

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

Alexander Kreiß, Ingrid Van Keilegom

September 3, 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.

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.

- *inc_likelihood* The Syntax is:

```
inc_likelihood <- function(x1,x2,w,rC,theta1_polar ,  
                           theta2_polar ,N=500)
```

This function computes the likelihood as specified in the paper for one single observation (S_1, S_2, W, C) . These four values are specified in $x1$, $x2$, w and rC , respectively. Note that rC is supposed to contain the exponential growth parameter related to location C . The tested model is specified in *theta1_polar* and *theta2_polar*. Note that their coordinates need to be specified in polar format without the radius (which is always equal to 1). So, only the angles are specified. In particular, it is not possible to specify $m_1 = 0$ and $\theta_1 = 1$.

The likelihood is computed by approximating the integrals through Riemann sums. The number of terms in the Riemann sums is specified via N . The default value is 500.

- *laguerre_density_pos* The syntax is

```
laguerre_density_pos <- function(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 <- function(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 <- function(theta, r)
```

This function computes the basic reproduction number R_0 when the distribution of the generation 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.

- *lag_quantile* The syntax is

```
lag_quantile <- function(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 a 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 <- function(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 <- function(theta, x1, x2, w, rC, m1, m2,
                                N=500, cluster=FALSE)
```

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 of them 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* is passed to *inc_likelihood*. Finally, *cluster* can contain a cluster identifier as returned from *makeCluster*. Then, the computations are done in a parallel fashion by using the *foreach* framework. Thus the packages *doParallel* and *froeach* are required in this case.

- *laguerre_approx* The syntax is

```
laguerre_approx <- function(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.

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)
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.

```
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
```

```
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. At first we specify the informations which are necessary for the optimizer and start a cluster.

```

m1 <- 2
m2 <- 2

opts <- list(algorithm="NLOPT_LN_SBPLX", print_level=0,
             xtol_rel=0.001, maxeval=10000)

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

```

In order to make sure that the optimizer does not get stuck due to a bad choice of starting value, we do the estimation starting from *trials* many random starting values. We also prepare a list which shall contain the estimation results

```

trials <- 5
out <- vector(mode="list", length=trials)

```

The following code does the actual optimisation:

```

currmin <- Inf
lowest <- 1
for(i in 1:trials) {
  out[[i]] <- nloptr(x0=runif(m1+m2, min=0, max=pi),
                   eval_f=whole_inc_likelihood,
                   lb=rep(0, m1+m2), ub=rep(pi, m1+m2),
                   opts=opts, x1=S1, x2=S2, w=w, rC=rC,
                   m1=m1, m2=m2, N=500, cluster=cl)

  if(out[[i]]$objective < currmin) {
    lowest <- i
    currmin <- out[[i]]$objective
  }
  cat("Trial_", i, ".\n")
}
stopCluster(cl)

```

The for-loop above repeats the same optimisation task for different random starting values (note that the starting values are specified in polar coordinates). The optimisa-

tion itself is performed using *nloptr*. The variable *lowest* contains in the end the index of the run which yielded the smallest result (i.e. the largest likelihood). We can now extract the estimates and transfer them to Cartesian coordinates.

```
estimate <- out[[lowest]]$solution
theta1_est <- polar2rect(1, estimate[, 1:(m1)])
theta2_est <- polar2rect(1, estimate[(m1+1):(m1+m2)])
```

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.

```
## 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 of the estimated Laguerre approximation, the function *lag_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] <- lag_quantile(best_theta_inc,tau[k])
  best_quant_gen[k] <- lag_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] <- lag_quantile(theta1_est,tau[k])
  est_quant_gen[k] <- lag_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*

```

## Compute estimated R0
R0_est <- laguerre_R0(theta2_est,r)

```

As a last step this file illustrates model selection. Here, certain maximal values of the degrees m_1 and m_2 are specified. For all lower degree combinations the best model is fit (in the same way as above). Afterwards, it is direct to compute BIC and AIC.

```

## Set Information
trials <- 5
m1_max <- 4
m2_max <- 4
out_matrix <- matrix(0,nrow=m1_max,ncol=m2_max)
opts <- list(algorithm="NLOPT_LN_SBPLX",print_level=3,
             xtol_rel=0.001,maxeval=10000)
out <- vector(mode="list",length=trials)

```

```

## Set-Up Cluster
noc <- detectCores()-1
cl <- makeCluster(noc,outfile="progress")
registerDoParallel(cl)

```

```

## Do the estimation for all allowed pairs of degrees
for(m1 in 1:m1_max) {
  for(m2 in 1:m2_max) {

```

```

for(i in 1:trials) {
  currmin <- Inf
  lowest <- 0
  out[[i]] <- nloptr(x0=runif(m1+m2,min=0,max=pi),
                    eval_f=whole_inc_likelihood,
                    lb=rep(0,m1+m2),ub=rep(pi,m1+m2),
                    opts=opts,x1=S1,x2=S2,w=w,rC=rC,m1=m1,
                    m2=m2,N=500,cluster=cl)
  if(out[[i]]$objective<currmin) {
    lowest <- i
    currmin <- out[[i]]$objective
  }
  cat(" Trial_",i,".\n")
}
out_matrix[m1,m2] <- out[[lowest]]$objective
}
}
stopCluster(cl)

## Compute AIC and BIC
BIC <- matrix(0,ncol=m2_max,nrow=m1_max)
AIC <- matrix(0,ncol=m2_max,nrow=m1_max)
for(m1 in 1:m1_max) {
  for(m2 in 1:m2_max) {
    BIC[m1,m2] <- (m1+m2+2)*log(n)-2*(-out_matrix[m1,m2])
    AIC[m1,m2] <- (m1+m2+2) -2*(-out_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>.