ddhazard

Benjamin Christoffersen

2017-06-07

Introduction

This note will cover the `ddhazard` function used for estimation in the `dynamichazard` library. You can install the version of the library used to make this vignette from github with the `devtools` library as follows:

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

```
## [1] "boennecd/dynamichazard@ed546aa20dd526a2286551bb2dd8b10ea94aaad4"
```

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

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

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

The `ddhazard` function estimates a dynamic binary regression model where the parameters are assumed to time-varying and follow a random walk.

Why and when to use this package

The package is intended for situation where you have a dynamic binary regression 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. Another use of this package is as an alternative to other methods of modelling time-varying coefficients for binary regression such as Generalized Additive models

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 censoring
3) It is fast due to the `C++` implementation which uses `Armadillo` library and use of multithreading through the standard library `thread`

All are important in the analysis of firm failures. Firstly, you can easily have 40-50.000 firms at risk at each point in time. Thus, point 1) is key to be able to fit the models. Moreover, the number of firms at risk will vary as time progress. Some firms default, some are opened, some merge, some are acquired etc. This relates to point 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. *Simulation study with logit model* presents a simulation study where the methods in this package are compared to each other and to Generalized Additive models. *Comparing methods for time-varying logistic models* applies the methods to a real world data set. Both vignettes illustrate how to use the estimation function `ddhazard` and other functions in this package. They only use the discrete time model. This vignette also describes the continuous time model. The *Bootstrap illustration* vignette shows how the wrapper `ddhazard_boot` for the function `boot` in the `boot` library works.

ddhazard Diagnostics illustrates how the `residuals` and `hatvalues` functions can be used to check the model fit

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 and 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 models we look at will allow for censoring to deal handle an individual like id `2`

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:

$$R_t = \left\{(i,j) \in \mathbb{Z}_+^2 : t_{i,j-1} \leq t-1 \leq t_{i,j}\right\}$$

where $\mathbb{Z}$ are the natural number $1, 2, \dots$. 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_s 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:

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

The equation for $\boldsymbol{y}_t$ is denoted the *observational equation*. $\sim (v, b)$ denotes a random variable(s) with mean (vector) v and variance (co-variance matrix) b. It needs not be a normal distribution. $\boldsymbol{\alpha}_t$ 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 define the observational equations covariance matrix as $\mathbf{H}_t(\boldsymbol{\alpha}_t) = \text{Var}\left(\boldsymbol{y}_t | \, \boldsymbol{\alpha}_t\right)$

The mean $\boldsymbol{z}_t(\boldsymbol{\alpha}_t)$ and variance $\mathbf{H}(\boldsymbol{\alpha}_t)$ are state dependent with:

$$\begin{aligned} z_{ft}(\boldsymbol{\alpha}_t) &= E\left(Y_{ijt} | \, \boldsymbol{\alpha}_t\right) = h(\boldsymbol{\alpha}_t^\top \boldsymbol{x}_{ijt}) \\ H_{ff't}(\boldsymbol{\alpha}_t) &= \begin{cases} \text{Var}\left(Y_{ijt} | \, \boldsymbol{\alpha}_t\right) & f = f' \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} z_{ft}(\boldsymbol{\alpha}_t)(1 - z_{ft}(\boldsymbol{\alpha}_t)) & f = f' \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where we assumed that individual i with covariate vector j was at the f'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. The vector in the state equation is ordered as $\boldsymbol{\alpha}_t = (\widetilde{\boldsymbol{\alpha}}_t^\top, \widetilde{\boldsymbol{\alpha}}_{t-1}^\top)^\top$ to match the definition of $\mathbf{F}$ and $\mathbf{R}$ where the tilde is added to indicate the coefficients used when computing the linear predictor. The likelihood of the model where $\boldsymbol{\alpha}_t$ are observed can be written as follows by application of the markovian property of the model:

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

which we can expand to:

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

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

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 the filtering. The method is chosen by passing a list to the `control` argument of `ddhazard` with `list(method = "EKF", ...)`, `list(method = "UKF", ...)` or `list(method = "SMA", ...)` respectively. 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 errors 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. As a last comment in this context, `Q_0` is not scaled and thus will exactly match $\mathbf{Q}_0$ in the estimation. The logic here 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 to call the `ddhazard` looks and to show that Q will be scaled by ψ_t. To do so, we use the data frame where we showed the first few entries from before. An estimation call will look like:

```
library(dynamichazard)
library(survival)
dd_fit_short <- ddhazard(
  Surv(tstart, tstop, event) ~ x1 + x2, # Formula like for coxph from survival
       data = simple_ex,
       by = 1,                          # Length of time intervals
       Q = diag(0.1, 3),                # Covariance matrix in state eqn
       Q_0 = diag(10000, 3),            # Covariance matrix for initial state
                                        # vector
       max_T = 28,                      # Last time we observe
       id = simple_ex$id                # id of individuals
  )

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

```
## (Intercept)          x1          x2
##       0.198       0.174       0.173
```

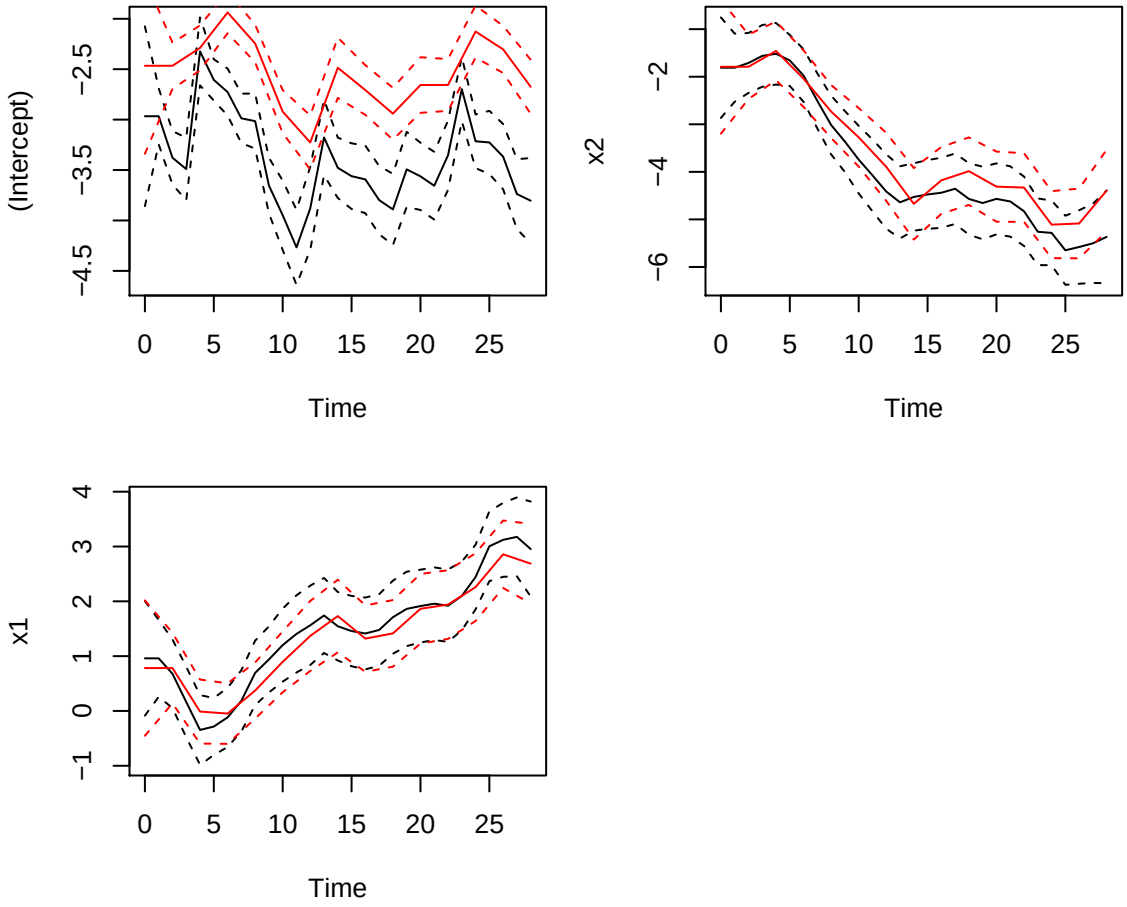
Above, we estimate the model with a time intervals of length `by = 1`. The model is the logistic model which we introduced later. For now, let us see what happens if we increase the interval length by changing the `by` argument:

```
library(dynamichazard)
library(survival)
dd_fit_wide <- ddhazard(
  Surv(tstart, tstop, event) ~ x1 + x2,
       data = simple_ex,
       by = 2,                          # increased
       Q = diag(0.1, 3),
       Q_0 = diag(10000, 3),
       max_T = 28,
       id = simple_ex$id)

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

```
## (Intercept)          x1          x2
##      -0.546      -0.115       0.281
```

We see that the diagonal entries are not "to-far" from each other with the two fits. Plots of the two predictions of the coefficients are similar in terms of width of the confidence bounds (black is the short interval width and red is long interval width):

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

for(i in 1:3){
  plot(dd_fit_short, cov_index = i, col = "Black")
  plot(dd_fit_wide, cov_index = i, col = "Red", 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. The section 'EM algorithm' will cover the EM algorithm. This is followed by the sections 'Extended Kalman Filter', 'Unscented Kalman Filter' and 'Approximation of the posterior mode' which respectively covers the EKF, UKF and approximation of the posterior mode used to make the filtering in the E-step of the EM algorithm. Next, the section 'Weigths' and 'Fixed effects' covers how the weights and estimation of fixed effects are done. The sections 'Logistic model' and 'Continuous time model' covers the models implemented in this package. Finally, we end with a section on diagnostics. You may find in it useful to make lookups of the notation in the appendix while reading

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

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 = list(est_Q_0 = T, ...)`. Though this is discourage as we are estimating more parameters than observations without penalization. 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 $s \geq t$. 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 $\mathbf{Q}$

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 either by using a filter to estimate:

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

Then the estimates are smoothed by computing:

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

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 $t = d$

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 test 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` element of the list passed to the `control` argument of `ddhazard` (e.g. `list(eps = 0.01, ...)`)

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 (1994) and Fahrmeir (1992) 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 $O(p)$ instead of $O(p_t^3)$ where $n_t = |R_t|$ denotes the dimension of the observation equation. That is, $n_t = |R_t|$. 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:

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

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

Divergence

Initial testing shows 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 datasets 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:

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

where δ is small like 10^{-9}. The arguments for the above are formulas are covered later on 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. See the vignette "Simulation study with logit model" for details

The value of ζ_0 and ϵ_{NR} are set by respectively setting the elements `LR` and `NR_eps` of the list passed to the `control` argument of `ddhazard`. 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 = list(LR = 0.75, NR_eps = 0.001)` for a learning rate of 0.75 and a threshold in the Fisher Scoring of 10^{-3}

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:

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

where $\xi > 0$ is a small number. The default can be changed by altering the `denom_term` in the list passed to the `control` argument of `ddhazard`. The approach is similar to how the `glmnet` package handles close to boundary estimates (see Friedman, Hastie, & Tibshirani (2010))

Parallel BLAS or LAPACK

All the computations use objects from the `Armadillo` library. Thus, an optimized version LAPACK and BLAS can speed up the computation. A multithreaded version of LAPACK or BLAS can cause issues with performance. The majority of the computation time is spent in the correction step of the EKF, where we compute $\boldsymbol{u}_t(\boldsymbol{\alpha}_t)$ and $\mathbf{U}_t(\boldsymbol{\alpha}_t)$, when the number of regression parameter is low and we have a lot of observations. For this reason, this part of the code is computed in parallel with the `C++` standard library `thread`. The reduction in computation time can be offset if a multithreaded version of LAPLACK or BLAS is used as the code already use multithreading

A specific solution to the issues is implemented for Windows users who compiles with openBLAS. The `src/Makevars.win` checks if there is `C:\OpenBLAS` folder. If so, we assume that the structure is:

```
C:/OpenBLAS/
|--lib/
   |--libopenblas.a
|--include/
   |--cblas.h
   |--f77blas.h
```

The code will be compiled with this `openBLAS` instead of the `BLAS` library used to compile `R`. This will allow parts of the matrix operations to be run in parallel by using `openBLAS` for multithreading. The number of threads openBLAS will use is set to 1 before the part that use the `thread` library is run and reset after the this part is completed.

Unscented Kalman Filter

The UKF selects state vectors called *sigma point* with given *sigma weigths* chosen to match the moments of observation equation. Thus, we approximate the density rather than approximating the observational equation. The idea is similar to a Monte Carlo method 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 & Uhlmann (1997) introduce a UKF that approximate the first two moments and up to fourth moment in certain settings. Julier & Uhlmann (2004) further develop the UKF and extended to what is later called *the Scaled Unscented Transformation*. We will cover the the Scaled Unscented Transformation with the parametrizion from E. A. Wan & 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 that computation of the Jacobian matrix. However, deriving the Jacobian matrix for the models in this package is not difficult

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:

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

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 & 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:

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

That is, we use the closed form solution. This version is both 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 $\widehat{\boldsymbol{a}}_0, \widehat{\boldsymbol{a}}_1, \dots, \widehat{\boldsymbol{a}}_{2q+1}$ according to:

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

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):

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

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

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

The correction step is then:

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

Re-writting

The above formulation has the drawback that we have to invert $\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}$ which is infeasible when the number of observations is large. We can re-write the correction step above by using the Woodbury matrix identity to get algorithm $O(n_t)$ instead of $O(n_t^3)$ where $n_t = |R_t|$ is the number of elements of the risk set. In other words, the new formulation is linear in time complexity with the dimension of the observational equation

The correction step can be computed as:

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

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.

The Square-root Unscented Kalman filter

Another idea could be to try the Square-root Unscented Kalman filter suggested in Merwe & Wan (2001). The idea is to use a QR decompositions and Cholesky updates to get a faster method and more stable method. While Merwe & 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} \\
&= \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 decompisition of $\mathbf{C}$ is $O((2q+1+n_t)n_t^2)$ as stated in Merwe & 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 postive. 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 reduce 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

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:

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

The addition extreme observation / outliers less influential

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 & Uhlmann (1997) state is 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 list element `kappa` in the list passed as the `control` argument to `ddhazard`. For example, `control = list(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 & Hendeby (2012) and Julier & 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 altered through the `alpha` element of the list passed to the argument `control` of `ddhazard`.

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 & 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 altered through the `beta` element of list passed to the argument `control` of `ddhazard`.

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 $b < q + 1$ sigma point is $\sqrt{q+\lambda}\sqrt{v}x_{ijb}$ where x_{ijb} 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_{ijb} 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.

Sequential approximation of the posterior mode (SMA)

Another idea is do sequential rank-one approximations of the posterior modes. This section will go into 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 would ideally like to 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 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 simplifies to a one dimensional problem of finding the constant $v \in \mathbb{R}$ that minimize:

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

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

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

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

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

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

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

This method is selected by setting `method = "SMA"` in the list passed to the `control` argument of `ddhazard`

Implementation

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

1. $-(y_{ijt} \log h(b) + (1 - y_{ijt}) \log(1 - h(b)))$ is convex in b
2. $-(y_{ijt} \log h(b) + (1 - y_{ijt}) \log(1 - h(b)))$ bounded from below

Number 1 and 2 are true for the binomial model and true for all exponential families where the form though will change. That is, we have to change the likelihood expression of $-\log P\left(y_{ijt}|\,\boldsymbol{\alpha}\right)$ which leads to $-\left(y_{ijt}\log h(b)+(1-y_{ijt})\log(1-h(b))\right)$ in the binary case.

As an example, point 2 is true for the logistic model since for $y_{ijt}=1$ the limit is 0 at $b\to\infty$ and for $y_{ijt}=0$ the limit is 0 for $b\to-\infty$. The convex polynomial will eventually dominate the slope of the logit function which tends towards zero. Thus, we get a unique finite minimum. 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:

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

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

0. 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 $k=1,2,\dots,n_t$:
1. Perform step 1 and 2 as before to find the constant v and update $\boldsymbol{a}_{t|t}^{(k)}$
2. Update $\mathbf{L}$ by a rank-one-update of $\boldsymbol{x}_{ij}g\boldsymbol{x}_{ij}^\top$ such that $\mathbf{L}\mathbf{L}^\top\leftarrow\mathbf{L}\mathbf{L}^\top+\boldsymbol{x}_{ij}g\boldsymbol{x}_{ij}^\top$. We use the left arrow, $\leftarrow$, to indicate that we make an update
3. Set $\mathbf{V}_{t|t}^{(k)}=\left(\mathbf{L}^{-1}\right)^\top\left(\mathbf{L}^{-1}\right)$

Step 0 comes at an $O(q^3)$ cost per interval $1,2,\dots,d$ due to the Cholesky decomposition. This is durable 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). It is the algorithm mentioned in Golub & Van Loan (2012) in section 6.5.4. Step 3 can be done in $O(q^2)$ using backward substitution when we compute $\left(\mathbf{L}^{-1}\right)^\top$. Thus, we end with an algorithm that scales as well as with the Woodbury matrix identity a part from the first step 0

Moreover, we can save computations throughout by storing $\widetilde{\mathbf{L}}=\left(\mathbf{L}^{-1}\right)^\top$ instead of storing $\mathbf{V}_{t|t}^{(k)}$. $\widetilde{\mathbf{L}}$ 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 $\boldsymbol{x}_{ij}\boldsymbol{x}_{ij}^\top$ 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 element `posterior_version = cholesky` in the list passed to the `control` argument of `ddhazard`. The former method using the Woodbury matrix identity is used if you set `posterior_version = woodbury`

Pros and cons

The pros of the SMA presented in this section are:

Better approximation	We may have a better approximation of the posterior mode because we avoid the linerization error of the EKF under the expectation operator
Cont. model	We avoid having the various definition of the continuous time model we introduce later. This is true since we directly maximize the likelihood and do not need residual outcome in the correction step as we do with the EKF and UKF
Scalability	This method is also $O(n_t)$

The cons are:

Sequential	The updates are sequential and hence cannot be done in parallel as the EKF. Further, the gain of doing the matrix and vector operations is minor due to the low dimensionality compared to the UKF
Ordering	The final outcome will depend on how we order the risk sets. Thus, the current implementation permute each risk set once with this method by default. You can switch this off by setting `permu = FALSE` to the `control` argument of `ddhazard`

Global approximation of the posterior mode (GMA)

We can directly minimize:

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

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 iteratively reweighted ridge regression. 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:

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

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

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

The box below shows the final algorithm for the correction step:

Algorithm 1 Correction step with global mode approximation by Newton Raphson.

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

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

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

repeat

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

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

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 $k \leftarrow k + 1$

This method is selected by setting `method = "GMA"` in the list passed to the `control` argument of `ddhazard`. You can change $k_{\max}$ and ϵ by respectively setting the elements `GMA_max_rep` and `GMA_NR_eps` to the `control` argument. 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. c' and c'' can be computed in parallel though this is not implemented at this point. Further, building with a multithreaded BLAS can decrease the computation time of $\mathbf{X}_t^\top c''(\boldsymbol{\alpha})\mathbf{X}_t$ along with the other matrix and vector products. We can make an implementation like the EKF which will reduce the memory requirement as we avoid storing the matrix $\mathbf{X}_t$

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:

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

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

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 $\boldsymbol{\alpha}^{(k)} = \left(\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\mathbf{X}}_t\right)^{-1}\widetilde{\mathbf{X}}_t^\top\widetilde{\mathbf{C}}\widetilde{\boldsymbol{b}}$. This problem can be solved with methods for least squares problem. Though, this is the EKF shown earlier. To see this, set $\boldsymbol{a}^{(k-1)} = \boldsymbol{a}_{t|t-1}$. Then,

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

Further,

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

Thus,

$$\left(\widetilde{\mathbf{X}}_t^\top \widetilde{\mathbf{C}} \widetilde{\mathbf{X}}_t\right)^{-1} \widetilde{\mathbf{X}}_t^\top \widetilde{\mathbf{C}} \widetilde{\boldsymbol{b}} = \boldsymbol{a}_{t|t-1} + \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}_{t|t-1})\right)^{-1} \boldsymbol{u}_t(\boldsymbol{a}_{t|t-1}) \tag{3}$$

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

$$\left(\widetilde{\mathbf{X}}_t^\top \widetilde{\mathbf{C}} \widetilde{\mathbf{X}}_t\right)^{-1} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}^{(k-1)})\right)^{-1} \tag{4}$$

$$\begin{aligned} \widetilde{\mathbf{X}}_t^\top \widetilde{\mathbf{C}} \widetilde{\boldsymbol{b}} &= \mathbf{X}_t^T \boldsymbol{h}' \left(\mathbf{X}_t \boldsymbol{a}^{(k-1)}\right)^2 \operatorname{Var}\left(\boldsymbol{Y}_t | \, \mathbf{X}_t \boldsymbol{\alpha}^{(k-1)}\right)^{-1} \mathbf{X}_t \boldsymbol{a}^{(k-1)} + \boldsymbol{u}_t(\boldsymbol{a}^{(k-1)}) + \mathbf{V}_{t|t-1}^{-1} \boldsymbol{a}_{t|t-1} \\ &= \mathbf{U}_t(\boldsymbol{a}^{(k-1)}) \boldsymbol{a}^{(k-1)} + \mathbf{V}_{t|t-1}^{-1} \boldsymbol{a}_{t|t-1} + \boldsymbol{u}_t(\boldsymbol{a}^{(k-1)}) \end{aligned} \tag{5}$$

$$\boldsymbol{a}^{(k)} = \left(\mathbf{V}_{t|t-1}^{-1} + \mathbf{U}_t(\boldsymbol{a}^{(k-1)})\right)^{-1} \left(\mathbf{U}_t(\boldsymbol{a}^{(k-1)}) \boldsymbol{a}^{(k-1)} + \mathbf{V}_{t|t-1}^{-1} \boldsymbol{a}_{t|t-1} + \boldsymbol{u}_t(\boldsymbol{a}^{(k-1)})\right) \tag{6}$$

The above may give ideas to different implementation of the EKF that are more numerically stable and/or where we can easily check for singularity. A review of numerical method for least square problems and numerical examples is in e.g. O'Leary (1990)

Pros and cons

The pros of the GMA presented in this section are:

Better approximation	We may have a better approximation of the posterior mode because we avoid the linerization error of the EKF under the expectation operator
Cont. model	We avoid having the various definition of the continuous time model we introduce later. This is true since we directly maximize the likelihood and do not need residual outcome in the correction step as we do with the EKF and UKF
Scalability	This method is also $O(n_t)$
Global	Performs global updates without effect of ordering of observations
Parallel	Can be computed in parallel

The cons are:

$\mathbf{Q}_0$	Sensitive to the choice of $\mathbf{Q}_0$ for some data sets
Iterative	May require more computations than the SMA. The computation time may be offset by the use of BLAS for matrix and vector product and parallel computation

Weights

Weights can be used in the EKF, UKF and posterior mode approximation. 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 $\boldsymbol{e}_t = (e_1, e_2, \dots, e_{n_t})$ 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

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

$$\boldsymbol{u}_t(\boldsymbol{\alpha}_t) = \sum_{f=1}^{n_t} e_f \boldsymbol{u}_{(R_t)_f t}(\boldsymbol{\alpha}_t), \qquad \mathbf{U}_t(\boldsymbol{\alpha}_t) = \sum_{f=1}^{n_t} e_f \mathbf{U}_{(R_t)_f t}(\boldsymbol{\alpha}_t)$$

where $(R_t)_f$ is the f'th element of R_t

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

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

SMA

Weights are handled in the SMA by replacing:

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

with

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

I.e. we change the weight on the likelihood term and scale the second derivative

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.

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

Estimation in the E-step

The fixed effect can be estimated in the E-step in a similar manner to Harvey & Phillips (1979). The method in Harvey & 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 $\mathbf{Q}_0$ (anything greater than 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 altered by setting the `Q_0_term_for_fixed_E_step` of the list passed to the `control` argument of `ddhazard`. Moreover, this method to estimate the fixed effect is used when you set the `fixed_terms_method = "E_step"` in the list passed to the `control` argument

Estimation in the M-step

We start be re-stating the log likelihood and introducing new notation in the EM-algorithm. We need the new notation to find the M-step for the model with fixed effects that are estimated in the M-step. The log likelihood up to a normalization constant is:

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

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

We perform the E-step by approximately integrating out the latent variables $\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d$ conditional on $\mathbf{Q}_0$ and the current estimates of $\mathbf{Q}^{(k-1)}$ and $\boldsymbol{a}_0^{(k-1)}$:

$$\begin{aligned}\widetilde{E}_k\left(\mathcal{L}\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right)\right) &= E\left(\mathcal{L}\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right) | \mathbf{Q}_0, \mathbf{Q}^{(k-1)}, \boldsymbol{a}_0^{(k-1)}\right) \\ &= \int_{\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d} \mathcal{L}\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right) f_{\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_d; \mathbf{Q}_0, \mathbf{Q}^{(k-1)}, \boldsymbol{a}_0^{(k-1)}) d\boldsymbol{x}_0 \cdots d\boldsymbol{x}_d\end{aligned}$$

where $f_{\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d}(\cdot; \mathbf{Q}_0, \mathbf{Q}^{(k-1)}, \boldsymbol{a}_0^{(k-1)})$ is the conditional density function of the latent variables $\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d$ given $\mathbf{Q}_0, \mathbf{Q}^{(k-1)}, \boldsymbol{a}_0^{(k-1)}$. The resulting expected likelihood can be summarized by the conditional means, $\boldsymbol{a}_{0|d}^{(k)}, \dots, \boldsymbol{a}_{d|d}^{(k)}$, covariance matrices $\mathbf{V}_{0|d}^{(k)}, \dots, \mathbf{V}_{d|d}^{(k)}$ and matrices $\mathbf{B}_1^{(k)}, \dots, \mathbf{B}_d^{(k)}$ when we update $\mathbf{Q}^{(k)}$ and $\boldsymbol{a}_0^{(k)}$

Notice that the entries in $\boldsymbol{\alpha}_0$, $\mathbf{Q}$ and $\mathbf{Q}_0$ only appears in the first three lines of the log likelihood $\mathcal{L}\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right)$. Hence, we only need these three sets of terms to update $\mathbf{Q}^{(k)}$ and $\boldsymbol{a}_0^{(k)}$. To stress this point, the conditional likelihood in the M-step is:

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

Suppose now that we assume that some of the effects are fixed such that we replace the linear predictor $\boldsymbol{x}_{it}^\top\boldsymbol{\alpha}_t$ by $\widetilde{\boldsymbol{x}}_{ij}^\top\widetilde{\boldsymbol{\alpha}}_t + \bar{\boldsymbol{x}}_{ij}^\top\boldsymbol{\gamma}$ where $\boldsymbol{\gamma}$ is the fixed coefficients and $\bar{\boldsymbol{x}}_{it}$ are the corresponding covariates. The new definition of l_{ijt} is:

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

Suppose that we fix $\boldsymbol{\gamma}^{(k-1)}$ doing the E-step and estimate $\boldsymbol{\gamma}^{(k)}$ doing the M-step. Then the new expected log likelihood is:

$$\widetilde{E}_k\left(\mathcal{L}\left(\widetilde{\boldsymbol{\alpha}}_0, \dots, \widetilde{\boldsymbol{\alpha}}_d\right)\right) = E\left(\mathcal{L}\left(\widetilde{\boldsymbol{\alpha}}_0, \dots, \widetilde{\boldsymbol{\alpha}}_d\right) | \mathbf{Q}_0, \mathbf{Q}^{(k-1)}, \widetilde{\boldsymbol{a}}_0^{(k-1)}, \boldsymbol{\gamma}^{(k-1)}\right)$$

We observe that:

1. The $\bar{\boldsymbol{x}}_{ij}^\top\boldsymbol{\gamma}^{(k-1)}$ term acts like offsets in the E-step where $\boldsymbol{\gamma}^{(k-1)}$ is fixed. Thus, we only need to add these offsets to the linear predictors
2. $\boldsymbol{\gamma}^{(k)}$ is estimated separately from $\widetilde{\boldsymbol{\alpha}}_0^{(k)}$ and $\mathbf{Q}^{(k)}$ in the M-step. Thus, no changes are needed in the update formulas for $\mathbf{Q}^{(k)}$ and $\widetilde{\boldsymbol{\alpha}}_0^{(k)}$

However, the update of $\boldsymbol{\gamma}^{(k)}$ requires that we optimize

$$\widetilde{E}_k\left(\sum_{t=1}^{d}\sum_{(i,j)\in R_t} l_{ijt}(\widetilde{\boldsymbol{\alpha}}_t, \boldsymbol{\gamma})\right)$$

with respect to $\boldsymbol{\gamma}$. The update formulas are not as simple as for $\widetilde{\boldsymbol{\alpha}}_0^{(k)}$ and $\mathbf{Q}^{(k)}$ as the terms of l_{ijt} are non-linear in the time-varying effects $\widetilde{\boldsymbol{\alpha}}_0 \dots, \widetilde{\boldsymbol{\alpha}}_d$. A simple way to overcome this is to make a zero order Taylor expansion around the mean estimates $\widetilde{\boldsymbol{a}}_{0|d}^{(k)}, \dots, \widetilde{\boldsymbol{a}}_{d|d}^{(k)}$:

$$\widetilde{E}_k \left(\sum_{t=1}^{d} \sum_{(i,j) \in R_t} l_{ijt}(\widetilde{\boldsymbol{\alpha}}_t, \boldsymbol{\gamma}) \right) \approx \sum_{t=1}^{d} \sum_{(i,j) \in R_t} l_{ijt}(\widetilde{\boldsymbol{a}}_{t|d}^{(k)}, \boldsymbol{\gamma})$$

This expansion coincides with a first order Taylor expansion as the first order terms are zero. The advantages are:

3. $\widetilde{\boldsymbol{x}}_{ij}^\top \widetilde{\boldsymbol{a}}_t$ acts like offsets in the M-step when we estimate $\boldsymbol{\gamma}^{(k)}$
4. $\boldsymbol{\gamma}^{(k)}$ is estimated in the M-step as a generalized linear model with offsets for distributions from the exponential family. Weights are treated as in typical generalized linear models

The fixed effects will be estimated in the M-step when you set `fixed_terms_method = M_step` in the list passed to the `control` argument

Implementation

Point number 4 above implies that we can use a typical Newton Raphson algorithm to update the estimate of $\boldsymbol{\gamma}$ when we are using a distribution from the exponential family. This can be solved by a QR decomposition as done in `glm`. However, point 3 implies that every observation will have a different offset in every time interval the observation is in. Thus, we can end with a large design matrix

To overcome the potential memory issue this can cause, this package use the same Fortran function that the `bigglm` function in the `biglm` package uses. The Fortran function recursively performs a QR update for each row in the design matrix. Hence, we do not need to store the entire design matrix at any given point. The Fortran code is described in Miller (1992) and written by Miller. It is an updated version of the algorithm described in Gentleman (1972) which has a time complexity of $O(|\boldsymbol{\gamma}|^2)$ for the QR-update of each row in the design matrix
The M-step recursively updates the $\boldsymbol{\gamma}$ starting with the previous estimated value. The estimation stops when $\|\boldsymbol{\gamma}^{(k)} - \boldsymbol{\gamma}^{(k-1)}\| / (\|\boldsymbol{\gamma}^{(k-1)}\| + \delta) < \epsilon$ where superscripts denote the iteration number, ϵ is the tolerance and δ is a small number. ϵ can be changed by setting `eps_fixed_params` element of the list passed to the `control` argument of `ddhazard`.

The estimation will stop if the criteria given by ϵ is not meet within a given number of iterations. The maximum number of iterations can be set by setting the `max_it_fixed_params` element of the `control` argument to `ddhazard`. The user is warned if the criteria is not meet within `max_it_fixed_params` iterations.

Surely, other methods to solve the QR problem or fit a generalized linear model could be used that does not require us to store the entire design matrix and are faster and/or more stable. An example could be the algorithm described in Hammarling & Lucas (2008). The current method is used since it has shown to work well in the `bigglm` function and as we assume that few parameters will be fixed. Thus, the $O(|\boldsymbol{\gamma}|^2)$ cost of doing the M-step should not be an issue. Other options are for example stochastic gradient descent methods or methods from Online learning.

Other options

Another option is to use higher order expansions of $\widetilde{E}_k \left(\sum_{t=1}^{d} \sum_{(i,j) \in R_t} l_{ijt}(\widetilde{\boldsymbol{\alpha}}_t) \right)$, approximate the expectation with an MCMC method using the conditional means $\widetilde{\boldsymbol{a}}_{0|d}^{(k)}, \dots, \widetilde{\boldsymbol{a}}_{d|d}^{(k)}$ and conditional covariance matrices $\mathbf{V}_{0|d}^{(k)}, \dots, \mathbf{V}_{d|d}^{(k)}$, or any other method to approximate $\widetilde{E}_k \left(\sum_{t=1}^{d} \sum_{(i,j) \in R_t} l_{ijt}(\widetilde{\boldsymbol{\alpha}}_t) \right)$. At this point, the zero order Taylor expansion is the only implemented method to estimate $\boldsymbol{\gamma}$ in the M-step

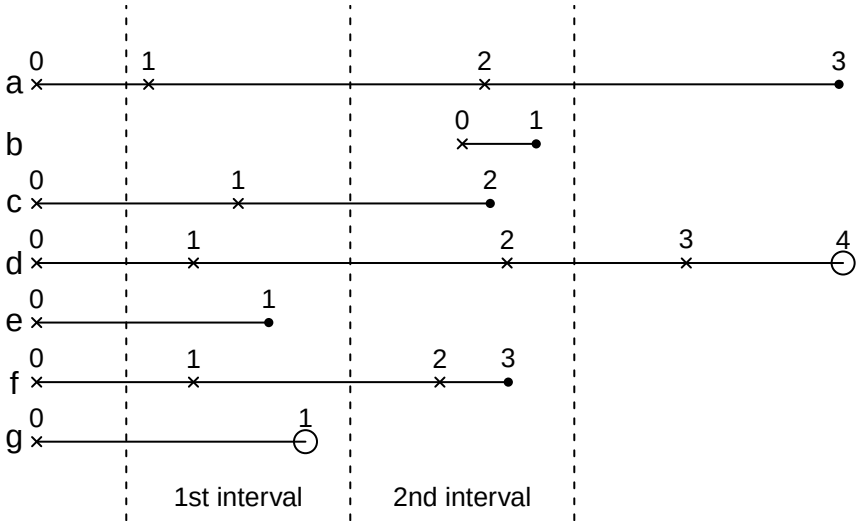


Figure 1: Illustration of going from event times to binary variables. Each horizontal line represents an individual. A cross indicates that new covariates are observed while a filled circle indicates that the individual have died. Open circles indicates that the individual is right censored. Vertical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

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.

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. It is important to stress that the logistic model yields similar estimates as a to Generalized Additive model as shown in the vignette *Comparing methods for time-varying logistic models* and *Simulation study with logit model*. Consequently, it is a valid alternative

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 intervals. 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 will count 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 is 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

Continuous time model

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

Assumptions

We make the following assumption in the continuous time model:

1. Coefficients (that is state variables $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_d$) change at the end of time intervals
2. The individuals covariates change at discrete times
3. We have piecewise constant instantaneous hazards given by $\exp(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t)$ given an individual's current co-covariate vector $\boldsymbol{x}_{ij}$ 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 up to a normalization constant is:

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

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

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

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:

$$\mathcal{R}_t = \left\{(i,j) \in \mathbb{Z}_+^2 : \, t_{i,j-1} < t \wedge t_{ij} > t - 1\right\}$$

In words, this 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. When we use the continuous time model we change the risk sets R_t and the likelihoods:

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

respectively to $\mathcal{R}_t$ and the new expression for l_{ijt} in all the previous part of this vignette . The following sections will introduce four methods to estimate the above model. They are:

1. Using a right clipped time variable
2. Using a binary variable
3. Using a right clipped time variable with a jump term
4. Combining method 1. and 2.

The methods with the binary outcome and the right clipped time variable with a jump term seem to do best on simulated data. It is needed to define clipping before we proceed since the term clipping is not commonly used as far as I am aware. By right clipping a random variable b at v we mean that the clipped variable $\tilde{b}$ is:

$$\tilde{b} = \begin{cases} b & b \leq v \\ c & b > v \end{cases}$$

Right clipped observations time

We start by defining the clipped observation time Δ_{ijt}:

$$\begin{aligned}
\Delta_{ijt} &= T_i - \max\{t_{i,j-1}, t-1\} + \left(\min\{t_{ij}, t\} - T_i\right)1_{\{T_i \geq \min\{t_{ij}, t\}\}} \\
&= \begin{cases} T_i - \max\{t_{i,j-1}, t-1\} & T_i \leq \min\{t_{ij}, t\} \\ \delta_{ijt} & T_i > \min\{t_{ij}, t\} \end{cases}
\end{aligned}$$

where $\delta_{ijt} = \min\{t_{ij}, t\} - \max\{t_{i,j-1}, t-1\}$ is the maximum length the stop time can take for observation j of individual i in interval t. Further, we omit the case where $T_i < \max\{t_{i,j-1}, t-1\}$ as this situation will not happens in the conditional means we look at. These time variables are connected to the stop time T_i as follows. Suppose that all the interval has length one such that $t_{i,t} - t_{i,t-1} = 1$. Then:

$$\begin{aligned}
P(T_i = t) &= P(T_i > 1) P(T_i > 2 | T_i > 1) \cdots P(T_i = t | T_i > \lceil t \rceil - 1) \\
&= P(\Delta_{i11} = 1) P(\Delta_{i22} = 1 | \Delta_{i11} = 1) \cdots P\left(\Delta_{i\lceil t \rceil \lceil t \rceil} = t - (\lceil t \rceil - 1) \middle| \Delta_{i,\lceil t \rceil - 1, \lceil t \rceil - 1} = 1\right) \\
P(T_i > t) &= P(T_i > 1) P(T_i > 2 | T_i > 1) \cdots P(T_i > t | T_i > \lceil t \rceil - 1) \\
&= P(\Delta_{i11} = 1) P(\Delta_{i22} = 1 | \Delta_{i11} = 1) \cdots P\left(\Delta_{i\lceil t \rceil \lceil t \rceil} > t - (\lceil t \rceil - 1) \middle| \Delta_{i,\lceil t \rceil - 1, \lceil t \rceil - 1} = 1\right)
\end{aligned}$$

The conditional probabilities simplifies into separate terms in the log likelihood due to the memoryless property of the exponential distribution. Thus, computing the conditional mean, h, can be done as follows. Assume for simplicity of notation that the observation $(t_{i,j-1}, t_{i,j}]$ has at most length one and is inside a time interval such that $\lceil t_{i,j} \rceil - 1 = \lfloor t_{i,j-1} \rfloor$ where $\lceil \cdot \rceil$ is the ceiling function and $\lfloor \cdot \rfloor$ is the floor function. Then:

$$\begin{aligned}
\tilde{z}(\boldsymbol{\alpha}) &= E\left(\Delta_{ijt} | \Delta_{i,j-1,\lceil t_{i,j-1} \rceil} = \delta_{i,j-1,\lceil t_{i,j-1} \rceil} \wedge \boldsymbol{\alpha}_{\lceil t_{ij} \rceil} = \boldsymbol{\alpha}_t\right) = h^{\Delta}_{ft}(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t) \\
&= (\bar{t}_{ij} - t_{i,j-1}) P(\tilde{T} \geq \bar{t}_{ij} - t_{i,j-1}) + \int_0^{\bar{t}_{ij} - t_{i,j-1}} r f_{\tilde{T}}(r) \mathrm{d}r, \qquad \bar{t}_{ij} = \lceil t_{ij} \rceil y_{ijt} + (1 - y_{ijt}) t_{ij}
\end{aligned}$$

where f is the index of observational equation at time t matching with individual i's j'th covariate vector, $\lceil t_{ij} \rceil = s$ is the time interval number that the observation is in, $\tilde{T} \sim \text{Exp}\left(\exp(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t)\right)$, $f_{\tilde{T}}$ is the density function of $\tilde{T}$ and h^{Δ}_{fj} is the inverse link function for the right clipped time variables for individual i's j'th observation. Set $\lambda = \exp(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t)$ and $\delta = \delta_{ijt}$. The resulting conditional mean is:

$$h^{\Delta}_{ft}(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t) = \frac{1 - \exp(-\lambda \delta)}{\lambda}$$

Moreover, we can show that the variance is:

$$\begin{aligned}
&\text{Var}\left(\Delta_{ijt} | \Delta_{i,j-1,\lceil t_{i,j-1} \rceil} = \delta_{i,j-1,\lceil t_{i,j-1} \rceil} \wedge \boldsymbol{\alpha}_{\lceil t_{ij} \rceil} = \boldsymbol{\alpha}_t\right) \\
&= \frac{1 - \exp(-2\delta\lambda) - 2\lambda\delta \exp(-\delta\lambda)}{\lambda^2}
\end{aligned}$$

We call these variables right clipped observation times because each Δ_{ijs} is a right clipped exponential variable conditional that individual i has survived up to time $t_{i,j-1}$. A point has to be made about how we compute the mean and the variance in the case of right censoring and in the case of death in the previous equations. Say that individual i has v covariates. Right censoring is treated by having $t_{iv} < \lceil t_{iv} \rceil$. We only know that the individual survived up to time t_{iv} and do not know if he survived the entire period (which is $\lceil t_{iv} \rceil$)

Further, we round up in the case of a death, $T_i = t_{iv}$, when we compute the mean and variance in the correction step of the filter (EKF or UKF). If we do not make the this adjustment then there is no difference in the model between a death, right censoring, new covariates or change of time intervals. Consequently, the state vectors will tend towards values such that the linear predictors goes to minus infinity

We will make a the following example to show the state vector will tend towards values such that the linear predictors goes to minus infinity. Suppose that we use the UKF and we only have an intercept such that $\boldsymbol{x}_{ij} = \boldsymbol{x} = (1)$ for all i and j. Further, we use a first order random walk such that $\boldsymbol{\alpha}_t = (b_t)$. This implies that all the linear predictors are given by $\boldsymbol{x}^\top \boldsymbol{\alpha}_t = b_t$. The predicted mean for given observation Δ_{ijt} can be $\delta_{ijt} = \min\{t_{ij}, t\} - \max\{t_{i,j-1}, t-1\}$ at most since we do not round up. Further, the predicted outcome will tend towards δ_{ijt} as the linear predictor, b_j, tend towards minus infinity.

Since the predicted mean can at most be δ_{ijt}, all the residuals in the correction step, $\boldsymbol{y}_t - \overline{\boldsymbol{y}}$, will be positive. Recall that the correction step in the UKF is:

$$b_t = \boldsymbol{a}_{t|t} = \boldsymbol{a}_{t|t-1} + \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1}(\boldsymbol{y}_t - \overline{\boldsymbol{y}}) = b_{t-1} + \mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1}(\boldsymbol{y}_t - \overline{\boldsymbol{y}})$$

where we can show that $\mathbf{P}_{\boldsymbol{\alpha}_t,\boldsymbol{y}_t}\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t}^{-1}$ (which is a scalar) is negative in the example given here. Hence, $b_1, b_2, \dots$ will decrease in every iterations of the UKF. On the other hand, if we do round up in the case of death then the residuals, $\boldsymbol{y}_t - \overline{\boldsymbol{y}}$, can be negative in the case of death. Consequently, a death can increase b_t value if the residual is negative. Another way to reason about this is that the individual could have survived the entire time period, $\lceil t_{ij} \rceil$, but only survived up to time t_{ij}. Thus, we use $\bar{t}_{ij}$

A draw back to this model is that it will not work if the reported time scale is coarse as we cannot distinguish between a change of time interval, new covariates vector, right censoring or death when the times coincides. As an extreme example, we cannot use this method if all times are reported on the grid of integers $1, 2, \dots$ and we use time intervals of length 1. You can estimate with the right clipped time method by setting the argument `model = "exp_clip_time"` in the call to `ddhazard`

Binary outcome

The next method is to replace the likelihood with binary variables y_{ijt} as the outcome. Then the likelihood given data has:

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

where the inverse link function is the inverse cloglog function. That is,

$$h_{ft}^Y(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t) = 1 - \exp\left(-\exp(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t) \left(\min\{t, t_{ij}\} - \max\{t-1, t_{i,j-1}\}\right)\right)$$

We omit the details of mapping between the indices of the observational equation, f, the indices of the individuals, (i, j). In the following, we assume that the observation $(t_{i,j-1}, t_{i,j}]$ has at most length one and is inside a time interval such that $\lceil t_{i,j} \rceil - 1 = \lfloor t_{i,j-1} \rfloor$

There are two points to be made here. Firstly, we do not round up by using $\bar{t}_{ij}$ in place of t_{ij} when we have a death. Thus, we do take into account the moment the person dies at in that $\log h_{ft}^Y(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t)$ is the likelihood of dying sometime in the period δ_{ijt} given that he survived up to $t_{i,j-1}$. Moreover, an individual can have multiple observations in the same time interval if he does not die but his covariate vectors change within the interval in the state space model. Thus, we do not have issue with going from event times to binary outcomes as with the logistic model

The con is that the term $\log h_{ft}^Y(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t)$ is only the log likelihood of dying *sometime* in the period δ_{ijt}. It is not the likelihood of dying after *exactly* δ_{ijt}. We do not loose this information with the right clipped observation time. This may be a minor drawback in settings where we have interval censoring. Here the exact time of death may be unknown and the given stop time is an upper bound for the death time. You can estimate with the binary outcome method by setting the argument `model = "exp_bin"` in the call to `ddhazard`

Right clipped observations time with jump

This section will cover the right clipped time with a jump term. The motivation is that the two previous methods are not flawless: the binary method has the drawback that we do not keep the information about the exact time of death and the right clipped time method cannot distinguish some deaths from censoring with a coarse time scale. A way to deal with the latter is to add a jump term in case of censoring. Particularly, we

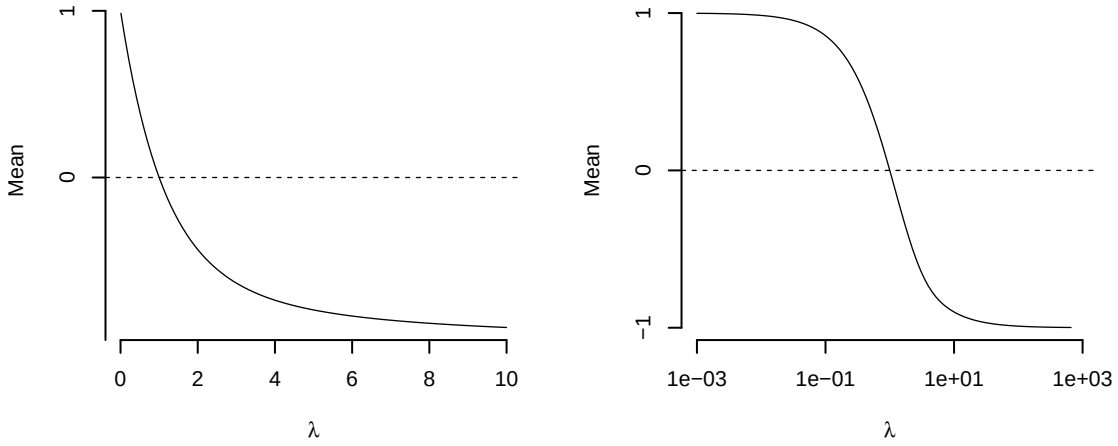
set:

$$\begin{aligned}\Lambda_{ijt} &= \delta_{ijt} 1_{\{T_i > \min\{t_{ij}, t\}\}} + (T_i - \max\{t_{i,j-1}, t-1\} - \delta_{ijt}) 1_{\{T_i \leq \min\{t_{ij}, t\}\}} \\ &= \delta_{ijt} 1_{\{T_i > \min\{t_{ij}, t\}\}} + (T_i - \min\{t_{ij}, t\}) 1_{\{T_i \leq \min\{t_{ij}, t\}\}} \\ &= \begin{cases} T_i - \min\{t_{ij}, t\} & T_i \leq \min\{t_{ij}, t\} \\ \delta_{ijt} & T_i > \min\{t_{ij}, t\} \end{cases}\end{aligned}$$

where we use Λ instead of Δ to distinguish between the two types of right clipped values. It is key to notice that we use $T_i - \min\{t_{ij}, t\}$ when $T_i \leq \min\{t_{ij}, t\}$ instead of $T_i - \max\{t_{i,j-1}, t-1\}$. Thus, the value is negative if an individual dies. The above implies that $\Lambda_{ijt} \in [-\delta_{ijt}, 0] \cup \{\delta_{ijt}\}$ conditional that the individual have survived up to time $\max\{t_{i,j-1}, t-1\}$. $\Lambda_{ijt} \leq 0$ implies that we have an event in interval t. Further, the smaller the value the sooner the event happened. $\Lambda_{ijt} = \delta_{ijt}$ implies that the variable is right clipped. The mean is given by:

$$h^{\Lambda}_{fs}(\boldsymbol{\alpha}) = \frac{(1 - \exp(-\delta\lambda))(\delta\lambda - 1)}{\lambda}$$

where the definition of δ and λ is the same as in the right clipped time variable section. I.e. we are looking at a particular observation for individual i and define $\lambda = \exp(\boldsymbol{x}_{ij}^\top \boldsymbol{\alpha}_t)$ and $\delta = \delta_{ijt}$. The mean is plotted below as a function of λ with $\delta = 1$



Lastly, the variance is given by:

$$\begin{aligned}&\operatorname{Var}\left(\Lambda_{i,s} \middle| \Lambda_{i,j-1} = \delta_{i,j-1} \wedge \boldsymbol{\alpha}_{\lceil t_{i,s} \rceil} = \boldsymbol{\alpha}, \boldsymbol{\alpha}_{\lceil t_{i,s} \rceil - 1}, \ldots, \boldsymbol{\alpha}_0\right) \\ &\quad = \frac{(1 - \exp(-\delta\lambda))(1 - \delta\lambda)\left(\delta^2\lambda^2(3\exp(-\delta\lambda) - \exp(-2\delta\lambda)) + \delta\lambda(2\exp(-2\delta\lambda) - 4\exp(-\delta\lambda)) - \exp(-2\delta\lambda) + 1\right)^2}{\lambda^5}\end{aligned}$$

The at risk length is computed in the same way as the right clipped time variable. That is, we use $\bar{t}_{ij}$ instead of the min term in case of a death. You can estimate with the right clipped time variable by setting the argument `model = "exp_clip_time_w_jump"` in the call to `ddhazard`

Combining the first two

Another idea is to use the binary variable and the first mentioned right clipped time at the same time. Though, this method has shown worse performance on simulated data. Thus, the details of the method is omitted and this version is not supported since version 0.3.0.

Fixed effects

Fixed effects in the M-step are estimated using a Poisson model with an offset equal to the logarithm of the time observed in each time interval plus the estimated offset from time-varying effects. That is, we use that if an arrival time T is exponential distributed with rate λ then having an outcome at at time t is Poisson distributed $Y \sim \text{Poisson}(\lambda t)$. For example, say that we fit the following model:

```
# The data we use
head(data_frame)
```

id	tstop	tstart	y	x1	x2
1	0	2	0	0.13	0.12
1	2	3	0	0.95	1.48
1	3	4	1	0.18	-1.08
2	0	2	0	-0.44	-1.08
2	2	4	0	-0.55	0.15

```
# The fit
fit <- ddhazard(Surv(tstart, tstop, y) ~ x1 + ddFixed(x2), data_frame,
                by = 1, # time interval lengths are 1
                id = data_frame$id, model = "exponential")
```

Take the individual with `id = 1`. As in the logistic model, he will yield four observations in the M-step. Each will have an offset of $\log(1) = 0$ plus a term form `x1` because the interval length is 1 plus $\boldsymbol{a}_{t|d}$ times the value of `x1`. Say instead that the data frame was:

```
head(data_frame_new)
```

id	tstop	tstart	y	x1	x2
1	0.0	0.5	0	0.43	0.33
1	0.5	2.0	0	0.13	0.12
1	2.0	3.0	0	0.95	1.48
1	3.0	4.0	1	0.18	-1.08
2	0.0	2.0	0	-0.44	-1.08
2	2.0	4.0	0	-0.55	0.15

Then individual 1 will yield five observations. The first row would only has an offset of $\log 0.5$ plus $\boldsymbol{a}_{1|d}$ times 0.43. The second row will yield two observations: one with an offset of $\log 0.5$ plus $\boldsymbol{a}_{1|d}$ times 0.13 and the other with an offset of $\log 1$ plus $\boldsymbol{a}_{2|d}$ times 0.13

Diagnostics

This section will cover diagnostics tools. These includes:

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

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,

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

Then we could:

- Plot residuals against time and highlight the individuals with atleast on 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 argument `type = "pearson"`

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:

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

where $\mathbf{X}_t$ is the design matrix in interval t. The above could motivate the following matrix as the "hat-like" matrix in each interval:

$$\widetilde{\mathbf{H}}_t(\boldsymbol{\alpha}_{t|d})^{1/2}\mathbf{X}_t \left(\mathbf{X}_t^\top \widetilde{\mathbf{H}}_t(\boldsymbol{\alpha}_{t|d})^{-1}\mathbf{X}_t + \mathbf{V}_{t|d}^{-1}\right)^{-1} \mathbf{X}_t^\top \widetilde{\mathbf{H}}_t(\boldsymbol{\alpha}_{t|d})^{1/2}, \qquad \widetilde{\mathbf{H}}_t(\boldsymbol{\alpha}_{t|d}) = \mathbf{H}_t(\boldsymbol{\alpha}_{t|d}) + \xi\mathbf{I}$$

where we have used the final smoothed estimators and adjusted for the ξ factor as done in the algorithm. Plotting cumulative values versus time may show influential observations. You can get these estimates by calling `hatvalues` with a `ddhazard` fit

Residuals from the state vector

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

$$\widehat{\boldsymbol{\eta}}_t = \mathbf{R}^\top \left(\boldsymbol{a}_{t|d} - \mathbf{F}\boldsymbol{a}_{t-1|d}\right) \sim N\left(\mathbf{0}, \mathrm{Var}\left(\boldsymbol{\eta}_t \middle| \boldsymbol{Y}_d\right)\right)$$

This will require that we find the smoothed covariance matrix $\mathrm{Var}\left(\boldsymbol{\eta}_t \middle| \boldsymbol{Y}_d\right) = \mathbf{R}^\top \mathrm{Var}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1} \middle| \boldsymbol{Y}_d\right)\mathbf{R}$ 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:

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

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

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

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

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

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

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

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

Further tasks and ideas

The last section will cover further task and ideas. Please, let me know what you think. Is it relevant, got ideas to the question I pose and how would you priorities? What can make the package more useful for you?

Confidence bounds

How do we construct confidence bounds both for the state vectors and for the predicted values? Bootstrapping data seems to be the way forward given the use of a random walk

Tests

Tests of whether the effects are time-varying or not would be useful. One idea is to test entries in $\mathbf{Q}$. Though, this involves tests on the boundary of the parameter space. Another idea is to make an F-test. This thread here suggest the idea when make all the parameter time invariant http://stats.stackexchange.com/a/161917

Other state equations

We can replace the state equation with other models then a given order random walk. For example, we can replace it with a stationary process:

$$\boldsymbol{\alpha}_t = \boldsymbol{\mu} + \mathbf{F}\boldsymbol{\alpha}_{t-1} + \mathbf{R}\boldsymbol{\eta}_t$$

where we require $\mathbf{F}$ is such that the process is stationary. $\mathbf{F}$ and $\boldsymbol{\mu}$ can be estimated in the M-step with closed form solutions when the noise is Gaussian. Models of this type could be AR, MA, ARMA etc. for each of the coefficients. The EM-algorithm with linear constraints shown in the vignette "EM Derivation" of the `MARSS` package can be used for this purpose. We can show that an efficient implementation will be cubic in time complexity of the number of parameters to estimate. Lastly, we can change the distribution of $\boldsymbol{\eta}$ or change make a non-linear dependence between $\boldsymbol{\alpha}_t$ and $\boldsymbol{\alpha}_{t-1}$

Other observational models

The methods here could easily be generalized to other than binary outcome. For example we could use competing risk models as in Fahrmeir & Wagenpfeil (1996) or counting models. Extension to real variable outcomes in each period is also straight forward

Active learning

The methods and models could be used for active learning setting as in Lee & Roberts (2010). Though, this do require an update formula for data set to quickly update estimates once a new set of observations is observed. This update could easily be implemented if we do not update the estimates $\mathbf{Q}$ and $\boldsymbol{\alpha}_0$.

A further point in this connection is that computing upper bounds for the predicted outcome given an input variable is straight forward if we the predicted point-wise covariance matrix. Thus, the method could be applied in a bandit setting

Appendix

Notation

General notation

Subscripts $\cdot_i$	Indices for individuals or indices of for state equation
Subscripts $\cdot_j$ and $\cdot_f$	Indices to differ between quantities
Subscripts $\cdot_t$ and $\cdot_s$	Time period indices
superscript $\cdot^{(k)}$	k'th iteration estimate of a given quantity in an algorithm which depends on the given context
d	Number of intervals we observe

$\boldsymbol{\alpha}_t$	State vector at interval t
$\boldsymbol{\gamma}$	Fixed (time in-variant) coefficients
m	is the number of time-varying coefficients
$q = \dim(\boldsymbol{\alpha})$	Dimension of state vector
$n_t = \lvert R_t \rvert$	Number of elements of the risk set at time t
$\boldsymbol{x}_{ij}$	The i'th individuals j'th covariate vector which is valid in period $(t_{i,j-1}, t_{ij}]$. This will be augmented by zeros if we use the second order random walk
$\mathbf{X}_t$	Design matrix in interval t
T_i	Event time or censoring time for individual i
R_t	Risk sets in interval t for the discrete model
$\mathcal{R}_t$	Risk set in interval t for the continuous model
y_{ijt}	Indicator for whether individual i has an event with the j'th covariate vector in interval t
$\boldsymbol{y}_t$	Outcomes in interval t. Whether they are binary or not depends on the context
$\boldsymbol{z}_t(\boldsymbol{\alpha})$	Mean function in state equation at time t given state vector $\boldsymbol{\alpha}$. It implicitly depends on the covariate matrices and risk sets
$h_{it}(\boldsymbol{\alpha})$	The indices of of i'th index of $\boldsymbol{z}_t(\boldsymbol{\alpha})$. Dropping the subscript implies that the function that depends on a scalar (i.e. a linear predictor)
$\mathbf{H}_t(\boldsymbol{\alpha})$	Covariance matrix of observational equation in interval t given the state vector $\boldsymbol{\alpha}$. It implicitly depends on the covariate matrices and risk sets
$\mathbf{F}$	Transition matrix in the state equation
$\mathbf{R}$	Matrix mapping innovations in the state equation
$\mathbf{Q}, \mathbf{Q}_0$	Covariance matrices in state equation

$\boldsymbol{\eta}_t, \boldsymbol{\epsilon}_t$	Innovation/error terms of respectively the state and observational equation in interval t
ψ_t	Length of interval t
$\mathbf{E}_t, \boldsymbol{e}_t$	Diagonal matrix with weigths in interval t. The diagonal entries are $\boldsymbol{e}_t$. They implicitly depends on the covariate matrices and risk sets
ξ	Parameter to reduce the effect of outliers/extreme observations
ζ ζ_0	Learning rate and factor to decrease the learning rate respectively
$\boldsymbol{a}_{t\mid s}$	Estimated state vector $\boldsymbol{\alpha}_t$ using information up to time s
$\mathbf{V}_{t\mid s}$	Estimated state covariance matrix of $\boldsymbol{\alpha}_t$ using information up to time s
$\mathbf{B}_t$	Intermediate matrix used in EM algorithm
$\mathcal{L}(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d)$	Log-likelihood up to a normalization constant given state vectors $\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d$. It implicitly depends on the covariate matrices and risk sets
$l_{ijt}(\boldsymbol{\alpha}_t)$	The likelihood term from i'th individual with the j'th covariate vector in interval t given state vector $\boldsymbol{\alpha}$. It implicitly depends on the covariate matrices and risk sets
δ	Small constants which values differs between contexts
ϵ	Convergence threshold which values differs between contexts
v, b, g, $\boldsymbol{v}, \boldsymbol{b}, \widetilde{\boldsymbol{y}}, \boldsymbol{c}$, $\mathbf{A}, \mathbf{C}, \mathbf{K}, \mathbf{G}, \mathbf{L}$	Scalars, vectors and matrices which depend on the context
$\mathbb{Z}_+$	The natural numbers $1, 2, \dots$
δ_{ijs}	Maximal length individual i's observation with the j'th covariate can take in interval s
Δ_{ijs}	Right clipped outcome in the continuous time model in interval s for individual i with the j'th covariate vector
Λ_{ijs}	Right clipped outcome with jump in the continuous time model in interval s for individual i with the j'th covariate vector

Notation in EKF

$\boldsymbol{u}_t(\boldsymbol{\alpha})$	Score vector in correction step
$\mathbf{U}_t(\boldsymbol{\alpha})$	Information matrix in correction step

Notation in UKF

$\widehat{\boldsymbol{a}}_j$	j'th sigma point
$W_j^{[f]}$	j'th sigma weight of type f
$\alpha, \beta, \kappa, \lambda$	Hyperparameters
$\Delta\mathbf{K}$	Deviation of matrix $\mathbf{K}$ with $\mathbf{K}$ defined in the context
$\mathbf{P}_{\boldsymbol{a}_t, \boldsymbol{b}_t}$	The correlation between a vector $\boldsymbol{a}_t$ and a vector $\boldsymbol{b}_t$ using the information up to time t

References

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.

Fahrmeir, L., & Wagenpfeil, S. (1996). Smoothing hazard functions and time-varying effects in discrete duration and competing risks models. *Journal of the American Statistical Association*, *91*(436), 1584–1594.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22. https://doi.org/10.18637/jss.v033.i01

Gentleman, W. M. (1972). Basic procedures for large, sparse, or weighted linear least squares problems.

Golub, G. H., & Van Loan, C. F. (2012). *Matrix computations* (Vol. 4). JHU Press.

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

Hammarling, S., & Lucas, C. (2008). Updating the qr factorization and the least squares problem.

Harvey, A. C., & Phillips, G. D. (1979). Maximum likelihood estimation of regression models with autoregressive-moving average disturbances. *Biometrika*, *66*(1), 49–58.

Julier, S. J., & Uhlmann, J. K. (1997). New extension of the kalman filter to nonlinear systems. In *AeroSense'97* (pp. 182–193). International Society for Optics; Photonics.

Julier, S. J., & Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the*

IEEE, *92*(3), 401–422.

Lee, S. M., & Roberts, S. J. (2010). Sequential dynamic classification using latent variable models. *The Computer Journal*, *53*(9), 1415–1429.

Menegaz, H. M. T. (2016). Unscented kalman filtering on euclidean and riemannian manifolds.

Merwe, R. V. der, & Wan, E. A. (2001). 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)* (Vol. 6, pp. 3461–3464 vol.6). https://doi.org/10.1109/ICASSP.2001.940586

Miller, A. J. (1992). 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.

O'Leary, D. P. (1990). Robust regression computation using iteratively reweighted least squares. *SIAM Journal on Matrix Analysis and Applications*, *11*(3), 466–480. https://doi.org/10.1137/0611032

Seeger, M. (2004). *Low rank updates for the cholesky decomposition.*

Shumway, R. H., & Stoffer, D. S. (2006). Time series analysis and its applications: With r examples. *Springer Texts in Statistics.*

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.