ddhazard

Benjamin Christoffersen

March 11, 2019

Introduction

This vignette will cover the `ddhazard` function used for estimation in the `dynamichazard` package You can install the version of the package used to make this vignette from github with the `devtools` package by calling `devtools::install_github("boennecd/dynamichazard@09bebe4")`. You can also get the latest version on CRAN by calling:

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

1.1 Why and when to use the ddhazard

The `ddhazard` function is intended for situation where you have a dynamic binary regression model or time-to-event model with time-varying coefficients. The advantage of the state spaces methods used here is that you can extrapolate to time periods beyond the data used in estimation. An example is forecasting firm failures given the firms present accounting data. The task is to use the present data to estimate a model and forecast the likelihood of default for the firms in the following year.

The estimation function `ddhazard` is implemented such that:

1. The time complexity of the computation is linear in the number of observations and in time.
2. The dimension of the observation equation can vary through time allowing for late entry and right censoring.
3. It is fast due to the `C++` implementation and supports multithreading.
4. The methods are fast compared to e.g., sequential Monte Carlo alternatives which are also implemented in the package (see `dynamichazard::PF_EM` and `dynamichazard::PF_forward_filter`).

1.2 Guide to vignettes

The vignette here is the primary vignette where the models and estimation methods are explained. The package also contains other supplementary vignettes. *Comparing methods for time-varying logistic models* shows the methods applied to a real world data set. The vignette illustrates how to use the estimation function `ddhazard` and other functions in this package. The vignette only uses the discrete time model. This vignette also describes the continuous time model. The *Bootstrap illustration* vignette shows how to use the `ddhazard_boot` function which is a wrapper for the `boot` function from the `boot` package. *ddhazard Diagnostics* illustrates how the `residuals` and `hatvalues` functions can be used to check the model fit.

1.3 Dynamic binary regression

We will introduce the setup and discrete model in the following paragraphs. We are observing individual $1, 2, \dots$ who each has an *event* at time $T_1, T_2, \dots$. We will also refer to an event as *death* as is typical in survival analysis. In addition we see covariate vectors $\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \dots$ for each individual i. Each covariate vector $\boldsymbol{x}_{ij}$ is valid in a period $(t_{i,j-1}, t_{ij}]$. Thus, a data frame may look as follows:

id	tstart	tstop	event	x1	x2
1	0.00	1.26	0	0.377	0.463
1	1.26	12.00	1	-0.241	-0.353
2	9.91	12.62	0	-0.140	0.122
2	12.62	16.94	0	0.188	0.352
2	16.94	18.45	0	-0.490	0.281
2	18.45	28.00	0	0.193	0.208
3	0.00	5.00	1	0.441	-0.476
4	0.00	1.09	0	0.297	0.352
4	1.09	7.00	1	-0.304	-0.324
5	24.71	28.00	0	-0.390	0.363

This is in the typical start-stop time format. The column `id` shows which individual the row belongs to. `tstart` is point at which the row is valid from and `tstop` is when the row is valid to. `event` is one if the individual dies at `tstop` and `x1` and `x2` are two covariates. Thus, the individual with id `1` dies at time 12 while id `2` survives all the periods we observe. The methods we look at will allow for right-censoring to handle an individuals like id `2`, and left-censoring (delayed entry).

We will put the observations into intervals $1, 2, \dots, d$ each with length $\psi_1, \psi_2, \dots, \psi_d$. That is, we observe a total of d intervals. Assume that each $\psi_t = 1$ for simplicity. Then we define the a series of indicators for each individual given by:

$$y_{ijt} = 1_{\{T_i \in (t_{i,j-1}, t_{ij}] \wedge t-1 < t_{ij} \leq t\}}$$

which denotes whether individual i has event with the j'th covariate vector in interval t. Next, the *risk set* in interval t is given by:

$$R_t = \{(i,j) \in \mathbb{Z}_+ \times \mathbb{Z} : \, t_{i,j-1} \leq t-1 \leq t_{ij}\}$$

where $\mathbb{Z}$ are the natural numbers. We will refer to this as the *discrete risk set* as we later introduce a continuous version. For simplicity we assume that we have removed all observation that are strictly inside an interval. I.e. those where:

$$\exists t \in \mathbb{Z}_+ : t-1 < t_{i,j-1} < t_{ij} < t$$

Further, we change the event flag for the last observation in case an individual has an event with a covariate vector inside an interval. Later, we introduce the continuous model where we can handle the information of such observations. For a given individual i who has covariate vector j in interval t, we model the chance of an event by:

$$P\left(Y_{ijt} = 1 | \, \boldsymbol{y}_1, \dots, \boldsymbol{y}_{t-1}, \boldsymbol{\alpha}_t\right) = h(\boldsymbol{\alpha}_t^\top \boldsymbol{x}_{ijt})$$

where $\boldsymbol{y}_t$ is the vector of outcomes given risk set R_t and h is the inverse link function. For example, this could be the inverse logistic function such that $h(\eta) = \exp(\eta)/(1+\exp(\eta))$. The `ddhazard` function estimates models in the state space form:

$$\begin{array}{l} \boldsymbol{y}_t \sim P\left(\boldsymbol{y}_t | \, \boldsymbol{\alpha}_t\right) \\ \boldsymbol{\alpha}_{t+1} = \mathbf{F}\boldsymbol{\alpha}_t + \mathbf{R}\boldsymbol{\eta}_t \quad \boldsymbol{\eta}_t \sim N(\mathbf{0}, \psi_t \mathbf{Q}) \end{array}, \qquad t = 1, \dots, d$$

$\boldsymbol{\alpha}_t$ is the state vector with the corresponding *state equation*. Again, we will fix $\psi_t = 1$. However, the `ddhazard` function is implemented to handle any equidistant interval length. That is, $\psi_t = \psi$ for a pre-specified constant ψ. Further, we let $\mathbf{H}_t(\boldsymbol{\alpha}_t) = \text{Var}\left(\boldsymbol{y}_t | \, \boldsymbol{\alpha}_t\right)$ denote the conditional covariance matrix of $\boldsymbol{y}_t \mid \boldsymbol{\alpha}_t$ and let $\boldsymbol{z}_t(\boldsymbol{\alpha}_t)$ denote the conditional mean. They are defined by:

$$z_{kt}(\boldsymbol{\alpha}_t) = E\left(Y_{ijt} | \, \boldsymbol{\alpha}_t\right) = h(\boldsymbol{\alpha}_t^\top \boldsymbol{x}_{ijt}) \tag{1}$$

$$\begin{aligned} H_{kk't}(\boldsymbol{\alpha}_t) &= \begin{cases} \text{Var}\left(Y_{ijt} | \, \boldsymbol{\alpha}_t\right) & f = f' \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} z_{kt}(\boldsymbol{\alpha}_t)(1 - z_{kt}(\boldsymbol{\alpha}_t)) & k = k' \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

where we assumed that individual i with covariate vector j was at the k'th index of the risk set at time t. The state equation is implemented with a 1. and 2. order random walk. For the first order random walk $\mathbf{F} = \mathbf{R} = \mathbf{I}_m$ where m is the number of time varying coefficients and $\mathbf{I}_m$ is the identity matrix with dimension m. As for the second order random walk, we have:

$$\mathbf{F} = \begin{pmatrix} 2\mathbf{I}_m & -\mathbf{I}_m \\ \mathbf{I}_m & \mathbf{0}_m \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} \mathbf{I}_m \\ \mathbf{0}_m \end{pmatrix}$$

where $\mathbf{0}_m$ is a $m \times m$ matrix with zeros in all entries. We let the linear predictor in equation (2) in the second order random walk as

$$(\mathbf{R}^\top \boldsymbol{\alpha}_t)^\top \boldsymbol{x}_{ijt} = \boldsymbol{\xi}_t^\top \boldsymbol{x}_{ijt}$$

where $\boldsymbol{\xi}_t = \mathbf{R}^\top \boldsymbol{\alpha}_t$ and we order state equation such that $\boldsymbol{\alpha}_t = (\boldsymbol{\xi}_t^\top, \boldsymbol{\xi}_{t-1}^\top)^\top$ to match the definition of $\mathbf{F}$ and $\mathbf{R}$. The likelihood of the model where $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_d$ are observed can be written as follows by application of the Markov property of the model:

$$\begin{aligned} P\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d, \boldsymbol{y}_t, \dots, \boldsymbol{y}_T\right) &= L\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right) \\ &= p(\boldsymbol{\alpha}_0) \prod_{t=1}^{d} P\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}\right) \prod_{(i,j) \in R_t} P\left(y_{ijt} \middle| \boldsymbol{\alpha}_t\right) \end{aligned}$$

which we can expand to (omitting a normalization constant):

$$\begin{aligned} \mathcal{L}\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right) = \log L\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right) = &- \frac{1}{2}\left(\boldsymbol{\alpha}_0 - \boldsymbol{a}_0\right)^\top \mathbf{Q}_0^{-1}\left(\boldsymbol{\alpha}_0 - \boldsymbol{a}_0\right) \\ &- \frac{1}{2}\sum_{t=1}^{d}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}\right)^\top \mathbf{R}^\top \psi_t^{-1} \mathbf{Q}^{-1} \mathbf{R}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}\right) \\ &- \frac{1}{2}\log\left|\mathbf{Q}_0\right| - \frac{d}{2}\log\left|\mathbf{Q}\right| \\ &+ \sum_{t=1}^{d}\sum_{(i,j) \in R_t} l_{ijt}(\boldsymbol{\alpha}_t) + \dots \end{aligned}$$

$$l_{ijt}(\boldsymbol{\alpha}_t) = y_{ijt} \log h(\boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha}_t) + (1 - y_{ijt}) \log\left(1 - h(\boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha}_t)\right)$$

The unknown parameters are the initial state vector $\boldsymbol{\alpha}_0$ and the covariance matrix $\mathbf{Q}$. We estimate these using an EM-algorithm. The E-step is carried out first by filtering with an Extended Kalman filter (EKF), an Unscented Kalman filter (UKF) or an approximation of the posterior modes. We apply a smoother after using the filter. The method is chosen by the `method` argument of the `ddhazard_control` function passed to the `control` argument of `ddhazard` (e.g., `ddhazard_control(method = "EKF", ...)`). All filtering methods requires an initial state vector $\boldsymbol{\alpha}_0$, co-variance matrix $\mathbf{Q}$ and initial co-variance matrix $\mathbf{Q}_0$ to start.

A key thing to notice (and a likely source of error if forgotten) is that the `Q` argument for $\mathbf{Q}$ is scaled by the length of the time interval, ψ_t. The motivation for this behavior is that you can alter ψ_t and get comparable estimates of `Q`. Further, it will also be useful if unequal intervals lengths are implemented later. `Q_0` is not scaled and thus will exactly match $\mathbf{Q}_0$ in the estimation. The reasoning is that $\mathbf{Q}_0$ is independent of our time interval length and reflects our uncertainty of $\boldsymbol{\alpha}_0$

We will make two fits to illustrate how a call to `ddhazard` looks and to show that $\mathbf{Q}$ will be scaled by ψ_t. To do so, we use the `pbc` data set from the `survival` package. The first call is:

```
library(dynamichazard)
dd_fit_short <- ddhazard(
  Surv(time, status == 2) ~ log(bili), # Formula like for coxph from survival
  data = pbc,
```

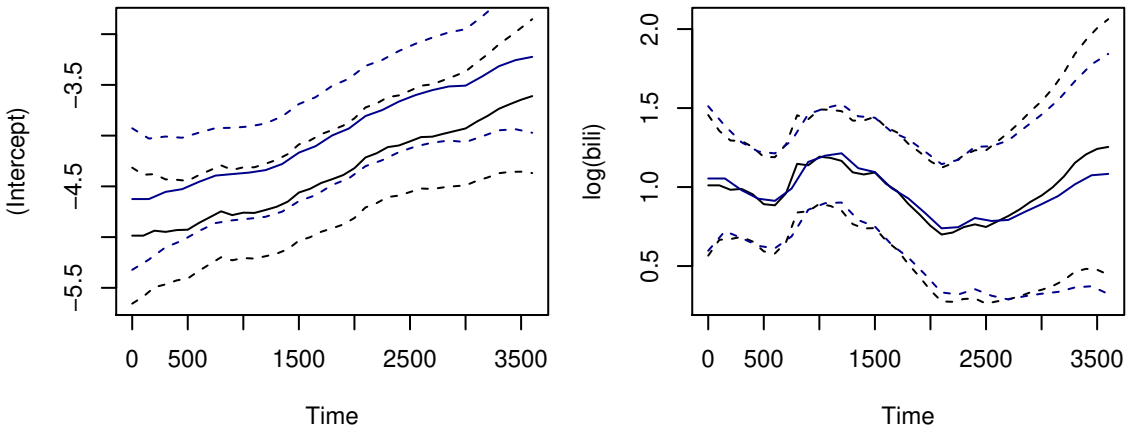
```
  by = 100,                               # Length of time intervals
  Q = diag(.3^2, 2),                      # Covariance matrix in state eqn
  Q_0 = diag(2^2, 2),                     # Covariance matrix for initial state
                                          # vector
  max_T = 3600,                           # Last time we observe
  id = pbc$id,                            # id of individuals
  control = ddhazard_control(eps = 1e-4))

# Print diagonal of covariance matrix
diag(dd_fit_short$Q)

## (Intercept)   log(bili)
##    0.000244    0.000211
```

Above, we estimate the model with a time intervals of length `by = 100`. The model is the logistic model which we introduced later. For now, let us see what happens if we increase the interval length by changing the `by` argument:

```
library(dynamichazard)
dd_fit_wide <- ddhazard(
  Surv(time, status == 2) ~ log(bili),
  data = pbc,
  by = 150,                           # Increased
  Q = diag(.3^2, 2),
  Q_0 = diag(2^2, 2),
  max_T = 3600,
  id = pbc$id,
  control = ddhazard_control(eps = 1e-4))

# Print relative differences between diagonal of covariance matrices
Q_short <- sqrt(diag(dd_fit_short$Q))
Q_wide <- sqrt(diag(dd_fit_wide$Q))
(Q_wide - Q_short) / Q_short

## (Intercept)   log(bili)
##    -0.00465    -0.08954
```

We see that the diagonal entries are not "too far" from each other with the two fits. Plots of the two sets of predicted coefficients are similar in terms of width of the confidence bounds (black is the short intervals and blue is long intervals):

```
par(mfcol = c(1, 2), mar = c(5, 4, 1, 1), cex = .75)

for(i in 1:2){
  plot(dd_fit_short, cov_index = i, col = "Black")
  plot(dd_fit_wide, cov_index = i, col = "DarkBlue", add = T)
}
```



Further, as expected the intercept is larger when we use longer intervals length. To ease the notation, we assume that $\psi_t = 1$ in the rest of the vignette. The rest of this vignette is structured as follows. Section 2 will cover the EM algorithm. This is followed by the sections 3, 4, and sections 5 and 6 which respectively covers the extended Kalman filter, unscented Kalman filter, and approximation of the posterior mode used to perform the filtering in the E-step of the EM algorithm. Next, the section 7 and 8 covers how estimation is done with weights or fixed effects. The sections 9 and 10 covers the models implemented in this package. Finally, we end with a section 11.

I encourage you to use the shiny app while reading this vignette. You can launch the shiny app by installing this package and running:

```
dynamichazard::ddhazard_app()
```

The app will allow you to compare the methods and models described here on simulated data sets.

2 EM algorithm

An EM algorithm is used to estimate the initial state space vector $\boldsymbol{\alpha}_0$ and the co-variance matrix $\mathbf{Q}$. Optionally $\mathbf{Q}_0$ is also estimated if `control = ddhazard_control(est_Q_0 = T, ...)`. Though, we do not have sufficient information to estimate this matrix. Define

$$\boldsymbol{a}_{t|s} = E\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_1, \dots, \boldsymbol{y}_s\right), \qquad \mathbf{V}_{t|s} = E\left(\mathbf{V}_t | \boldsymbol{y}_1, \dots, \boldsymbol{y}_s\right)$$

for the conditional mean and co-variance matrix. Notice that the letter 'a' is used for mean estimates while 'alpha' is used for the unknown state as is typical in the state space literature. The notation above both covers filter estimates in the case where $s \leq t$ and smoothed estimates when $s > t$. We suppress the dependence of the covariates $(\boldsymbol{x}_{ijt})$ here to simplify the notation. The initial values for $\boldsymbol{\alpha}_0$, $\mathbf{Q}$ and $\mathbf{Q}_0$ can be set by passing a vector for the `a_0` argument of `ddhazard` for $\boldsymbol{\alpha}_0$ and matrices to `Q_0` and `Q` argument of `ddhazard` for respectively $\mathbf{Q}_0$ and $\mathbf{Q}$.

2.1 E-step

The outcome of the E-step are the smoothed estimates:

$$\boldsymbol{a}_{t|d}^{(k)}, \qquad \mathbf{V}_{t|d}^{(k)}, \qquad t = 0, 1, \dots, d$$

where d is the number of periods we observe. Superscripts $\cdot^{(k)}$ is used to distinguish between the estimates from each iteration of the EM-algorithm. Thus, $\boldsymbol{a}_{t|d}^{(k)}$ is the smoothed state space vector for interval t in iteration k of the EM algorithm. The required input to start the E-step is an initial mean vector $\widehat{\boldsymbol{a}}_0^{(k-1)}$ and co-variance matrix $\widehat{\mathbf{Q}}^{(k-1)}$. Given these input, we compute the following estimates by using a filter:

$$\boldsymbol{a}_{j|j-1}, \quad \boldsymbol{a}_{i|i}, \quad \mathbf{V}_{j|j-1}, \quad \mathbf{V}_{i|i}, \quad i = 0, 1, \dots, d \wedge j = 1, 2, \dots, d$$

Then the estimates are smoothed by computing:

$$\begin{aligned}
\mathbf{B}_t^{(k)} &= \mathbf{V}_{t-1|t-1}\mathbf{F}\mathbf{V}_{t|t-1}^{-1} \\
\boldsymbol{a}_{t-1|d}^{(k)} &= \boldsymbol{a}_{t-1|t-1} + \mathbf{B}_t(\boldsymbol{a}_{t|d}^{(k)} - \boldsymbol{a}_{t|t-1}) \qquad\qquad t = d, d-1, \dots, 1 \\
\mathbf{V}_{t-1|d}^{(k)} &= \mathbf{V}_{t-1|t-1} + \mathbf{B}_t(\mathbf{V}_{t|d}^{(k)} - \mathbf{V}_{t|t-1})\mathbf{B}_t^\top
\end{aligned}$$

2.2 Kalman Filter

The standard Kalman filter is carried out by recursively doing a prediction step and a correction step. This also applies for all the implemented filters. Thus, this paragraph is included to introduce general notions. The first step in the Kalman Filter is the *prediction step* where we estimate $\boldsymbol{a}_{t|t-1}$ and $\mathbf{V}_{t|t-1}$ based on $\boldsymbol{a}_{t-1|t-1}$ and $\mathbf{V}_{t-1|t-1}$. Secondly, we carry out the *correction step* where we estimate $\boldsymbol{a}_{t|t}$ and $\mathbf{V}_{t|t}$ based on $\boldsymbol{a}_{t|t-1}$ and $\mathbf{V}_{t|t-1}$ and the observations. We repeat the process until $t = d$.

2.3 M-step

The M-step updates the mean $\widehat{\boldsymbol{a}}_0^{(k-1)}$ and co-variance matrices $\widehat{\mathbf{Q}}^{(k-1)}$ and $\widehat{\mathbf{Q}}_0^{(k-1)}$ (the latter being optional). These are computed by:

$$\begin{aligned}
\widehat{\boldsymbol{\alpha}}_0^{(k)} &= \boldsymbol{a}_{0|d}^{(k)}, \qquad \widehat{\mathbf{Q}}_0^{(k)} = \mathbf{V}_{0|d}^{(k)} \\
\widehat{\mathbf{Q}}^{(k)} &= \frac{1}{d}\sum_{t=1}^{d} \mathbf{R}^\top \Bigg(\left(\boldsymbol{a}_{t|d}^{(k)} - \mathbf{F}\boldsymbol{a}_{t-1|d}^{(k)}\right)\left(\boldsymbol{a}_{t|d}^{(k)} - \mathbf{F}\boldsymbol{a}_{t-1|d}^{(k)}\right)^\top \\
&\qquad + \mathbf{V}_{t|d}^{(k)} - \mathbf{F}\mathbf{B}_t^{(k)}\mathbf{V}_{t|d}^{(k)} - \left(\mathbf{F}\mathbf{B}_t^{(k)}\mathbf{V}_{t|d}^{(k)}\right)^\top + \mathbf{F}\mathbf{V}_{t-1|d}^{(k)}\mathbf{F}^\top \Bigg)\mathbf{R}
\end{aligned}$$

We check the relative norm of the change in the state vectors to check for convergence. You can select the threshold for convergence by setting the `eps` argument of `ddhazard_control` which is passed to the `control` argument of `ddhazard` (e.g., `ddhazard_control(eps = 0.001, ...)`). Algorithm 1 shows the method.

3 Extended Kalman Filter

The idea of the Extended Kalman filter is to replace the observational equation with a first order Taylor expansion. This approximated model can then be estimated with a regular Kalman Filter. The EKF presented here is originally described in Fahrmeir [1992] and Fahrmeir [1994] where the EM-algorithm as shown above is also from. The formulation in Fahrmeir [1994] differs from the standard Kalman Filter by re-writing the correction step using the Woodbury matrix identity. This has two computational advantages. The first one is that the time complexity is $O(n_t)$ instead of $O(n_t^3)$ where $n_t = |R_t|$ denotes the cardinality of the risk set. Secondly, we do not have store an intermediate $n_t \times n_t$ matrix. The EKF starts with prediction step where we compute:

$$\begin{aligned}
\boldsymbol{a}_{t|t-1} &= \mathbf{F}\boldsymbol{a}_{t-1|t-1}, \\
\mathbf{V}_{t|t-1} &= \mathbf{F}\mathbf{V}_{t-1|t-1}\mathbf{F}^\top + \mathbf{R}\mathbf{Q}\mathbf{R}^\top
\end{aligned}$$

Secondly, we perform the correction step by:

$$\begin{aligned}
\mathbf{V}_{t|t} &= \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}_{t|t-1})\right)^{-1} \\
\boldsymbol{a}_{t|t} &= \boldsymbol{a}_{t|t-1} + \mathbf{V}_{t|t}\boldsymbol{u}_t(\boldsymbol{a}_{t|t-1})
\end{aligned}$$

where $\boldsymbol{u}_t(\boldsymbol{a}_{t|t-1})$ and $\mathbf{U}_t(\boldsymbol{a}_{t|t-1})$ are given by:

Algorithm 1 EM algorithm with unspecified filter.

Input:

$\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{X}_1, \dots, \mathbf{X}_d, \boldsymbol{y}_1, \dots, \boldsymbol{y}_d, R_1, \dots, R_d$

Convergence threshold ϵ

Set $\boldsymbol{a}_{0|0}^{(0)} = \boldsymbol{a}_0$ and $\mathbf{Q}^{(0)} = \mathbf{Q}$

for $k = 1, 2, \dots$ **do**

  procedure E-STEP

    Apply filter with $\boldsymbol{a}_{0|0}^{(k-1)}$, $\mathbf{Q}^{(k-1)}$ and $\mathbf{Q}_0$ to get

      $\boldsymbol{a}_{1|0}$, $\boldsymbol{a}_{1|1}$, $\boldsymbol{a}_{2|1}, \dots, \boldsymbol{a}_{d|d-1}$, $\boldsymbol{a}_{d|d}$ and

      $\mathbf{V}_{1|0}$, $\mathbf{V}_{1|1}$, $\mathbf{V}_{2|1}, \dots, \mathbf{V}_{d|d-1}$, $\mathbf{V}_{d|d}$

    Apply smoother by computing

    for $t = d, d-1, \dots, 1$ **do**

      $\mathbf{B}_t^{(k)} = \mathbf{V}_{t-1|t-1}\mathbf{F}\mathbf{V}_{t|t-1}^{-1}$

      $\boldsymbol{a}_{t-1|d}^{(k)} = \boldsymbol{a}_{t-1|t-1} + \mathbf{B}_t^{(k)}(\boldsymbol{a}_{t|d}^{(k)} - \boldsymbol{a}_{t|t-1})$

      $\mathbf{V}_{t-1|d}^{(k)} = \mathbf{V}_{t-1|t-1} + \mathbf{B}_t^{(k)}(\mathbf{V}_{t|d}^{(k)} - \mathbf{V}_{t|t-1})\left(\mathbf{B}_t^{(k)}\right)^\top$

  procedure M-STEP

    Update the initial state and the covariance matrix by

    $\boldsymbol{a}_{0|0}^{(k)} = \boldsymbol{a}_{0|d}^{(k)}$

    $$\mathbf{Q}^{(k)} = \frac{1}{d}\sum_{t=1}^{d}\mathbf{R}^\top\left(\left(\boldsymbol{a}_{t|d}^{(k)} - \mathbf{F}\boldsymbol{a}_{t-1|d}^{(k)}\right)\left(\boldsymbol{a}_{t|d}^{(k)} - \mathbf{F}\boldsymbol{a}_{t-1|d}^{(k)}\right)^\top + \mathbf{V}_{t|d}^{(k)} - \mathbf{F}\mathbf{B}_t^{(k)}\mathbf{V}_{t|d}^{(k)} - \left(\mathbf{F}\mathbf{B}_t^{(k)}\mathbf{V}_{t|d}^{(k)}\right)^\top + \mathbf{F}\mathbf{V}_{t-1|d}^{(k)}\mathbf{F}^\top\right)\mathbf{R}$$

    Stop the if sum of relative norm of changes is below the threshold

    $\sum_{t=0}^{d}\frac{\left\|\boldsymbol{a}_{t|d}^{(k)} - \boldsymbol{a}_{t|d}^{(k-1)}\right\|}{\left\|\boldsymbol{a}_{t|d}^{(k-1)}\right\|} < \epsilon$

$$\boldsymbol{u}_t(\boldsymbol{\alpha}_t) = \sum_{(i,j)\in R_t}\boldsymbol{u}_{ijt}(\boldsymbol{\alpha}_t), \quad \boldsymbol{u}_{ijt}(\boldsymbol{\alpha}_t) = \boldsymbol{x}_{ijt}\frac{\partial h(\eta)/\partial\eta}{H_{kkt}(\boldsymbol{\alpha}_t)}\left(y_{ijt} - h(\eta)\right)\Big|_{\eta=\boldsymbol{x}_{ijt}^\top\boldsymbol{\alpha}_t}$$

$$\mathbf{U}_t(\boldsymbol{\alpha}_t) = \sum_{(i,j)\in R_t}\mathbf{U}_{ijt}(\boldsymbol{\alpha}_t), \quad \mathbf{U}_{ijt}(\boldsymbol{\alpha}_t) = \boldsymbol{x}_{ijt}\boldsymbol{x}_{ijt}^\top\frac{\left(\partial h(\eta)/\partial\eta\right)^2}{H_{kkt}(\boldsymbol{\alpha}_t)}\Big|_{\eta=\boldsymbol{x}_{ijt}^\top\boldsymbol{\alpha}_t}$$

R_t is the set of indices of individuals who are at risk in time interval t. Further, the ks in $H_{kkt}(\boldsymbol{\alpha}_t)$ are set such that the match with the i'th individuals j'th covariate matrix. Algorithm 2 shows the EKF method.

3.1 Divergence

Initial testing showed that the EKF has issues with divergence for some data set. The cause of divergence seems to be overstepping in the correction step where we update $\boldsymbol{a}_{t|t}$. In particular, the signs of the elements of $\boldsymbol{a}_{t|t}$ tends to alter between $t-1, t, t+1$ etc. and the absolute values tends to increase in each iteration when the algorithm diverges. The following section describes solutions to this issue. Fahrmeir [1992] mentions that the correction step can be viewed as a single Fisher Scoring step. This motivates:

1. To take multiple steps if $\boldsymbol{a}_{t|t}$ is far from $\boldsymbol{a}_{t|t-1}$.
2. Introduce a learning rate.

Simulated data sets show that the learning rate solves the issues with divergence. Let $1 \geq \zeta_0 > 0$ denote the learning rate and ϵ_{NR} denote the tolerance for convergence in the correction step. Then set $\boldsymbol{a} = \boldsymbol{a}_{t|t-1}$ and compute:

Algorithm 2 Extended Kalman filter (EKF).

Input:

$\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{X}_1, \dots, \mathbf{X}_d, \boldsymbol{y}_1, \dots, \boldsymbol{y}_d, R_1, \dots, R_d$

Set $\boldsymbol{a}_{0|0} = \boldsymbol{a}_0$ and $\mathbf{V}_{0|0} = \mathbf{Q}_0$

for $t = 1, 2, \dots, d$ **do**

  procedure PREDICTION STEP

    $\boldsymbol{a}_{t|t-1} = \mathbf{F}\boldsymbol{a}_{t-1|t-1}$

    $\mathbf{V}_{t|t-1} = \mathbf{F}\mathbf{V}_{t-1|t-1}\mathbf{F}^\top + \mathbf{R}\mathbf{Q}\mathbf{R}^\top$

  procedure CORRECTION STEP

    Compute the score vector and the information matrix and set

    Let $\boldsymbol{a} = \boldsymbol{a}_{t|t-1}$

    $\boldsymbol{u}_t(\boldsymbol{a}) = \sum_{(i,j)\in R_t} \boldsymbol{u}_{ijt}(\boldsymbol{a}), \quad \boldsymbol{u}_{ijt}(\boldsymbol{a}) = \boldsymbol{x}_{ijt} \frac{\partial h(\eta)/\partial \eta}{H_{kkt}(\boldsymbol{a})} \left(y_{ijt} - h(\eta)\right)\Big|_{\eta = \boldsymbol{x}_{ijt}^\top \boldsymbol{a}}$

    $\mathbf{U}_t(\boldsymbol{a}) = \sum_{(i,j)\in R_t} \mathbf{U}_{ijt}(\boldsymbol{a}), \quad \mathbf{U}_{ijt}(\boldsymbol{a}) = \boldsymbol{x}_{ijt}\boldsymbol{x}_{ijt}^\top \frac{(\partial h(\eta)/\partial \eta)^2}{H_{kkt}(\boldsymbol{a})}\Big|_{\eta = \boldsymbol{x}_{ijt}^\top \boldsymbol{a}}$

    $\mathbf{V}_{t|t} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a})\right)^{-1}$

    $\boldsymbol{a}_{t|t} = \boldsymbol{a}_{t|t-1} + \mathbf{V}_{t|t}\boldsymbol{u}_t(\boldsymbol{a})$

$$
\begin{aligned}
&\mathbf{V}_{t|t} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a})\right)^{-1} \\
&\boldsymbol{a}_{t|t} = \mathbf{V}_{t|t}\left(\mathbf{U}_t(\boldsymbol{a})\boldsymbol{a} + \mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} + \zeta_0 \boldsymbol{u}_t(\boldsymbol{a})\right) \\
&\text{if } \|\boldsymbol{a}_{t|t} - \boldsymbol{a}\|/(\|\boldsymbol{a}\| + \delta) < \epsilon_{\text{NR}} \text{ then exit} \\
&\text{else set } \boldsymbol{a} = \boldsymbol{a}_{t|t} \text{ and repeat}
\end{aligned}
$$

where δ is small like 10^{-9}. The arguments for the above formulas are covered later in the global mode approximation section. Selecting $\zeta_0 < 1$ in case of divergence can solve the non-convergence issue. Thus, the following procedure is used if the algorithm fails with initial learning rate ζ_0: try a learning of ζ_0 for given $0 < \zeta_0 \leq 1$ and define $0 < \zeta < 1$. If that fails then try a rate of $\zeta_0\zeta^1$. If that fails then try a rate of $\zeta_0\zeta^2$ etc. The process is stopped when we succeed to fit the model or we fail to estimate the model with a learning rate of $\zeta_0\zeta^b$ for a given integer b.

While Fahrmeir [1992] does not observe improvements with multiple iterations, we find improvements in terms of out-of-sample prediction (for example by setting $\epsilon_{\text{NR}} = 10^{-2}$ or lower) with a moderate or large amount of observations.

The value of ζ_0 and ϵ_{NR} are set by respectively the arguments `LR` and `NR_eps` to `ddhazard_control`. By default, `LR = 1` and `NR_eps = NULL` which yields a learning rate of 1 and a single Fischer scoring step. These arguments can be altered by setting e.g. `control = ddhazard_control(LR = 0.75, NR_eps = 0.00001)` for a learning rate of 0.75 and a threshold in the Fisher Scoring of 10^{-5}.

In addition, a minor term is added covariance matrix to reduce the influence of extreme values. Thus, the score and information matrix are computed with:

$$
\begin{aligned}
\boldsymbol{u}_t(\boldsymbol{\alpha}_t) &= \sum_{(i,j)\in R_t} \boldsymbol{u}_{ijt}(\boldsymbol{\alpha}_t), \quad \boldsymbol{u}_{ijt}(\boldsymbol{\alpha}_t) = \boldsymbol{x}_{ijt} \frac{\partial h(\eta)/\partial \eta}{H_{kkt}(\boldsymbol{\alpha}_t) + \xi} \left(y_{ijt} - h(\eta)\right)\Big|_{\eta = \boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha}_t} \\
\mathbf{U}_t(\boldsymbol{\alpha}_t) &= \sum_{(i,j)\in R_t} \mathbf{U}_{ijt}(\boldsymbol{\alpha}_t), \quad \mathbf{U}_{ijt}(\boldsymbol{\alpha}_t) = \boldsymbol{x}_{ijt}\boldsymbol{x}_{ijt}^\top \frac{\left(\partial h(\eta)/\partial \eta\right)^2}{H_{kkt}(\boldsymbol{\alpha}_t) + \xi}\Big|_{\eta = \boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha}_t}
\end{aligned}
$$

where $\xi > 0$ is a small number. The default can be changed with the `denom_term` argument of `ddhazard_control`. The approach is similar to how the `glmnet` package handles close to boundary estimates [see Friedman et al., 2010]. Algorithm 3 shows the new E-step.

Algorithm 3 EKF with extra correction steps, learning rate, and hyperparameter ξ replacing lines 8-11 of Algorithm 2.

Input:

Threshold ϵ_{NR}, learning rate ζ_0 and small numbers δ and ξ

1: $\boldsymbol{a}_{t|t} = \boldsymbol{a}_{t|t-1}$
2: **repeat**
3: $\quad \boldsymbol{a} = \boldsymbol{a}_{t|t}$
4: $\quad \boldsymbol{u}_t(\boldsymbol{a}) = \sum_{(i,j)\in R_t} \boldsymbol{u}_{ijt}(\boldsymbol{a}), \quad \boldsymbol{u}_{ijt}(\boldsymbol{a}) = \boldsymbol{x}_{ijt} \frac{\partial h(\eta)/\partial \eta}{H_{kkt}(\boldsymbol{a})+\xi} \left(y_{ijt} - h(\eta)\right)\Big|_{\eta=\boldsymbol{x}_{ijt}^\top \boldsymbol{a}}$
5: $\quad \mathbf{U}_t(\boldsymbol{a}) = \sum_{(i,j)\in R_t} \mathbf{U}_{ijt}(\boldsymbol{a}), \quad \mathbf{U}_{ijt}(\boldsymbol{a}) = \boldsymbol{x}_{ijt}\boldsymbol{x}_{ijt}^\top \frac{(\partial h(\eta)/\partial \eta)^2}{H_{kkt}(\boldsymbol{a})+\xi}\Big|_{\eta=\boldsymbol{x}_{ijt}^\top \boldsymbol{a}}$
6: $\quad \mathbf{V}_{t|t} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a})\right)^{-1}$
7: $\quad \boldsymbol{a}_{t|t} = \mathbf{V}_{t|t}\left(\mathbf{U}_t(\boldsymbol{a})\boldsymbol{a} + \mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} + \zeta_0 \boldsymbol{u}_t(\boldsymbol{a})\right)$
8: **until** $\left\|\boldsymbol{a}_{t|t} - \boldsymbol{a}\right\| / (\|\boldsymbol{a}\| + \delta) < \epsilon_{\text{NR}}$

4 Unscented Kalman Filter

The UKF selects state vectors called *sigma point* with given *sigma weigths* chosen to match the moments of observation equation. The idea is similar to a Monte Carlo methods for state space models but where the state vectors are chosen deterministically rather than randomly drawn.

The motivation to use the UKF in place of the EKF is that we avoid the linerization error in the EKF. Julier and Uhlmann [1997] introduce a UKF that approximate the first two moments and up to fourth moment in certain settings. Julier and Uhlmann [2004] further develop the UKF and extends it to what is later called *the Scaled Unscented Transformation*. We will cover the the Scaled Unscented Transformation with the parametrizion from Wan and Van Der Merwe [2000] and formulas from Menegaz [2016].

One of the reasons the UKF has received a lot of attention (especially in engineering) is for settings where the observation equation is complicated since the UKF does not require computation of the Jacobian matrix. However, deriving the Jacobian matrix for the models in this package is not difficult.

4.1 The usual UKF formulation

We start by introducing a common notation used in the UKF literature. For two random vectors $\boldsymbol{v}_t$ and $\boldsymbol{b}_t$, let:

$$\mathbf{P}_{\boldsymbol{v}_t,\boldsymbol{b}_t} = \text{Cov}\left(\boldsymbol{v}_t, \boldsymbol{b}_t | \, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\right)$$

Notice that $\mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{\alpha}_t} = \mathbf{V}_{t|t}$. The UKF start with the prediction step. As pointed out in Julier and Uhlmann [2004] and Menegaz [2016], the regular Kalman filter prediction step can be used when the state equation is a linear Gaussian model. Thus, the prediction step is:

$$\begin{aligned} \boldsymbol{a}_{t|t-1} &= \mathbf{F}\boldsymbol{a}_{t-1|t-1}, \\ \mathbf{V}_{t|t-1} &= \mathbf{F}\mathbf{V}_{t-1|t-1}\mathbf{F}^\top + \mathbf{R}\mathbf{Q}\mathbf{R}^\top \end{aligned}$$

This is exact given the previous estimates $\boldsymbol{a}_{t-1|t-1}$ and $\mathbf{V}_{t-1|t-1}$ and computationally less demanding. Then we select $2q+1$ so-called *sigma points* (where q is the dimension of the state equation) denoted by $\widehat{\boldsymbol{a}}_0, \widehat{\boldsymbol{a}}_1, \ldots, \widehat{\boldsymbol{a}}_{2q+1}$ according to:

$$\begin{aligned} \widehat{\boldsymbol{a}}_0 &= \boldsymbol{a}_{t|t-1} \\ \widehat{\boldsymbol{a}}_j &= \boldsymbol{a}_{t|t-1} + \sqrt{q+\lambda}\left(\sqrt{\mathbf{V}_{t|t-1}}\right)_j \qquad j = 1, 2, \ldots, q \\ \widehat{\boldsymbol{a}}_{j+q} &= \boldsymbol{a}_{t|t-1} - \sqrt{q+\lambda}\left(\sqrt{\mathbf{V}_{t|t-1}}\right)_j \end{aligned}$$

where $\left(\sqrt{\mathbf{V}_{t|t-1}}\right)_j$ is the j'th column of the lower triangular matrix of the Cholesky decomposition of $\mathbf{V}_{t|t-1}$. We assign the following weights to each sigma point (we will cover selection of the hyperparameters α, β

and κ shortly):

$$
\begin{aligned}
W_0^{[m]} &= \frac{\lambda}{q+\lambda} \\
W_0^{[c]} &= \frac{\lambda}{q+\lambda} + 1 - \alpha^2 + \beta \\
W_0^{[cc]} &= \frac{\lambda}{q+\lambda} + 1 - \alpha \\
W_j^{[m]} &= W_j^{[c]} = \frac{1}{2(q+\lambda)}, \qquad j = 1, \ldots, 2q \\
\lambda &= \alpha^2(q+\kappa) - q
\end{aligned}
$$

Next, we proceed to the correction step. We start by defining the following intermediates:

$$
\begin{aligned}
\widehat{\boldsymbol{y}}_j &= \boldsymbol{z}_t\left(\widehat{\boldsymbol{a}}_j\right), \qquad j = 0, 1, \ldots, 2q \\
\widehat{\mathbf{Y}} &= (\widehat{\boldsymbol{y}}_0, \ldots, \widehat{\boldsymbol{y}}_{2q}) \\
\overline{\boldsymbol{y}} &= \sum_{j=0}^{2q} W_j^{[m]} \boldsymbol{y}_j, \qquad \Delta\widehat{\mathbf{Y}} = \widehat{\mathbf{Y}} - \overline{\boldsymbol{y}}\mathbf{1}^\top, \qquad \widehat{\mathbf{H}} = \sum_{j=0}^{2q} W_j^{[c]} \mathbf{H}_t(\widehat{\boldsymbol{a}}_j) \\
\Delta\widehat{\mathbf{A}} &= (\widehat{\boldsymbol{a}}_0, \ldots, \widehat{\boldsymbol{a}}_{2q}) - \boldsymbol{a}_{t|t-1}\mathbf{1}^\top \\
\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t} &= \sum_{j=0}^{2q} W_j^{[c]} \left((\widehat{\boldsymbol{y}}_j - \overline{\boldsymbol{y}})(\widehat{\boldsymbol{y}}_j - \overline{\boldsymbol{y}})^\top + \widehat{\mathbf{H}} \right) = \Delta\widehat{\mathbf{Y}} \mathrm{diag}\Big(\boldsymbol{W}^{(c)}\Big) \Delta\widehat{\mathbf{Y}}^\top + \widehat{\mathbf{H}} \\
\mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t} &= \sum_{j=0}^{2q} W_j^{[cc]} (\widehat{\boldsymbol{a}}_j - \boldsymbol{a}_{t|t-1})(\widehat{\boldsymbol{y}}_j - \overline{\boldsymbol{y}})^\top = \Delta\widehat{\mathbf{A}} \mathrm{diag}\Big(\boldsymbol{W}^{(cc)}\Big) \Delta\widehat{\mathbf{Y}}^\top
\end{aligned}
$$

Then the correction step is:

$$
\begin{aligned}
\boldsymbol{a}_{t|t} &= \boldsymbol{a}_{t|t-1} + \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t} \mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1} (\boldsymbol{y}_t - \overline{\boldsymbol{y}}) \\
\mathbf{V}_{t|t} &= \mathbf{V}_{t|t-1} - \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t} \mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1} \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}^\top
\end{aligned}
$$

4.2 Re-writting

The above formulation has the drawback that we have to invert $\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}$ which is in-feasible when the number of observations is large. We can re-write the correction step above by using the Woodbury matrix identity to get an algorithm $O(n_t)$ instead of $O(n_t^3)$ where $n_t = |R_t|$ is the number of elements of the risk set. The correction step can be computed as:

$$
\begin{aligned}
\tilde{\boldsymbol{y}} &= \Delta\widehat{\mathbf{Y}}^\top \widehat{\mathbf{H}}^{-1} (\boldsymbol{y}_t - \overline{\boldsymbol{y}}) \\
\mathbf{G} &= \Delta\widehat{\mathbf{Y}}^\top \widehat{\mathbf{H}}^{-1} \Delta\widehat{\mathbf{Y}} \\
\boldsymbol{c} &= \tilde{\boldsymbol{y}} - \mathbf{G} \left(\mathrm{diag}\Big(\boldsymbol{W}^{(c)}\Big)^{-1} + \mathbf{G} \right)^{-1} \tilde{\boldsymbol{y}} \\
\mathbf{L} &= \mathbf{G} - \mathbf{G} \left(\mathrm{diag}\Big(\boldsymbol{W}^{(c)}\Big)^{-1} + \mathbf{G} \right)^{-1} \mathbf{G} \\
\boldsymbol{a}_{t|t} &= \boldsymbol{a}_{t|t-1} + \Delta\widehat{\mathbf{A}} \mathrm{diag}\Big(\boldsymbol{W}^{(cc)}\Big) \boldsymbol{c} \\
\mathbf{V}_{t|t} &= \mathbf{V}_{t|t-1} - \Delta\widehat{\mathbf{A}} \mathrm{diag}\Big(\boldsymbol{W}^{(cc)}\Big) \mathbf{L} \mathrm{diag}\Big(\boldsymbol{W}^{(cc)}\Big) \Delta\widehat{\mathbf{A}}^\top
\end{aligned}
$$

where $\tilde{\boldsymbol{y}}$, $\mathbf{G}$, $\mathbf{L}$ and $\boldsymbol{c}$ are intermediates. The above algorithm is $O(n_t)$ since $\widehat{\mathbf{H}}$ is a diagonal matrix. Algorithm 4 shows the UKF method.

Algorithm 4 The unscented Kalman filter (UKF) where $\mathbf{V}_{t|t-1}^{1/2}$ denotes the "square root" matrix of $\mathbf{V}_{t|t-1}$ and $\left(\mathbf{V}_{t|t-1}^{1/2}\right)_j$ denotes the j'th column of the "square root" matrix. diag$(\cdot)$ gives a diagonal matrix with the entries of the argument vector in the diagonal. q is the dimension of the state vector. $\boldsymbol{W}^{(\cdot)}$ is the vector with elements $W_0^{[\cdot]}, W_1^{[\cdot]}, \ldots, W_{2q}^{[\cdot]}$.

Input:

$\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{X}_1, \ldots, \mathbf{X}_d, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_d, R_1, \ldots, R_d$

Hyperparameters α, β, and κ

Set $\boldsymbol{a}_{0|0} = \boldsymbol{a}_0$ and $\mathbf{V}_{0|0} = \mathbf{Q}_0$

Compute *sigma weights* with $\lambda = \alpha^2(q + \kappa) - q$

$W_0^{[m]} = \frac{\lambda}{q+\lambda}$

$W_0^{[c]} = \frac{\lambda}{q+\lambda} + 1 - \alpha^2 + \beta$

$W_0^{[cc]} = \frac{\lambda}{q+\lambda} + 1 - \alpha$

$W_j^{[m]} = W_j^{[c]} = \frac{1}{2(q+\lambda)}, \qquad j = 1, \ldots, 2q$

for $t = 1, 2, \ldots, d$ **do**

  procedure PREDICTION STEP

    $\boldsymbol{a}_{t|t-1} = \mathbf{F}\boldsymbol{a}_{t-1|t-1}$

    $\mathbf{V}_{t|t-1} = \mathbf{F}\mathbf{V}_{t-1|t-1}\mathbf{F}^\top + \mathbf{R}\mathbf{Q}\mathbf{R}^\top$

  procedure CORRECTION STEP

    Compute *sigma points*

    $\widehat{\boldsymbol{a}}_0 = \boldsymbol{a}_{t|t-1}$

    $\widehat{\boldsymbol{a}}_j = \boldsymbol{a}_{t|t-1} + \sqrt{q+\lambda}\left(\mathbf{V}_{t|t-1}^{1/2}\right)_j \qquad j = 1, 2, \ldots, q$

    $\widehat{\boldsymbol{a}}_{j+q} = \boldsymbol{a}_{t|t-1} - \sqrt{q+\lambda}\left(\mathbf{V}_{t|t-1}^{1/2}\right)_j$

    Compute intermediates

    $\widehat{\boldsymbol{y}}_j = \boldsymbol{z}_t(\widehat{\boldsymbol{a}}_j) \qquad j = 0, 1, \ldots, 2q$

    $\widehat{\mathbf{Y}} = (\widehat{\boldsymbol{y}}_0, \ldots, \widehat{\boldsymbol{y}}_{2q})$

    $\overline{\boldsymbol{y}} = \sum_{j=0}^{2q} W_j^{[m]} \boldsymbol{y}_j$

    $\Delta\widehat{\mathbf{Y}} = \widehat{\mathbf{Y}} - \overline{\boldsymbol{y}}\mathbf{1}^\top$

    $\widehat{\mathbf{H}} = \xi\mathbf{I} + \sum_{j=0}^{2q} W_j^{[c]} \mathbf{H}_t(\widehat{\boldsymbol{a}}_j)$

    $\Delta\widehat{\mathbf{A}} = (\widehat{\boldsymbol{a}}_0, \ldots, \widehat{\boldsymbol{a}}_{2q}) - \boldsymbol{a}_{t|t-1}\mathbf{1}^\top$

    $\widetilde{\boldsymbol{y}} = \Delta\widehat{\mathbf{Y}}^\top \widehat{\mathbf{H}}^{-1}(\boldsymbol{y}_t - \overline{\boldsymbol{y}})$

    $\mathbf{G} = \Delta\widehat{\mathbf{Y}}^\top \widehat{\mathbf{H}}^{-1} \Delta\widehat{\mathbf{Y}}$

    $\boldsymbol{c} = \widetilde{\boldsymbol{y}} - \mathbf{G}\left(\text{diag}\left(\boldsymbol{W}^{(c)}\right)^{-1} + \mathbf{G}\right)^{-1} \widetilde{\boldsymbol{y}}$

    $\mathbf{L} = \mathbf{G} - \mathbf{G}\left(\text{diag}\left(\boldsymbol{W}^{(c)}\right)^{-1} + \mathbf{G}\right)^{-1} \mathbf{G}$

    Compute updates

    $\boldsymbol{a}_{t|t} = \boldsymbol{a}_{t|t-1} + \Delta\widehat{\mathbf{A}}\text{diag}\left(\boldsymbol{W}^{(cc)}\right)\boldsymbol{c}$

    $\mathbf{V}_{t|t} = \mathbf{V}_{t|t-1} - \Delta\widehat{\mathbf{A}}\text{diag}\left(\boldsymbol{W}^{(cc)}\right)\mathbf{L}\text{diag}\left(\boldsymbol{W}^{(cc)}\right)\Delta\widehat{\mathbf{A}}^\top$

4.3 The Square-root Unscented Kalman filter

Another idea could be to try the Square-root Unscented Kalman filter suggested in der Merwe and Wan [2001]. The idea is to use a QR decompositions and Cholesky updates to get a more stable method. While der Merwe and Wan [2001] shows that this scales equally well in the dimension of the state vector we show below that it does not scale well with the number of individuals at risk, n_t. The prediction step is as before.

Next, let

$$\begin{aligned}
&\text{qr}\left(\mathbf{B}\right) = \mathbf{C}, \qquad \mathbf{C} \text{ is the Choleksy decomposition of } \mathbf{B}\mathbf{B}^\top = \mathbf{C}^\top\mathbf{C} \\
&\text{cholupdate}\left(\mathbf{C}, \boldsymbol{c}, v\right) = \widetilde{\mathbf{C}}, \qquad \widetilde{\mathbf{C}} \text{ is the updated Cholesky factor } \widetilde{\mathbf{C}}^\top\widetilde{\mathbf{C}} = \mathbf{C}^\top\mathbf{C} + v\boldsymbol{c}\boldsymbol{c}^\top
\end{aligned}$$

Whether we make an update or a downdate depends on the sign of v. The correction step can done as follows:

$$\begin{aligned}
\mathbf{C} &= \text{qr}\left(\left[\sqrt{W_1^{[c]}}\left(\widehat{\boldsymbol{y}}_1 - \overline{\boldsymbol{y}}\right) \quad \sqrt{W_2^{[c]}}\left(\widehat{\boldsymbol{y}}_2 - \overline{\boldsymbol{y}}\right) \quad \dots \quad \sqrt{W_{2q}^{[c]}}\left(\widehat{\boldsymbol{y}}_{2q} - \overline{\boldsymbol{y}}\right) \quad \sqrt{\widehat{\mathbf{H}}}\right]\right) \\
\mathbf{C} &\leftarrow \text{cholupdate}\left(\mathbf{C}, \widehat{\boldsymbol{y}}_0 - \overline{\boldsymbol{y}}, \text{sign}\left(W_0^{[c]}\right)\sqrt{|W_0^{[c]}|}\right) \\
\mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t} &= \Delta\widehat{\mathbf{A}}\text{diag}\Big(\boldsymbol{W}^{(cc)}\Big)\Delta\widehat{\mathbf{Y}}^\top \\
\mathbf{K} &= \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{C}^{-1}\left(\mathbf{C}^{-1}\right)^\top \\
\boldsymbol{a}_{t|t} &= \boldsymbol{a}_{t|t-1} + \mathbf{K}\left(\boldsymbol{y}_t - \overline{\boldsymbol{y}}\right) \\
\mathbf{V}_{t|t} &= \mathbf{V}_{t|t-1} - \mathbf{K}\mathbf{C}^\top\mathbf{C}\mathbf{K}^\top
\end{aligned}$$

where we use the left arrow, $\leftarrow$, to indicate an update and all definitions of matrices and vectors are as in the beginning of this section. We will show that this is equivalent to the first method. First, assume that the first weight is positive, $W_0^{[c]} > 0$, such that we do not need the Cholesky update. Then:

$$\begin{aligned}
&\mathbf{C} = \text{qr}\left(\left[\sqrt{W_0^{[c]}}\left(\widehat{\boldsymbol{y}}_1 - \overline{\boldsymbol{y}}\right) \quad \sqrt{W_2^{[c]}}\left(\widehat{\boldsymbol{y}}_2 - \overline{\boldsymbol{y}}\right) \quad \dots \quad \sqrt{W_{2q}^{[c]}}\left(\widehat{\boldsymbol{y}}_{2q} - \overline{\boldsymbol{y}}\right) \quad \sqrt{\widehat{\mathbf{H}}}\right]\right) \\
&\Rightarrow \mathbf{C}^\top\mathbf{C} = \left[\sqrt{W_0^{[c]}}\left(\widehat{\boldsymbol{y}}_1 - \overline{\boldsymbol{y}}\right) \quad \sqrt{W_2^{[c]}}\left(\widehat{\boldsymbol{y}}_2 - \overline{\boldsymbol{y}}\right) \quad \dots \quad \sqrt{W_{2q}^{[c]}}\left(\widehat{\boldsymbol{y}}_{2q} - \overline{\boldsymbol{y}}\right) \quad \sqrt{\widehat{\mathbf{H}}}\right] \begin{bmatrix} \sqrt{W_0^{[c]}}\left(\widehat{\boldsymbol{y}}_1 - \overline{\boldsymbol{y}}\right)^\top \\ \sqrt{W_2^{[c]}}\left(\widehat{\boldsymbol{y}}_2 - \overline{\boldsymbol{y}}\right)^\top \\ \vdots \\ \sqrt{W_{2q}^{[c]}}\left(\widehat{\boldsymbol{y}}_{2q} - \overline{\boldsymbol{y}}\right)^\top \\ \sqrt{\widehat{\mathbf{H}}} \end{bmatrix} \\
&\qquad = \Delta\widehat{\mathbf{Y}}\text{diag}\Big(\boldsymbol{W}^{(c)}\Big)\Delta\widehat{\mathbf{Y}}^\top + \widehat{\mathbf{H}} = \mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t} \\
&\mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t} = \Delta\widehat{\mathbf{A}}\text{diag}\Big(\boldsymbol{W}^{(cc)}\Big)\Delta\widehat{\mathbf{Y}}^\top \\
&\mathbf{K} = \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{C}^{-1}\left(\mathbf{C}^{-1}\right)^\top = \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\left(\mathbf{C}^\top\mathbf{C}\right)^{-1} = \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1} \\
&\boldsymbol{a}_{t|t} = \boldsymbol{a}_{t|t-1} + \mathbf{K}\left(\boldsymbol{y}_t - \overline{\boldsymbol{y}}\right) = \boldsymbol{a}_{t|t-1} + \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1}\left(\boldsymbol{y}_t - \overline{\boldsymbol{y}}\right) \\
&\mathbf{V}_{t|t} = \mathbf{V}_{t|t-1} - \mathbf{K}\mathbf{C}^\top\mathbf{C}\mathbf{K}^\top = \mathbf{V}_{t|t-1} - \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1}\mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}^\top
\end{aligned}$$

Next, we look at the computational cost of the case where $W_0^{[c]} > 0$. Since $\mathbf{C} \in \mathbb{R}^{(2q+1+n_t)\times n_t}$, the cost of finding the decomposition of $\mathbf{C}$ is $O((2q+1+n_t)n_t^2)$ as stated in der Merwe and Wan [2001]. Consequently, we end with a $O(n_t^3)$ cost in every iteration of the filter making this method of little use when n_t is large. I have some ideas how we might change the method. Lets continue with the assumption that the first weight is positive. Then one idea is to compute:

$$\begin{aligned}
\mathbf{C} &= \text{qr}\left(\left[\sqrt{W_0^{[c]}}\left(\widehat{\boldsymbol{y}}_1 - \overline{\boldsymbol{y}}\right) \quad \sqrt{W_2^{[c]}}\left(\widehat{\boldsymbol{y}}_2 - \overline{\boldsymbol{y}}\right) \quad \dots \quad \sqrt{W_{2q}^{[c]}}\left(\widehat{\boldsymbol{y}}_{2q} - \overline{\boldsymbol{y}}\right)\right]\right) \\
\mathbf{C} &\leftarrow \text{cholupdate}\left(\mathbf{C}, \widehat{\mathbf{H}}, 1\right)
\end{aligned}$$

In this case, we first find $\mathbf{C} \in \mathbb{R}^{(2q+1)\times n_t}$ and then make a rank-n_t update. The rest of the computations are in-expensive relative to the number of observations, n_t, since we have reduced the dimension of $\mathbf{C} \in \mathbb{R}^{(2q+1)\times n_t}$. In general, the key is that we want a Cholesky decomposition (or another decomposition) of $\Delta\widehat{\mathbf{Y}}\text{diag}\big(\boldsymbol{W}^{[c]}\big)\Delta\widehat{\mathbf{Y}}^\top + \widehat{\mathbf{H}}$ which is easy to compute and ease the rest of the computations and storage requirements. None of the above is implemented in the package.

4.4 Extreme values

As with the EKF, a minor addition is made to the covariance matrix of the observational equation such that we replace $\widehat{\mathbf{H}}$ by:

$$\widetilde{\widehat{\mathbf{H}}} = \widehat{\mathbf{H}} + \xi \mathbf{I}$$

4.5 Selecting hyperparameters

We still need to select the hyperparameters κ, α and β. We will cover these in the given order. κ is usually set to $\kappa = 0$ or $\kappa = 3 - m$. Julier and Uhlmann [1997] argue that the latter is a "*useful heuristic*" when the state equation is Gaussian and $\alpha = 1$.

The default in this package is $\kappa = q(1+\alpha^2(0.1-1))/(\alpha^2(1-0.1))$ and can be altered by setting the `kappa` argument in the `ddhazard_control` call which is passed to the `control` argument of `ddhazard`. For example, `control = ddhazard_control(kappa = 1, ...)` yields $\kappa = 1$. The default makes $W_0^{[m]} = 0.1$ such that all weights are positive. This ensures that $\mathbf{V}_{t|t-1}$ and $\mathbf{P}_{\boldsymbol{y}_t, \boldsymbol{y}_t}$ are positive semi-definite. This follows since both are sum of outer products with positive weights and as $\widehat{\mathbf{H}}$ is a diagonal matrix with positive entries.

$0 < \alpha \leq 1$ controls the spread of the sigma points. Notice that $\lambda + q \to 0^+$, $W_0^{[c]}, W_0^{[m]} \to -\infty$ and $W_j^{[c]}, W_j^{[m]} \to \infty$ $(j > 0)$ as $\alpha \to 0^+$. Thus, the lower the value of α, the lower the spread but the higher the absolute weights. It is generally suggested to choose α small [see Gustafsson and Hendeby, 2012, Julier and Uhlmann, 2004]. However, initial simulation studies showed that $\alpha = 1$ yields the smallest mean square error of estimated coefficients. Thus, this is the default. The parameter can be set with the `alpha` argument of `ddhazard_control`.

Lastly, β is a correction term to match the fourth-order term in the Taylor series expansion of the covariance of the observational equation. Julier and Uhlmann [2004] show in the appendix that the optimal value with a Gaussian state equation is $\beta = 2$. Though, initial simulation showed that $\beta = 0$ yielded the best results and is therefore the default. It can be set with the `beta` argument of `ddhazard_control`.

4.6 Selecting starting values

Experience with different data sets and the UKF shows that the method is sensitive to the starting values of $\mathbf{Q}$ and $\mathbf{Q}_0$ (where the latter may be fixed). The reason for divergence can be illustrated by the effect of $\mathbf{Q}_0$. We start the filter by setting $\mathbf{V}_{0|0} = \mathbf{Q}_0$. Say that we set $\mathbf{Q}_0 = v\mathbf{I}_m$ and $\boldsymbol{a}_0 = \mathbf{0}$. Then the j'th column of the Cholesky decomposition $\mathbf{V}_{0|0}$ is a vector with $\sqrt{v}$ in the j'th entry and zero in the rest of the entries. Suppose that we set v large. Then the linear predictors computed with the $b \leq q + 1$ sigma point is $\sqrt{q + \lambda}\sqrt{v}x_{ij1b}$ where x_{ij1b} is the b'th entry of individual i's j'th covariate vector at time 1. This can be potentially quite large in absolute terms if x_{ij1b} is moderately different from zero. This seems to lead to divergence in some cases where all the predicted values becomes either zero or one with variance close to zero. The later is an issue as we divide by the weighted average of the variances in the correction step.

$\mathbf{Q}$ has a similar effect although it is harder to illustrate with a small example as it occurs in an intermediate computations in the UKF. Based on experience, it seems that $\mathbf{Q}_0$ should be a diagonal matrix with *"somewhat"* large values and $\mathbf{Q}$ should be a diagonal matrix with small values. Though, what is *"somewhat"* large and what is small dependent on the data set.

5 Sequential approximation of the posterior mode (SMA)

Another idea is do sequential rank-one approximations of the posterior modes. This section cover the details of this method, the implementation and the pros and cons. Say we are at a given iteration t of the filtering in the E-step. First, we carry out the prediction step with the closed form solution:

$$\begin{aligned} \boldsymbol{a}_{t|t-1} &= \mathbf{F}\boldsymbol{a}_{t-1|t-1}, \\ \mathbf{V}_{t|t-1} &= \mathbf{F}\mathbf{V}_{t-1|t-1}\mathbf{F}^\top + \mathbf{R}\mathbf{Q}\mathbf{R}^\top \end{aligned}$$

Next, we replace the correction step with finding the mode that minimize the negative log-likelihood:

$$\arg\min_{\boldsymbol{\alpha}} \ -\log P\left(\boldsymbol{\alpha}|\, \boldsymbol{a}_{t|t-1}, \mathbf{V}_{t|t-1}\right) - \sum_{(i,j)\in R_t} \log P\left(y_{ijt}|\, \boldsymbol{\alpha}\right)$$

However, we replace this problem with a series of update with one for each of the $n_t = |R_t|$ observations in interval t. First, we set:

$$\boldsymbol{a}_{t|t}^{(0)} = \boldsymbol{a}_{t|t-1}, \qquad \mathbf{V}_{t|t}^{(0)} = \mathbf{V}_{t|t-1}$$

Then for $k = 1, 2, \dots, n_t$ we:

1. Set (i, j) to the k'th element of the risk set R_t
2. Update
$$\boldsymbol{a}_{t|t}^{(k)} = \arg\min_{\boldsymbol{\alpha}} \ -\log P\left(\boldsymbol{\alpha}|\, \boldsymbol{a}_{t|t}^{(k-1)}, \mathbf{V}_{t|t}^{(k-1)}\right) - \log P\left(y_{ijt}|\, \boldsymbol{\alpha}\right)$$
3. Update the covariance matrix by computing the inverse of the Hessian at $\boldsymbol{a}_{t|t}^{(k)}$:
$$\mathbf{V}_{t|t}^{(k)} = \left(\left(\mathbf{V}_{t|t}^{(k-1)}\right)^{-1} + \left.\frac{\partial \log P\left(y_{ijt}|\, \boldsymbol{\alpha}\right)}{\partial \boldsymbol{\alpha} \partial \boldsymbol{\alpha}^\top}\right|_{\boldsymbol{\alpha}=\boldsymbol{a}_{t|t}^{(k)}}\right)^{-1}$$

Step 2 is a one dimensional problem of finding the constant $v \in \mathbb{R}$ that minimize:

$$\begin{aligned}
v &= \arg\min_b b^2 \frac{1}{2} \frac{1}{\boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}} - b \frac{\boldsymbol{x}_{ijt}^\top \boldsymbol{a}_{t|t}^{(k-1)}}{\boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}} - \left(y_{ijt} \log h(b) + (1 - y_{ijt}) \log(1 - h(b))\right) \\
&= \arg\min_b b^2 \frac{1}{2} d_1 - b d_1 d_2 - \left(y_{ijt} \log h(b) + (1 - y_{ijt}) \log(1 - h(b))\right) \\
d_1 &= \frac{1}{\boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}}, \qquad d_2 = \boldsymbol{x}_{ijt}^\top \boldsymbol{a}_{t|t}^{(k-1)}
\end{aligned}$$

The update of the state vector given the constant v is done by:

$$\boldsymbol{a}_{t|t}^{(k)} = \boldsymbol{a}_{t|t}^{(k-1)} - (d_1 - v) d_2 \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}$$

Further, step 3. can be re-written to:

$$\mathbf{V}_{t|t}^{(k)} = \left(\left(\mathbf{V}_{t|t}^{(k-1)}\right)^{-1} + \boldsymbol{x}_{ijt} g \boldsymbol{x}_{ijt}^\top\right)^{-1}, \qquad g = -\left.\frac{\log P\left(y_{ijt}|\, \boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha} = b\right)}{\partial b^2}\right|_{b=v}$$

Further, we can apply Woodbury matrix identity (or in this case the less general Sherman–Morrison formula) to avoid the inversions and get:

$$\mathbf{V}_{t|t}^{(k)} = \mathbf{V}_{t|t}^{(k-1)} - \frac{\mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt} g \boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)}}{1 + g \boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}} = \mathbf{V}_{t|t}^{(k-1)} - \frac{\mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt} g \boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)}}{1 + g/d_1}$$

This method is selected by setting `method = "SMA"` in the `ddhazard_control` call.

5.1 Implementation

Finding the constant v in step 2 can be done by the Newton Raphson method to find a unique minimum when:

1. $-\log P\left(y_{ijt}|\, \boldsymbol{\alpha}\right)$ is convex in b.
2. $-\log P\left(y_{ijt}|\, \boldsymbol{\alpha}\right)$ bounded from below.

which is true for the implemented models. Further, the learning rate ζ_0 the decrease factor ζ as in the EKF can be used here to by changing the correction step to:

$$\boldsymbol{a}_{t|t}^{(k)} = \boldsymbol{a}_{t|t}^{(k-1)} - (d_1 - v) d_2 \zeta_0 \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}$$

The Woodbury matrix identity can perform poorly for ill-conditioned matrices. This motivates the following algorithm:

1. Set
$$\boldsymbol{a}_{t|t}^{(0)} = \boldsymbol{a}_{t|t-1}, \qquad \mathbf{V}_{t|t}^{(0)} = \mathbf{V}_{t|t-1}, \qquad \mathbf{L}\mathbf{L}^\top = \left(\mathbf{V}_{t|t-1}\right)^{-1}$$
where $\mathbf{L}$ is the lower triangular matrix from the Cholesky decomposition of $\left(\mathbf{V}_{t|t-1}\right)^{-1}$. For $k = 1, 2, \ldots, n_t$:

2. Perform step 1 and 2 as before to find the constant v and update $\boldsymbol{a}_{t|t}^{(k)}$.

3. Update $\mathbf{L}$ by a rank-one-update of $\boldsymbol{x}_{ijt} g \boldsymbol{x}_{ijt}^\top$ such that $\mathbf{L}\mathbf{L}^\top \leftarrow \mathbf{L}\mathbf{L}^\top + \boldsymbol{x}_{ijt} g \boldsymbol{x}_{ijt}^\top$. We use the left arrow, $\leftarrow$, to indicate that we make an update.

4. Set $\mathbf{V}_{t|t}^{(k)} = \left(\mathbf{L}^{-1}\right)^\top \left(\mathbf{L}^{-1}\right)$.

Step 1 comes at an $O(q^3)$ cost per interval $1, 2, \ldots, d$. This is doable if we do not have too many coefficients. Step 2 can be performed in $O(q^2)$. The current implementation use the Fortran code from the post here `http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=2&t=2646` based on Seeger [2004].

Moreover, we can reduce computations time by storing $\widetilde{\mathbf{L}} = \left(\mathbf{L}^{-1}\right)^\top$ and using that it is a triangular matrix. $\widetilde{\mathbf{L}}$ is also a triangular matrix. Thus, we need to do less operations when doing the matrix multiplications. Another advantage is that the rank-one update yields a positive semi definite matrix $\mathbf{L}\mathbf{L}^\top$. This is true since $\boldsymbol{x}_{ijt}\boldsymbol{x}_{ijt}^\top$ is a vector outer product and as g is positive with distributions from the exponential family. Hence, $\mathbf{V}_{t|t}^{(k)} = \left(\mathbf{L}^{-1}\right)^\top \left(\mathbf{L}^{-1}\right)$ will remains a positive semi definite matrix. The method described here is used if you set the argument `posterior_version = "cholesky"` in the call to `ddhazard_control`. The former method using the Woodbury matrix identity is used if you set `posterior_version = "woodbury"`. Algorithm 5 shows the Woodbury matrix identity version and Algorithm 6 shows the Cholesky decomposition version.

Algorithm 5 Sequential approximation of the posterior mode. Left-arrow, $\leftarrow$, indicates an update instead of an equality.

Input:

$\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{X}_1, \ldots, \mathbf{X}_d, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_d, R_1, \ldots, R_d$

Learning rate ζ_0

Set $\boldsymbol{a}_{0|0} = \boldsymbol{a}_0$ and $\mathbf{V}_{0|0} = \mathbf{Q}_0$

1: **for** $t = 1, 2, \ldots, d$ **do**

2: **procedure** PREDICTION STEP

3: $\boldsymbol{a}_{t|t-1} = \mathbf{F}\boldsymbol{a}_{t-1|t-1}$

4: $\mathbf{V}_{t|t-1} = \mathbf{F}\mathbf{V}_{t-1|t-1}\mathbf{F}^\top + \mathbf{R}\mathbf{Q}\mathbf{R}^\top$

5: **procedure** CORRECTION STEP

Set $\mathbf{V}_{t|t}^{(0)} = \mathbf{V}_{t|t-1}$, $k = 0$ and $\boldsymbol{a}_{t|t}^{(0)} = \boldsymbol{a}_{t|t-1}$

6: **for** $(i, j) \in R_t$ **do**

7: $k \leftarrow k + 1$

8: $d_1 = \frac{1}{\boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}}$

9: $d_2 = \boldsymbol{x}_{ijt}^\top \boldsymbol{a}_{t|t}^{(k-1)}$

10: $v = \arg\min_b b^2 \frac{1}{2} d_1 - b d_1 d_2 - \log P\left(y_{ijt} | \boldsymbol{\alpha}_t^\top \boldsymbol{x}_{ijt} = b\right)$

11: $g = -\left. \frac{\log P\left(y_{ijt} | \boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha} = b\right)}{\partial b^2} \right|_{b=v}$

12: $\boldsymbol{a}_{t|t}^{(k)} = \boldsymbol{a}_{t|t}^{(k-1)} - (d_1 - v) d_2 \zeta_0 \mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt}$

13: $\mathbf{V}_{t|t}^{(k)} = \mathbf{V}_{t|t}^{(k-1)} - \frac{\mathbf{V}_{t|t}^{(k-1)} \boldsymbol{x}_{ijt} g \boldsymbol{x}_{ijt}^\top \mathbf{V}_{t|t}^{(k-1)}}{1 + g/d_1}$

Set $\boldsymbol{a}_{t|t} = \boldsymbol{a}_{t|t}^{(n_t)}$ and $\mathbf{V}_{t|t} = \mathbf{V}_{t|t}^{(n_t)}$

Algorithm 6 Alternative correction step in the procedure at line 5 of Algorithm 5 with a Cholesky decomposition. Left-arrow, $\leftarrow$, indicates an update instead of an equality.

Compute the Cholesky decomposition $\mathbf{L}\mathbf{L}^\top = \left(\mathbf{V}_{t|t-1}\right)^{-1}$ and $\widetilde{\mathbf{L}} = \left(\mathbf{L}^{-1}\right)^\top$

for $(i, j) \in R_t$ **do**

$k \leftarrow k + 1$

$\widetilde{\boldsymbol{x}}_{ijt} = \widetilde{\mathbf{L}}^\top \boldsymbol{x}_{ijt}$

$d_1 = \frac{1}{\widetilde{\boldsymbol{x}}_{ijt}^\top \widetilde{\boldsymbol{x}}_{ijt}}$

$d_2 = \boldsymbol{x}_{ijt}^\top \boldsymbol{a}_{t|t}^{(k-1)}$

$v = \arg\min_b b^2 \frac{1}{2} d_1 - b d_1 d_2 - \log P\left(y_{ijt} | \boldsymbol{\alpha}_t^\top \boldsymbol{x}_{ijt} = b\right)$

$g = -\left. \frac{\log P\left(y_{ijt} | \boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha} = b\right)}{\partial b^2} \right|_{b=v}$

$\boldsymbol{a}_{t|t}^{(k)} = \boldsymbol{a}_{t|t}^{(k-1)} - (d_1 - v) d_2 \zeta_0 \widetilde{\mathbf{L}} \widetilde{\boldsymbol{x}}_{ijt}$

$\mathbf{L}\mathbf{L}^\top \leftarrow \mathbf{L}\mathbf{L}^\top + \boldsymbol{x}_{ijt} g \boldsymbol{x}_{ijt}^\top$

$\widetilde{\mathbf{L}} = \left(\mathbf{L}^{-1}\right)^\top$

Set $\boldsymbol{a}_{t|t} = \boldsymbol{a}_{t|t}^{(n_t)}$ and $\mathbf{V}_{t|t} = \widetilde{\mathbf{L}}\widetilde{\mathbf{L}}^\top$

6 Global approximation of the posterior mode (GMA)

We can directly minimize:

$$\arg\min_{\boldsymbol{\alpha}} \; -\log P\left(\boldsymbol{\alpha} | \, \boldsymbol{a}_{t|t-1}, \mathbf{V}_{t|t-1}\right) - \sum_{(i,j) \in R_t} \log P\left(y_{ijt} | \, \boldsymbol{\alpha}\right)$$

Algorithm 7 Correction step with the global mode approximation by Newton-Raphson.

Set $\boldsymbol{a}^{(0)} = \boldsymbol{a}_{t|t-1}$ and define:

$$c'(\boldsymbol{\alpha}) = \left.\frac{\partial \log P\left(\boldsymbol{y}_t|\boldsymbol{e}'\right)}{\partial \boldsymbol{e}'}\right|_{\boldsymbol{e}'=\mathbf{X}_t\boldsymbol{\alpha}}$$

$$c''(\boldsymbol{\alpha}) = \left.\frac{\partial \log P\left(\boldsymbol{y}_t|\boldsymbol{e}'\right)}{\partial \boldsymbol{e}'\partial(\boldsymbol{e}')^\top}\right|_{\boldsymbol{e}'=\mathbf{X}_t\boldsymbol{\alpha}}$$

1: **repeat**

2: $$\boldsymbol{a}^{(k)} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{X}_t^\top(-c''(\boldsymbol{\alpha}^{(k-1)})\mathbf{X}_t\right)^{-1}\left(\zeta_0\mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} + \zeta_0\mathbf{X}_t^\top c'(\boldsymbol{\alpha}^{(k-1)})\right.$$
$$\left. + \left(\mathbf{X}_t^\top\left(-c''(\boldsymbol{\alpha}^{(k-1)})\right)\mathbf{X}_t + (1-\zeta_0)\mathbf{V}_{t|t-1}^{-1}\right)\boldsymbol{a}^{(k-1)}\right)$$

3: **until** $\left\|\boldsymbol{a}^{(k)} - \boldsymbol{a}^{(k-1)}\right\| / (\left\|\boldsymbol{a}^{(k-1)}\right\| + \delta) < \epsilon$ **or** $k \geq k_{\max}$ **else** set $k \leftarrow k+1$

4: $\mathbf{V}_{t|t} = \left(\widetilde{\mathbf{G}}(\boldsymbol{a}^{(k-1)})\right)^{-1}$

This is equivalent to a L2 penalized Generalized Linear Models (GLM) since we only use models from the exponential family. This can be done with the usual penalized iteratively reweighted least squares. Every iteration can be done in $O\left(n_t q^2 + q^3\right)$. We will go through computations in the following paragraphs. First, we derive the gradient and the Hessian:

$$\tilde{h}(\boldsymbol{\alpha}) = -\log P\left(\boldsymbol{\alpha}|\,\boldsymbol{a}_{t|t-1}, \mathbf{V}_{t|t-1}\right) - \sum_{(i,j)\in R_t}\log P\left(y_{ijt}|\,\boldsymbol{\alpha}\right)$$

$$\begin{aligned}\widetilde{\boldsymbol{g}}(\boldsymbol{\alpha}) = \tilde{h}'(\boldsymbol{\alpha}) &= \mathbf{V}_{t|t-1}^{-1}\left(\boldsymbol{\alpha} - \boldsymbol{a}_{t|t-1}\right) - \sum_{(i,j)\in R_t}\left.\frac{\partial \log P\left(y_{ijt}|\,\boldsymbol{\alpha}'\right)}{\partial \boldsymbol{\alpha}'}\right|_{\boldsymbol{\alpha}'=\boldsymbol{\alpha}} \\ &= \mathbf{V}_{t|t-1}^{-1}\left(\boldsymbol{\alpha} - \boldsymbol{a}_{t|t-1}\right) - \mathbf{X}_t^\top \underbrace{\left.\frac{\partial \log P\left(\boldsymbol{y}_t|\,\boldsymbol{e}'\right)}{\partial \boldsymbol{e}'}\right|_{\boldsymbol{e}'=\mathbf{X}_t\boldsymbol{\alpha}}}_{c'(\boldsymbol{\alpha})}\end{aligned}$$

$$\begin{aligned}\widetilde{\mathbf{G}}(\boldsymbol{\alpha}) = \tilde{h}''(\boldsymbol{\alpha}) &= \mathbf{V}_{t|t-1}^{-1} - \sum_{(i,j)\in R_t}\left.\frac{\partial^2 \log P\left(y_{ijt}|\,\boldsymbol{\alpha}'\right)}{\partial \boldsymbol{\alpha}'\partial\left(\boldsymbol{\alpha}'\right)^\top}\right|_{\boldsymbol{\alpha}'=\boldsymbol{\alpha}} \\ &= \mathbf{V}_{t|t-1}^{-1} - \mathbf{X}_t^\top \underbrace{\left.\frac{\partial \log P\left(\boldsymbol{y}_t|\,\boldsymbol{e}'\right)}{\partial \boldsymbol{e}'\partial\left(\boldsymbol{e}'\right)^\top}\right|_{\boldsymbol{e}'=\mathbf{X}_t\boldsymbol{\alpha}}}_{c''(\boldsymbol{\alpha})}\mathbf{X}_t\end{aligned}$$

Thus, the update equation with a learning rate, ζ_0, is:

$$\begin{aligned}\boldsymbol{a}^{(k)} &= \boldsymbol{a}^{(k-1)} + \zeta_0\left(\widetilde{\mathbf{G}}(\boldsymbol{a}^{(k-1)})\right)^{-1}\left(-\widetilde{\mathbf{g}}(\boldsymbol{a}^{(k-1)})\right) \\ &= \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{X}_t^\top(-c''(\boldsymbol{\alpha}^{(k-1)})\mathbf{X}_t\right)^{-1}\left(\zeta_0\mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} + \zeta_0\mathbf{X}_t^\top c'(\boldsymbol{\alpha}^{(k-1)})\right. \\ &\quad \left. + \left(\mathbf{X}_t^\top\left(-c''(\boldsymbol{\alpha}^{(k-1)})\right)\mathbf{X}_t + (1-\zeta_0)\mathbf{V}_{t|t-1}^{-1}\right)\boldsymbol{a}^{(k-1)}\right)\end{aligned}$$

Algorithm 7 shows the final algorithm for the correction step with the GMA.

This method is selected by passing `method = "GMA"` in the `ddhazard_control` call. You can change $k_{\max}$ and ϵ by respectively with the arguments `GMA_max_rep` and `GMA_NR_eps` of `ddhazard_control`. The above is sensitive to the choice of $\mathbf{Q}_0$. An extreme example is if we have no events in the first interval and only an intercept. Then setting $\mathbf{Q}_0$ to a diagonal matrix with large entries (in this case $\mathbf{Q}_0$ is a scalar) implies almost no restrictions on the intercept. Thus, it will be optimal to select a value tending towards minus infinity.

6.1 Alternative implementation

An alternative to the above algorithm for the exponential family is to re-write the original problem to get a weighted least squares problem of the form:

$$\boldsymbol{b} = \mathbf{X}_t\boldsymbol{a}^{(k-1)} + \boldsymbol{h}'\left(\mathbf{X}_t\boldsymbol{a}^{(k-1)}\right)^{-1}\left(\boldsymbol{y}_t - \boldsymbol{h}\left(\mathbf{X}_t\boldsymbol{a}^{(k-1)}\right)\right)$$

$$\arg\min_{\boldsymbol{\alpha}} \left\| \underbrace{\begin{pmatrix} \boldsymbol{h}'\left(\mathbf{X}_t\boldsymbol{a}^{(k-1)}\right)\operatorname{Var}\left(\boldsymbol{Y}_t|\,\mathbf{X}_t\boldsymbol{\alpha}^{(k-1)}\right)^{-1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{t|t-1}^{-1/2} \end{pmatrix}}_{\widetilde{\mathbf{C}}^{1/2}} \left(\underbrace{\begin{pmatrix} \mathbf{X}_t \\ \mathbf{I} \end{pmatrix}}_{\widetilde{\mathbf{X}}_t} \boldsymbol{\alpha} - \underbrace{\begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{\alpha}_{t|t-1} \end{pmatrix}}_{\widetilde{\boldsymbol{b}}} \right) \right\|$$

where $\boldsymbol{b}$ is the working responses, $\boldsymbol{h}$ temporarily denotes the inverse link function at time t, $\boldsymbol{h}'$ is the derivative w.r.t. the linear predictor $\mathbf{X}_t\boldsymbol{a}^{(k-1)}$ and the inverse link function $\boldsymbol{h}$ implicitly depends on the risk set at time t. The minimum w.r.t. $\boldsymbol{\alpha}$ is $\boldsymbol{\alpha}^{(k)} = \left(\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\mathbf{X}}_t\right)^{-1}\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\boldsymbol{b}}$. Though, this is the EKF shown earlier. To see this, set $\boldsymbol{a}^{(k-1)} = \boldsymbol{a}_{t|t-1}$. Then,

$$\left(\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\mathbf{X}}_t\right)^{-1} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}_{t|t-1})\right)^{-1}$$

Further,

$$\begin{aligned}\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\boldsymbol{b}} &= \mathbf{X}_t^T\boldsymbol{h}'\left(\mathbf{X}_t\boldsymbol{a}_{t|t-1}\right)^2\operatorname{Var}\left(\boldsymbol{Y}_t|\,\mathbf{X}_t\boldsymbol{a}_{t|t-1}\right)^{-1}\mathbf{X}_t\boldsymbol{a}_{t|t-1} + \boldsymbol{u}_t(\boldsymbol{a}_{t|t-1}) + \mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} \\ &= \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}_{t|t-1})\right)\boldsymbol{a}_{t|t-1} + \boldsymbol{u}_t(\boldsymbol{a}_{t|t-1})\end{aligned}$$

Thus,

$$\left(\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\mathbf{X}}_t\right)^{-1}\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\boldsymbol{b}} = \boldsymbol{a}_{t|t-1} + \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}_{t|t-1})\right)^{-1}\boldsymbol{u}_t(\boldsymbol{a}_{t|t-1})$$

This is the update equation in the EKF. Taking multiple steps give us:

$$\left(\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\mathbf{X}}_t\right)^{-1} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}^{(k-1)})\right)^{-1}$$

$$\begin{aligned}\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\boldsymbol{b}} &= \mathbf{X}_t^T\boldsymbol{h}'\left(\mathbf{X}_t\boldsymbol{a}^{(k-1)}\right)^2\operatorname{Var}\left(\boldsymbol{Y}_t|\,\mathbf{X}_t\boldsymbol{a}^{(k-1)}\right)^{-1}\mathbf{X}_t\boldsymbol{a}^{(k-1)} + \boldsymbol{u}_t(\boldsymbol{a}^{(k-1)}) + \mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} \\ &= \mathbf{U}_t(\boldsymbol{a}^{(k-1)})\boldsymbol{a}^{(k-1)} + \mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} + \boldsymbol{u}_t(\boldsymbol{a}^{(k-1)})\end{aligned}$$

$$\boldsymbol{a}^{(k)} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}^{(k-1)})\right)^{-1}\left(\mathbf{U}_t(\boldsymbol{a}^{(k-1)})\boldsymbol{a}^{(k-1)} + \mathbf{V}_{t|t-1}^{-1}\boldsymbol{a}_{t|t-1} + \boldsymbol{u}_t(\boldsymbol{a}^{(k-1)})\right)$$

7 Weights

Weights can be used in the EKF, UKF and posterior mode approximations. This can reduce the computation in the logistic model with only categorical covariates or if we want to bootstrap the estimates. The following section covers how the weights are handled in the previous filters. We will denote the weights at time t by $\boldsymbol{e}_t = (e_1, e_2, \ldots, e_{n_t})$ where $n_t = |R_t|$ is the number of observations at risk at time t. Further, we denote $\mathbf{E}_t$ as the diagonal matrix with $\boldsymbol{e}_t$ as the diagonal

7.1 EKF

Weights are handled in the EKF by replacing

$$\boldsymbol{u}_t(\boldsymbol{\alpha}_t) = \sum_{(i,j)\in R_t} \boldsymbol{u}_{ijt}(\boldsymbol{\alpha}_t), \qquad \mathbf{U}_t(\boldsymbol{\alpha}_t) = \sum_{(i,j)\in R_t} \mathbf{U}_{ijt}(\boldsymbol{\alpha}_t)$$

with

$$\boldsymbol{u}_t(\boldsymbol{\alpha}_t) = \sum_{k=1}^{n_t} e_k \boldsymbol{u}_{(R_t)_k t}(\boldsymbol{\alpha}_t), \qquad \mathbf{U}_t(\boldsymbol{\alpha}_t) = \sum_{k=1}^{n_t} e_k \mathbf{U}_{(R_t)_k t}(\boldsymbol{\alpha}_t)$$

where $(R_t)_k$ is the k'th element of R_t.

7.2 UKF

Weights are handled in the UKF by replacing:

$$\tilde{\boldsymbol{y}} = \Delta\widehat{\mathbf{Y}}^\top \widehat{\mathbf{H}}^{-1}(\boldsymbol{y}_t - \overline{\boldsymbol{y}}) \qquad \mathbf{G} = \Delta\widehat{\mathbf{Y}}^\top \widehat{\mathbf{H}}^{-1}\Delta\widehat{\mathbf{Y}}$$

with

$$\tilde{\boldsymbol{y}} = \Delta\widehat{\mathbf{Y}}^\top \mathbf{E}_t\widehat{\mathbf{H}}^{-1}(\boldsymbol{y}_t - \overline{\boldsymbol{y}}) \qquad \mathbf{G} = \Delta\widehat{\mathbf{Y}}^\top \mathbf{E}_t\widehat{\mathbf{H}}^{-1}\Delta\widehat{\mathbf{Y}}$$

7.3 SMA

Weights are handled in the SMA by replacing:

$$v = \arg\min_b b^2\frac{1}{2}d_1 - bd_1d_2 - (y_{ijt}\log h(b) + (1-y_{ijt})\log(1-h(b))$$
$$\mathbf{V}_{t|t}^{(k)} = \mathbf{V}_{t|t}^{(k-1)} - \frac{\mathbf{V}_{t|t}^{(k-1)}\boldsymbol{x}_{ijt}g\boldsymbol{x}_{ijt}^\top\mathbf{V}_{t|t}^{(k-1)}}{1+g\boldsymbol{x}_{ijt}^\top\mathbf{V}_{t|t}^{(k-1)}\boldsymbol{x}_{ijt}}$$

with

$$v = \arg\min_b b^2\frac{1}{2}d_1 - bd_1d_2 - e_f\left(y_{ijt}\log h(b) + (1-y_{ijt})\log(1-h(b))\right)$$
$$\mathbf{V}_{t|t}^{(k)} = \mathbf{V}_{t|t}^{(k-1)} - \frac{\mathbf{V}_{t|t}^{(k-1)}\boldsymbol{x}_{ij}e_fg\boldsymbol{x}_{ijt}^\top\mathbf{V}_{t|t}^{(k-1)}}{1+e_fg\boldsymbol{x}_{ijt}^\top\mathbf{V}_{t|t}^{(k-1)}\boldsymbol{x}_{ijt}}$$

7.4 GMA

Weights are handled in the same way as for the SMA by multiplying the $c'(\boldsymbol{\alpha})$ and $c''(\boldsymbol{\alpha})$ with the weights of the observation.

8 Fixed effects

This section will cover how fixed effects (non time-varying effects) are estimated. We will denote the coefficients for the fixed effects by $\boldsymbol{\gamma}$. The fixed effects can be estimated with two methods. The first one is by adding the fixed effects to state equation with their elements of the covariance matrix $\mathbf{Q}$ set to zero. That is, we estimate the fixed effects in the E-step. The second method is to estimate the fixed effects in the M-step.

8.1 Estimation in the E-step

The fixed effect can be estimated in the E-step in a similar manner to Harvey and Phillips [1979]. The method in Harvey and Phillips [1979] is similar to recursive least squares where some of the effects are time-varying. The elements with the fixed effects have a large value in the diagonal of $\mathbf{Q}_0$ (say 10^6) and zero in the elements of the covariance matrix $\mathbf{Q}$. Thus, we end with recursive least squares for the linear model if all effects are fixed.

In this package, we set the entries of $\mathbf{Q}_0$ and $\mathbf{Q}$ in the same way. Nothing else is changed in the E-step. Further, we set the all rows and columns of the fixed effects in $\mathbf{Q}$ to zero after the update in the M-step. This seems to work with the EKF for a large range of large diagonal elements $\mathbf{Q}_0$ (say 10^5). However, the choice of the diagonal entry in $\mathbf{Q}_0$ for fixed effects do have an impact with the UKF. "*Large*" but not "*too large*" values tends to work. Though, what is large depends data set and model. The default for the diagonal elements of $\mathbf{Q_0}$ for the fixed effects can be set with `Q_0_term_for_fixed_E_step` argument of `ddhazard_control`. Setting `fixed_terms_method = "E_step"` in the `ddhazard_control` call yields this method.

8.2 Estimation in the M-step

The other method is to estimate the fixed parameters in the M-step. For this section, I define $\boldsymbol{\gamma}$ as the fixed parameters, $\boldsymbol{x}_{ijt}$ as the covariates corresponding to the time-varying parameters, and $\boldsymbol{z}_{ijt}$ as the covariates corresponding to the fixed parameters. The dot product between the fixed parameters and the corresponding covariates act as offsets in the filters because the linear predictor is $\boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha}_t + \boldsymbol{z}_{ijt}^\top \boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ is fixed. Moreover, the formulas for $\boldsymbol{a}_0$ and $\mathbf{Q}$ in the M-step are not affected because the only relevant terms for fixed effects in the M-step are in the last line of the log-likelihood. However, the optimization is not easily solved exactly in the M-step for the fixed parameters. The log-likelihood we need to maximize in the M-step in the k'th iteration is

$$\arg\max_{\boldsymbol{\gamma}} E\left(\sum_{t=1}^{d}\sum_{(i,j)\in R_t} l_{ijt}(\boldsymbol{\alpha}_t, \boldsymbol{\gamma}) \middle| \boldsymbol{a}_0^{(k-1)}, \boldsymbol{\gamma}^{(k-1)}, \mathbf{Q}^{(k-1)}, \mathbf{Q}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_d\right)$$

where I temporarily add an additional argument in the log-likelihood terms, l_{ijt}, for the fixed effects, and use the superscripts to differentiate between the hyperparameter estimates of different iterations of the EM-algorithm. The above is not easy to optimize. Thus, I make a zero-order Taylor expansion about $\boldsymbol{a}_{1|d}, \ldots, \boldsymbol{a}_{d|d}$ in the current implementation to get

$$\arg\max_{\boldsymbol{\gamma}} \sum_{t=1}^{d}\sum_{(i,j)\in R_t} l_{ijt}(\boldsymbol{a}_{t|d}, \boldsymbol{\gamma})$$

One advantage of doing this is that the problem can be solved with regular methods for GLMs when the model is from the exponential family. However, the design matrix will be big, as each individual will yield multiple rows because of different offsets from the time-varying parameters given by $\boldsymbol{x}_{ijt}^\top \boldsymbol{a}_{t|d}$. To overcome this problem, I use the Fortran code from Miller [1992] to do a series of rank-one updates of the QR-decomposition used to solve the iteratively re-weighted least square problem. This is the same approach as in the biglm package. The computational complexity of each update is $\mathcal{O}\left(c^2\right)$, where c is the dimension of $\boldsymbol{\gamma}$.

8.3 Which method to use

Neither the method that use the recursive least squares like method in the E-step, nor the zero order Taylor expansion in the M-step have performed consistently better on the data sets seen so far. Hence, both are valid alternatives at this point. Fixed terms can be estimated by wrapping the covariates in the formula of `ddhazard` in the `ddFixed` function. As an example, `ddhazard(Surv(tstart, tstop, y) ~x1 + ddFixed(x2), ...)` will fit a model where `x1` is time-varying and `x2` is not.

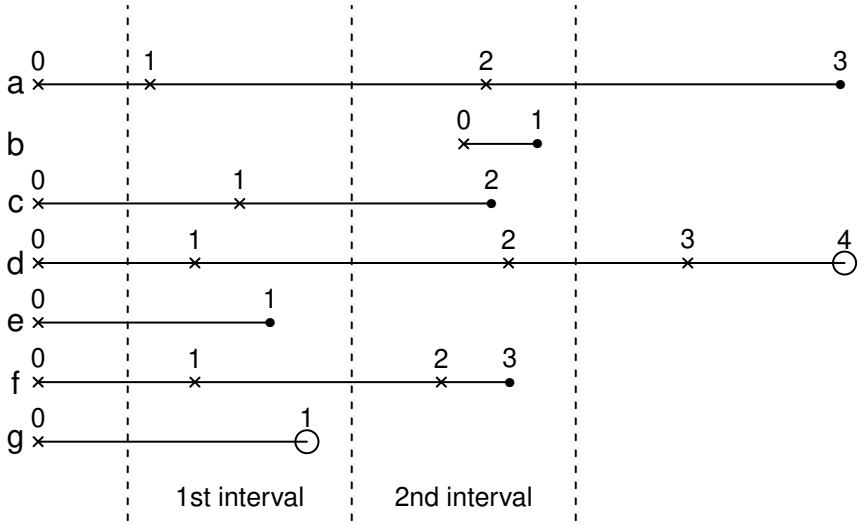


Figure 1: Illustration of going from event times to binary variables. Each horizontal line represents an individual. A cross indicates that new covariates are observed while a filled circle indicates that the individual have died. Open circles indicates that the individual is right-censored. Vertical dashed lines are time interval borders.

9 Logistic model

The logistic model uses the inverse logit function as the inverse link function h. That is $h(\eta) = \exp(\eta)/(1 + \exp(\eta))$. The logistic model is fitted by setting `model = "logit"` in the call to `ddhazard`. The following paragraphs will cover the "loss" of information due to using time intervals instead of event times which motivates the continuous time model.

9.1 Event times to binary variables

This section will illustrate how we go from event time to binary variables for the logistic model and how this can lead to "loss" of information. It is elementary but included to stress this point and motivate the continuous time model. We will use figure 1 as the illustration. Each horizontal line represent an individual. A cross represents when the covariate values change for the individual and a filled circle represents the death of an individual. Lines that ends with an open circle are right-censored.

We will return to the vertical lines shortly. First, we notice that the example is where we assume that the covariates are step functions. An example hereof is a medical trial where patients get tests taken at different point in time (when they have a time at their doctor, visit the hospital or similar). As an example, ideally we would like to model that individual one has a blood pressure of x at time 0, re-visits at time 1.5 and has a blood pressure y and dies at time 2.5 whereas individual 2 has a blood pressure if z at time zero, never visits the doctor again and we know that he have not died by time 2.25 (he is right-censored).

However, we do not model event times in the logistic model. Instead, we model binary outcomes in each time interval. The vertical dashed lines in the figure represents the time interval borders. The first vertical line from the left is where we start our estimation, the second vertical line is where the first time interval ends and the second time intervals starts and the third vertical line is where the time interval ends. Thus, we only have two time intervals in this example.

We can now cover how the individuals (horizontal lines) are used in the estimation:

a is a control in both time intervals. We use the covariates from 0 in the first time interval and the covariates from 1 in the second time interval.

b is not included in any of the time intervals. We do not know the covariates values at the start of the second time interval so we cannot include him.

c is a control in the first time interval with the covariates from 0. He counts as a death in the second time interval with the covariates from 1.

d acts like a.

e is a death in the first time interval with covariates from 0.

f is a control in the first time interval with the covariates from 0. He counts as a death in the second time interval with the covariates from 1.

g is not included in any time intervals. We do not know if he survived the entire period of the first time interval and thus we cannot include him.

The example illustrates that:

1. We loose information about covariates that are updated within time intervals. For instance, a, c, d and f all use the covariates from 0 for the entire period of the first time interval despite that the covariates change at 1. Moreover, we never use the information at 2 from a, d and f.
2. We loose information when we have right censoring. For instance, g is not included at all since we only know that he survives parts of the first time interval.
3. We loose information for observation that only occurs within time intervals as is the case for b.

The above motivates the continuous time model that will be covered in the next sections where we go from modelling binary outcomes to event times.

10 Continuous time model

The following section introduce the continuous time model. We start by describing the assumption of the continuous time model. Then we turn to different estimation methods.

10.1 Assumptions

We make the following assumption in the continuous time model:

1. Coefficients (that is state variables $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d$) change at the end of time intervals.
2. The individuals covariates change at discrete times.
3. We have piecewise constant instantaneous hazards given by $\exp(\boldsymbol{x}_{ijt}^\top \boldsymbol{\alpha}_t)$ given an individual's current co-covariate vector $\boldsymbol{x}_{ijt}$ and state variable $\boldsymbol{\alpha}_t$ (assuming that individual i's j'th covariate is within time interval t).

The instantaneous hazard change when either the individuals covariates change or the coefficients change when we change time interval. Thus, each individual's stop time is piecewise constant exponential distributed event time given the state vectors. The log-likelihood omitting a normalization constant:

$$\begin{aligned}
\mathcal{L}\left(\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_d\right) = &- \frac{1}{2}\left(\boldsymbol{\alpha}_0 - \boldsymbol{a}_0\right)^\top \mathbf{Q}_0^{-1}\left(\boldsymbol{\alpha}_0 - \boldsymbol{a}_0\right) \\
&- \frac{1}{2}\sum_{t=1}^{d}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}\right)^\top \mathbf{Q}^{-1}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}\right) \\
&- \frac{1}{2}\log|\mathbf{Q}_0| - \log\frac{d}{2}|\mathbf{Q}| \\
&+ \sum_{t=1}^{d}\sum_{(i,j)\in\mathcal{R}_t} l_{ijt}(\boldsymbol{\alpha}_t) + \ldots \\
l_{ijt}(\boldsymbol{\alpha}_t) = &y_{ijt}\boldsymbol{x}_{ijt}^\top\boldsymbol{\alpha}_t - \exp\left(\boldsymbol{x}_{ijt}^\top\boldsymbol{\alpha}_t\right)\left(\min\{t, t_{ij}\} - \max\{t-1, t_{i,j-1}\}\right)
\end{aligned}$$

where the l_{ijt} terms come from the log-likelihood:

$$\log\left(P\left(t_i|\,\boldsymbol{\alpha}_0,\ldots,\boldsymbol{\alpha}_d\right)\right) = \boldsymbol{x}_i(t_i)^\top\boldsymbol{\alpha}(t_i) - \int_0^{t_i}\exp\left(\boldsymbol{x}_i(u)^\top\boldsymbol{\alpha}(u)\right)\,du$$

which simplifies into the terms of l_{ijt}s when both the covariates $\boldsymbol{x}_i(t)$ and state space parameters $\boldsymbol{\alpha}(t)$ are piecewise constant. Further, $\mathcal{R}_t$ is the continuous risk set given by:

$$\mathcal{R}_t = \{(i,j)\in\mathbb{Z}_+\times\mathbb{Z}:\, t_{i,j-1} < t \wedge t_{ij} > t-1\}$$

In words, the condition is that the j'th observation of individual i is in the risk set if the observations 1) starts before the intervals ends and 2) ends after the interval starts. The EKF and UKF uses Poisson counts with a offset equal to the log of the at risk length of the covariate vector and coefficients pair for the continuous time model. The SMA and GMA works directly with log-likelihood as shown above.

11 Diagnostics

This section will cover diagnostics tools. These includes:

- Residuals from the observations.
- Hat values.
- Residuals from the state vector.

11.1 Residuals from the observations

For the binary outcomes in the logistic model, one idea is to look at the Pearson residuals which we denote r^P_{ijt} which is the i'th individual's Pearson residual with covariate vector j in interval t. That is,

$$\begin{aligned}\hat{y}_{ijt} &= \exp\left(\boldsymbol{x}_{ijt}^\top\boldsymbol{a}_{t|d}\right)\big/\left(1+\exp\left(\boldsymbol{x}_{ijt}^\top\boldsymbol{a}_{t|d}\right)\right)\\ r^P_{ijt} &= \frac{y_{ijt}-\hat{y}_{ijt}}{\sqrt{H_{kkt}(\boldsymbol{a}_{t|d})}} = \frac{y_{ijt}-\hat{y}_{ijt}}{\sqrt{\hat{y}_{ijt}(1-\hat{y}_{ijt})}}\end{aligned}$$

Then we can:

- Plot residuals against time and highlight the individuals with at least one "high" residual.
- Accumulate residuals for each individual i and plot against t. Any individuals with large or small values may worth looking at.
- Stratify a covariate values into factors and plot accumulated residuals versus time. Any structural deviations may show a missing covariate or incorrect transformation of the covariate on the linear predictor scale.
- Accumulate residuals across intervals t and plot these.

You can get the Pearson residuals by calling `residuals` with a `ddhazard` fit and with argument `type = "pearson"`.

11.2 Hat values

Finding the influence matrix (also known as the hat matrix) does not seem to be possible in a computationally efficient way. Thus, we will look at an approximation. We will focus on the logistic model. In the filters in the E-step, each correction step in itself can be viewed as an logistic regression with an L2 penalty. Say we at time t in the filter in the correction step with estimates $\boldsymbol{a}_{t|t-1}$ and $\mathbf{V}_{t|t-1}$. Then the penalty could be interpreted as a prior $N(\boldsymbol{a}_{t|t-1},\mathbf{V}_{t|t-1})$. With this view, regular hat values would be computed by:

$$\mathbf{C}_t(\boldsymbol{\alpha}_{t|t-1})^{1/2}\mathbf{X}_t\left(\mathbf{X}_t^\top\mathbf{C}_t(\boldsymbol{\alpha}_{t|t-1})^{-1}\mathbf{X}_t+\mathbf{V}_{t|t-1}^{-1}\right)^{-1}\mathbf{X}_t^\top\mathbf{C}_t(\boldsymbol{\alpha}_{t|t-1})^{1/2}$$

where $\mathbf{X}_t$ is the design matrix in interval t and $\mathbf{C}_t(\boldsymbol{\alpha})$ is the working weights as in `glm`. The above could motivate the following matrix as the "hat-like" matrix in each interval:

$$\widetilde{\mathbf{C}}_t(\boldsymbol{\alpha}_{t|d})^{1/2}\mathbf{X}_t\left(\mathbf{X}_t^\top\widetilde{\mathbf{C}}_t(\boldsymbol{\alpha}_{t|d})^{-1}\mathbf{X}_t+\mathbf{V}_{t|d}^{-1}\right)^{-1}\mathbf{X}_t^\top\widetilde{\mathbf{C}}_t(\boldsymbol{\alpha}_{t|d})^{1/2}$$

where we have used the final smoothed estimators. Plotting cumulative values versus time may show influential observations. You can get these estimates by calling `hatvalues` with a `ddhazard` object as the argument.

11.3 Residuals from the state vector

We may be interested in looking at the predicted state error. The predicted state errors are given by:

$$\widehat{\boldsymbol{\eta}}_t=\mathbf{R}^\top\left(\boldsymbol{a}_{t|d}-\mathbf{F}\boldsymbol{a}_{t-1|d}\right)$$

which we may check if they are $N\left(\mathbf{0},\text{Var}\left(\boldsymbol{\eta}_t|\,\boldsymbol{Y}_d\right)\right)$. This will require that we find the smoothed covariance matrix $\text{Var}\left(\boldsymbol{\eta}_t|\,\boldsymbol{Y}_d\right)=\mathbf{R}^\top\text{Var}\left(\boldsymbol{\alpha}_t-\mathbf{F}\boldsymbol{\alpha}_{t-1}|\,\boldsymbol{Y}_d\right)\mathbf{R}$ in order to standardize the predicted errors. We will explain how this can be estimated in the following paragraphs when the EKF have been used. Standard results yields:

$$\begin{aligned}\text{Var}\left(\boldsymbol{\alpha}_t-\mathbf{F}\boldsymbol{\alpha}_{t-1}|\,\boldsymbol{Y}_d\right)=&\text{Var}\left(\boldsymbol{\alpha}_t|\,\boldsymbol{Y}_d\right)+\mathbf{F}\text{Var}\left(\boldsymbol{\alpha}_{t-1}|\,\boldsymbol{Y}_d\right)\mathbf{F}^\top\\&-\mathbf{F}\text{Cov}\left(\boldsymbol{\alpha}_t,\boldsymbol{\alpha}_{t-1}|\,\boldsymbol{Y}_d\right)-\text{Cov}\left(\boldsymbol{\alpha}_t,\boldsymbol{\alpha}_{t-1}|\,\boldsymbol{Y}_d\right)^\top\mathbf{F}^\top\end{aligned}$$

Thus, we need smoothed correlation matrices $\text{Cov}\left(\boldsymbol{\alpha}_t,\boldsymbol{\alpha}_{t-1}|\,\boldsymbol{Y}_d\right)$. We can estimate these recursively by first setting [e.g., see Shumway and Stoffer, 2006]:

$$\text{Cov}\left(\boldsymbol{\alpha}_d,\boldsymbol{\alpha}_{d-1}|\,\boldsymbol{Y}_d\right)=\left(\mathbf{I}-\mathbf{K}_d\dot{\boldsymbol{z}}_d(\boldsymbol{a}_{d|d})\right)\mathbf{F}\mathbf{V}_{d-1|d-1},\qquad\dot{\boldsymbol{z}}_d(\boldsymbol{a}_{d|d})=\left.\frac{\partial\boldsymbol{z}_d(\alpha)}{\partial\boldsymbol{\alpha}}\right|_{\boldsymbol{\alpha}=\boldsymbol{a}_{d|d}}$$

where $\mathbf{K}_d$ is the Kalman gain given by:

$$\mathbf{K}_d=\mathbf{F}\mathbf{V}_{d|d-1}\dot{\boldsymbol{z}}_d(\boldsymbol{a}_{d|d-1})^\top\mathbf{C}^{-1},\qquad\mathbf{C}=\text{Var}\left(\boldsymbol{Y}_d|\,\boldsymbol{y}_{d-1}\right)=\dot{\boldsymbol{z}}_d(\boldsymbol{a}_{d|d-1})^\top\mathbf{V}_{d|d-1}\dot{\boldsymbol{z}}_d(\boldsymbol{a}_{d|d-1})+\mathbf{H}_d(\boldsymbol{a}_{d|d-1})$$

Next, for $t=d,d-1,\ldots,2$ we recursively compute:

$$\begin{aligned}&\text{Cov}\left(\boldsymbol{\alpha}_{t-1},\boldsymbol{\alpha}_{t-2}|\,\boldsymbol{Y}_d\right)\\&\quad=\mathbf{V}_{t-1|t-1}\mathbf{B}_{t-1}^\top+\mathbf{B}_t\left(\text{Cov}\left(\boldsymbol{\alpha}_t,\boldsymbol{\alpha}_{t-1}|\,\boldsymbol{Y}_d\right)-\mathbf{T}\mathbf{V}_{t-1|t-1}\right)\mathbf{B}_{t-1}^\top\end{aligned}$$

Though, $\mathbf{C}$ will be a large square matrix when we have a lot of observation and possibly singular. However, we can apply the Woodbury matrix identity to get:

$$\mathbf{C}=\mathbf{H}_d(\boldsymbol{a}_{d|d-1})^{-1}-\mathbf{X}_d\mathbf{V}_{d|d-1}\left(\mathbf{I}+\mathbf{U}_d(\boldsymbol{a}_{d|d-1})\mathbf{V}_{d|d-1}\right)^{-1}\mathbf{X}_d^\top$$

where $\mathbf{X}_d$ is the design matrix in the final interval. This is easy to compute when $\mathbf{H}_d$ is a diagonal matrix and the dimension of the state equation is low. You can get the standardized predicted state errors by calling `residuals` with a `ddhazard` fit and `type = "std_space_error"` if you have used the EKF.

References

R. Van der Merwe and E. A. Wan. The square-root unscented kalman filter for state and parameter-estimation. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 6, pages 3461–3464 vol.6, 2001. doi: 10.1109/ICASSP.2001.940586.

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

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

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. ISSN 1548-7660. doi: 10.18637/jss.v033.i01. URL `https://www.jstatsoft.org/index.php/jss/article/view/v033i01`.

Fredrik Gustafsson and Gustaf Hendeby. Some relations between extended and unscented kalman filters. *IEEE Transactions on Signal Processing*, 60(2):545–555, 2012.

Andrew C Harvey and Gary DA Phillips. Maximum likelihood estimation of regression models with autoregressive-moving average disturbances. *Biometrika*, 66(1):49–58, 1979.

Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997.

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

Henrique Marra Taira Menegaz. *Unscented kalman filtering on euclidean and Riemannian manifolds*. PhD thesis, The University of Brasília, 2016. Available on `http://repositorio.unb.br/handle/10482/21617`.

Alan J Miller. Algorithm as 274: Least squares routines to supplement those of gentleman. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):458–478, 1992.

Matthias Seeger. Low rank updates for the cholesky decomposition. Available on webpage `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.585.5275&rep=rep1&type=pdf`, 2004.

Robert H Shumway and David S Stoffer. Time series analysis and its applications: with r examples. *Springer texts in statistics*, 2006.

Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000.