Documentation to R-Code

Alexander Kreiss

July 27, 2022

Abstract

This file contains remarks and explanations about the R-Code in *functions.R* and *example.R* which is available from the authors github page in the repository *Baseline Estimation*: https://github.com/akreiss/Baseline-Estimation.

In the following we give extra information about the functions provided in *functions.R*. These functions can be used to compute the parametric and non-parametric baseline estimators together with test statistic as introduced in Kreiss et al. (2021). This test statistic can be centred and scaled as if it was on the hypothesis or as if it was on the alternative. In the R-code we will always use exponential link functions, i.e., $\alpha(\theta;t) = \exp(Z(t)'\theta)$ and $\Psi(x,\beta) = \exp(x'\beta)$. For theory about the model and a detailed theoretical description of the procedures we refer to the paper. Moreover, we will use the same notation in this documentation as in the paper. Please report any bugs you find or suggestions about the code to me (my contact details can be found on my website: https://agkreiss.uber.space/).

Below we will firstly introduce the necessary data structure and illustrate then how the functions can be used. In the file *functions.R* we provide details about the functionality and syntax of the functions.

Data Structures

The input data for the estimators and the test statistics needs to be provided as

- *events* which contains the events which happen in the network,
- *covariates* which contains the covariates $X_{n,ij}(t) \in \mathbb{R}^p$ for the hazard and
- *alpha_covariates* which contains the covariates $Z(t) \in \mathbb{R}^d$ for the baseline.

The structure of *events* is the same as in https://github.com/akreiss/Estimate-Event-Network, please see the documentation in that repository for a detailed description. The structure of *covariates* is very similar, however, slightly different to https://github.com/akreiss/Estimate-Event-Network. Therefore, we repeat it here completely:
The structure of *covariates* is as follows: It is a list of four elements: *time* (vector), *index* (list of Sparse Matrices), *edgelist* (list of matrices of two columns), *covars* (list of matrices

with p columns). The lengths of all four elements must be equal. The general philosophy is that the covariates $X_{n,ij}(t) \in \mathbb{R}^p$ and the indicators $C_{n,ij}(t)$ are piecewise constant. The k-th (constant) segment of these functions is described by the k-th entries of *time*, *index*, *edgelist*, and *covars*. In particular:

- *covariates$time[k]* contains the start point of the k-th segment and the variables *covariates$index[[k]]*, *covariates$edgelist[[k]]* and *covariates$covars[[k]]* contain the information about this segment which is valid until time *covariates$times[k+1]* or until the end of the observation period if $k+1$ exceeds the length of *covariates$times*.
- The variable *covariates$index[[k]]* is a Sparse Matrix containing entries at those places (i, j) for which $C_{n,ij}$(covariates$times[k]) $= 1$. The covariate of such an edge can be found in *covariates$covars[[k]]*. This is a matrix of all covariates and the entry in *covariates$index[[k]]* gives the row-index of the corresponding edge in *covariates$covars[[k]]*. So if you would like to know whether the edge (i, j) is active in the k-th segment, you would check *covariates$index[[k]][i,j]*. If it equals zero, then the edge is inactive (i.e. $C_{n,ij} = 0$) and no covariate information is available. If it equals $r \neq 0$, then *covariates$covars[[k]][r,]* contains the corresponding covariate vector.
- The variable *covariates$edgelist[[k]]* is reversing the information in *covariates$index[[k]]*, i.e., the covariate vector *covariates$covars[[k]][r,]* corresponds to the edge *covariates$edgelist[[k]][r,]*.

The covariates for the baseline are the same for all pairs. The structure of *alpha_covariates* is therefore the same as the structure of *covariates* but with only one pair of vertices. As a consequence *alpha_covariates* does not need the entries *index* and *edgelist*. Note that the time discretisations of the two covariate data structures may be different.

Compute Estimate

Suppose that we have data given for two observation periods which we assume to have the same length for simplicity. We thus have the following variables

- *covariates1*, *covariates2*: Covariate lists (as above) of two data-sets.
- *events1*, *events2*: Event lists of the same two data-sets (as above)
- T: End of observation horizons for the two data-sets (we suppose that they have the same length).
- *alpha_covariates1*, *alpha_covariates2*: Covariates for the baselines on the two data-sets.

The file *example_data.RData* contains examples for these variables. If you load these functions first to your R-environment the following code should run without errors (when using R 4.1.2 or newer). We briefly explain the data contained in *example_data.RData*: The

dataset is one instance from the simulation in Section 4.3 of (Kreiss et al., 2021). Thus, we have two datasets of length $T = 168 = 7 \cdot 24$, thus in the scale of hours it covers one week. The underlying network changes every hour. As only pairwise covariate $X_{n,ij}(t)$ the data contains the number of paths of length two (i.e. two edges) from i to j. The only global covariate is the intercept. Hence, we have $p = 1$ and $d = 1$. The network is of size 527 and the events are contained in the two matrices *events1, events2*. The data was generated on the hypothesis using $\beta = 0.178$ and $\theta = 0.193$.

Moreover, we suppose to have the following functions for the kernel

- *myK*(u) returns the value of the kernel $K(u)$
- *myKintegrate*(x) return $\int_{-\infty}^{x} K(u)du$

and a variable *Ksq* which contains the correspoding value of $K^{(2)}$. For the weight function we need the following two functions:

- *myw*(u) equal $w(u)$
- *mywIntegrate*(x) equal $\int_{-\infty}^{x} w(u)^2 du$

Examples for those functions are given in *example.R*.

We need to load the following library:

```
library(Matrix)
```

The following commands compute the partial likelihood estimator based on the first dataset:

```
## Compute Partial Likelihood Estimator Based on data-set 1
epp1 <- events_per_pair(covariates1,events1)
beta_cpl_out <- nlm(cpl,runif(p),print.level=2,
        covariates=covariates1,events=events1,T=T,epp=epp1)
beta_cpl <- beta_cpl_out$estimate
```

The first line extracts the number of events per pair and covariate segment from the data. This needs to be done once in the beginning and can then be passed to the likelihood function. This significantly speeds up the optimisation. We choose here *nlm* to do the optimisation but any other optimisation package can be used as well. Similarly the random starting values may be replaced by anything else. In the next step, we optimise the full likelihood

```
## Compute parametric Estimator based on data-set 1
param_out <- nlm(cfl,rnorm(p+d),print.level=2,iterlim=1000,
        betadim=p,covariates=covariates1,
        alpha_covariates=alpha_covariates1,events=events1,
        T=T,epp=epp1)
```

```
theta_hat <- param_out$estimate[(p+1):(p+d)]
```

Again *epp1* is specified in order to speed up the optimisation (we can simply use it from the previous step). By using the partial likelihood estimate, we compute next the Nelson-Aalen Estimator

```
## Compute Nelson-Aalen Estimator based on data-set 2
t <- seq(from=0,to=T,length.out=1000)
h <- 0.5
alpha_hat_NP <- NAE(t,h,beta_cpl,covariates2,events2,T,myK)
```

and the parametric estimator evaluated at the same time points t as the Nelson-Aalen Estimator (this step uses the previously computed full maximum likelihood estimator)

```
## Get parametric Baseline based on data-set 2
alpha_hat_P <- alpha_eval(t,alpha_covariates2,theta_hat,T)
```

Having the two estimators we can plot them. Note that the code above evaluates both estimators at the points given in t. In the plot we choose to display days on the x-axis rather than hours.

```
## Plot the two estimators
dev.new()
plot(t/24,alpha_hat_NP,type="l",xlab="Time in Days",ylab="",
                              main="Baseline Estimates")
lines(t/24,alpha_hat_P,lty=2)
legend("topleft",lty=c(1,2),legend=c("NP","P"))
```

And finally, we compute the test statistic. The finer we have chosen the evaluation grid t, the better is the approximation of the test-statistic in this function: The integral is approximated by assuming that the estimators are constant between the grid points with value being the average of the values at the adjacent grid points. The function call is

```
## Compute the test statistic
test <- baseline_test(alpha_hat_NP,alpha_hat_P,myw,t,events2,
        covariates2,beta_cpl,myK,h,T,myKintegrate,
        alpha_covariates2,theta_hat,mywIntegrate,Ksq)
```

The value of the test statistic can be found in *test$Tstat*. The value of the scaled and centred test statistic, i.e., the value that can be compared with a standard normal distribution, can be found in *test$Tscaled*.

References

Alexander Kreiss, Enno Mammen, and Wolfgang Polonik. Testing for a parametric baseline-intensity in dynamic interaction networks. *arXiv*, 2103.14668, 2021.