ddhazard

Benjamin Christoffersen

2016-12-28

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@6d6885222287339bd524ddad0ce47058b3d9b094"
```

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

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

Why and when to use this package

The package is implemented 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 in 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
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 two 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 logit model.

Dynamic binary regression

We will introduce the model in the following paragraphs. Let $\boldsymbol{x}_{it}$ denote the co-variate vector for individual i at time t and let Y_{it} be the random variable for whether the i'th individual dies within time $(t-1,t]$. Another way to phrase this is whether the i'th individual dies in the t'th bin. Next, denote the parameters at time t by $\boldsymbol{\alpha}_t$. For given parameters at time t the probability of death is:

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

where 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:

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

$\boldsymbol{y}_t$ is the vector of the binary outcomes and the associated equation is the *observational equation*. $\sim (a,b)$ denotes a random variable(s) with mean (vector) a 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*. ψ_t is the length of the bin interval number t. Thus, $\psi_i=\psi$ when we use equidistant bins which is the only option at this point. Further, we define the observational equations covariance matrix as $\mathbf{H}_t(\boldsymbol{\alpha}_t)=\text{Var}\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t\right)$

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

$$\begin{aligned} z_{it}(\boldsymbol{\alpha}_t) &= E\left(Y_{it} \middle| \boldsymbol{\alpha}_t\right) = h(\boldsymbol{\alpha}_t^T\boldsymbol{x}_{it}) \\ H_{ijt}(\boldsymbol{\alpha}_t) &= \begin{cases} \text{Var}\left(Y_{it} \middle| \boldsymbol{\alpha}_t\right) & i=j \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} z_{it}(\boldsymbol{\alpha}_t)(1-z_{it}(\boldsymbol{\alpha}_t)) & i=j \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The state equation is implemented with a 1. and 2. order random walk. For the first order random walk $\mathbf{F}=\mathbf{R}=\mathbf{I}_q$ where q is the number of regression parameters and $\mathbf{I}_q$ is the identity matrix with dimension q. As for the second order random walk, we have:

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

where $\mathbf{0}_q$ is a $q\times q$ matrix with zeros in all entries. The vector in the state equation is ordered as $\widetilde{\boldsymbol{\alpha}}_t=(\boldsymbol{\alpha}_t^T,\boldsymbol{\alpha}_{t-1}^T)^T$ to match the definition of $\mathbf{F}$ and $\mathbf{R}$. 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:

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

which we can expand to:

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

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

The unknown parameters are the initial state vector $\boldsymbol{\alpha}_0$ and the covariance matrix $\mathbf{Q}$. We estimate these with with an EM-algorithm. The E-step is carried out by an Extended Kalman filter (EKF) or an Unscented Kalman filter (UKF). The method is chosen by passing a list to the `control` argument of `ddhazard` with `list(method = "EKF", ...)` or `list(method = "UKF", ...)` respectively. Both the UKF and EKF require 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 bins, 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 bin length 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 binning length and reflects our uncertainty of $\boldsymbol{\alpha}_0$

We will use a small example to illustrate how to fit a model and illustrate that the lengths of the bins does not have a big effect on $\mathbf{Q}$. The data frame we use is in the usual start and stop time format:

```r
knitr::kable(head(simple_ex, 10), digits = 4)
```

id	tstart	tstop	event	x1	x2
1	0.00	1.00	1	0.785	0.8468
2	0.00	16.96	0	0.563	0.2664
2	16.96	19.54	0	0.959	0.7323
2	19.54	24.06	0	0.623	0.2150
2	24.06	28.00	0	0.031	0.0985
3	12.02	23.03	0	0.182	0.2629
3	23.03	28.00	0	0.339	0.5262
4	0.00	3.51	0	0.294	0.7766
4	3.51	8.81	0	0.568	0.7893
4	8.81	10.96	0	0.282	0.3075

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, individual with id `1` dies at time 1 while id `2` survives all the periods we observe. Next, we can fit a model as follows:

```r
library(dynamichazard)
library(survival)
dd_fit_short_bins <- ddhazard(
  Surv(tstart, tstop, event) ~ x1 + x2, # Formula like for cox.ph from survival
       data = simple_ex,
       by = 1,                          # Length of bin intervals
       Q = diag(0.1, 3),                # Covariance matrix in state eqn
       Q_0 = diag(10, 3),               # Covariance matrix for initial state
                                        # vector
       max_T = 28,                      # Last time we observe
       id = simple_ex$id,               # id of individuals
       control =
        list(ridge_eps = 0.0001)        # Penalty term explained later
  )

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

```
## (Intercept)          x1          x2 
##       0.549       0.779       0.451
```

Above, we estimate the model with a binning length of `by = 1`. It is not important in this scope but the model is the logistic model which we introduced later and we will return to the `ridge_eps` shortly. For now, let us see what happens if we increase the binning interval length by changing the `by` argument:

```
library(dynamichazard)
library(survival)
dd_fit_wide_bins <- ddhazard(
  Surv(tstart, tstop, event) ~ x1 + x2,
       data = simple_ex,
       by = 2,                          # increased
       Q = diag(0.1, 3),
       Q_0 = diag(10, 3),
       max_T = 28,
       id = simple_ex$id,
       control =
        list(ridge_eps = 0.0002))       # increased

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

```
## (Intercept)          x1          x2 
##     -0.3972      0.0093      0.2767
```

We see that the diagonal entries are not far from each other with the two fits. 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' and 'Unscented Kalman Filter' which respectively covers the EKF and UKF used in the E-step of the EM algorithm. Finally, we end with the sections 'Logistic model' and 'Continuous time model' which cover the models implemented in this package

I encourage you to use the shiny app while reading this vignette. You can lunch 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, ...)`. Define

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

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 $s \le t$ and smoothed estimates when $s > t$. We suppress the dependence of the co-variates ($\boldsymbol{x}_{it}$) 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:

$$\boldsymbol{a}_{i|d}^{(k)}, \quad \mathbf{V}_{i|d}^{(k)}, \quad \mathbf{B}_j^{(k)} = \mathbf{V}_{j-1|j-1}\mathbf{F}\mathbf{V}_{j|j-1}^{-1}, \quad i = 0, 1, \dots, d \wedge j = 1, 2, \dots, d$$

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, $\boldsymbol{a}_{i|d}^{(k)}$ is the smoothed state space vector for bin i 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 the EKF or UKF:

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

Then the estimates are smoothed by computing:

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

Kalman Filter

The standard Kalman filter is carried out by recursively doing a filter step and a correction step. This also applies for the EKF and UKF used in the E-step. Thus, this paragraph is included to introduce general notions. The first step in the Kalman Filter is the *filter 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 carny 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:

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

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 = 0.0001, ...)`)

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)

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 $O(p^3)$ where p denotes the dimension of the observation equation. Secondly, we do not have store an intermediate $p \times p$ matrix

The EKF starts with filter step where we compute:

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

Secondly, we perform the correction step by:

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

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

$$\begin{aligned}\boldsymbol{u}_t(\boldsymbol{\alpha}_t) &= \sum_{i\in R_t}\boldsymbol{u}_{it}(\boldsymbol{\alpha}_t), \quad \boldsymbol{u}_{it}(\boldsymbol{\alpha}_t) = \boldsymbol{z}_{it}\frac{\partial h(\eta)/\partial\eta}{H_{iit}(\boldsymbol{\alpha}_t)}\left(y_{it} - h(\eta)\right)\Big|_{\eta=\boldsymbol{z}_{it}^T\boldsymbol{\alpha}_t} \\ \mathbf{U}_t(\boldsymbol{\alpha}_t) &= \sum_{i\in R_t}\mathbf{U}_{it}(\boldsymbol{\alpha}_t), \quad \mathbf{U}_{it}(\boldsymbol{\alpha}_t) = \boldsymbol{z}_{it}\boldsymbol{z}_{it}^T\frac{\left(\partial h(\eta)/\partial\eta\right)^2}{H_{iit}(\boldsymbol{\alpha}_t)}\Bigg|_{\eta=\boldsymbol{z}_{it}^T\boldsymbol{\alpha}_t}\end{aligned}$$

R_t is the set of indices of individuals who are at risk in bin t. It is commonly referred to as the risk set. Thus, the dimension the observational equation can vary as individual dies or are right censored

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 $l > 0$ denote the learning rate and ϵ_{NR} denote the tolerance in the correction step. We then set $\boldsymbol{a} = \boldsymbol{a}_{t|t-1}$ and compute:

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

where δ is small like 10^{-9}. Selecting $l < 1$ in case of divergence can solve the issue. Thus, the following procedure is used if the algorithm fails with initial learning rate l: try a learning of $l\zeta$ in place of l above for given $0 < \zeta < 1$. If that fails then try a rate of $l\zeta^2$. The process is stopped when we succeed to fit the model or we fail to estimate the model with a learning rate of $l\zeta^w$ for a given integer w.

While Fahrmeir (1992) does not observe improvements with multiple repetitions, 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 l 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 as in ridge linear regression. Thus, the score and information matrix are computed with:

$$\begin{aligned}
\boldsymbol{u}_{it}(\boldsymbol{\alpha}_t) &= \boldsymbol{z}_{it} \frac{\partial h(\eta)/\partial\eta}{H_{iit}(\boldsymbol{\alpha}_t) + \xi} \left(y_{it} - h(\eta)\right)\Big|_{\eta = \boldsymbol{z}_{it}^T \boldsymbol{\alpha}_t} \\
\mathbf{U}_{it}(\boldsymbol{\alpha}_t) &= \boldsymbol{z}_{it}\boldsymbol{z}_{it}^T \frac{\left(\partial h(\eta)/\partial\eta\right)^2}{H_{iit}(\boldsymbol{\alpha}_t) + \xi}\Bigg|_{\eta = \boldsymbol{z}_{it}^T \boldsymbol{\alpha}_t}
\end{aligned}$$

where $\xi > 0$ is a small number. The default can be changed by altering the `ridge_eps` in the list passed to the `control` argument of `ddhazard`

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 spend 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.

Uncented 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 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 $\boldsymbol{a}_t$ and $\boldsymbol{b}_t$, let:

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

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

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

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:

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

where $\left(\sqrt{\mathbf{V}_{t|t-1}}\right)_i$ is the i'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):

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

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

$$
\begin{aligned}
\widehat{\boldsymbol{y}}_i &= \boldsymbol{z}_t\left(\widehat{\boldsymbol{a}}_i\right), \qquad i = 0,1,\dots,2q \\
\widehat{\mathbf{Y}} &= (\widehat{\boldsymbol{y}}_0,\dots,\widehat{\boldsymbol{y}}_{2q}) \\
\overline{\boldsymbol{y}} &= \sum_{i=0}^{2q} W_i^{(m)}\boldsymbol{y}_i, \qquad \Delta\widehat{\mathbf{Y}} = \widehat{\mathbf{Y}} - \overline{\boldsymbol{y}}\mathbf{1}^T, \qquad \widehat{\mathbf{H}} = \sum_{i=0}^{q} W_i^{(c)}\mathbf{H}_t(\widehat{\mathbf{a}}_i) \\
\Delta\widehat{\mathbf{A}} &= (\widehat{\boldsymbol{a}}_0,\dots,\widehat{\boldsymbol{a}}_{2q}) - \boldsymbol{a}_{t|t-1}\mathbf{1}^T \\
\mathbf{P}_{\boldsymbol{y}_t,\boldsymbol{y}_t} &= \sum_{i=0}^{2q} W_i^{(c)}\left((\widehat{\boldsymbol{y}}_i - \overline{\boldsymbol{y}})(\widehat{\boldsymbol{y}}_i - \overline{\boldsymbol{y}})^T + \widehat{\mathbf{H}}\right) = \Delta\widehat{\mathbf{Y}}\mathrm{diag}\left(\boldsymbol{W}^{(c)}\right)\Delta\widehat{\mathbf{Y}}^T + \widehat{\mathbf{H}} \\
\mathbf{P}_{\boldsymbol{x}_t,\boldsymbol{y}_t} &= \sum_{i=0}^{2q} W_i^{(cc)}(\widehat{\boldsymbol{a}}_i - \boldsymbol{a}_{t|t-1})(\widehat{\boldsymbol{y}}_i - \overline{\boldsymbol{y}})^T = \Delta\widehat{\mathbf{A}}\mathrm{diag}\left(\boldsymbol{W}^{(cc)}\right)\Delta\widehat{\mathbf{Y}}^T
\end{aligned}
$$

The correction step is then:

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

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 (say greater than 1000). We can re-write the correction step above by using the Woodbury matrix identity to get algorithm $O(|R_t|)$ instead of $O(|R_t|^3)$ where R_t is the indices at risk in the i'th interval. 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:

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

where $\tilde{\boldsymbol{y}}$, $\mathbf{G}$, $\mathbf{L}$ and $\boldsymbol{c}$ are intermediates. The above algorithm is $O(|R_t|)$ since $\widehat{\mathbf{H}}$ is a diagonal matrix and all products involves at most multiplications of $m \times |R_t|$ or $|R_t| \times m$ matrices

Ridge regression

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:

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

The addition makes divergence less common and shrinks the coefficient estimates

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 $\kappa = m/\alpha^2 - m$ 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 $W_0^{(m)} = 0$ 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. Notice though that this means that α only affects $W_0^{(c)} = 1 - \alpha^2 + \beta$ and $W_0^{(cc)} = 1 - \alpha$

$0 < \alpha \leq 1$ controls the spread of the sigma points. Notice that $\lambda + m \to 0^+$, $w_0^{(c)}, w_0^{(m)} \to -\infty$ and $w_i^{(c)}, w_i^{(m)} \to \infty$ $(i > 0)$ 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 $\mathbf{Q}_0 = k\mathbf{I}_m$ and $\boldsymbol{a}_0 = \mathbf{0}$. Then the i'th column of the Cholesky decomposition $\mathbf{V}_{0|0}$ is a vector with $\sqrt{k}$ in the i'th entry and zero in the rest of the entries. Suppose that we set k large. Then the linear predictors computed with the $l < q + 1$ sigma point is $\sqrt{q + \lambda}\sqrt{k}x_{j1l}$ where x_{j1l} is the l'th entry of individuals j's co-variate vector at time 1. This can be potentially quite large in absolute terms if x_{kjt} 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.

Fixed effects

This section will cover how fixed effects (non time-varying effects) are estimated. 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 has 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 diagonal elements (anything greater than 10^5 in the diagonal of $\mathbf{Q}_0$ for the fixed effects). 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:

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

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

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 $\boldsymbol{a}_0^{(k-1)}$:

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

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:

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

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

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

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:

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

We observe that:

1. The $\bar{\boldsymbol{x}}_{it}^T\boldsymbol{\gamma}^{(k-1)}$ 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 in the UKF or EKF in the implementation
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

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

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_{it} 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 $\widetilde{\boldsymbol{a}}_{0|d}^{(k)},\dots,\widetilde{\boldsymbol{a}}_{d|d}^{(k)}$:

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

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

3. $\widetilde{\boldsymbol{x}}_{it}^T\widetilde{\boldsymbol{a}}_t$ 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

Point 2., 3. and 4. are apparent by noticing that the conditional log likelihood in the M-step differentiated with respect to $\boldsymbol{\gamma}$ is:

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

when we use the zero order Taylor expansion for $\widetilde{E}_k \left(\sum_{t=1}^d \sum_{i \in R_t} l_{it}(\widetilde{\boldsymbol{\alpha}}_t) \right)$. This is a score equation for a generalized linear model if we use a distribution function from the exponential family. This method will be used to estimate the fixed effects 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 bin 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 $\widetilde{E}_k \left(\sum_{t=1}^d \sum_{i \in R_t} l_{it}(\widetilde{\boldsymbol{\alpha}}_t) \right)$, approximate $\widetilde{E}_k \left(\sum_{t=1}^d \sum_{i \in R_t} l_{it}(\widetilde{\boldsymbol{\alpha}}_t) \right)$ with an MC like 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 $\widetilde{E}_k \left(\sum_{t=1}^d \sum_{i \in R_t} l_{it}(\widetilde{\boldsymbol{\alpha}}_t) \right)$. At this point, the zero order Taylor expansion is the only implemented method to estimate $\boldsymbol{\gamma}$ in the M-step

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 co-variates 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 UKF and EKF are implemented as mentioned above without any complications. The following paragraphs will cover the loss of information due to binning 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

Binning

This section will illustrate how binning is performed 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 co-variate 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

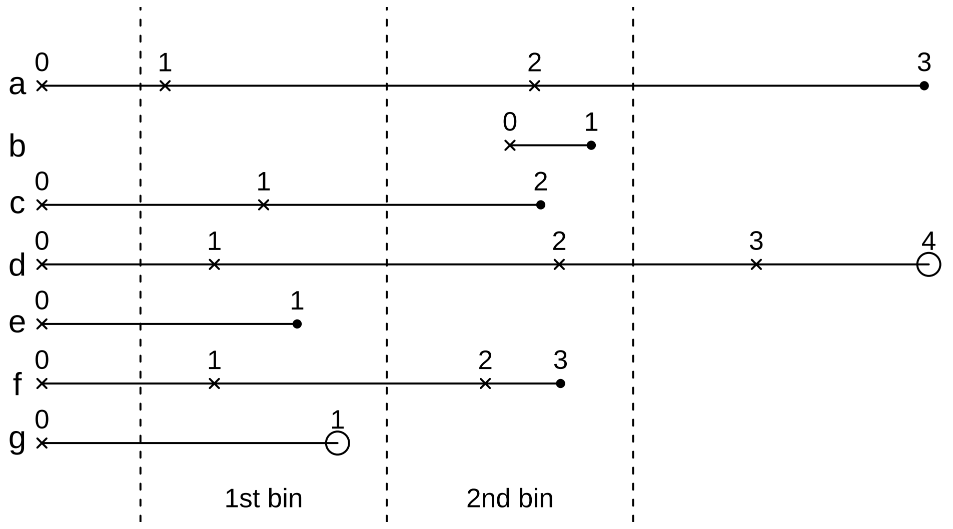


Figure 1: Illustration of binning. Each horizontal line represents an individual. A cross indicates that new covariates are observed while a filled circle indicates that the individual have died. Open circles indicates that the individual is right censored. Vertical dashed lines are bin borders

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). Ideally we would like to model that we know that for example individual a has the covariates from 0 to 1 and survives, the covariates from 1 to 2 and survives etc. That is, we would like to model the stop times

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

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

a. Is a control in both bins. We use the co-variates from 0 in the first bin and the co-variates from 1 in the second bin
b. Is not included in any of the bins. We do not know the co-variates values at the start of the second bin so we cannot include him
c. Is a control in the first bin with the co-variates from 0. He will count as a death in the second bin with the co-variates from 1
d. Acts like a.
e. Is a death in the first bin with co-variates from 0
f. Is a control in the first bin with the co-variates from 0. He is a death in the second bin with the co-variates from 1
g. Is not included in any bins. We do not know if he survived the entire period of the first bin and thus we cannot include him

The example illustrates that:

1. We loose information about co-variates that are updated within bins. For instance, a, c, d and f all use the co-variates from 0 for the entire period of the first bin despite that the co-variates 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 survives parts of the first bin
3. We loose information for observation that only occurs within bins as is the case for b

The above motivates the continuous time model that will be covered in the next sections where binning is not an issue

Continous 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 end of bin $1, 2, \dots, d$
2. The individuals co-variates change at discrete times
3. We have a piecewise constant exponential distributed arrival time for each individual when we condition on the state variables and co-variates

These assumption means that we have piecewise constant hazards given by $\exp(\boldsymbol{x}_{it}^T \boldsymbol{\alpha})$. The instantaneous hazard change when either the individuals co-variates change or the coefficients change when we change bin. We make the following definitions to formalize the assumptions above. Let $\boldsymbol{x}_{ij}$ denote the i'th individuals j'th co-variate vector. For each individual we observe $j = 1, 2, \dots, l_i$ values of the co-variate vector. Each co-variate vector is valid in a period $(t_{i,j-1}, t_{i,j}]$. This definition differs from the previous definition of $\boldsymbol{x}_{ij}$ where the subscript j referred to the bin number. The new notation is used to cover the cases where update of co-variate values do not coincide with end or start time of bin intervals. For instance, this is the case for the examples in the figure in the 'Binning' section before

Let T_i denote the random event time of the i'th individual and let $y_{ij} = 1_{\{T_i \in (t_{i,j-1}, t_{i,j}]\}}$ be the indicator for whether the i'th individual dies in period $(t_{i,j-1}, t_{i,j}]$. Further, define the indicator $\tilde{y}_{i,j,s} = y_{i,j} 1_{\{s-1 < t_{i,j} \leq s\}}$ which is one if individual i dies in bin s with covariates j. The log likelihood up to a normalization constant is:

$$\begin{aligned}
\mathcal{L}\left(\boldsymbol{\alpha}_0, \dots, \boldsymbol{\alpha}_d\right) = & -\frac{1}{2}\left(\boldsymbol{\alpha}_0 - \boldsymbol{a}_0\right)^T \mathbf{Q}_0^{-1}\left(\boldsymbol{\alpha}_0 - \boldsymbol{a}_0\right) \\
& - \frac{1}{2}\sum_{t=1}^{d}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}\right)^T \mathbf{Q}^{-1}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}\right) \\
& - \frac{1}{2}\left|\mathbf{Q}_0\right| - \frac{1}{2d}\left|\mathbf{Q}\right| \\
& + \sum_{s=1}^{d} \sum_{(i,j) \in \left\{(i,j) \middle| \substack{t_{i,j-1} < s \\ t_{i,j} > s-1}\right\}} l_{i,j,s}(\boldsymbol{\alpha}_s) \\
l_{i,j,s}(\boldsymbol{\alpha}_s) = & (\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s)^{y_{i,j,s}} - \exp\left(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s\right)\left(\min\{s, t_{i,j}\} - \max\{s-1, t_{i,j-1}\}\right)
\end{aligned}$$

where the $l_{i,j,s}$ terms come from the log likelihood:

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

which simplifies into the terms of $l_{i,j,s}$s when both the covariates $\boldsymbol{x}_i(t)$ and state space parameters $\boldsymbol{\alpha}_s$ are step functions. Thus, the first sum from $s = 1, \dots, n$ is for the change in state space parameters and the inner sum is for the changes in covariate vectors. The following sections will introduce four methods to estimate the above model. They are:

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

The methods with the binary outcome and the right truncated time variable with a jump term seem to do best on simulated data

Right truncated observations time

We start by defining the truncated observation time Δ_{is}:

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

The conditional probabilities simplifies into separate terms on the log likelihood scale due to the memoryless property of the exponential distribution. Thus, computing the conditional mean, h, can 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 bin such that $\lceil t_{i,j} \rceil - 1 = \lfloor t_{i,j-1} \rfloor$. Then:

$$\begin{aligned}
\tilde{z}(\boldsymbol{\alpha}) &= E\left(\Delta_{i,j} \middle| \Delta_{i,j-1} = t_{i,j-1} - t_{i,j-2} \wedge \boldsymbol{\alpha}_{\lceil t_{i,j} \rceil} = \boldsymbol{\alpha}, \boldsymbol{\alpha}_{\lceil t_{i,j} \rceil - 1}, \dots, \boldsymbol{\alpha}_0\right) \\
&= h_{i,j}^{\Delta}(\eta)\Big|_{\eta = \boldsymbol{x}_{is}^T\boldsymbol{\alpha}} \\
&= (\bar{t}_{i,j} - t_{i,j-1})P(\tilde{T} \geq \bar{t}_{i,j} - t_{i,j-1}) + \int_0^{\bar{t}_{i,j} - t_{i,j-1}} r f_{\tilde{T}}(r)\mathrm{d}r, \qquad \bar{t}_{i,j} = \lceil t_{i,j} \rceil y_{i,j,s} + (1 - y_{i,j,s})t_{i,j}
\end{aligned}$$

where s is the bin number that the observation is in, $\tilde{T} \sim \text{Exp}\left(\exp(\boldsymbol{x}_{ij}^T\boldsymbol{\alpha})\right)$, $f_{\tilde{T}}$ is the density function of $\tilde{T}$ and $h_{i,j}^{\Delta}$ is the inverse link function for the right truncated time variables for individual i's jth observation. Set $\lambda = \exp(\boldsymbol{x}_{ij}^T\boldsymbol{\alpha})$ and $\delta = t_{i,j} - t_{i,j-1}$. The resulting conditional mean is:

$$\tilde{z}(\boldsymbol{\alpha}) = \frac{1 - \exp\left(-\lambda\delta\right)}{\lambda}$$

Moreover, we can show that the variance is:

$$\begin{aligned}&\text{Var}\left(\Delta_{i,s} \middle| \Delta_{i,j-1} = t_{i,j-1} - t_{i,j-2} \wedge \boldsymbol{\alpha}_{\lceil t_{i,s}\rceil} = \boldsymbol{\alpha}, \boldsymbol{\alpha}_{\lceil t_{i,s}\rceil - 1}, \dots, \boldsymbol{\alpha}_0\right) \\ &\quad = \frac{1 - \exp\left(-2\delta\lambda\right) - 2\lambda\delta\exp\left(-\delta\lambda\right)}{\lambda^2}\end{aligned}$$

Right censoring is treated by having $t_{i,l_i} < \lceil t_{i,l_i} \rceil$ in the case of censoring. We only know that the individual survived up to time t_{i,l_i} and do not know if he survived the entire bin. Further, we round up in the case where $T_i = t_{i,l_i}$ (in the case of a death) when we compute the inverse link function in the prediction step of the filter (EKF or UKF). The logic is that the individual could have survived the entire bin ($\lceil t_{i,l_i} \rceil$) but only survived t_{i,l_i}. Thus, we use $\bar{t}_{i,j}$. Notice if we do not make the above adjustment then there is no difference in the model between a death, a right censoring, new covariates or change of bins

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 bin, 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 bins of length 1. You can estimate with the right truncated time method by setting the argument `model = "exp_trunc_time"` in the call to `ddhazard`

Binary outcome

The next method is to replace the likelihood with binary variables $y_{i,j,s}$ as the outcome. Then the likelihood given data has:

$$l_{i,j,s}(\boldsymbol{\alpha}_s) = y_{i,j,s}\log h_{i,j}^Y(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s) + (1 - y_{i,j,s})\log\left(1 - h_{i,j}^Y(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s)\right)$$

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

$$h_{i,j}^Y(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s) = 1 - \exp\left(-\exp(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s)\left(\min\{s, t_{i,j}\} - \max\{s-1, t_{i,j-1}\}\right)\right)$$

if we assume that the observation $(t_{i,j-1}, t_{i,j}]$ has at most length one and is inside a bin such that $\lceil t_{i,j} \rceil - 1 = \lfloor t_{i,j-1} \rfloor$. This is assumed to in the following paragraphs

There are two points to be made here. Firstly, we do not round up by using $\bar{t}_{i,j}$ in place of $t_{i,j}$ when we have a death. Thus, we do take into account the moment the person dies at in that $\log h_{i,j}^Y(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s)$ is the likelihood of dying sometime in the period $t_{i,j} - t_{i,s-1}$. Moreover, an individual can have multiple observation in the same bin. Thus, we do not have the binning issue as with the logistic model

Secondly, the cons is that the term $\log h_{i,j}^Y(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s)$ is only the log likelihood of dying sometime in the period $t_{i,j} - t_{i,s-1}$. It is not the likelihood of dying after exactly $t_{i,j} - t_{i,s-1}$. Contrary, we do prevail this information with the right truncated observation time. It is worth stressing that this may be 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 truncated observations time with jump

This section will cover the right truncated 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 prevail the exact time of death and the right truncated time method cannot distinguish some deaths from censoring with a coarse time scale

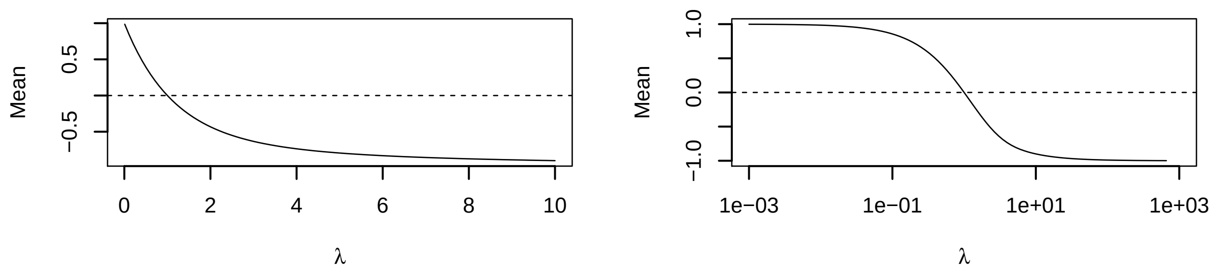
A way to deal with the latter is to add a jump term in case of censoring. Particulary, we set:

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

where we use Λ instead of Δ to distinguish between the two time variables. The above implies that $\Lambda_{i,j} \in [-\delta_{i,j}, 0] \cup \{\delta_{i,j}\}$. $\Lambda_{i,j} \leq 0$ implies that we have an event and the smaller the value the soner the event happend. $\Lambda_{i,j} = \delta_{i,j}$ implies that the variable is right truncated. The mean is given by:

$$\tilde{z}(\boldsymbol{\alpha}) = \frac{(1 - \exp(-\delta\lambda))(\delta\lambda - 1)}{\lambda}$$

where the definition of δ and λ is the same as in the right truncated time variable section. The mean is plotted below as a function of λ with $\delta = 1$



Lastly, the variance is given by:

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

The at risk length is computed in the same way as the right truncated time varialbe. That is, we use $\bar{t}_{i,j}$ instead of the min term in case of a death

You can estimate with the right truncated time variable by setting the argument `model = "exp_trunc_time_w_jump"` in the call to `ddhazard`

Combining the two

Another idea is to use the binary variable and the first mentioned right truncated time at the same time. Though, this method has shown worse performance on simulated data. Thus, the description of the method is left as an appendix

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 bin 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:

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

```
##   id tstop tstart y    x1    x2
## 1  1     0      2 0  0.13  0.12
## 2  1     2      3 0  0.95  1.48
## 3  1     3      4 1  0.18 -1.08
## 4  2     0      2 0 -0.44 -1.08
## 5  2     2      4 0 -0.55  0.15
```

```r
# The fit
fit <- ddhazard(Surv(tstart, tstop, y) ~ x1 + ddFixed(x2), data_frame,
                by = 1, # bin 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:

```r
head(data_frame_new)
```

```
##   id tstop tstart y    x1    x2
## 1  1   0.0    0.5 0  0.43  0.33
## 2  1   0.5    2.0 0  0.13  0.12
## 3  1   2.0    3.0 0  0.95  1.48
## 4  1   3.0    4.0 1  0.18 -1.08
## 5  2   0.0    2.0 0 -0.44 -1.08
## 6  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. Note that this is not the case with the logistic model as we have the same binning issue as described in the Binning section

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. An extension would be to make functions that makes this easy

Diagnostics

I am thinking of making a another vignettes with diagnostics. It will contain raw residuals, Pearson residuals, non-standardized and standardized state space error. They are all ready implemented with `s3` method for `predict`. See `?predict.fahrmeier_94`

Further, I need to look into tests the effects are time-varying or not. 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. Another idea is to generalize to ARMA models. Further, 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

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

Combining the two

An idea is to combine the binary method and the first mentioned time variable for the continous time model. Thus, we can use tuples $(y_{i,j,s}, \Delta_{i,j})$ for each observation. Further, we change the inverse link function for the binary outcome to:

$$h^{\tilde{Y}}_{i,j}(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s) = 1 - \exp\left(-\exp(\boldsymbol{x}_{i,j}^T\boldsymbol{\alpha}_s)\left(\bar{t}_{i,j} - \max\{s-1, t_{i,j-1}\}\right)\right)$$

such that we round the stop time for the binary in case of death (we use $\bar{t}_{i,j}$ instead of the min term). The motivation is to ease derivation and as we can compute the mean without knowing the actual outcome time.

The use of tuples means that we get covariance terms in the observational equation. This will affect the score vector and information matrix ($\boldsymbol{u}_t(\boldsymbol{a}_{t|t-1})$ and $\mathbf{U}_t(\boldsymbol{a}_{t|t-1})$) in the EKF. Further, it will affect the covariance matrix $\widehat{\mathbf{H}}$ in the UKF. First though, we have to derive the covariance. Assume that all observations in the bin

s that focus on have $(t_{i,j-1}, t_{i,j}]$ with at most length one and are inside a bin such that $\lceil t_{i,j} \rceil - 1 = \lfloor t_{i,j-1} \rfloor$. Then, we can drop the third subscript on the binary outcomes. Further we define the following variables to ease the notation:

$$\delta = \bar{t}_{i,j} - t_{i,j-1}, \qquad \lambda = \exp\left(\boldsymbol{x}_{i,j}^T \boldsymbol{\alpha}_s\right), \qquad Z \sim \text{Exp}\left(\lambda\right)$$

Consequently, we have the following relation (conditional on having survived up to time $t_{i,j-1}$):

$$Y_{i,j} \sim 1_{\{Z \in [0,\delta\}\}}, \qquad \Delta_{i,j} \sim Z + (\delta - Z) 1_{\{Z \in [\delta, \infty\}\}}$$

where $\sim$ denotes "similarly distributed" and $1_{\{\cdot\}}$ is one if the condition in the braces are satisfied and zero otherwise. Hence, we can find the covariance by:

$$\begin{aligned}
&\text{Cov}\left(1_{\{Z \in [0,\delta\}\}}, Z + (\delta - Z) 1_{\{Z \in [\delta,\infty\}\}}\right) \\
&\quad = \text{Cov}\left(1_{\{Z \in [0,\delta\}\}}, Z\right) + \delta \text{Cov}\left(1_{\{Z \in [0,\delta\}\}}, 1_{\{Z \in [\delta,\infty\}\}}\right) - \text{Cov}\left(1_{\{Z \in [0,\delta\}\}}, Z 1_{\{Z \in [\delta,\infty\}\}}\right) \\
&\quad = -\exp(-\lambda\delta)\delta + \delta\left(-\left(1 - \exp(-\lambda\delta)\right)\exp(-\lambda\delta)\right) - \frac{-\left(1 - \exp(-\lambda\delta)\right)\exp(-\lambda\delta)(1 + \delta\lambda)}{\lambda} \\
&\quad = -\frac{\exp(-2\lambda\delta)\left(1 + \lambda\delta\exp(\lambda\delta) - \exp(\lambda\delta)\right)}{\lambda}
\end{aligned}$$

Next, define,

$$\text{Var}\left(Y_{i,j}\right) = \sigma^2_{Y_{i,j}}, \qquad \text{Var}\left(\Delta_{i,j}\right) = \sigma^2_{\Delta_{i,j}}, \qquad \text{Cov}\left(Y_{i,j}, \Delta_{i,j}\right) = \xi_{i,j}$$

where we suppress the dependents on $\boldsymbol{\alpha}_s$. Further, order the observational equation such that

$$\boldsymbol{y}_t = \left(y_{1,i_{1,1}}, \Delta_{1,i_{1,1}}, y_{1,i_{1,2}}, \Delta_{1,i_{1,2}} \dots y_{m,i_{m,k}}, \Delta_{m,i_{m,k}}\right)^T$$

where $i_{j,i}$ is an index that correspond to the index of the j'th individuals i'th observation in this bin. Although this notation is tedious, it is included to stress that any given individual could have none, one or multiple entries in this bin s. Further, notice that $\boldsymbol{y}_t$ refers to the whole observation vector (including truncated stop times) while $y_{l,j}$ refers to the indicator for whether individual l dies at his j'th observations. The co-variance matrix $\mathbf{H}_s(\boldsymbol{\alpha}_s)$ is then a block diagonal matrix with the form:

$$\mathbf{H}_s(\boldsymbol{\alpha}_s) = \begin{pmatrix} \mathbf{H}_{1,i_{1,1}}(\boldsymbol{\alpha}_s) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{1,i_{1,2}}(\boldsymbol{\alpha}_s) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{H}_{m,i_{m,k}}(\boldsymbol{\alpha}_s) \end{pmatrix}, \qquad \mathbf{H}_{l,j}(\boldsymbol{\alpha}_s) = \begin{pmatrix} \sigma^2_{Y_{l,j}} & \xi_{l,j} \\ \xi_{l,j} & \sigma^2_{\Delta_{l,j}} \end{pmatrix}$$

You can estimate with the tuples method by setting the argument `model = "exp_combined"` in the call to `ddhazard`

EKF

Next, we turn to the EKF. In order to update the EKF we have to go back to the formulas for the score vector and information matrix. They are:

$$\dot{\boldsymbol{z}}_s(\alpha_s) = \left.\frac{\partial \boldsymbol{z}_s(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}}\right|_{\boldsymbol{\eta} = \boldsymbol{\alpha}_s}$$

$$\boldsymbol{u}_s(\boldsymbol{\alpha}_s) = \dot{\boldsymbol{z}}_s(\alpha_t)\mathbf{H}_s(\boldsymbol{\alpha}_s)^{-1}\left(\boldsymbol{y}_s - \boldsymbol{z}_s(\boldsymbol{\alpha}_s)\right), \qquad \mathbf{U}_s(\boldsymbol{\alpha}_s) = \dot{\boldsymbol{z}}_s(\alpha_s)\mathbf{H}_s(\boldsymbol{\alpha}_s)^{-1}\dot{\boldsymbol{z}}_s(\alpha_s)^T$$

where we get the simpler expression we saw previously when $\mathbf{H}_s(\boldsymbol{\alpha}_s)$ is a diagonal matrix. In the present case the inverse covariance matrix is block diagonal matrix given by:

$$\mathbf{H}_s(\boldsymbol{\alpha}_s)^{-1} = \begin{pmatrix} \mathbf{H}_{1,i_{1,1}}(\boldsymbol{\alpha}_s)^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{1,i_{1,2}}(\boldsymbol{\alpha}_s)^{-1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{H}_{m,i_{m,k}}(\boldsymbol{\alpha}_s)^{-1} \end{pmatrix}$$

Setting $\mathbf{K}_{l,j} = \mathbf{H}_{l,j}(\boldsymbol{\alpha}_s)^{-1}$ gives us the following score matrix:

$$\begin{aligned} \boldsymbol{u}_s(\boldsymbol{\alpha}_s) = \sum_{l,j} & \boldsymbol{x}_{l,j} \left(h^{\tilde{Y}}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \left((\mathbf{K}_{l,j})_{1,1} + (\mathbf{K}_{l,j})_{1,2}\right) (y_{l,j} - h^{\tilde{Y}}_{l,j}(\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s)) \\ & + \boldsymbol{x}_{l,j} \left(h^{\Delta}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \left((\mathbf{K}_{l,j})_{2,2} + (\mathbf{K}_{l,j})_{1,2}\right) (\Delta_{l,j} - h^{\Delta}_{l,j}(\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s)) \end{aligned}$$

where $\left(h^{\Delta}_{l,j}\right)'$ and $\left(h^{\tilde{Y}}_{l,j}\right)'$ are the first derivatives of the inverse link functions w.r.t. the linear predictor for the truncated waiting time and the binary outcome. Similarly, $h^{\Delta}_{l,j}$ and $h^{\tilde{Y}}_{l,j}$ are the inverse link functions the truncated waiting time and the binary outcome. We need the subscript because each observation can be at risk for different amount of time in the bin we are focusing on. Lastly, $(\cdot)_{ij}$ is the (i, j)'th entry of the matrix in the parenthesis. Finally, the information matrix can be computed by:

$$\begin{aligned} \mathbf{U}_s(\boldsymbol{\alpha}_s) = \sum_{i=1}^{r} & \boldsymbol{x}_{l,j} \left(\left(\left(h^{\tilde{Y}}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s)\right)^2 (\mathbf{K}_{l,j})_{1,1} + \left(h^{\tilde{Y}}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \cdot \left(h^{\Delta}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) (\mathbf{K}_{l,j})_{1,2} \right) \boldsymbol{x}_{l,j}^T \\ & + \boldsymbol{x}_{l,j} \left(\left(h^{\tilde{Y}}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \cdot \left(h^{\Delta}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) (\mathbf{K}_{l,j})_{1,2} + \left(\left(h^{\Delta}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s)\right)^2 (\mathbf{K}_{l,j})_{2,2} \right) \boldsymbol{x}_{l,j}^T \end{aligned}$$

The method with the tuples is more prone to diverge due to the correlation between the tuples. Thus, a ridge regression like solution is used where $\mathbf{H}_s(\boldsymbol{\alpha}_t)$ is replaced by:

$$\widetilde{\mathbf{H}}_s(\boldsymbol{\alpha}_s) = \mathbf{H}_s(\boldsymbol{\alpha}_s) + \xi\mathbf{I}$$

Below, the formulas are given for each of elements of the block diagonal matrices of the inverse covariance matrix with the ridge regression like term:

$$\left(\widetilde{\mathbf{K}}_{l,j}\right)_{11} = \frac{-2\delta_{l,j}\lambda_{l,j}\exp\left(-\delta_{l,j}\lambda_{l,j}\right) - \exp\left(-2\delta_{l,j}\lambda_{l,j}\right) + \xi\lambda_{l,j}^2 + 1}{\exp\left(-\delta_{l,j}\lambda_{l,j}\right)\left(-2\xi\delta_{l,j}\lambda_{l,j} + \xi\lambda_{l,j}^2 + 1\right) + \exp\left(-2\delta_{l,j}\lambda_{l,j}\right)\left(-\delta_{l,j}^2\lambda_{l,j}^2 - \xi\lambda_{l,j}^2 - \xi - 2\right) + \exp\left(-3\delta_{l,j}\lambda_{l,j}\right) + \xi^2\lambda_{l,j}^2 + \xi}$$

$$\left(\widetilde{\mathbf{K}}_{l,j}\right)_{2,2} = \frac{\lambda_{l,j}^2\left(-\exp\left(-2\delta_{l,j}\lambda_{l,j}\right)\right) + \lambda_{l,j}^2\exp\left(-\delta_{l,j}\lambda_{l,j}\right) + \xi\lambda_{l,j}^2}{\exp\left(-\delta_{l,j}\lambda_{l,j}\right)\left(-2\xi\delta_{l,j}\lambda_{l,j} + \xi\lambda_{l,j}^2 + 1\right) + \exp\left(-2\delta_{l,j}\lambda_{l,j}\right)\left(-\delta_{l,j}^2\lambda_{l,j}^2 - \xi\lambda_{l,j}^2 - \xi - 2\right) + \exp\left(-3\delta_{l,j}\lambda_{l,j}\right) + \xi^2\lambda_{l,j}^2 + \xi}$$

$$\left(\widetilde{\mathbf{K}}_{l,j}\right)_{1,2} = \frac{\lambda_{l,j}\exp\left(-2\delta_{l,j}\lambda_{l,j}\right) + \exp\left(-\delta_{l,j}\lambda_{l,j}\right)\left(\delta_{l,j}\lambda_{l,j}^2 - \lambda_{l,j}\right)}{\exp\left(-\delta_{l,j}\lambda_{l,j}\right)\left(-2\xi\delta_{l,j}\lambda_{l,j} + \xi\lambda_{l,j}^2 + 1\right) + \exp\left(-2\delta_{l,j}\lambda_{l,j}\right)\left(-\delta_{l,j}^2\lambda_{l,j}^2 - \xi\lambda_{l,j}^2 - \xi - 2\right) + \exp\left(-3\delta_{l,j}\lambda_{l,j}\right) + \xi^2\lambda_{l,j}^2 + \xi}$$

where $\lambda_{l,j} = \exp(\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s)$. Though, it is numerically more stable to reduce the following three factors where the two first are used for the score vector and the last factor is used for the information matrix

$$\left(h^{\tilde{Y}}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \left(\left(\left(\widetilde{\mathbf{K}}_{l,j}\right)_{11} \right)_{ii} + \left(\widetilde{\mathbf{K}}_{l,j}\right)_{1,2} \right)$$

$$\left(h^{\Delta}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \left(\left(\widetilde{\mathbf{K}}_{l,j}\right)_{2,2} + \left(\widetilde{\mathbf{K}}_{l,j}\right)_{1,2} \right)$$

$$\left(\left(h^{\tilde{Y}}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s)\right)^2 \left(\left(\widetilde{\mathbf{K}}_{l,j}\right)_{11} \right)_{ii} + 2\left(h^{\tilde{Y}}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \cdot \left(h^{\Delta}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s) \left(\widetilde{\mathbf{K}}_{l,j}\right)_{1,2} + \left(\left(h^{\Delta}_{l,j}\right)' (\boldsymbol{x}_{l,j}^T\boldsymbol{\alpha}_s)\right)^2 \left(\widetilde{\mathbf{K}}_{l,j}\right)_{2,2}$$

UKF

As with the UKF for the logit model, the multiplication by the inverse of $\widehat{\mathbf{H}}$ can be an issue. More so, the covariance terms does not help in this regard as the matrix can become (even) closer to singular. Hence, we replace the $\widehat{\mathbf{H}}$ by:

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

The matrix inversion of the matrix is easily computed since it only involves inversions of 2×2 matrices

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.

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

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.

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.

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.