# Documentation to R-Code

Alexander Kreiss

March 30, 2021

**Abstract**

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

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). 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 fie *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 $X_{n,ij}$ for the hazard and
- *alpha_covariates* which contains the covariates for the baseline $Z$.

The structure of *events* and *covariates* is the same as in `https://github.com/akreiss/Estimate-Event-Network`, please see the documentation in that repository for a detailed description of the two data structures. 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. We assume that the covariates have dimension 2
- *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). We suppose that the time is given in the scale oh hours, i.e., $T = 48$ would be an observation horizon of 2 days.
- *alpha_covariates1*, *alpha_covariates2*: Covariates for the baselines on the two data-sets.

Moreover, we suppose to have functions for the kernel

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

and the weight function

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

Moreover, we suppose to have a variable *Ksq* which contains the value of $K^{(2)}$. The following commands compute the partial likelihood estimator based on the first data-set:

```
## Compute Partial Likelihood Estimator Based on data-set 1
epp1 <- events_per_pair(covariates1,events1)
beta_cpl_out <- nlm(cpl,runif(2),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
dbeta <- length(beta_cpl)
d <- length(beta_cpl)+length(alpha_covariates1$covars[[1]])
param_out <- nlm(cfl,rnorm(d),print.level=2,iterlim=1000,betadim=2,
                         covariates=covariates1 ,
                         alpha_covariates=alpha_covariates1 ,events=events1 ,
                         T=T, epp=epp1)

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

Again *epp1* is specified in order to speed up the optimisation. 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 have displayed 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
baseline_test(alpha_hat_NP,alpha_hat_P,myw,t,events2,covariates2,
              beta_cpl,myK,h,T,myKintegrate,alpha_covariates2,
              theta_hat,mywIntegrate,Ksq)
```

## References

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