Documentation to R-Code

Alexander Kreiss

March 1, 2020

Abstract

This file contains remarks and explanations about the R-Code in *MLEfunctions.R* which is available from the authors github page.

For theory about the model and a detailed theoretical description of the procedures we refer to Kreiss et al. (2019); Kreiss (2019). In this documentation we use the same notation as in these papers. 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/) The structure of the file and the R-Code is as follows: The file *MLEfunctions.R* contains all the relevant R-Code. Moreover, there is one function which is written in C in the file *c_functions.C*. This needs to be compiled. However, it is only used when it comes to simulations. All other functions should work without it. We begin in the following by introducing how the data has to be provided in order to use it with the functions in the R-File. Afterwards we give simple examples on how the functions can be used. The detailed syntax of the functions is given in the R-Code itself along with a detailed description of the output.

Data Structures

The data on which the model is supposed to be applied needs to be provided in two variables: *events* and *covariates*. Their structure is explained in the following.

- **covariates** The variable covariates is a list of four elements each of which is a list itself: *time* (a vector), *index* (a list of Sparse Matrices), *edgelist* (list of matrices of two columns), *covars* (list of vectors). All four lists must have the same length. The general philosophy is that the covariates $X_{n,ij}(t)$ 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 the four sub-lists *time*, *index*, *edgelist*, *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.

- 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 list of all covariates and the entry in *covariates$index[[k]]* gives the index of the corresponding edge in *covariates$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,]*.

- **events** It is a matrix and every row corresponds to an event. The meaning of the columns is as follows:
  1. Time of the event
  2. Starting vertex of the event
  3. Ending vertex of the event
  4. Number of events at that time point (currently only value 1 is supported, so you cannot have multiple events)
  5. Index of the covariates segment which is relevant for that event (see above), i.e., if the fifth column has value k it means that *covariates$covars[[k]]* and so forth are used for that event. There is the function *add_covariate_index* in the R-Code file which creates this column for you.

Compute Estimate

In order to compute the estimator of the parameter function θ the function

```
est <- MLE(mu_init, t_0, h, events, covariates, T, K, KIntegrate, par,
           iterlim, gradtol)
```

is used. Here *events* and *covariates* need to have the structure which we described above. All other input variables are explained in the R-file. Note that the file *MLEfunctions.R* contains two examples for the kernel functions: The functions *K* and *KIntegrate* are implemented for the triangular kernel and *Kuniform* and *KuniformIntegrate* for the uniform kernel.

Bandwidth Selection

In order to perform bandwidth selection one firstly needs to computed the local linear estimates for different bandwidths. Suppose there is a vector H which contains all the bandwidths which should be considered in the procedure. The variable *test_times* contains all time points t for which $\hat{\theta}_h(t)$ should be computed where the bandwidth $h \in H$ is used.

```
for(k in 1:length(H)) {
  MLEs[[k]] <- LL_MLE(as.matrix(mu_init),test_times,H[k],events,
  covariates,T,par,iterlim)
}
```

Next, we have to compute the real number of events: We generate therefore a list of graphs *real_events*. Each element of this list is a graph with edge attribute *count* which gives the number of events which happened on the edge in the time intervals defined by *test_times*, i.e., in the intervals which endpoints are given in *test_times*):

```
for(t in 1:(length(test_times)-1)) {
  real_events[[t]] <- graph_from_events(events,test_times[t],
  test_times[t+1],n)
}
```

In the last step predictions which use the estimates in *MLEs* have to be computed and compared to *real_events*. In this example we assume that *test_times* is equispaced and we define *mesh=test_times[2]-test_times[1]*. Note also that in the following code snippet *MLEs[[k]]$t_0* yields the same vector of times (namely *test_times*) for all k.

```
MSEs <- NULL
for(k in 1:length(H)) {
  ## Create Time List
  ee_times <- 1:(2*length(MLEs[[k]]$t_0))
  for(k in 1:length(MLEs[[k]]$t_0)) {
    ee_times[2*k-1] <- MLEs[[k]]$t_0[k]
    ee_times[2*k] <- MLEs[[k]]$t_0[k]+mesh
  }

  ## Compute estimated number of events
  cat("Test bandwidth h=",H[k],"\n")
  ee <- event_estimates(ee_times,MLEs,covariates,n)

  ## Compare the Estimated number of events to the observed reality
  cat("Evaluate its fit\n")
  MSEs <- c(MSEs,compare_estimate_reality(ee,real_events))
```

```
}
```

Now the vector *MSEs* contains the mean squared errors of the predictions computed based on the bandwidths in *H*. This bandwidth was computed based on a locally linear estimator by using a one-sided kernel. In order to transform the bandwidth to another kernel we refer to the procedure described in Kreiss et al. (2019).

Testing

To test for a constant parameter function one first has to compute a constant estimator. Here we use the maximum likelihood estimator which assumes a constant parameter function. This is then compared to the non-parametric estimator (see above)

```
## Compute Parametric Estimate
param_est <- parametric_MLE(mu_init,events,covariates,T)

## Compute Non-Parametric Estimate
est <- MLE(mu_init,t_0,h,events,covariates,T,K,KIntegrate,par,
        iterlim,gradtol)

## Compute Test statistic
test <- L2_test_statistic(est,param_est,covariates,events,h)
```

The format of the variable *test* is described in the R-Code in front of the function *L2_test_statistic*.

Simulation

Simulations from a certain model can be performed by the function *simulate_network*. Currently the function can only perform simulations which start at a specified time point t with a given parameter vector *theta* and on a specified covariate segment k. The simulations run until time *t+Deltat*.

```
simulate_network(theta,n,covariates,k,t,t+Deltat,reason="time")
```

References

A. Kreiss. Correlation bounds, mixing and m-dependence under random time-varying network distances with an application to cox-processes, 2019.

A. Kreiss, E. Mammen, and W. Polonik. Nonparametric inference for continuous-time event counting and link-based dynamic network models. *Electron. J. Statist.*, 13(2):2764–2829, 2019.