Documentation to R-Code for Semi-Parametric Estimation Using Laguerre Polynomials

Alexander Kreiß, Ingrid Van Keilegom

December 21, 2021

Abstract

Comments and explanations to the R-Code used in our paper Kreiss and Van Keilegom [1]. The code itself can be download from the author's github account (akreiss).

1 General Remarks

In this documentation we explain the basic functionalities of the R code in the files *functions.R* and *example.R*. The file *functions.R* contains the functions which are necessary to implement the estimation procedure outlined in our paper Kreiss and Van Keilegom [1]. Afterwards we show how the functions can be applied by going through the code provided in*example.R*. For theoretical properties and justifications we refer to our paper. In this documentation we use the same notation as in the paper. The code relies on some c-code. A compiled version of the code is part of the repository. If you work on windows, you might be able to download *c_function.dll* and load it as in the example provided below. Otherwise you need to compile the c code in *c_function.c* yourself.

If you find any mistakes in the implementation or have any other questions or comments to the code, please contact us (contact details can be found on https://agkreiss.uber.space/)

2 The functions in *functions.R*

This file contains the following functions.

- *fit_laguerre* The syntax is:

```
fit_laguerre <- function(S1,S2,w,rC,x0=NULL,m1,m2,
                         glob_repeats,xtol_rel=0.0001,maxeval=10000,
                         int_prec=500,print=TRUE)
```

  This function fits the Laguerre model to the observations: Symptom onset times of the index cases *S1*, symptom onset times of the secondary cases *S2*, end of exposure windows *w* and the exponential growth parameters *rC*. All of these are vectors and they must have the same length. The degrees of the approximation are given in *m1* (for the incubation time) and *m2* (for the generation time).

The internal optimisation is based on the *nloptr* package. At first there will be *glob_repeats* many global optimisations, the first of which starts from *x0* if provided or a random value if *x0=NULL*. The other instances will always be started from random locations. The parameters used by nloptr are *xtol_rel=1000*xtol_rel* and *maxeval=maxeval*. The used algorithm is *NLOPT_GN_CRS2_LM*. The best value from all these optimisation tasks is subsequently provided to a local optimizer (*NLOPT_LN_SBPLX*) which uses *xtol_rel* and *maxeval* as provided by the user. Finally, the value in *int_prec* gives the length of the Riemann sum which is used to approximate the integrals in the likelihood and *print* can be set to *FALSE* to avoid status messages.

The function returns a list with elements *theta1_est* and *theta2_est* which contain the estimators for the incubation time and the generation time, respectively. Furthermore the list contains the output of the best global optimizer in *glob_opt* and of the local optimizer in *loc_opt*.

- *model_selection_laguerre* The syntax is:

```
model_selection_laguerre(S1,S2,w,rC,m1_max,m2_max,
        print=TRUE,glob_repeats=5,xtol_rel=0.0001,
        maxeval=10000,int_prec=500,cluster=FALSE)
```

This function computes the optimal fits for all pairs of degrees (m_1, m_2) for which $m_1 \leq$*m1_max* and $m_2 \leq$*m2_max*. The data is provided in *S1, S2, w, rC* in the same way as for *fit_laguerre*. The values of *print, glob_repeats, xtol_rel, maxeval* and *int_prec* have the same meaning as for *fit_laguerre*. If you have previously started a cluster using *makeCluster*, you can provide the cluster object here to have the optimisations for different degrees of approximation carried out in parallel. If *cluster=FALSE* (the default) the computations are carried out serially.

The function returns a list which contains the matrix *L_matrix* which contains the optimal negative log-likelihoods for all combinations (m_1, m_2) with $m_1 \leq$*m1_max* and $m_2 \leq$*m2_max*. More precisely, the element *L_matrix*$[m_1, m_2]$ contains the negative optimal log-likelihood corresponding to the approximating degrees m_1 and m_2. The output contains furthermore the elements *fit_results, dim_list* and *index_matrix* which can be used as follows to access the specific estimators: For a given pair of degrees (m_1, m_2) find the value $r =$*index_matrix*$[m_1, m_2]$, the returned object of *fit_laguerre* for the degrees (m_1, m_2) can be found in *fit_results*$[[r]]$. The two-columned matrix *dim_list* has the property that *dim_list*$[r,] = (m_1, m_2)$, where m_1, m_2 and r are as before. Finally, the elements *m1_max* and *m2_max* are also contained in the list as a reminder.

Note lastly that the optimisations are carried out independently from each other. Therefore it can happen due to numerical issues that a strictly larger model does not yield a larger likelihood. In order to solve this issue, we recommend to call *force_monotonicity* after each call of *model_selection_laguerre*.

- *force_monotonicity* The syntax is:

```
force_monotonicity(MSL,S1,S2,w,rC,xtol_rel=0.0001,
        maxeval=10000,int_prec=500,print=TRUE)
```

With this function the output of *model_selection_laguerre* can be post-processed to guarantee an increase in the likelihood for larger models. This is achieved by

looking for inconsistencies in the output of *model_selection_laguerre* which is specified in *MSL*. When an inconsistency is found, it is resolved by calling *fit_laguerre* with one global optimisation starting from the smaller model which causes the inconsistency. The remaining parameters are identical to the ones in *fit_laguerre* and in particular the dataset provided in *S1, S2, w* and *rC* has to be the same as the one used in the call of *model_selection_laguerre* to provide meaningful results. The output has the same format as the output of *model_selection_laguerre*.

- *laguerre_density_pos* The syntax is

  ```
  laguerre_density_pos(z,theta)
  ```

  The function simply computes the density specified by the parameter *theta* evaluated at *z*. Here *theta* is specified in Cartesian coordinates and must have Euclidean norm equal to one.

- *laguerre_distr_pos* The syntax is

  ```
  laguerre_distr_pos(z,theta)
  ```

  This function computes $\int_0^z f_{theta}(x)dx$ where f_θ is a Laguerre density which can be computed by the previous function *laguerre_density_pos*. Hence, the output is the distribution function of the corresponding Laguerre density evaluated at the values in *z*. The Laguerre density is specified through *theta* which has the same format as for *laguerre_density_pos*.

- *laguerre_R0* The syntax is

  ```
  laguerre_R0(theta,r)
  ```

  This function computes the basic reproduction number R_0 when the distribution of the generation time is a Laguerre density specified through *theta* (cf. *laguerre_density_pos* for a description). The value of r equals the growth rate of the expected incidence.

- *laguerre_quantile* The syntax is

  ```
  laguerre_quantile(theta,tau,ub=20,L=10000)
  ```

  This function computes the quantiles specified in *tau* of a given Laguerre density (specified through *theta*, cf. *laguerre_density_pos*). This is done by numerically evaluating the density between 0 and a user specified upper bound *ub* on an equidistant grid of length *L*. The returned value equals the average of those two grid points which lie just above and below of the target quantile. Thus, it is necessary that *ub* is larger than the true value of the quantile.

- *laguerre_polynomial* The syntax is

  ```
  laguerre_polynomial(z,k)
  ```

  This function computes the Laguerre polynomial (which may not be confused with the corresponding Laguerre density) of a given degree *k*. The vector *z* contains the points at which the polynomial is supposed to be evaluated. The output of the function is a vector of the same length containing the values of the polynomial at the corresponding places.

- *whole_inc_likelihood* The syntax is

  ```
  whole_inc_likelihood(theta,x1,x2,w,rC,m1,m2,
          N=500)
  ```

  This function computes the negative log-likelihood for a given set of observations $(S_{1,i}, S_{2,i}, \widetilde{W}_i, C_i)_{i=1,\dots,n}$. The four types of observations are specified in the vectors *x1*, *x2*, *w* and *rC*, respectively. Note that the vector *rC* has to contain the exponential growth coefficients of the respective locations rather than the locations themselves. Thus, all four vectors must have the same length. The first parameter *theta* contains both parameter sets, for the incubation period in the entries 1 to *m1* and for the generation time in the entries *m1+1* to *m1+m2*. Both parameters have to be specified in polar coordinates (only the angles). The value of N has the same meaning as *int_prec* in other functions, e.g. *fit_laguerre*.

  The output of this function is the un-normalized (i.e. not divided by the number of observations) negative log-likelihood corresponding to the given observations and models.

- *laguerre_approx* The syntax is

  ```
  laguerre_approx(fdens,m,...)
  ```

  This function can be used to compute the closest Laguerre type density of degree m to a given density *fdens*. The function *fdens* is required to allow as a first argument a vector of evaluation points and it must return the values of the density evaluated at these points. The additional arguments *...* are passed to *fdens*. The output of this function is a normalized vector of length $m + 1$ which contains the Cartesian coordinates of the parameter specifying the closest density.

- *hellinger_distance* The syntax is:

  ```
  hellinger_distance(theta,td,...)
  ```

  The function returns the squared Hellinger distance between a Laguerre density specified through *theta* and an arbitrary density specified in *td*. The format of *td* needs to be the same as the format of *fdens* in *laguerre_approx*.

- *hellinger_distance_lag* The syntax is:

  ```
  hellinger_distance_lag(theta1,theta2)
  ```

  The function returns the squared Hellinger distance between the two Laguerre densities given through *theta1* and *theta2*.

3 Example

In this section we illustrate the usage of the functions by going through the code provided in *example.R*. In this example the following packages are required. Also we set the seed to get reproducible results.

```
## Libraries
library(doParallel)
library(foreach)
```

```
library(nloptr)
library(SphericalCubature)

## Load functions
dyn.load("c_function.dll")
source('./functions.R')

## Set seed for reproducability
set.seed(2020)
```

We begin by generating an example data set which we will work with. This requires no functions specific to our methodology. We decide here to restrict to a small dataset with $n = 10$ observations to obtain an example which runs quickly on most systems.

```
## Set information
n <- 10 # Number of observations
r <- log(2)/5 # Exponential growth rate

## Generate Data
w <- rexp(n,rate=0.3820225)
location<- sample(c(0,1),n,replace=TRUE,prob=c(26/40,14/40))
rC <- rep(0,n)
rC[location==1] <- r

no_exp_growth <- which(location==0)
exp_growth    <- which(location==1)

first_infection  <- rep(0,n)
first_infection[no_exp_growth] <- runif(length(no_exp_growth),
        min=0,max=w[no_exp_growth])
first_infection[   exp_growth] <- w[exp_growth]-
        rexp(length(exp_growth),rate=r) %% w[exp_growth])

second_infection <- first_infection
        +rweibull(n,shape=2.826,scale=5.665)

S1 <- first_infection +rlnorm(n,meanlog=1.644,sd=0.363)
S2 <- second_infection+rlnorm(n,meanlog=1.644,sd=0.363)

w <- pmin(w,S1)
```

Next we perform estimation for given degrees of the Laguerre polynomials. In this example we choose $m_1 = m_2 = 2$. Below, we discuss model selection too. Here we let the optimizer start from 10 random starting locations and stick otherwise with the default choices.

```
m1 <- 2
m2 <- 2

model_fit <- fit_laguerre(S1,S2,w,rC,NULL,m1,m2,10,0.0001,
```

```
10000,500,TRUE)
```

In the next step we use the functions *laguerre_density_post* and *laguerre_approx* in order to compute the estimated density as well as that Laguerre type density which come closest to the true model which we used in the simulations.

```
## Extract Estimate
theta1_est <- model_fit$theta1
theta2_est <- model_fit$theta2

## Compute estimated densities
x <- seq(from=0,to=15,length.out=1000)
inc_est <- laguerre_density_pos(x,theta1_est)
gen_est <- laguerre_density_pos(x,theta2_est)

## Compute true density
inc <- dlnorm(x,meanlog=1.644,sd=0.363)
gen <- dweibull(x,shape=2.826,scale=5.665)

## Compute the best Laguerre approximation to the truth
best_theta_inc <- laguerre_approx(dlnorm,m=m1,meanlog=1.644,
                  sd=0.363)
best_theta_gen <- laguerre_approx(dweibull,m=m2,shape=2.826,
                  scale=5.665)

best_inc <- laguerre_density_pos(x,best_theta_inc)
best_gen <- laguerre_density_pos(x,best_theta_gen)
```

Finally, all results are plotted.

```
dev.new()
par(mfrow=c(1,2))
plot(x,inc,main="Incubation_Period",type="l")
lines(x,inc_est,lty=2)
lines(x,best_inc,lty=3)
plot(x,gen,main="Generation_Time",type="l")
lines(x,gen_est,lty=2)
lines(x,best_gen,lty=3)
```

As an example of feature estimation, we show next how quantiles and reproduction numbers can be obtained from the estimates. We begin with quantiles. Our interests lies in the following quantiles

```
tau <- c(0.3,0.5,0.7,0.9)
```

The true quantiles are found through the corresponding R-functions

```
true_quant_inc <- qlnorm(tau,meanlog=1.644,sd=0.363)
true_quant_gen <- qweibull(tau,shape=2.826,scale=5.665)
```

In order to get the quantiles of the best Laguerre approximation and the quantiles of the estimated Laguerre approximation, the function *laguerre_quantile* can be used as follows:

```
## Quantiles of the best Laguerre Approximation
best_quant_inc <- rep(0,length(tau))
best_quant_gen <- rep(0,length(tau))
for(k in 1:length(tau)) {
  best_quant_inc[k] <- laguerre_quantile(best_theta_inc,tau[k])
  best_quant_gen[k] <- laguerre_quantile(best_theta_gen,tau[k])
}

## Estimated Quantiles
est_quant_inc <- rep(0,length(tau))
est_quant_gen <- rep(0,length(tau))

for(k in 1:length(tau)) {
  est_quant_inc[k] <- laguerre_quantile(theta1_est,tau[k])
  est_quant_gen[k] <- laguerre_quantile(theta2_est,tau[k])
}
```

Let us now compute the basic reproduction numbers. The true R_0 can be found by using the equation from the paper:

```
helpf <- function(x) {
  return(exp(-r*x)*dweibull(x,shape=2.826,scale=5.665))
}
R0 <- 1/integrate(helpf,lower=0,upper=Inf)$value
```

The estimated R_0 can be found by supplying the Laguerre approximation to the function *laguerre_R0*

```
R0_est <- laguerre_R0(theta2_est,r)
```

As a last step this file illustrates model selection. Here, we try all degrees m_1 and m_2 which are lower than or equal to 8. For all degree combinations (m_1, m_2) the corresponding model is fit. This is dine by using the function *model_selection_laguerre*. In order to save time, we use here parallel computations as well. Afterwards we call *force_monotonicity* to have a decrease in the negative log-likelihoods.

```
noc <- detectCores()-1
cl <- makeCluster(noc,outfile="progress")
registerDoParallel(cl)

MSL <- model_selection_laguerre(S1,S2,w,rC,8,8,TRUE,10,0.0001,
                    10000,500,cl)
MSLmon <- force_monotonicity(MSL,S1,S2,w,rC,0.0001,10000,
                    500,TRUE)

stopCluster(cl)
```

With the output from *force_monotonicity*, it is direct to compute BIC and AIC:

```
BIC <- matrix(0,ncol=8,nrow=8)
AIC <- matrix(0,ncol=8,nrow=8)
for(m1 in 1:8) {
```

```
  for(m2 in 1:8) {
    BIC[m1,m2] <- (m1+m2+2)*log(n)-2*(-MSLmon$L_matrix[m1,m2])
    AIC[m1,m2] <- (m1+m2+2)       -2*(-MSLmon$L_matrix[m1,m2])
  }
}
```

References

[1] A. Kreiss and I. Van Keilegom. Semi-parametric estimation of incubation and generation times by means of Laguerre polynomials. *arxiv*, 2020. URL https://arxiv.org/abs/2012.15611.