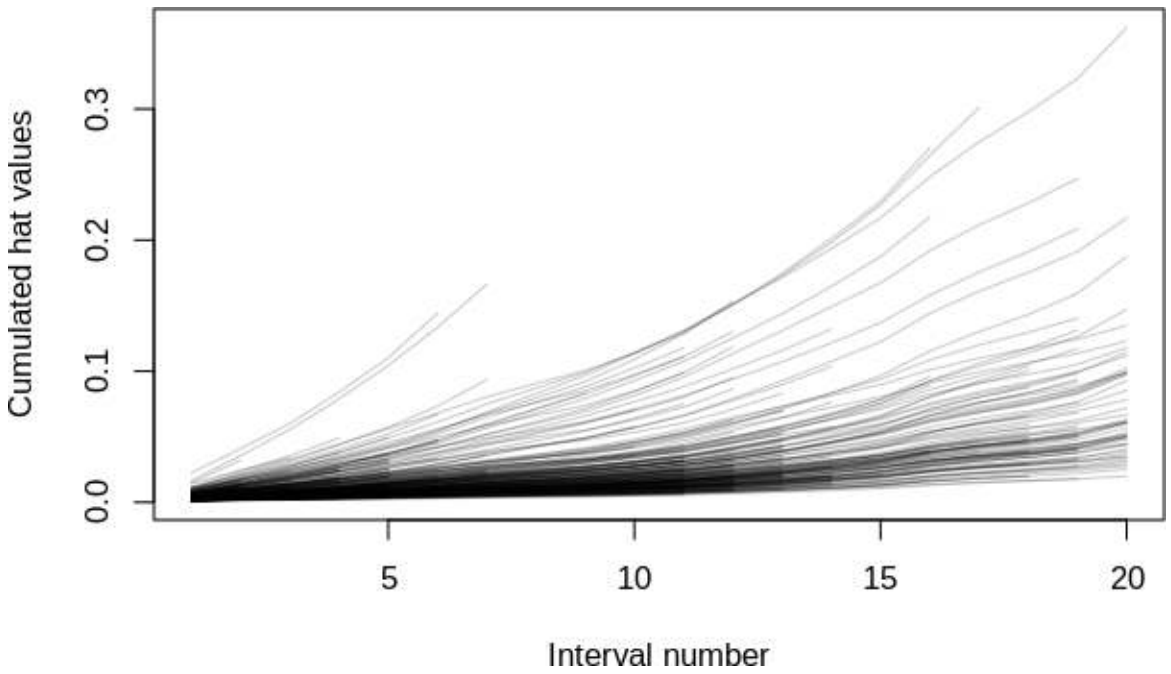
## Hat values for Worcester Heart Attack Study

In the following paragraphs, we check hat values for Worcester Heart Attack Study data set. First, we compute them:

```
hats <- hatvalues(dd_whas)
hats_stacked <- stack_hats(hats)
```

Then we compute the cumulated hat values for each individuals and plot against time

```
# Compute cumulated hat values
hats_stacked$hats_cum <- unlist(tapply(
  hats_stacked$hat_value, hats_stacked$id, cumsum))

# Plot the cumulated residuals for each individual
plot(c(1, 20), range(hats_stacked$hats_cum), type = "n",
     xlab = "Interval number", ylab = "Cumulated hat values")
invisible(
  tapply(hats_stacked$hats_cum, hats_stacked$id, lines,
         col = gray(0, alpha = .2)))
```




Three individuals seems to stand out. We look at these in the next line:

```
max_cum <- tapply(hats_stacked$hats_cum, hats_stacked$id, max)
is_max <- order(max_cum, decreasing = T)[1:3]
is_max
```

```
## [1] 112  12  89
```

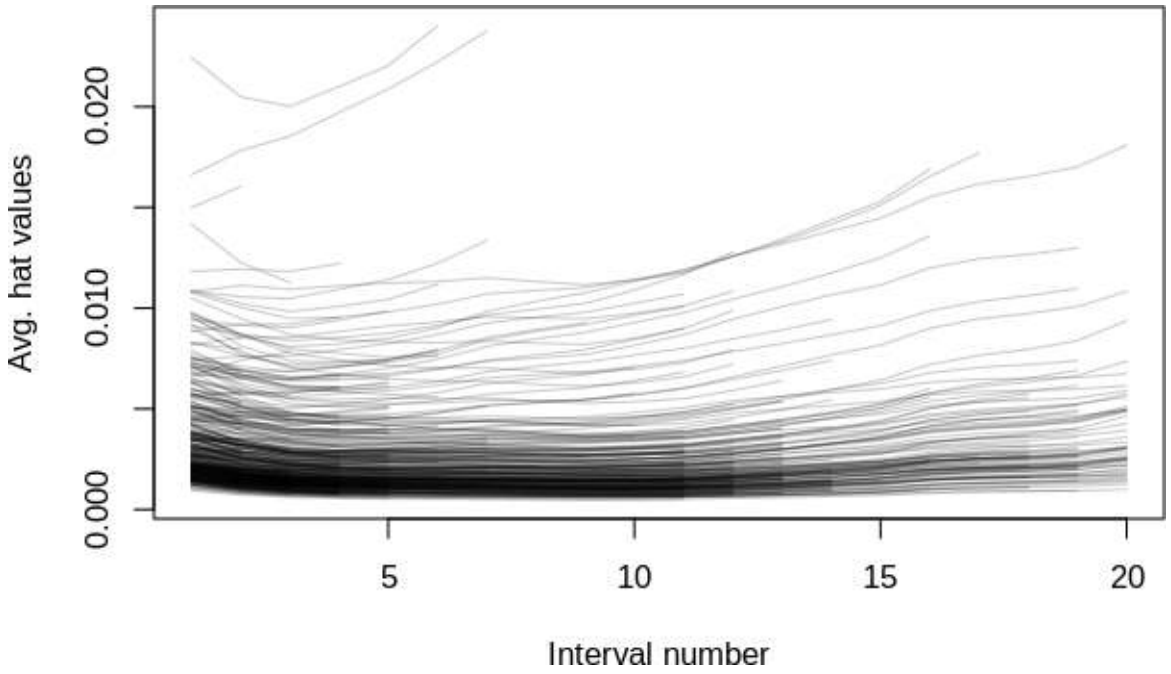
```
# The records for these
whas500[is_max, ]
```

```
##      id lenfol fstat gender age  bmi  hr cvd
## 112 112   2123     0      1  87 14.8 105   1
## 12   12   1671     1      0  75 28.7 154   1
## 89   89   1553     1      0  95 15.9  62   1
```

One of them has a high heart rate while the two others have a low BMI (below 18.5 is underweight). Another idea is to look at the average up to the given time of the hat values. The motivation is the two lines that end around the 5'th interval either because they die or are right censored. Hence, we normalize by the interval

number and plot against time

```
# Averages of hat values
hats_stacked$hats_avg <- hats_stacked$hats_cum / hats_stacked$interval_n

# Plot against time
plot(c(1, 20), range(hats_stacked$hats_avg), type = "n",
     xlab = "Interval number", ylab = "Avg. hat values")
invisible(
  tapply(hats_stacked$hats_avg, hats_stacked$id, lines,
         col = gray(0, alpha = .2)))
```




Indeed the two stands. Hence, we look further at the five largest values:

```
max_avg <- tapply(hats_stacked$hats_avg, hats_stacked$id, max)
is_max_avg <- order(max_avg, decreasing = T)[1:5]
is_max_avg
```

```
## [1] 472 389 112  12  99
```

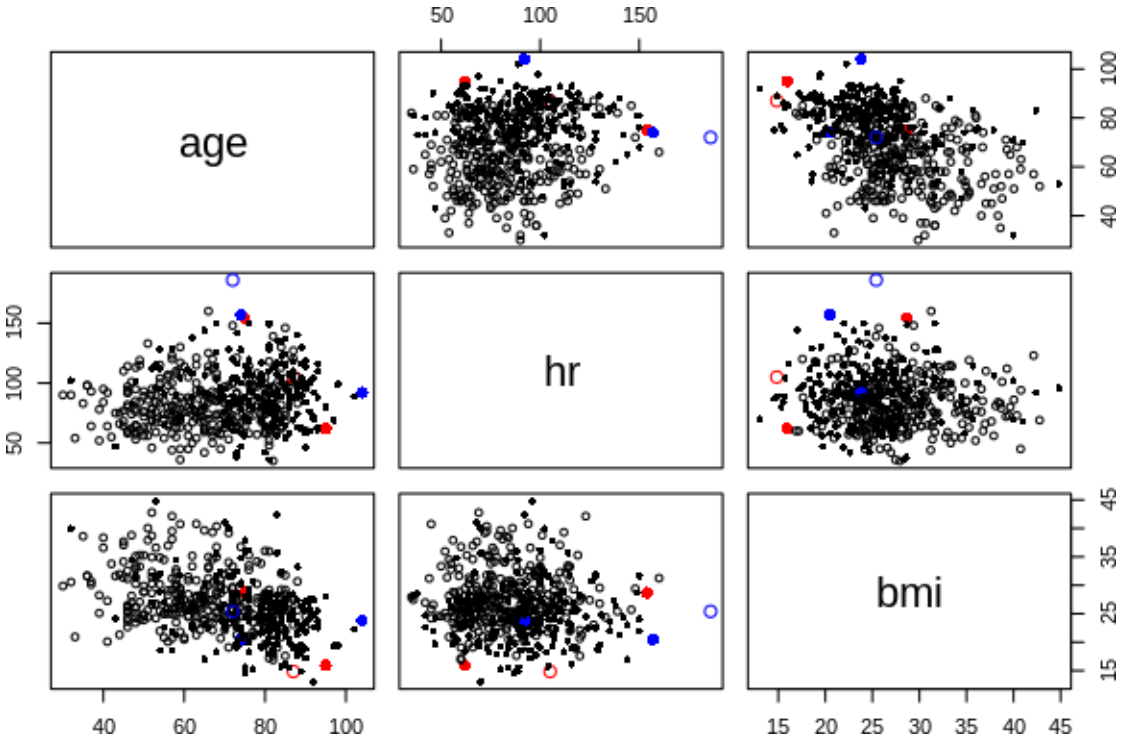
```
# The records for these
whas500[is_max_avg, ]
```

```
##      id lenfol fstat gender age  bmi  hr cvd
## 472 472    626     0      0  72 25.4 186   1
## 389 389    646     1      1 104 23.8  92   0
## 112 112   2123     0      1  87 14.8 105   1
## 12   12   1671     1      0  75 28.7 154   1
## 99   99      7     1      0  74 20.5 157   0
```

The two new ones with the highest values are one who is old and another with an extreme heart rate (a typical rule rule of thumb is that the maximum heart rate is 220 less your age!). In order to show this, we make the following plots:

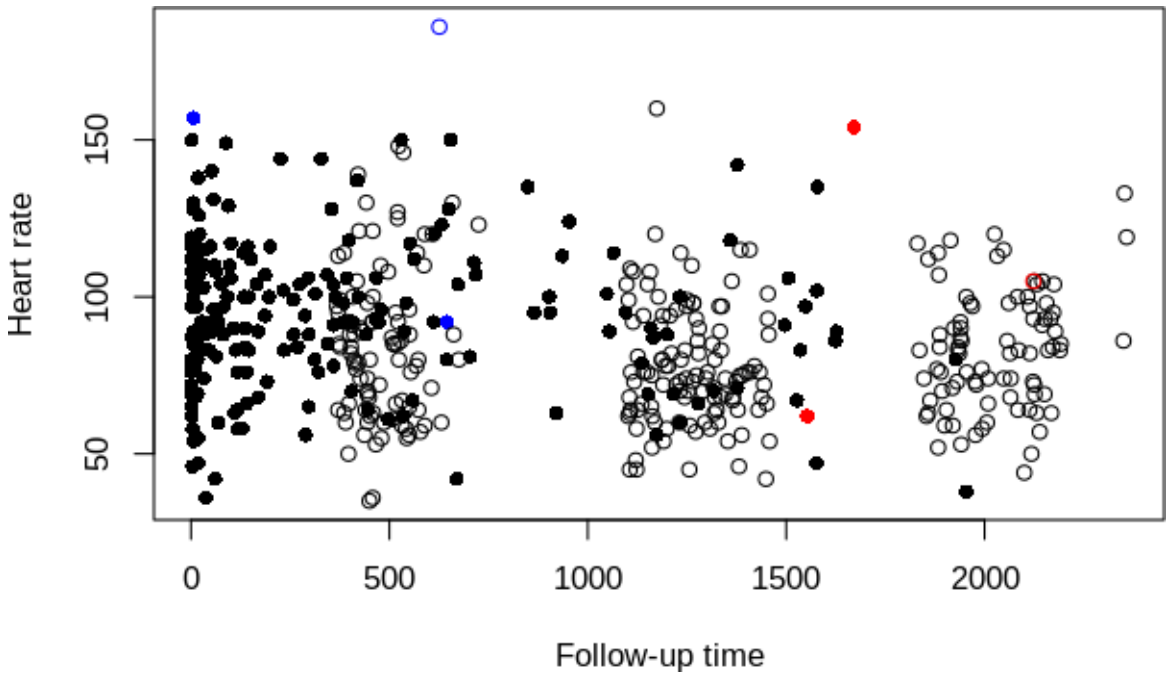
```
# Setup parameters for the plot
cols <- rep("Black", 500)
```

```
cols[1:500 %in% is_max_avg] <- "Blue"
cols[1:500 %in% is_max] <- "Red"

cexs <- ifelse(cols != "Black", par()$cex * 1.25, par()$cex * .75)
pchs <- ifelse(whas500$fstat == 1 & whas500$lenfol <= 2000, 16, 1)

plot(whas500[, c("age", "hr", "bmi")], pch = pchs, cex = cexs, col = cols)
```

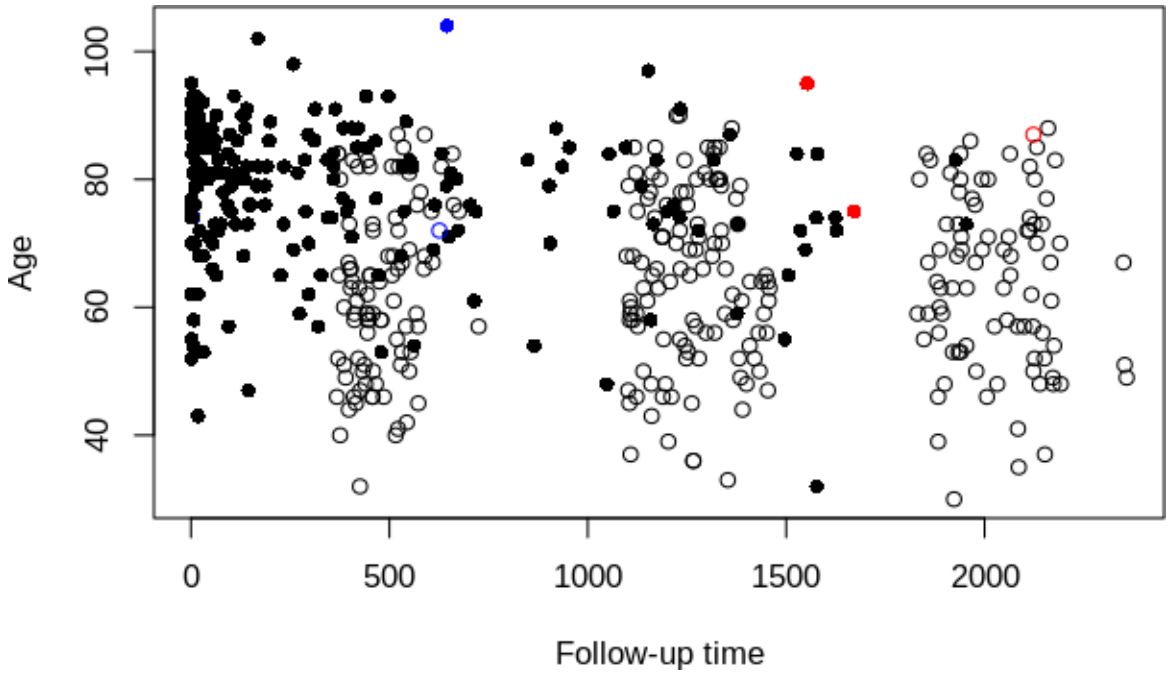



Filled circles are cases and non-filled circles are censored. The blue dots are the ones with a high maximum average hat value while the red one have both a high maximum average and a high cumulative hat values. Plotting against time shows the censoring is is clustered in time as shown in the next two plots:

```
plot(whas500$lenfol, whas500$hr, col = cols, pch = pchs,
     xlab = "Follow-up time", ylab = "Heart rate")
```




```
plot(whas500$lenfol, whas500$age, col = cols, pch = pchs,
     xlab = "Follow-up time", ylab = "Age")
```



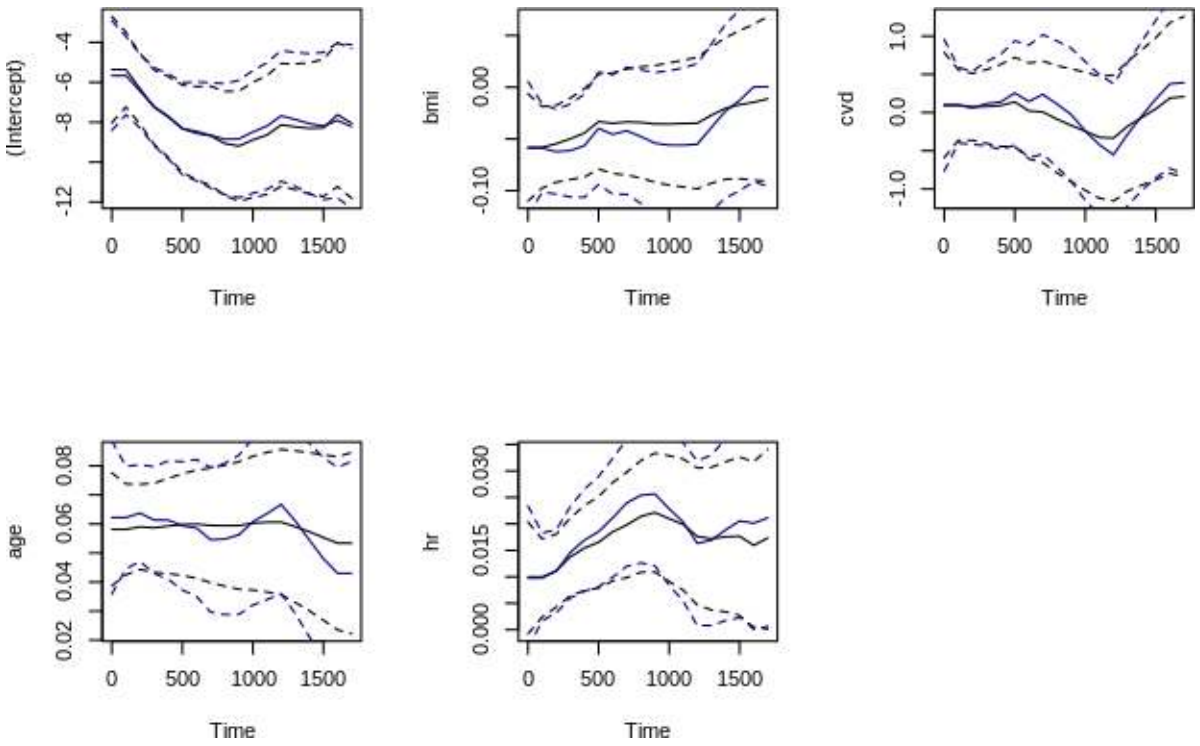

This could motivate us to stop the slightly before the last cluster of censoring of individuals occurs. Thus, we set the final time (the `max_T` argument to `ddhazard`) to 1700 instead of 2000 in the next code block where we re-estimate the model. Further, we make a fit without the "extreme" individuals:

```
dd_whas <- ddhazard(
  Surv(lenfol, fstat) ~ age + bmi + hr + cvd,
  data = whas500, by = 100, max_T = 1700,
  Q_0 = diag(10000, 5), Q = diag(.1, 5),
  control = ddhazard_control(eps = .001))
```

```
dd_whas_no_extreme <- ddhazard(
  Surv(lenfol, fstat) ~ age + bmi + hr + cvd,

  data = whas500[-c(is_max, is_max_avg), ], # we exclude the "extreme" persons

  by = 100, max_T = 1700,
  Q_0 = diag(10000, 5), Q = diag(.1, 5))
```

We plot the two sets of predicted coefficients next:

```
par(mfcol = c(2,3))
for(i in 1:5){
  plot(dd_whas, cov_index = i)
  plot(dd_whas_no_extreme, cov_index = i, add = T, col = "DarkBlue")
}
```




The blue line is the predicted coefficients without the "extreme" individuals. The difference seems minor

## Function definitions

The functions used in this vignette that are no included in the package are defined below:

```
stack_hats <- function(hats){
  # Stack
  resids_hats <- data.frame(do.call(rbind, hats), row.names = NULL)

  # Add the interval number
  n_rows <- unlist(lapply(hats, nrow))
  interval_n <- unlist(sapply(1:length(n_rows), function(i) rep(i, n_rows[i])))
```

```
  resids_hats <- cbind(resids_hats, interval_n = interval_n)

  # Sort by id and interval number
  resids_hats <-
    resids_hats[order(resids_hats$id, resids_hats$interval_n), ]

  resids_hats
}
```

```
stack_residuals <- function(fit, resids){
  if(!inherits(resids, "ddhazard_residual"))
    stop("Residuals must have class 'ddhazard_residual'")
  if(!inherits(fit, "ddhazard"))
    stop("fit must have class 'ddhazard'")

  # Stack the residuals
  resids_stacked <-
    data.frame(do.call(rbind, resids[[1]]), row.names = NULL)

  # Add the interval number and id
  n_rows <- unlist(lapply(resids$residuals, nrow))
  interval_n <- unlist(sapply(1:length(n_rows), function(i) rep(i, n_rows[i])))

  resids_stacked <- cbind(
    resids_stacked,
    interval_n = interval_n,
    id = fit$id[resids_stacked$row_num])

  # Sort by id and interval number
  resids_stacked <-
    resids_stacked[order(resids_stacked$id, resids_stacked$interval_n), ]

  resids_stacked
}
```

## References

Rossi, P. H., Berk, R. A., & Lenihan, K. J. (1980). Money, work and crime: Some experimental results. New York: Academic Press.