# Bootstrap illustration

*Benjamin Christoffersen*

*6 January 2017*

## Introduction

This vignette will show how to bootstrap the confidence intervals of a `ddhazard` call. This vignette builds on the vignettes 'ddhazard' and 'Comparing methods for time varying logistic models'. Thus, it is recommended to read these first. You can get the version used to make this vignette by calling:'

```
current_version # The string you need to pass devtools::install_github
```

```
## [1] "boennecd/dynamichazard@f79de934e169aa6bc9f1f9341c5f5bc601f5f348"
```

```
devtools::install_github(current_version)
```

You can also get the latest version on CRAN by calling:

```
install.packages("dynamichazard")
```

## PBC data set

We start by settings up the data set. We will use the pbc2 data set from the `survival` package as in the vignette 'Comparing methods for time varying logistic models':

```
# PBC data set from survival with time variying covariates
# Details of tmerge are not important in this scope. The code is included
# to make you able to reproduce the results
# See: https://cran.r-project.org/web/packages/survival/vignettes/timedep.pdf
library(survival)
temp <- subset(pbc, id <= 312, select=c(id, sex, time, status, edema, age))
pbc2 <- tmerge(temp, temp, id=id, death = event(time, status))
pbc2 <- tmerge(pbc2, pbcseq, id=id, albumin = tdc(day, albumin),
               protime = tdc(day, protime), bili = tdc(day, bili))
pbc2 <- pbc2[, c("id", "tstart", "tstop", "death", "sex", "edema",
                 "age", "albumin", "protime", "bili")]
```

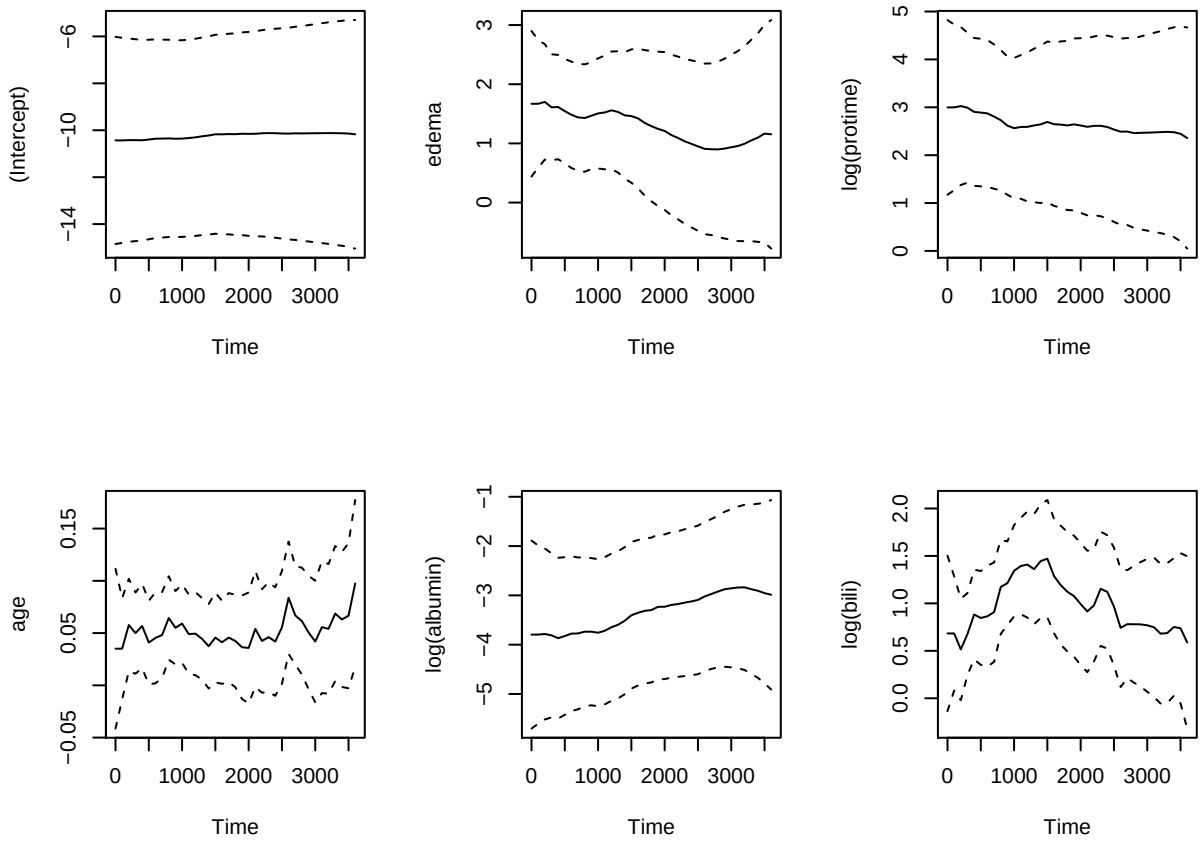
Next, we fit the model as in the vignette 'Comparing methods for time varying logistic models':

```
library(dynamichazard)
dd_fit <- ddhazard(Surv(tstart, tstop, death == 2) ~ age + edema +
                       log(albumin) + log(protime) + log(bili), pbc2,
                   id = pbc2$id, by = 100, max_T = 3600,
                   Q_0 = diag(rep(10000, 6)), Q = diag(rep(0.001, 6)),
                   control = list(save_risk_set = T, save_data = T, eps = .1))
```

```
## a_0 not supplied. One iteration IWLS of static glm model is used
```

A plot of the estimates is given below. The dashed lines are 95% point-wise confidence intervals using the variances estimates from the Extended Kalman filter with smoothing:

```
plot(dd_fit)
```




## Sampling individuals

We can bootstrap the estimates in the model by using `ddhazard_boot` function as done below:

```
set.seed(7451)
R <- 10000
boot_out <- ddhazard_boot(
  dd_fit,
  do_sample_weights = F,       # should re-sampeling be by weights or by
                               # sampling each individual discreetly
  do_stratify_with_event = F, # stratify on whether the individual is an event
                               # or not
  R = R                        # Number of bootstrap samples
  )
```

```
## Warning in ddhazard_boot(dd_fit, do_sample_weights = F,
## do_stratify_with_event = F, : Failed to estimate 330 times
```

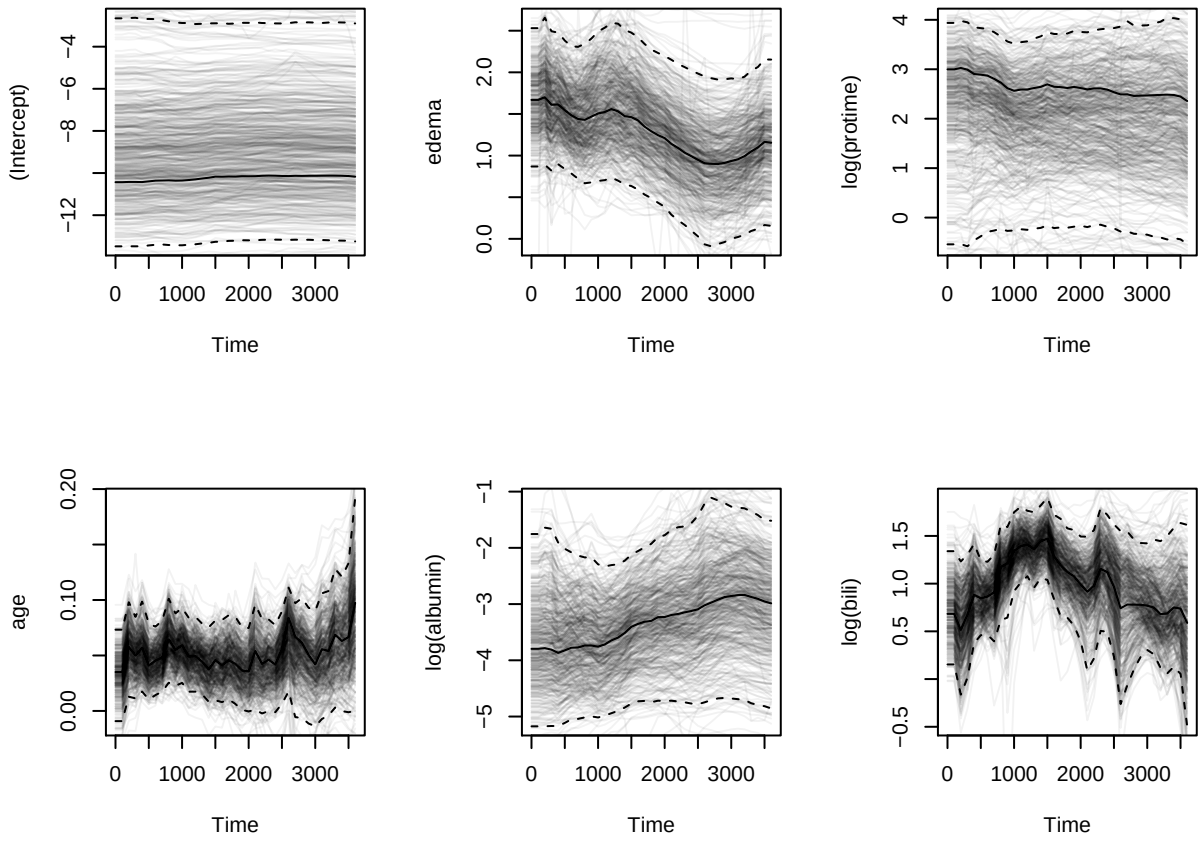
```
# The list has the same structure and class as the list returned by boot::boot
# Though, a few elements are added
class(boot_out)
```

```
## [1] "ddhazard_boot" "boot"
```

Above, we bootstrap the model by sampling the individuals. I.e. individuals will have weights of $0, 1, 2, \ldots$ in the estimation. We can plot 95% confidence bounds from the bootstrap coefficients with the Percentile Bootstrap method as follows:

```
plot(dd_fit, ddhazard_boot = boot_out)
```

```
## Only plotting 500 of the boot sample estimates
```




The completely black line is the original estimates, the dashed lines are 2.5% and 97.5% quantiles of the bootstrap coefficient taken at each point and the transparent black lines each represent a bootstrap estimate

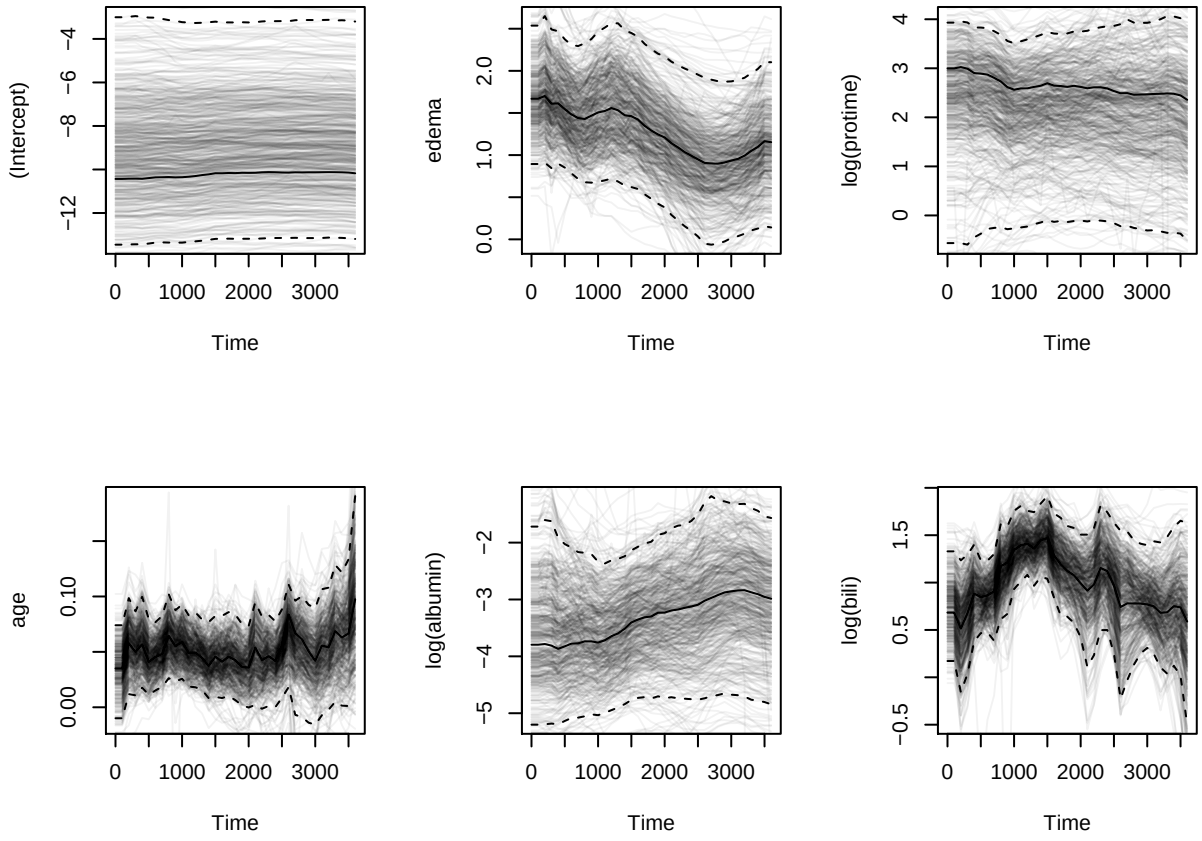
We can check if it affects the results if we perform stratified sampling between those individuals who has an event and those who does not as follows:

```
set.seed(8524)
boot_out_stratify_by_events <- ddhazard_boot(
  dd_fit,
  do_sample_weights = F,
  do_stratify_with_event = T, # changed
  R = R)
```

```
## Warning in ddhazard_boot(dd_fit, do_sample_weights = F,
## do_stratify_with_event = T, : Failed to estimate 332 times
```

```
plot(dd_fit, ddhazard_boot = boot_out_stratify_by_events)
```

```
## Only plotting 500 of the boot sample estimates
```




A motivation for sampling this way is if we have a small data set with few cases or controls. Thus, we could end with a sample with e.g. no or few cases. It seems to have no impact in this example

## Sampling weights

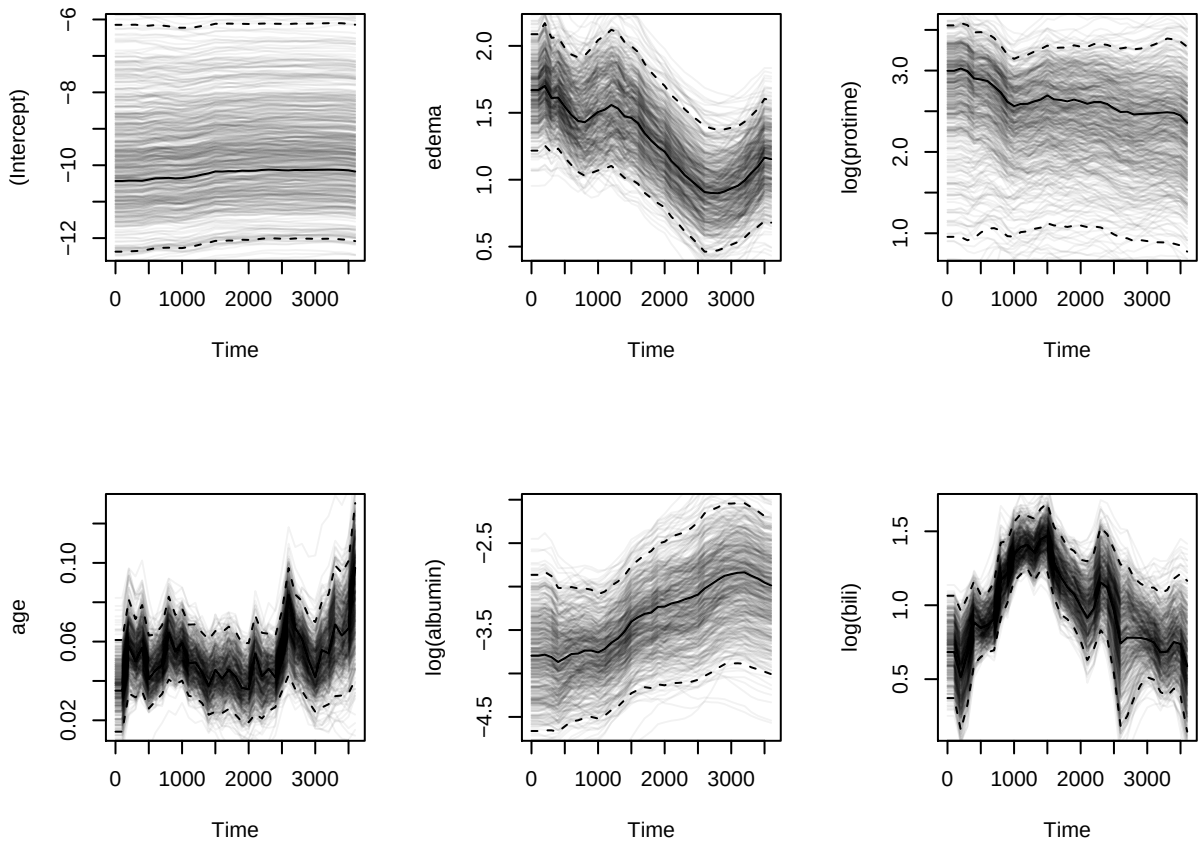
We can also sample the weights. This is done as follows: within each stratum $j$ (e.g. those who have an event and those who do not) let $r_j$ denote the number of individuals. Then we sample $r_j$ uniform variables $l_i \sim \text{Unif}(0, 1)$ for $i = 1, \ldots, r_j$ and normalize with a constant $c$ such that $\sum_{i=1}^{r_j} l_i/c = r_j$. The code below will sample the weights as described above:

```
set.seed(401)
boot_out_by_weights <- ddhazard_boot(
  dd_fit,
  do_sample_weights = T,       # changed
  do_stratify_with_event = F, # set back to false
  R = R)
```

```
## Warning in ddhazard_boot(dd_fit, do_sample_weights = T,
## do_stratify_with_event = F, : Failed to estimate 1 times
```

```
plot(dd_fit, ddhazard_boot = boot_out_by_weights)
```

```
## Only plotting 500 of the boot sample estimates
```




## Other strata

You can also provide your own strata to perform stratified sampling with. This is done by setting the `strata` argument in the call to `ddhazard_boot`. Notice that this has to be on an individual level (one indicator variable per individual) not observation level (not one indicator variable per row in the data set). Further, you can use the `unique_id` argument to match the individual entries with the entries in `strata`. As an example, we stratify by the `age` at the start of the study period with the code below:

```
# Individuals have different number of rows in the dataset
xtabs(~xtabs(~pbc2$id))
```

```
## xtabs(~pbc2$id)
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## 27 27 34 48 32 30 18 22 21 22  9  9  8  5
```

```
# Though all the individual have the same age for all periods
# This age is the age at the start of the study
unique(tapply(pbc2$age, pbc2$id, function(x) length(unique(x))))
```

```
## 1
## 1
```

```r
# Next, we find the age for each individual
unique_id <- unique(pbc2$id)
age <- sapply(unique_id, function(x) pbc2$age[pbc$id == x][1])
summary(age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   33.63   44.52   52.04   50.69   56.22   70.56
```

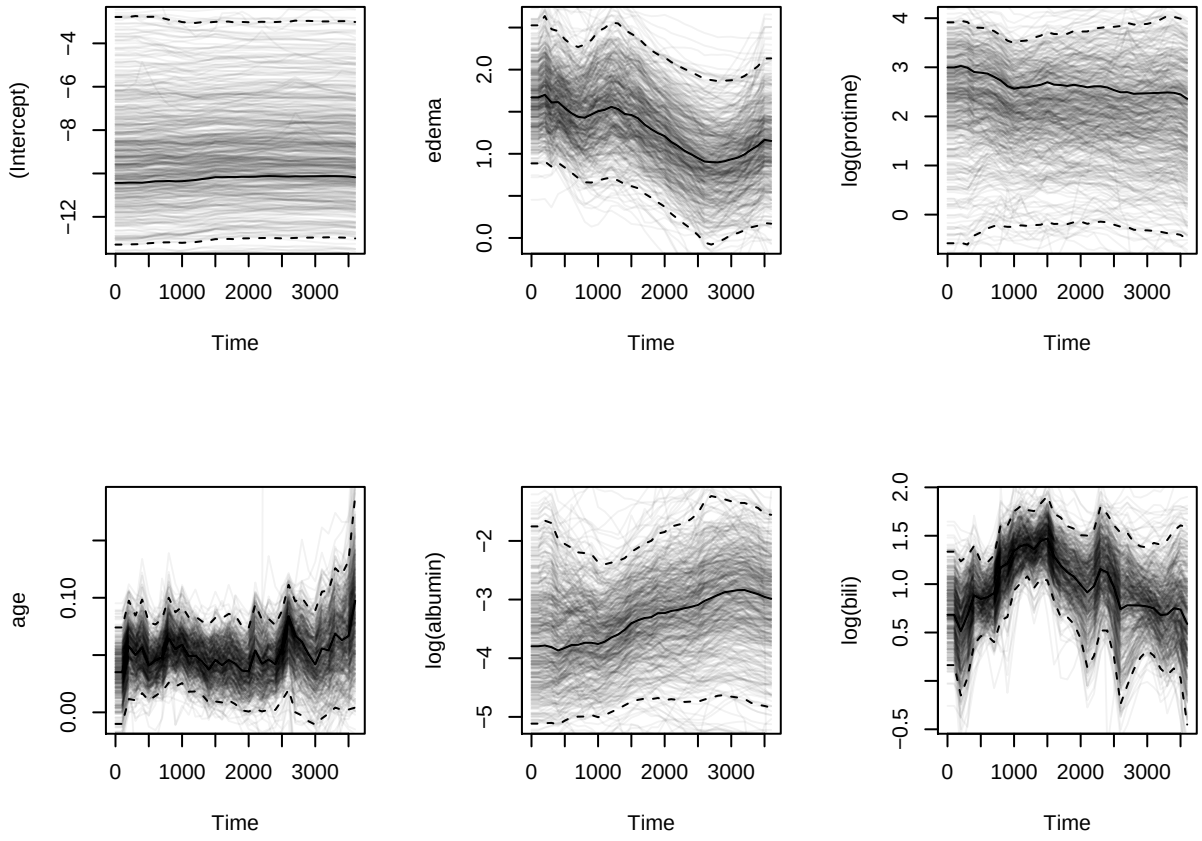
```r
# We define a strata variable for those less than age 50
is_less_than_50 <- age < 50

# We perform stratified sampling over this variable as follows
set.seed(101)
boot_out_with_strata <- ddhazard_boot(
  dd_fit,
  unique_id = unique_id,
  strata = is_less_than_50,
  R = R)
```

```
## Warning in ddhazard_boot(dd_fit, unique_id = unique_id, strata =
## is_less_than_50, : Failed to estimate 309 times
```

```r
plot(dd_fit, ddhazard_boot = boot_out_with_strata)
```

```
## Only plotting 500 of the boot sample estimates
```




The above code is only provided for illustrative purposes. There is no reason to do stratified sampling over the age variable (as far as I gather). However, it may be useful if you have e.g. categorical variables in your model and want to ensure that each bootstrap sample has a given amount of observation in each category. Lastly, setting `do_stratify_with_event = T` will yield an interaction factor between the passed `strata` and whether or not the given individual has an event. Stratified sampling will then be performed over this variable

## Fixed effects

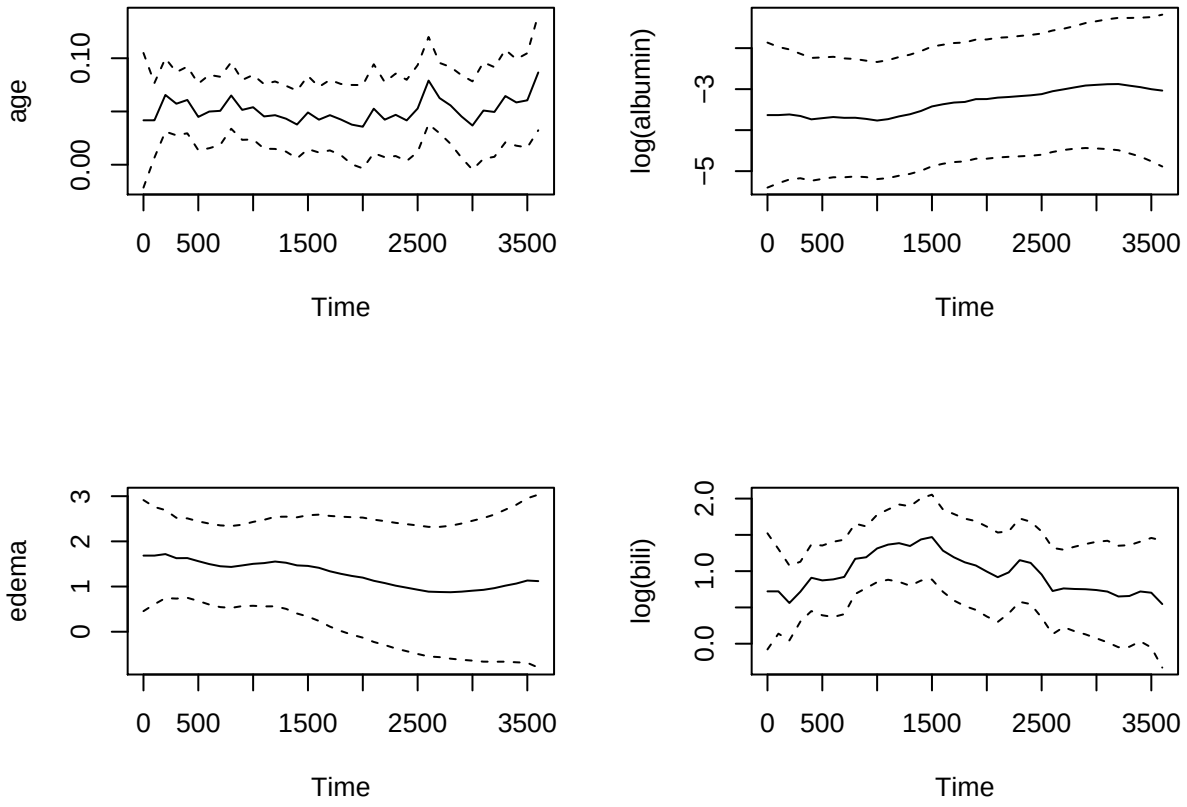
Fixed effects (time invariant effects) can also be bootstrap to get confidence bounds. The fixed effects bootstrap coefficients are added as the last entries of the element `t` of the returned object by `ddhazard_boot`. As an example we will estimate a model below where `log(protime)` and the intercept are fixed

```
dd_fit <- ddhazard(Surv(tstart, tstop, death == 2) ~
                     ddFixed(1) + ddFixed(log(protime)) # changed to fixed
                     + age + edema + log(albumin) +  + log(bili), pbc2,
                   id = pbc2$id, by = 100, max_T = 3600,
                   Q_0 = diag(rep(10000, 4)), Q = diag(rep(0.001, 4)),
                   control = list(
                     save_risk_set = T, save_data = T, eps = .1,
                     fixed_terms_method = "E_step")     # Fixed effects are
                                                        # esimated in E-step
                   )
```

```
## a_0 not supplied. One iteration IWLS of static glm model is used
```

The time varying effects are plotted below:

```
plot(dd_fit)
```




The fixed effects are estimated to:

```
dd_fit$fixed_effects
```

```
##  (Intercept) log(protime)
##   -10.406120     2.725937
```

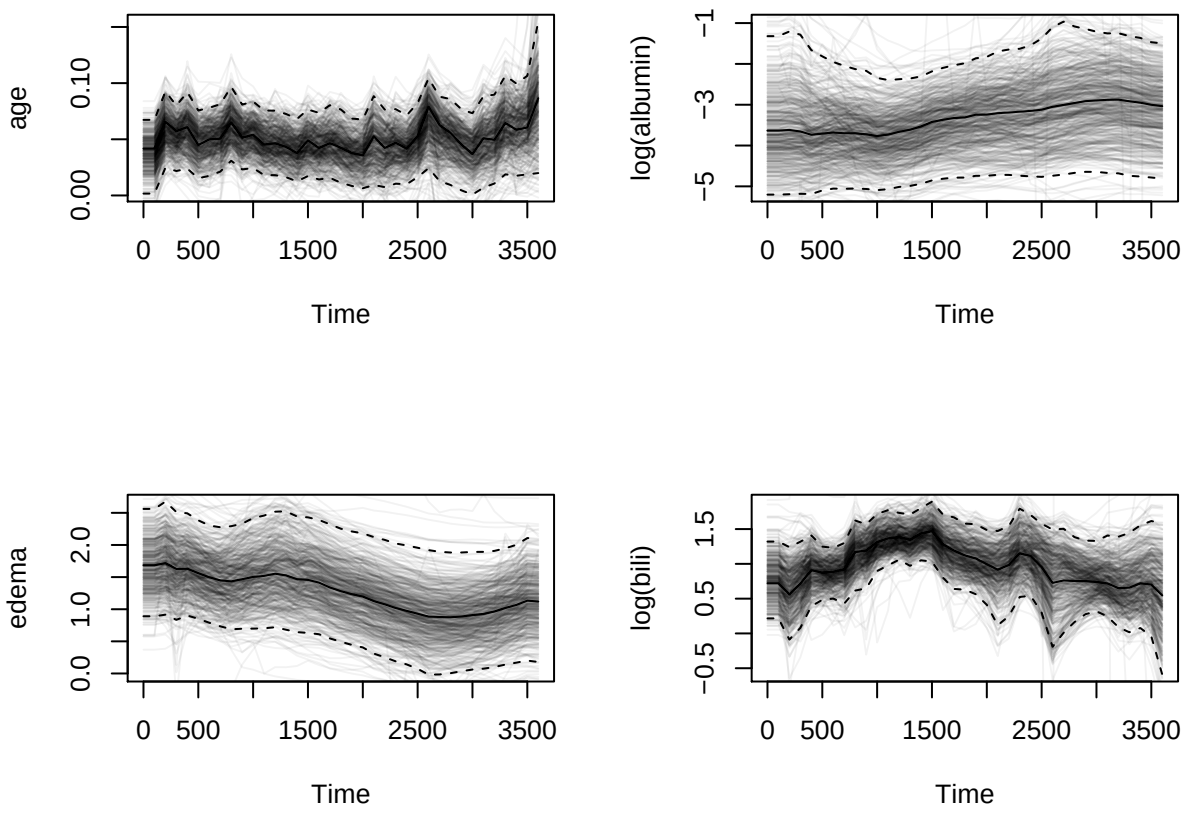
We can bootstrap the estimates with a call similar to those we made before:

```
set.seed(9001)
boot_out <- ddhazard_boot(
  dd_fit,
  do_sample_weights = F,       # dont sample weights
  do_stratify_with_event = F, # dont stratify by event
  R = R)
```

```
## Warning in ddhazard_boot(dd_fit, do_sample_weights = F,
## do_stratify_with_event = F, : Failed to estimate 321 times
```

```
# Plot time varying effects
plot(dd_fit, ddhazard_boot = boot_out)
```

```
## Only plotting 500 of the boot sample estimates
```




We then turn the bootstrap confidence intervals of the fixed effects. These can be computed with the `boot.ci` function from the `boot` library as shown below:

```
library(boot)

# We start by printing confidence intervals for
colnames(boot_out$t)[ncol(boot_out$t) - 1]
```

```
## [1] "(Intercept)"
```

```
boot.ci(boot_out, index = ncol(boot_out$t) - 1,
        # We specify the types of confidence intervals estimates here:
        type = c(
          "norm",  # A matrix of intervals calculated using the normal
                   # approximation.
          "basic", # The intervals calculated using the basic bootstrap method.
          "perc",  # The intervals calculated using the bootstrap percentile
                   # method.
          "bca")   # The intervals calculated using the adjusted bootstrap
                   # percentile (BCa) method.
        )
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
```

```
## Based on 9679 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_out, type = c("norm", "basic", "perc",
##     "bca"), index = ncol(boot_out$t) - 1)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-17.22,  -6.61 )   (-17.96,  -7.50 )
##
## Level     Percentile            BCa
## 95%   (-13.20,  -2.74 )   (-16.26,  -7.24 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```
# Then we print confidence intervals for
colnames(boot_out$t)[ncol(boot_out$t)]
```

```
## [1] "log(protime)"
```

```
boot.ci(boot_out, index = ncol(boot_out$t) - 0, type = c(
  "norm", "basic", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 9679 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_out, type = c("norm", "basic", "perc",
##     "bca"), index = ncol(boot_out$t) - 0)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.138,  6.762 )   ( 1.802,  5.677 )
##
## Level     Percentile            BCa
## 95%   (-0.263,  3.612 )   ( 1.605,  4.646 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```