Comparing methods for time varying logistic models

Benjamin Christoffersen

2016-12-28

Intro

This note will compare the dynamic logistic model in `dynamichazard` with others methods within the package and methods from the `timereg` and `mgcv` packages. Further this note will serve as an illustration of how to use the `ddhazard` function for the logistic model. We will use the `pbc2` dataset from the `survival` package. The motivation is that the `pbc2` data set is commonly used in survival analysis for illustrations. It is suggested first to read the ddhazard vignette to get an introduction to the models and estimation methods in this package

The note is structured as follows: First, we cover the `pbc2` data set. Then we estimate two non-dynamic logistic regression models using `glm`. This is followed by a fit using a Generalized Additive model with the `gam` function in the `mgcv` package. Next, we will estimate a cox-model with time varying parameters using the `timecox` function in the `timereg` package. Finally, we will end by illustrating the methods in this package for time varying parameters in a logistic regression

You can install the version of the library used to make this vignettes from github with the `devtools` library as follows:

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

```
## [1] "boennecd/dynamichazard@6d6885222287339bd524ddad0ce47058b3d9b094"
```

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

The pbc data set

The `pbc` data set contains data from the Mayo Clinic trial on primary biliary cirrhosis. We use this dataset to compare with results previously found analyzing the data set. We will focus on a the subset of co-variates used in Martinussen & Scheike (2007). The dataset can be created by:

```
# 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")]
```

as described in the vignette *Using Time Dependent Covariates and Time Dependent Coefficients in the Cox Model* in the `survival` package. The resulting data frame is structured as follows:

```
head(pbc2)
```

```
##   id tstart tstop death sex edema  age albumin protime bili
## 1  1      0   192     0   f     1 58.8    2.60    12.2 14.5
## 2  1    192   400     2   f     1 58.8    2.94    11.2 21.3
## 3  2      0   182     0   f     0 56.4    4.14    10.6  1.1
## 4  2    182   365     0   f     0 56.4    3.60    11.0  0.8
## 5  2    365   768     0   f     0 56.4    3.55    11.6  1.0
## 6  2    768  1790     0   f     0 56.4    3.92    10.6  1.9
```

The data set is in the usual start and stop time format used in survival analysis. Each individual in the trial has one or more observations. The `id` column identifies the individual. The `tstart` column indicates when the row is valid from and the `tstop` column indicates when the row is valid to. The `death` column is the outcome which is 2 when the individual dies at `tstop` (1 indicates that the individual gets a transplant). The `sex`, `edema` and `age` are baseline variables while `albumin`, `protime` and `bili` are laboratory values updated at follow ups with the patient. As an example, we can look individual `282`:

```
(ex <- pbc2[pbc2$id == 282, ])
```

```
##       id tstart tstop death sex edema  age albumin protime bili
## 1708 282      0   258     0   f     0 33.9    3.52     9.5  1.3
## 1709 282    258   370     0   f     0 33.9    3.39     9.5  1.1
## 1710 282    370   744     0   f     0 33.9    3.28    10.1  1.2
## 1711 282    744  1455     0   f     0 33.9    3.04    10.2  1.3
```

She (`sex` is `f`) has had four laboratory values measured at time 0, 258, 370 and 744. Further, she does not die as the `death` column is zero in the final row

Logistic model

We can start of with a simple logistic model where we ignore `tstart` and `tstop` variable using `glm`:

```
glm_simple <- glm(death == 2 ~ age + edema + log(albumin) + log(protime) +
                    log(bili), binomial, pbc2)

glm_simple$coefficients
```

```
##  (Intercept)          age        edema log(albumin) log(protime)    log(bili)
##      -9.3834       0.0537       1.4354      -4.4379       2.9274       1.1925
```

Though, we want to account for the fact that say the second the row of individual 282 has a length of 112 days (370 minus 258) while the fourth row has a length 711 days. A way to incorporate this information is to bin the observations into periods of a given length. Example of such binning is found in Tutz & Schmid (2016) and Shumway (2001)

Say that we set the bin interval lengths to 100 days. Then the first row for `id = 282` will yield three observation: one from 0 to 100, one from 100 to 200 and one from 200 to 300 (since we know that she survives beyond time 300). That is, she survives from time 0 to time 100, survives from time 100 to time 200 etc. We can make such a data frame with the `get_survival_case_weigths_and_data` function in this package:

```
library(dynamichazard)
pbc2_big_frame <- get_survival_case_weigths_and_data(
  Surv(tstart, tstop, death == 2) ~ age + edema + log(albumin) + log(protime) +
    log(bili), data = pbc2, id = pbc2$id, by = 100, max_T = 3600,
  use_weights = F)
pbc2_big_frame <- pbc2_big_frame$X
```

The code above uses the `Surv` function on the left hand side of the formula. The `Surv` function needs a start time, a stop time and a outcome. The right hand side is as before. The `by` argument specifies the interval length (here 100 days) and the `max_T` specify the last time we want to include. We will comment on `use_weights` argument shortly. As an example, we can look at individual 282 in the new data frame:

```
pbc2_big_frame[pbc2_big_frame$id == 282, ]
```

```
##       id tstart tstop death sex edema  age albumin protime bili     Y    t
## 282  282      0   258     0   2     0 33.9    3.52     9.5  1.3 FALSE  100
## 590  282      0   258     0   2     0 33.9    3.52     9.5  1.3 FALSE  200
## 703  282      0   258     0   2     0 33.9    3.52     9.5  1.3 FALSE  300
## 1207 282    258   370     0   2     0 33.9    3.39     9.5  1.1 FALSE  400
## 1410 282    370   744     0   2     0 33.9    3.28    10.1  1.2 FALSE  500
## 1667 282    370   744     0   2     0 33.9    3.28    10.1  1.2 FALSE  600
## 1939 282    370   744     0   2     0 33.9    3.28    10.1  1.2 FALSE  700
## 2201 282    370   744     0   2     0 33.9    3.28    10.1  1.2 FALSE  800
## 2556 282    744  1455     0   2     0 33.9    3.04    10.2  1.3 FALSE  900
## 2791 282    744  1455     0   2     0 33.9    3.04    10.2  1.3 FALSE 1000
## 3029 282    744  1455     0   2     0 33.9    3.04    10.2  1.3 FALSE 1100
## 3209 282    744  1455     0   2     0 33.9    3.04    10.2  1.3 FALSE 1200
## 3425 282    744  1455     0   2     0 33.9    3.04    10.2  1.3 FALSE 1300
## 3646 282    744  1455     0   2     0 33.9    3.04    10.2  1.3 FALSE 1400
## 3862 282    744  1455     0   2     0 33.9    3.04    10.2  1.3 FALSE 1500
##      weights
## 282        1
## 590        1
## 703        1
## 1207       1
## 1410       1
## 1667       1
## 1939       1
## 2201       1
## 2556       1
## 2791       1
## 3029       1
## 3209       1
## 3425       1
## 3646       1
## 3862       1
```

Notice that three new columns have been added: `Y` which is the outcome, `t` which is the stop time in the bin and `weights` which is the weight to be used in a regression. We see that the first row in the initial data frame for individual 282 has three rows now since the row is in three bins (the bins at times $(0, 100]$, $(100, 200]$ and $(200, 300]$). We could just use weights instead. This is what we get if we set `use_weights = T`:

```
pbc2_small_frame <- get_survival_case_weigths_and_data(
  Surv(tstart, tstop, death == 2) ~ age + edema + log(albumin) + log(protime) +
    log(bili), data = pbc2, id = pbc2$id, by = 100, max_T = 3600,
  use_weights = T)
pbc2_small_frame <- pbc2_small_frame$X
```

The new data rows for individual 282 looks as follows:

```
pbc2_small_frame[pbc2_small_frame$id == 282, ]
```

```
##      Y  id tstart tstop death sex edema  age albumin protime bili weights
## 1708 0 282      0   258     0   f     0 33.9    3.52     9.5  1.3       3
## 1709 0 282    258   370     0   f     0 33.9    3.39     9.5  1.1       1
## 1710 0 282    370   744     0   f     0 33.9    3.28    10.1  1.2       4
```

```
## 1711 0 282    744  1455     0    f     0 33.9    3.04    10.2  1.3       7
```

Further, individuals who do die are treated a bit differently. For instance, take individual 268:

```
pbc2[pbc2$id == 268, ] # the orginal data
```

```
##       id tstart tstop death sex edema  age albumin protime bili
## 1652 268      0   164     0   f   0.5 55.4    2.75      11  6.4
## 1653 268    164  1191     2   f   0.5 55.4    2.78      11  6.8
```

```
pbc2_small_frame[pbc2_small_frame$id == 268, ] # new data
```

```
##       Y  id tstart tstop death sex edema  age albumin protime bili weights
## 1652  0 268      0   164     0   f   0.5 55.4    2.75      11  6.4       2
## 1653  0 268    164  1191     2   f   0.5 55.4    2.78      11  6.8       9
## 16531 1 268    164  1191     2   f   0.5 55.4    2.78      11  6.8       1
```

Notice, that we have to add an additional row with weight `1` where `Y = 1` as it would be wrong to give a weight of `10` to a the row with `Y = 1`. She survives for 11 bins and dies in the 12th bin

Finally, we can fit the model with the `glm` function using either of the two data frames as follows:

```
glm_fit_big <- glm(Y ~ age + edema + log(albumin) + log(protime) +
                     log(bili), binomial, pbc2_big_frame)
glm_fit_small <- glm(Y ~ age + edema + log(albumin) + log(protime) +
                       log(bili), binomial, pbc2_small_frame,
                      weights = pbc2_small_frame$weights) # <-- we use weights
```

We can confirm that the two models give the same estimate:

```
all.equal(glm_fit_big$coefficients, glm_fit_small$coefficients)
```

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

Further, the binning affects the estimates as shown below. In particular, it affects the estimates for `edema` and `log(albumin)`. The standard errors from the simple fit are also printed. However, these standard errors do not account for the dependence as we use multiple observations from the same individual

```
knitr::kable(
  rbind("glm with bins" = glm_fit_big$coefficients,
        "glm without bins" = glm_simple$coefficients,
        "Sds with bins" =
          summary(glm_fit_big)[["coefficients"]][, "Std. Error"],
        "Sds without bins" =
          summary(glm_simple)[["coefficients"]][, "Std. Error"]),
  row.names = T)
```

	(Intercept)	age	edema	log(albumin)	log(protime)	log(bili)
glm with bins	-10.34	0.046	1.066	-3.661	2.824	1.057
glm without bins	-9.38	0.054	1.435	-4.438	2.927	1.193
Sds with bins	2.20	0.010	0.282	0.548	0.787	0.110
Sds without bins	2.25	0.012	0.370	0.624	0.815	0.129

To end this section, you can skip making data frame with `get_survival_case_weigths_and_data` by calling the `static_glm` function from `dynamichazard` package. The call below yields the same results as shown:

```
static_glm_fit <- static_glm(
  Surv(tstart, tstop, death == 2) ~ age + edema + log(albumin) + log(protime) +
```

```r
    log(bili), data = pbc2, id = pbc2$id, by = 100, max_T = 3600)

all.equal(static_glm_fit$coefficients, glm_fit_big$coefficients)
```

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

For details, see the help file by typing `?static_glm`

Generalized Additive Models using mgvc

The first method we will compare with is Generalized Additive Models (GAM) by using the `gam` function in the `mgcv` package. The model we fit is of the form:

$$\begin{aligned}\text{logit}(\pi_i) = &\boldsymbol{\gamma}_{\text{time}}\boldsymbol{f}_{\text{time}}(t_i) + \boldsymbol{\gamma}_{\text{age}}\boldsymbol{f}_{\text{time}}(t_i)a_i + \boldsymbol{\gamma}_{\text{ede}}\boldsymbol{f}_{\text{ede}}(t_i)e_i + \boldsymbol{\gamma}_{\text{alb}}\boldsymbol{f}_{\text{alb}}(t_i)\log al_{it} \\ &+ \boldsymbol{\gamma}_{\text{pro}}\boldsymbol{f}_{\text{alb}}(t_i)\log p_{it} + \boldsymbol{\gamma}_{\text{biil}}\boldsymbol{f}_{\text{bili}}(t_i)\log b_{it}\end{aligned}$$

where π_{it} is the probability that the i'th individual dies of cancer, t is the stop time of the bin and a_i, e_i, al_{it}, p_{it} and b_{it} are respectively the i'th individual's age, edema, albumin, protime and bilirunbin. The extra subscript t is added to refer to the level of the covariate in the bin at time t. It is important to notice that we will use the same bins as in later fits. In particular, we will use the `pbc2_big_frame` data frame from before. $\boldsymbol{f}_{\cdot}$ is a smooth function. We will use cubic regression splines with knots spread evenly through the covariate values. We fit the model with the following call:

```r
library(mgcv, quietly = T)
spline_fit <- gam(
  formula = Y ~
    # cr is cubic spline with k knots
    s(t, bs = "cr", k = 3, by = age) +
    s(t, bs = "cr", k = 3, by = edema) +
    s(t, bs = "cr", k = 5, by = log(albumin)) +
    s(t, bs = "cr", k = 5, by = log(protime)) +
    s(t, bs = "cr", k = 5, by = log(bili)),
  family = binomial, data = pbc2_big_frame,
  method = "GCV.Cp") # estimate smoothing parameters with generalized cross
                     # validation
```

The above estimates the GAM model where the likelihood is penalized by a L2 norm for each of the spline functions. The tuning parameters are chosen by generalized cross validation. Below are plots of the estimates

```r
plot(spline_fit, scale = 0, page = 1, rug = F)
```

Further, we compare the result with static model. Recall that our static model had the following estimates:

```r
glm_fit_big$coefficients
```

```
##  (Intercept)          age        edema log(albumin) log(protime)    log(bili)
##     -10.3437       0.0461       1.0659      -3.6611       2.8239       1.0571
```

These do seem to correspond with the plots. Further, the intercept in the spline model is:

```r
spline_fit$coefficients["(Intercept)"]
```

```
## (Intercept)
##       -10.2
```

which again seems to match. The plot suggest that there may be time varying effects for `bili` particularly

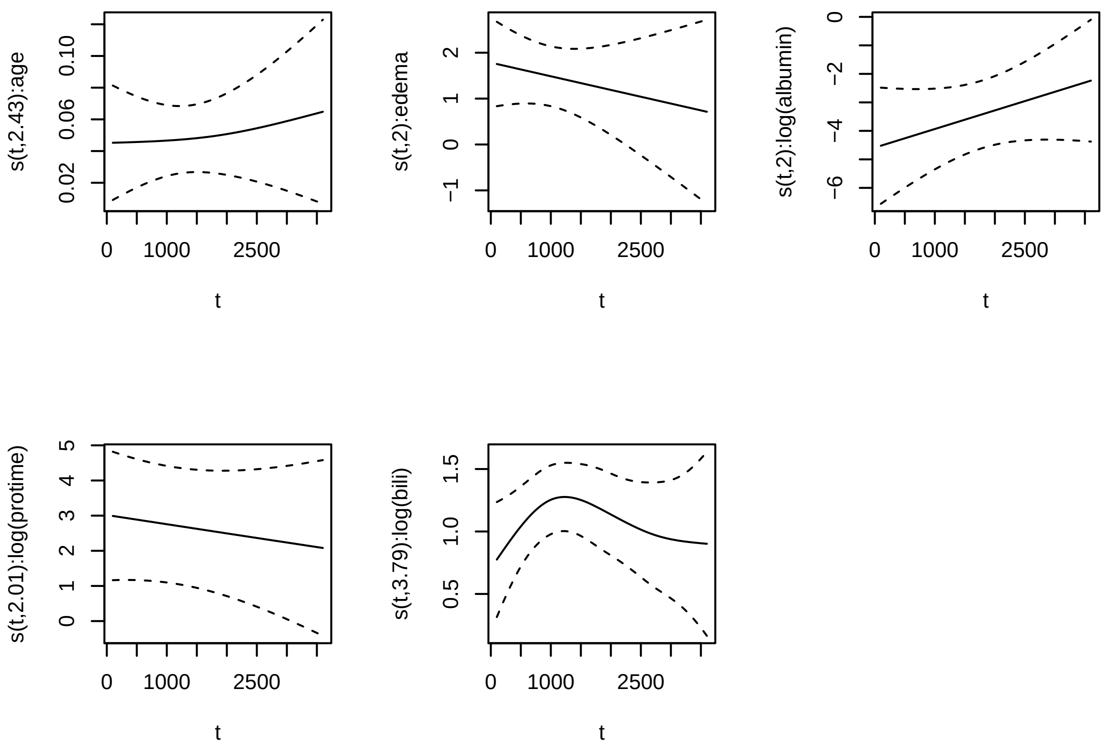


Figure 1: Plots of estimated effects in the GAM model. The effective degree of freedom is noted in the parentheses on the y axis and is computed given the number knots and final tuning parameter for spline function. For instance, `s(t,2.43):age` means that the effective degrees of freedom for `age` is `2.43`

Time varying cox model from `timereg`

Another method we can try is a time varying effects Cox model. We will use the Cox model from the package `timereg` based on Martinussen & Scheike (2007). The model differs from a regular Cox model (e.g. by using the `coxph` function from the `survival` package) by replacing the coefficient $\boldsymbol{\beta}$ by $\boldsymbol{\beta}(t)$ such that the instantaneous hazard is:

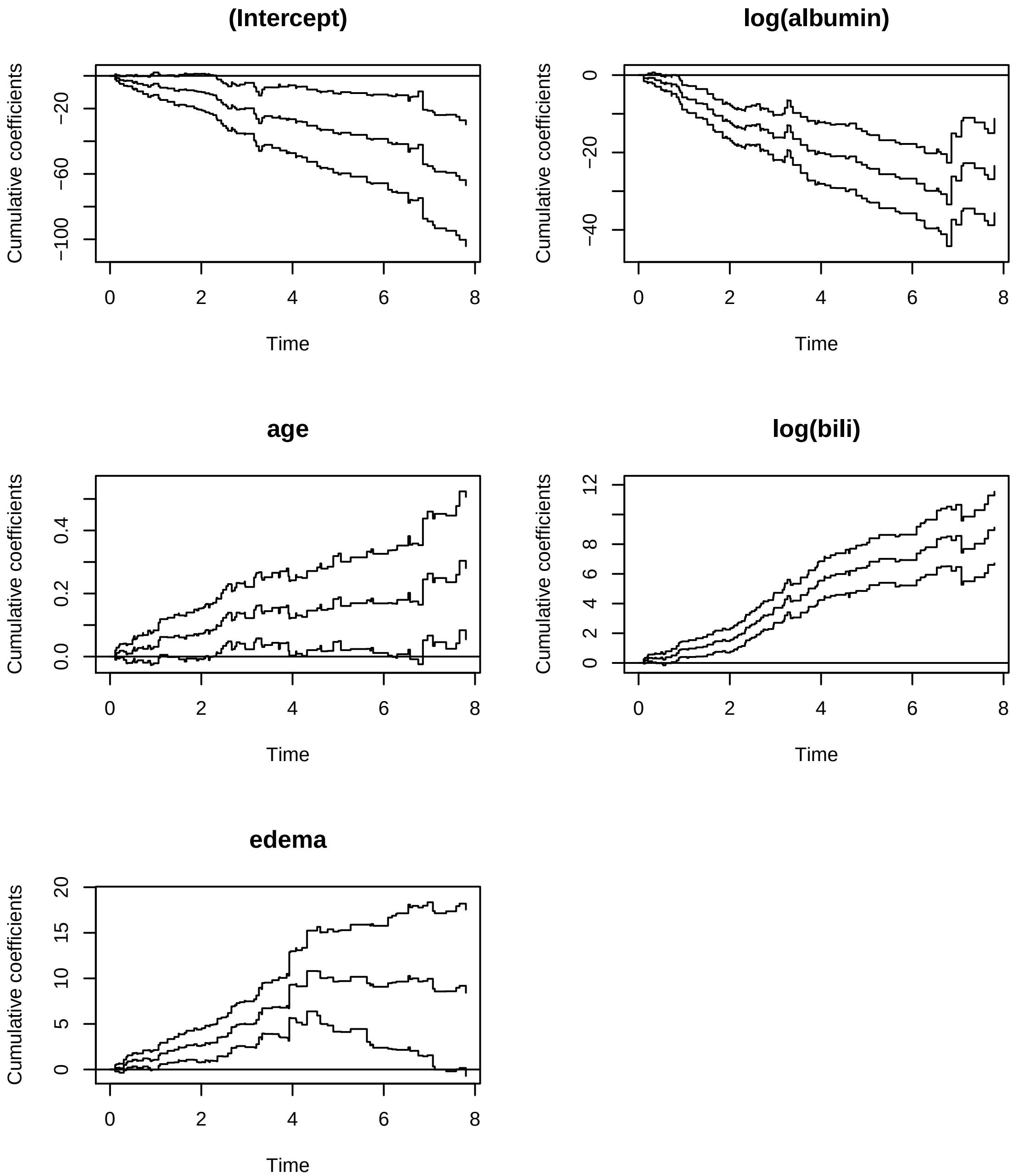
$$\lambda(t) = \lambda_0(t)\exp\left(\boldsymbol{x}\boldsymbol{\beta}(t)\right)$$

where each element of $\boldsymbol{\beta}(t)$ is estimated recursively with an update equation. The update equation is simplified through a first order Taylor expansion on the score function and by adding a smoothness by using weighting depending on time changes with a uniform continuous kernel. For details see Martinussen & Scheike (2007) in chapter 6. The estimation method differs from other alternatives in `R` that use splines for the time varying effects (like the `cox.ph` in the `mgcv` package). The baseline is $\lambda_0(t) = \exp(\alpha_0(t))$ where $\alpha_0(t)$ is estimated in a similar to way to $\boldsymbol{\beta}(t)$. Below we estimate a model similar to the previously fitted models

We set the last observation time (`max.time`) lower than in the previous model as there are issues with converge if we do not. For the same reason we specify the effect of `log(protime)` to be constant (non-time varying). To inspect the fitted model, the cumulated coefficients $B(t) = \int_0^t \beta(t)dt$ are plotted. Thus, a constant effect should be roughly linear

```
library(timereg)
cox_fit<- timecox(Surv(tstart / 365, tstop / 365, death == 2) ~ age + edema +
                        log(albumin) + const(log(protime)) + log(bili), pbc2,
                  start.time=0,
                  max.time = 3000 / 365, # <-- decreased
                  id = pbc2$id, bandwidth = 0.35)

par(mfcol = c(3, 2))
plot(cox_fit)
```



The timecox packages provides two test for whether the coefficient is time invariant or not:

```r
summary(cox_fit)
```

```
## Multiplicative Hazard Model
##
## Test for nonparametric terms
##
## Test for non-significant effects
##                 Supremum-test of significance p-value H_0: B(t)=0
## (Intercept)                              5.53               0.000
## age                                      3.37               0.014
```

```
## edema                                  4.65              0.000
## log(albumin)                           6.16              0.000
## log(bili)                              8.53              0.000
##
## Test for time invariant effects
##                    Kolmogorov-Smirnov test p-value H_0:constant effect
## (Intercept)                       15.9000                       0.211
## age                                0.0785                       0.828
## edema                              6.1500                       0.058
## log(albumin)                      13.1000                       0.199
## log(bili)                          1.0700                       0.703
##                      Cramer von Mises test p-value H_0:constant effect
## (Intercept)                      3.61e+02                       0.321
## age                              5.44e-03                       0.956
## edema                            5.80e+01                       0.086
## log(albumin)                     3.72e+02                       0.211
## log(bili)                        2.28e+00                       0.589
##
## Parametric terms :
##                     Coef.    SE Robust SE    z P-val lower2.5% upper97.5%
## const(log(protime))  2.43 0.815     0.966 2.51 0.012     0.829       4.02
##
##   Call:
## timecox(formula = Surv(tstart/365, tstop/365, death == 2) ~ age +
##     edema + log(albumin) + const(log(protime)) + log(bili), data = pbc2,
##     start.time = 0, max.time = 3000/365, id = pbc2$id, bandwidth = 0.35)
```

The above test suggest that only `edema` might be "border line" time varying

Dynamic hazard model

In this section, we will cover the dynamic hazard model with the logistic link function that is implemented in this package. The model which is estimated with EM algorithm which are from Fahrmeir (1994) and Fahrmeir (1992) when an Extended Kalman Filter is used in the E-step are. Firstly, we will briefly cover the model. See the ddhazard vignette for a more detailed explanation of the models. Secondly, we will turn to different ways of designing the model and fitting the model

The idea is that we discretize the outcomes into $1, 2, \dots, d$ bins. In each bin, we observe whether the individual dies or survives. The state space model we are applying is of the form:

$$\begin{array}{ll} \boldsymbol{y}_t = \boldsymbol{z}_t(\boldsymbol{\alpha}_t) + \boldsymbol{\epsilon}_t & \boldsymbol{\epsilon}_t \sim (\boldsymbol{0}, \text{Var}(\boldsymbol{y}_t|\boldsymbol{\alpha}_t)) \\ \boldsymbol{\alpha}_{t+1} = \mathbf{F}\boldsymbol{\alpha}_t + \mathbf{R}\boldsymbol{\eta}_t & \boldsymbol{\eta}_t \sim N(\boldsymbol{0}, \psi_t\mathbf{Q}) \end{array}, \qquad t = 1, \dots, d$$

where $y_{it} \in \{0, 1\}$ is an indicator for whether the i'th individual dies in interval t. $\cdots \sim (a, b)$ denotes a random variable with mean (or mean vector) a and variance (or co-variance matrix) b. It needs not to be normally distributed. $z_{it}(\boldsymbol{\alpha}_t) = h(\boldsymbol{\alpha}_t \boldsymbol{x}_{it})$ is the non-linear map from state space variables to mean outcomes where h is the inverse link function. We use the logit model in this note. Thus, $h(x) = \text{logit}^{-1}(x)$. The current implementation supports first and second order random walk for the state equation. Further, we define the conditional covariance matrix in the observational equation as $\mathbf{H}_t(\boldsymbol{\alpha}_t) = \text{Var}(\boldsymbol{y}_t|\boldsymbol{\alpha}_t)$. ψ_t is the length of the bin and is assumed equal for values of t since we use equidistant bins

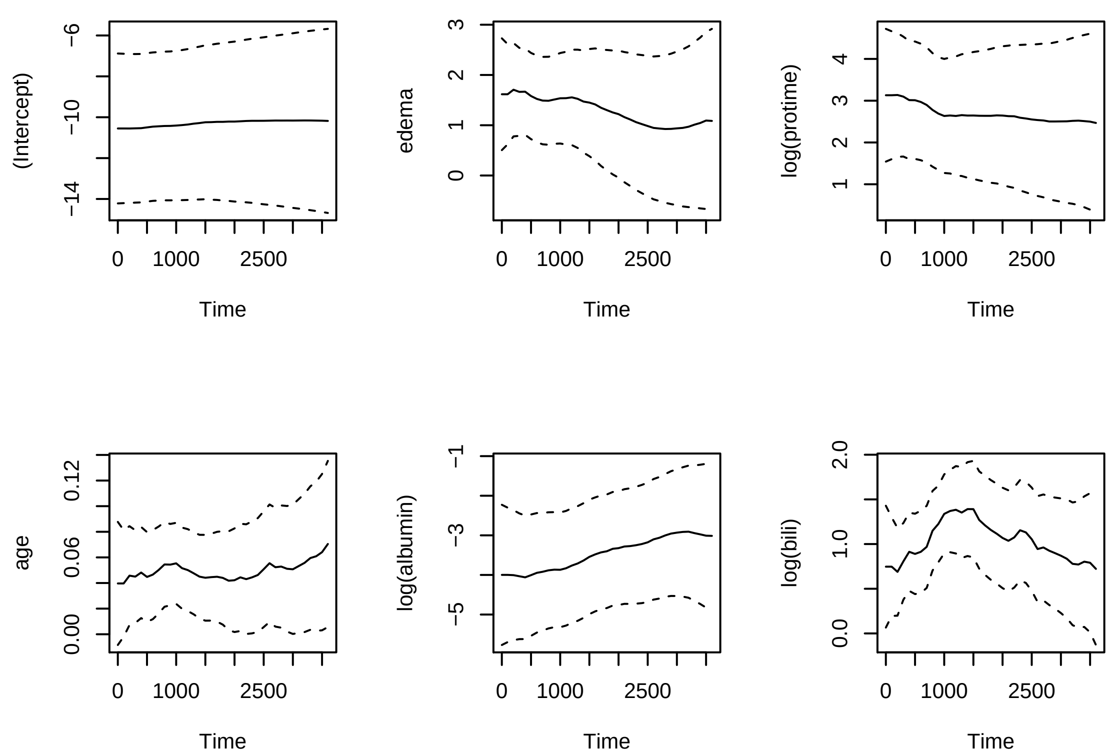
The unknown parameters are the starting value $\boldsymbol{\alpha}_0$ and co-variance matrix $\mathbf{Q}$. These are estimated with an EM-algorithm. The E-step is calculated using a Extended Kalman filter (EKF) or Unscented Kalman filter (UKF). Both are followed by a smoothing step. The result is smoothed predictions of $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_d$, smoothed co-variance matrix and smoothed correlation matrices that we need for the M-step to update $\boldsymbol{\alpha}_0$ and $\mathbf{Q}$. Further, we assume that $\mathbf{Q}_0 = \kappa\mathbf{I}$ is fixed to a large value κ in the examples. For more details see the ddhazard vignette

Estimation with Extended Kalman Filter

We start by estimating the model using the EKF where we let all coefficients follow a first order random walk:

```r
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(10, 6)), Q = diag(rep(0.001, 6)))

plot(dd_fit)
```



We start by estimating the model. The arguments `Q_0` and `Q` corresponds to the co-variance matrix at time zero and the co-variance matrix in the state equation. `Q_0` will remain fixed while `Q` is the starting value in the first iteration of the EM algorithm after which we update $\mathbf{Q}$

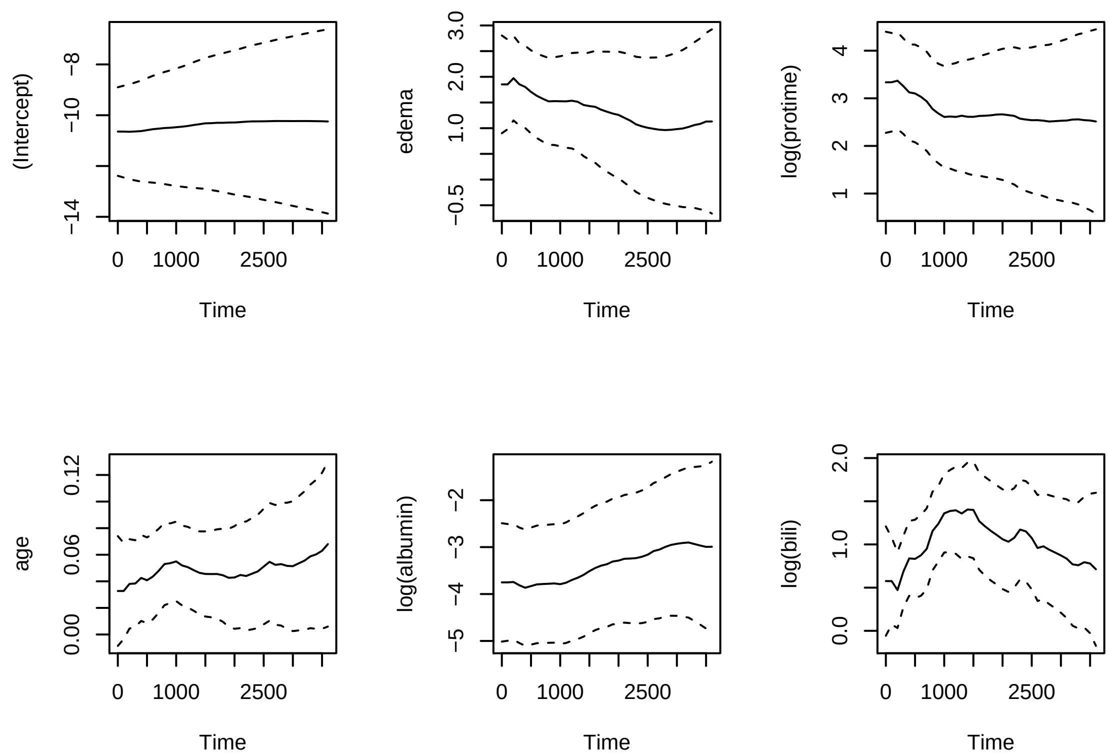
Next, we plot the coefficient. That is, we plot the predicted latent variables $\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d$. Notice that the predicted coefficient are close to the estimates we saw previously for the GAM model

Extra iteration in the correction step

Another idea is to take extra iterations in the correction step of the EKF. The motivation is that this step has the form of a Newton Rapshon algorithm as pointed out in Fahrmeir (1992) and Fahrmeir (1994). For more details see the ddhazard vignette. One of the motivation to use extra iterations is that experimental simulation have shown decrease predicted mean square error for one step ahead forecasts (see the vignette 'Simulation study with logit model'). Below, we estimate the model with potentially extra steps in the correction step.

```
dd_fit <- ddhazard(Surv(tstart, tstop, death == 2) ~ age + edema +
                        log(albumin) + log(protime) + log(bili), pbc2,
                   id = pbc2$id, by = 100, max_T = 3600,
                   a_0 = glm_fit_big$coefficients,
                   control = list(NR_eps = 0.1), # <-- tolerance in correction step
                   Q_0 = diag(rep(1, 6)),        # <-- decreased Q_0
                   Q = diag(rep(0.001, 6)))

# Plot result
plot(dd_fit)
```



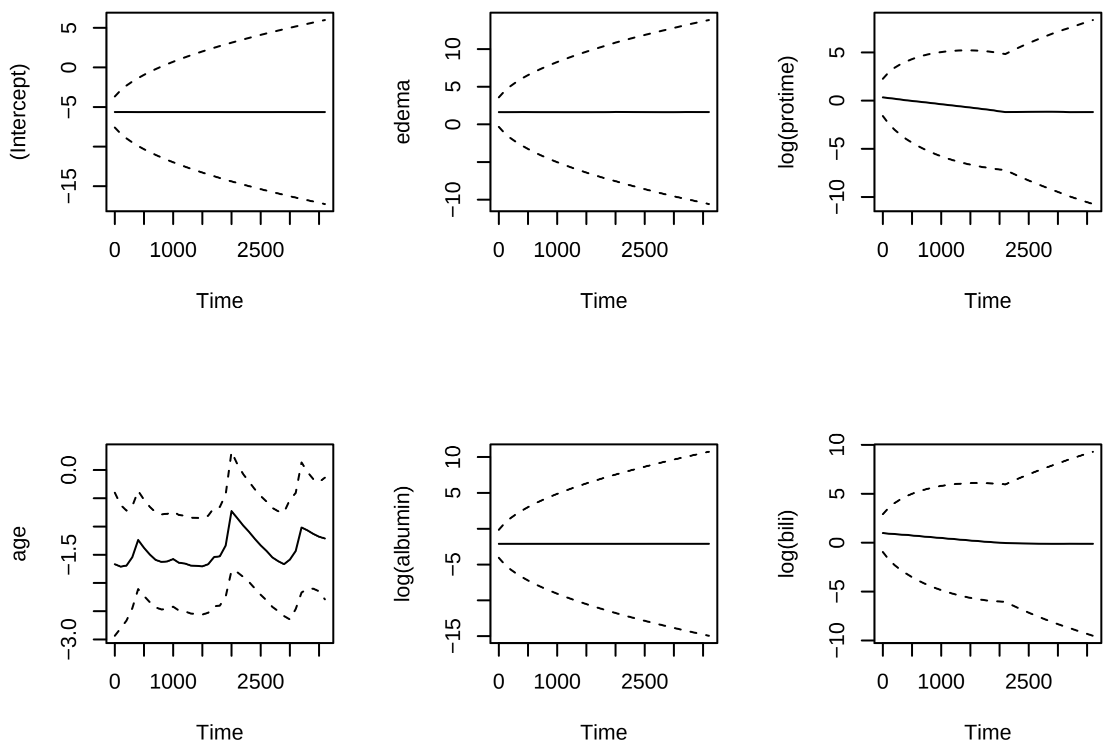
First, we run the code with the `NR_eps` element of the list passed to the `control` argument set to something that is finite. The value is the threshold for the relative change of in the state vector in correction step of the EKF. We end the code above by creating plots of the new estimates. The curves seems to be less smooth. Notice that we decrease `Q_0`. The algorithm diverges if we do not

Estimation with Unscented Kalman Filter

Another option is to perform the E-step using an unscented Kalman filter. This is done below. We start by setting the initial co-variance matrix $\mathbf{Q}$ large:

```
dd_fit_UKF <- 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(1, 6)), Q = diag(rep(0.01, 6)),
                       control = list(method = "UKF", beta = 0, alpha = 1,
                                      eps = 0.1, n_max = 1e4))

plot(dd_fit_UKF)
```



Clearly, the plots of the estimates are not what we expected. The reason is that $\mathbf{Q}_0$'s diagonal entries are quite large. The square root of the diagonal entries are used to form the sigma points in the first iteration. Hence, we mainly get estimates that are either zero or one when $\mathbf{Q}_0$ is a diagonal matrix with large entries. You can run the code below to see how the algorithm progress:

```
# Not run
tmp_file <- file("pick_some_file_name.txt")
sink(tmp_file)
dd_fit_UKF <- 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(1, 6)), Q = diag(rep(0.01, 6)),
                       control =
```

```r
                        list(method = "UKF", beta = 0, alpha = 1,
                             debug = T)) # <-- prints information in each iteration
sink()
close(tmp_file)
```

It will print quite a lot of information and hence it is recommended to use `sink` to write the output to a file. The main take away is that the conditional co-variance matrices accumulate in each iteration will the state vectors does not move. This motivates us to pick $\mathbf{Q}$ and $\mathbf{Q}_0$ more carefully. Our estimates from the EKF was:

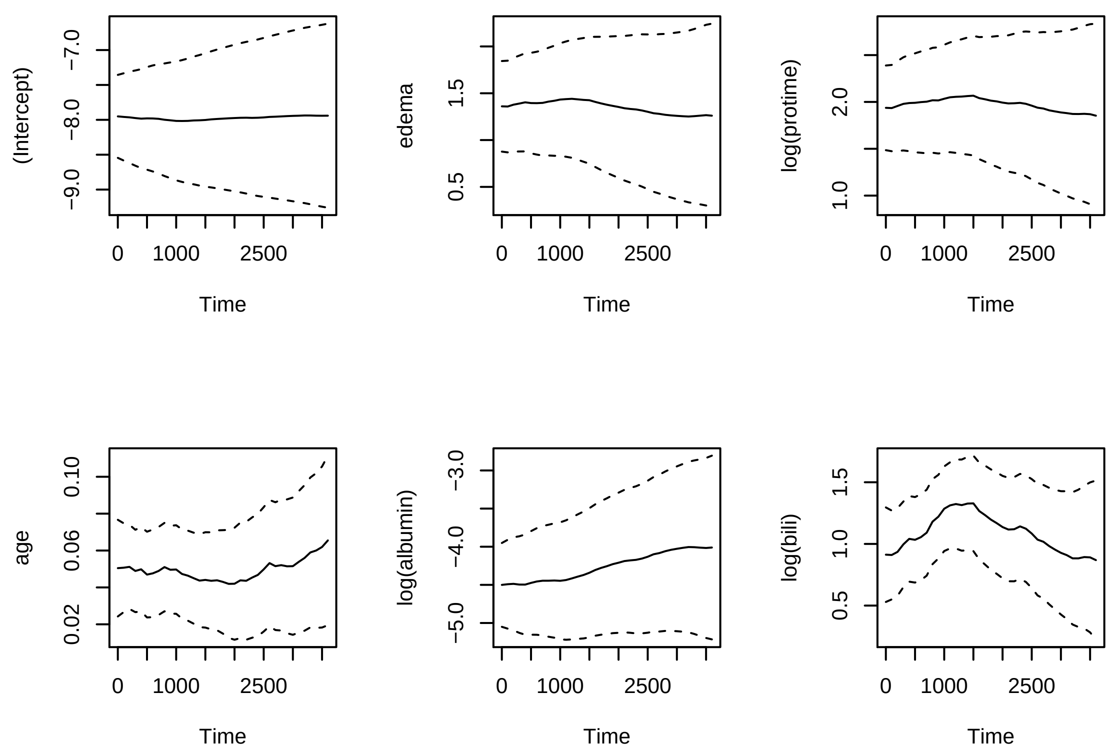
```r
diag(dd_fit$Q)
```

```
##  (Intercept)          age        edema log(albumin) log(protime)    log(bili)
##     8.90e-04     1.19e-06     6.39e-04     6.88e-04     5.90e-04     4.79e-04
```

which could motivate us to make the following choices:

```r
dd_fit_UKF <- 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(c(0.001, 0.00001, rep(0.001, 4))) * 100, # <-- decreased
  Q = diag(rep(0.0001, 6)),                            # <-- decreased
  control =
    list(method = "UKF", beta = 0, alpha = 1, eps = 0.01))

plot(dd_fit_UKF)
```



This is comparable to the fits from the EKF and GAM model. The main point here is:

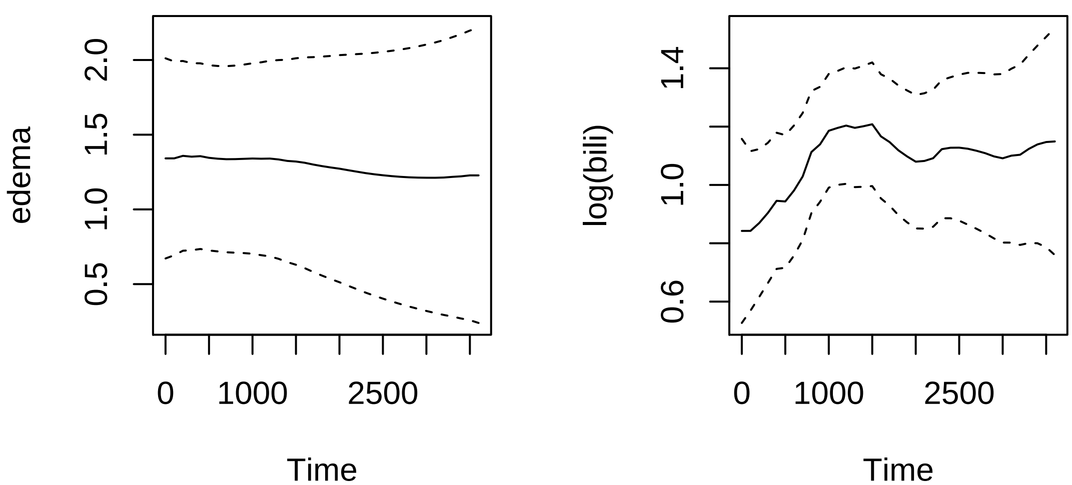
1. The UKF may work for certain data set. It may require "more care" for some data sets
2. The algorithm is sensitive to the choice of $\mathbf{Q}$ and $\mathbf{Q}_0$. Further, there is dependence on hyper parameters α, κ and β which we have not explored
3. The output you get by setting `debug` in the list passed to the `control` argument can be useful. Combine this with `sink` because the output may be long
4. The UKF has shown better performs than the EKF previously. Examples includes Romanenko & Castro (2004), Kandepu, Foss, & Imsland (2008), Julier & Uhlmann (2004), Wan & Van Der Merwe (2000) and chapter 11 of Durbin & Koopman (2012)

Estimation with fixed effects

We may want to fit a model where we assume that some of the co-variates are fixed. For instance, we may want to fit a model where `age`, the intercept, `log(protime)` and `log(albumin)` are fixed. We fix these based on the previous plots where the effects seems no to be time-varying. The model can be fitted as by the following call:

```
dd_fit <- ddhazard(
  Surv(tstart, tstop, death == 2) ~ ddFixed(1) +
    ddFixed(age) + ddFixed(log(albumin)) + edema +ddFixed(log(protime)) + log(bili),
  pbc2, id = pbc2$id, by = 100, max_T = 3600,
  Q_0 = diag(rep(1, 2)), Q = diag(rep(0.0001, 2)), control = list(eps = 0.02))
```

```
plot(dd_fit)
```



The estimates seems similar. Moreover, the fixed effects are in agreement with the previous fits (they are printed below):

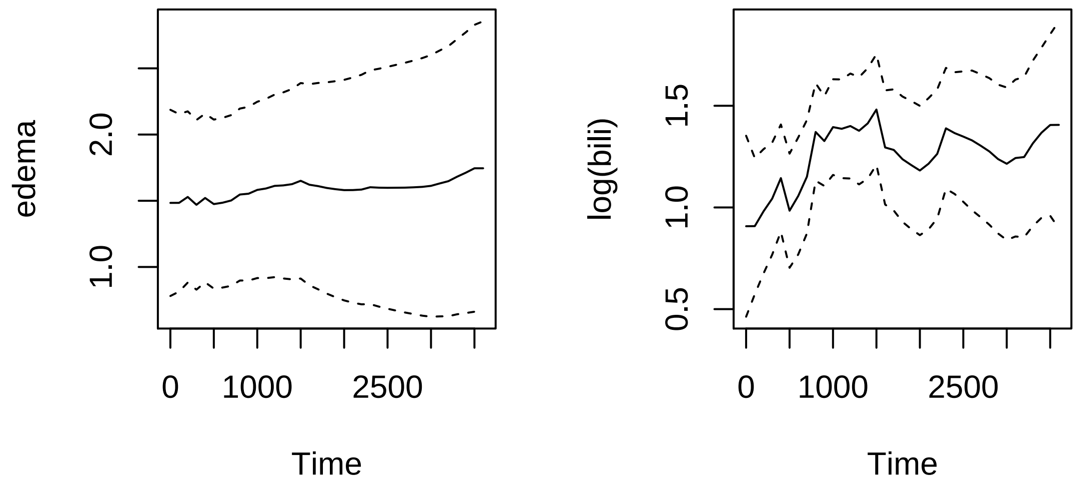
```
dd_fit$fixed_effects
```

```
## rep(1, nrow(data))                age        log(albumin)        log(protime)
##           -10.3102             0.0482             -3.6238              2.7188
```

We can also try to both fix some of terms and potentially take extra iterations in the correction step. This is done below:

```r
dd_fit <- ddhazard(
  Surv(tstart, tstop, death == 2) ~ ddFixed(1) +
    ddFixed(age) + ddFixed(log(albumin)) + edema +ddFixed(log(protime)) + log(bili),
  pbc2, id = pbc2$id, by = 100, max_T = 3600,
  Q_0 = diag(rep(1, 2)), Q = diag(rep(0.0001, 2)),
  control = list(eps = 0.02, NR_eps = 0.1))

plot(dd_fit)
```



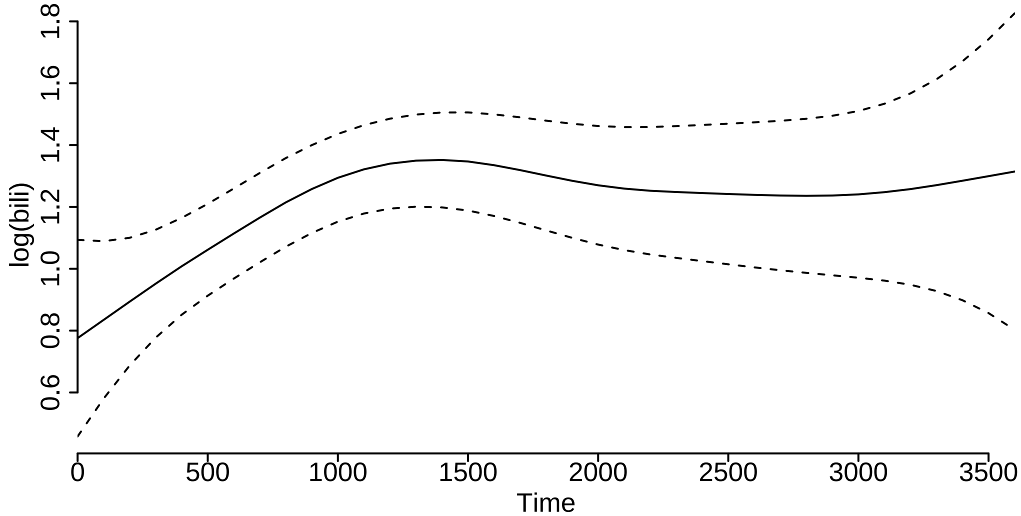
We see that the curve is less smooth similar to what we saw previously

Second order random walk

We will end by fitting a second order random walk to model where only `log(bili)` effect is time varying. The motivation is that the second order random walk tend to diverge more easily especially for small data sets. Further, the previous models seems to suggest that it is the only covariate where we may have a time varying coefficient. We fit the model below:

```r
dd_fit <- ddhazard(Surv(tstart, tstop, death == 2) ~
                     ddFixed(1) + ddFixed(age) +
                     ddFixed(edema) + ddFixed(log(albumin)) +
                     ddFixed(log(protime)) + log(bili), pbc2,
                   id = pbc2$id, by = 100, max_T = 3600,
                   order = 2,            # <-- second order
                   Q_0 = diag(c(10, 10)), # <-- needs more elements
                   Q = matrix(0.001))

plot(dd_fit)
```




```
dd_fit$fixed_effects
```

```
## rep(1, nrow(data))                 age               edema       log(albumin)
##           -10.5457              0.0523              1.4558            -3.5583
##       log(protime)
##             2.5986
```

We plot the curve after the fit. We see that the estimate is more smooth as expected with a second order random walk. Further, we can confirm that the fixed effects are comparable with our previous fits

Summary

We have estimated a model using Generalized additive models with the `mgcv` package and a time varying Cox model with the `timereg` package. Further, we have illustrated how the `ddhazard` function in the `dynamichazard` package can be used. In particular, we have used the logit link function and showed how to estimate with fixed effects, taking extra Newton Raphson steps in the correction step, fitting second order random walks and using the Unscented Kalman filter in the E-step. All the fits have been comparable with the Generalized Additive model

An unaddressed question is why you should this package. Some of the advantageous of the state space model here are among other:

1. You can extrapolate beyond the last observation time. An example hereof could be any time series where the underlying time is the calendar time such a medical trail where we may suspect a different effects of a drug in 2015 than in 1990
2. The implementation is linear in time complexity with the number of observations. Further, the Extended Kalman filter computes the correction step in parallel. Consequently, it is very fast even for large number observations
3. All estimation is made in `c++` with use of the `Armadillo` library. Thus, an optimized version of `Blas` or `Lapack` can decrease the computation time
4. It is an alternative to spline models. This relates to 1. The main tools for including time varying coefficient in `R` seems to apply of splines. Thus, this is an alternative estimation method

References

Durbin, J., & Koopman, S. J. (2012). *Time series analysis by state space methods*. Oxford University Press.

Fahrmeir, L. (1992). Posterior mode estimation by extended kalman filtering for multivariate dynamic generalized linear models. *Journal of the American Statistical Association*, *87*(418), 501–509.

Fahrmeir, L. (1994). Dynamic modelling and penalized likelihood estimation for discrete time survival data. *Biometrika*, *81*(2), 317–330.

Julier, S. J., & Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, *92*(3), 401–422.

Kandepu, R., Foss, B., & Imsland, L. (2008). Applying the unscented kalman filter for nonlinear state estimation. *Journal of Process Control*, *18*(7), 753–768.

Martinussen, T., & Scheike, T. H. (2007). *Dynamic regression models for survival data*. Springer Science & Business Media.

Romanenko, A., & Castro, J. A. (2004). The unscented filter as an alternative to the ekf for nonlinear state estimation: A simulation case study. *Computers & Chemical Engineering*, *28*(3), 347–355.

Shumway, T. (2001). Forecasting bankruptcy more accurately: A simple hazard model*. *The Journal of Business*, *74*(1), 101–124.

Tutz, G., & Schmid, M. (2016). Nonparametric modeling and smooth effects. In *Modeling discrete time-to-event data* (pp. 105–127). Springer.

Wan, E. A., & Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Adaptive systems for signal processing, communications, and control symposium 2000. as-spcc. the ieee 2000* (pp. 153–158). Ieee.