Appendix C

AMPL, MAPLE, and S-Plus Programmes

C.1 AMPL programmes

Programming in AMPL requires several files: the model file including parameters, variables, objective function, and restrictions, the data file containing the data, and the optional programme file. If the Kestrel interface, described in Section 1.3.2, is used, it has to be invoked in the programme file and the NEOS solver has to be specified.

In this section, we present two AMPL programmes: the one used for the Weibull regression model in Chapter 6, and the one of the simulation study. Any programme to implement simultaneous maximization for the models in Chapter 4, will be similar to the one for the data from Can Ruti; the main difference concerns the objective function. The command lines for the implementation of these objective functions are shown in Section C.1.2.

C.1.1 Programme for the evaluation of data from Can Ruti

The Model File

```
set V;        # set for variables
set DIM;      # set for boundaries

param bounds{V,DIM};
param M ;     # number of possible covariate values
param N ;     # number of observations

param datmat{1..N,1..6};   # data matrix
```

```
# Definition of parameters containing data
param hivn{i in 1..N} = datmat[i,1];
param hivp{i in 1..N} = datmat[i,2];
param y{i in 1..N}    = datmat[i,3];
param epsi{i in 1..N} = datmat[i,4];
param xi_1{i in 1..N} = datmat[i,5];
param xi_2{i in 1..N} = datmat[i,6];

param s{j in 1..M} = j;                  # covariate values
param alfa{i in 1..N,j in 1..M}= if s[j]
           in [hivn[i],hivp[i]] then 1;  # censoring patterns of covariate
param logsum{i in 1..N,j in 1..M} = if
      alfa[i,j]==1 and epsi[i]==1
      then log(y[i]+hivp[i]-s[j]);       # log(Y|Z)

param RR_zeta;          # relative risk
param acc_fac_z;        # accelerating factor

# Model parameters
var mu;
var beta;
var sigma >= 0, :=1;
var omega{j in 1..M} >=0;

param num_run:= 1;
param sum_log{i in 1..N};
param sum_ome;

# Objective function
maximize logLikelihood:
sum{i in 1..N}
 (epsi[i]*xi_1[i]*log(1/sigma*sum{j in 1..M}
  alfa[i,j]*exp((logsum[i,j]-mu-beta*log(s[j]))/sigma-
            exp((logsum[i,j]-mu-beta*log(s[j]))/sigma))*omega[j])+
  epsi[i]*xi_2[i]*log(sum{j in 1..M} alfa[i,j]*
       exp(-exp((logsum[i,j]-mu-beta*log(s[j]))/sigma))*omega[j])+
  epsi[i]*(1-xi_1[i])*(1-xi_2[i])*log(sum{j in 1..M} alfa[i,j]*(1-
       exp(-exp((logsum[i,j]-mu-beta*log(s[j]))/sigma)))*omega[j])+
 (1-epsi[i])*log(sum{j in 1..M} alfa[i,j]*omega[j]));

#Restrictions
subject to sum1:
    sum {i in 1..M} omega[i] = 1;
```

The data file

Within the following file, an ASCII file, `canrutidat.txt` is invoked, which contains the data.

```
set V:= mu bet sig;
set DIM:= low up;

param  bounds:
        low  up:=
  mu     -5   5
  bet    -5   5
  sig     0   5;

param M := 204;
param N := 361;

close canrutidat.txt;

read {i in 1..N,j in 1..6} datmat[i,j] < canrutidat.txt;
```

The programme file

The command `option solver kestrel` (line 4) in the following programme file invokes the Kestrel interface, which itself calls the solver SNOPT (line 5). Results are displayed at the end of the programme (lines 28 through 34) and can be read into a local data file.

```
1   reset;
    model weibull.mod;
    data canruti.dat;
    option solver kestrel;
5   option kestrel_options 'solver=snopt';
    option snopt_options 'ftimes 1';

    let mu:=Uniform(bounds["mu","low"],bounds["mu","up"]);
    let beta:= Uniform(bounds["bet","low"],bounds["bet","up"]);
10  let sigma:=Uniform(bounds["sig","low"],bounds["sig","up"]);
    let{j in 1..M} omega[j] := 1/M;
    repeat{
     solve;
     if solve_result =='solved' then break;
15   let num_run:= num_run +1;
     if num_run == 15 then break;
     let mu:= Uniform(bounds["mu","low"],bounds["mu","up"]);
     let beta:=  Uniform(bounds["bet","low"],bounds["bet","up"]);
     let sigma:= Uniform(bounds["sig","low"],bounds["sig","up"]);
```

```
20    let{j in 1..M} omega[j]:= Uniform(0,1);
      let sum_ome:= sum{j in 1..M} omega[j];
      let{j in 1..M} omega[j]:=omega[j]/sum_ome;
     }
     let RR_zeta:= exp(-beta/sigma);
25   let acc_fac_z:= exp(beta);

     printf "The estimated values of the model are (mu, beta, sigma)
                        = (%6.4f, %6.4f, %6.4f).\n",mu, beta, sigma;
     printf "The relative risk amounts to %5.4f,", RR_zeta; printf
30          " the acceleration factor is %5.4f.\n", acc_fac_z;
     option omit_zero_rows 1;
     display omega;
```

C.1.2 Objective functions of further AMPL programmes

Herein, we show the command lines of AMPL's model files containing the objective functions of the three models described in Chapter 4. These are the following:

1. Linear regression model: objective function (4.3)

```
# Objective function
maximize linregmod_logLikelihood:
sum{i in 1..N} log(sum{j in 1..M}
  gamma[i,j]*exp(-(y[i]-alpha-beta*s[j]-kappa*x[i])^2/
        (2*sigma^2))*omega[j])-N*log(sqrt(2*pi)*sigma);
```

2. Logistic regression: objective function (4.4)

```
# Objective function
maximize logistic_logLikelihood:
sum{i in 1..N} (y[i]*(alpha+kappa*x[i])+log(sum{j in 1..M}gamma[i,j]*
   exp(y[i]*beta*s[j])/(1+exp(alpha+beta*s[j]+kappa*x[i]))*omega[j]));
```

3. Proportional hazard model: objective function (4.11)

```
# Objective function
maximize Cox_logLikelihood:
sum{i in 1..N}(delta[i]*log(lambda[i])+log(sum{j in 1..M}alfa[i,j]*
  exp(delta[i]*beta*s[j])*exp(-sum{l in 1..i}lambda[l]*exp(beta*s[j]))*omega[j]));
```

C.1.3 Programmes for simulation study

The files of the AMPL programme for the simulation study are designed to be used from the WWW-submission tool of SNOPT. This makes it possible to sent one single programme for the

D maximization problems of a particular simulation setting. Here, the use of the Kestrel interface is not recommendable, because each maximization problem would have to be sent separately to NEOS. We show the model and the programme file.

The model file

```
set V;   # variables set
set Q;   # quantiles set
set DIM; # boundaries set
set EST; # estimation results set

param bounds{V,DIM};    # bounds for variables
param estresul{V,EST};  # Estimation results
param estresulq{Q,EST}; # Estimation results for quantiles

param D; # number of data sets
param N; # number of observations
param feasol;             # number of feasible solutions
param RRfeas;
param med20feas;
param med50feas;
param nc := 4*D;          # column number of data matrix
param datmat{1..N,1..nc}; # data matrix
param k;

# Definition of parameters containing data
param y{i in 1..N} = datmat[i,k*4-3];
param del{i in 1..N}= datmat[i,k*4-2];
param zl{i in 1..N}= datmat[i,k*4-1];
param zr{i in 1..N}= datmat[i,k*4];
param M{j in 1..D} = round(max{i in 1..N} datmat[i,j*4]);
param Mmax = max{j in 1..D} M[j];
param s{j in 1 .. Mmax} = j;                    # covariate values
param alfa{i in 1..N, j in 1..M[k]}= if s[j] in (zl[i],zr[i]] then 1;
param xi_1{i in 1..N}= if del[i] == 1 then 1;  # 1 Y observed exactly
param xi_2{i in 1..N}= if del[i] == 0 then 1;  # 1 Y right-censored

param xq{Q};            # quantiles to be calculated
param quant{Q};         # vector of quantiles
param qj;               # quantile counter when calculating them
param quantall{Q,1..D};
param quantorg{Q};

param Result{1 .. D} symbolic;
param num_run{1..D};
param sum_ome;

var mu;
```

```
var beta;
var sigma >= 0;
var omega{j in 1..Mmax} >=0;

param thetall{V,1..D}; # Estimation values for each run
param thetorg{V};      # original model parameters

# Objective function
maximize Weibull_logLikelihood:
 sum{i in 1..N} (xi_1[i]*log(1/sigma*sum{j in 1..M[k]}
  alfa[i,j]*exp((log(y[i])-mu-beta*log(s[j]))/sigma-
            exp((log(y[i])-mu-beta*log(s[j]))/sigma))*omega[j])+
   xi_2[i]*log(sum{j in 1..M[k]} alfa[i,j]*
            exp(-exp((log(y[i])-mu-beta*log(s[j]))/sigma))*omega[j])+
  (1-xi_1[i])*(1-xi_2[i])*log(sum{j in 1..M[k]} alfa[i,j]*(1-
            exp(-exp((log(y[i])-mu-beta*log(s[j]))/sigma)))*omega[j]));

subject to sum_omega:
    sum {i in 1..M[k]} omega[i] = 1;
```

The programme file

```
data;
set V:= mu bet sig rr med20 med50;
set Q:= q10 q25 q50 q75 q90;
set DIM:= low up;
set EST:= mean std bias rbias mse;

param  bounds:
      low   up:=
 mu   0    5
 bet  0    2
 sig  0    5;

param thetorg:=
 mu     4       #  3       #
 bet    0.25    #  0.45    #
 sig    0.45    #  0.65    #
 rr     0.574   #  0.5     #
 med20  97.905  #  60.937  #
 med50  123.11; #  92.036; #

param xq:=
 q10   0.1
 q25   0.25
 q50   0.5
 q75   0.75
 q90   0.9;
```

```
param quantorg:=
 q10   16.23      # 37.185
 q25   26.818     # 43.255
 q50   41.628     # 50           50: Z normal; 41.628 Z Weibull
 q75   58.871     # 56.745
 q90   75.871;    # 62.816;

param D:= 500;
param N:= 50;

option snopt_options "version";
let feasol:= 0;
let RRfeas:= D;
let med20feas:= D;
let med50feas:= D;

for{K in 1..D} {
    let k:= K;
    let num_run[k]:= 1;
    let mu:=Uniform(bounds["mu","low"],bounds["mu","up"]);
    let beta:= Uniform(bounds["bet","low"],bounds["bet","up"]);
    let sigma:=Uniform(bounds["sig","low"],bounds["sig","up"]);
    let{m in {M[k]}, i in 1..m} omega[i] := 1/m;
    option snopt_options "version";
    repeat{
        solve;
        let Result[k] := solve_result;
        if Result[k] ==’solved’ then break;
        let num_run[k]:= num_run[k] +1;
        if num_run[k] == 15 then break;
        let mu:= Uniform(bounds["mu","low"],bounds["mu","up"]);
        let beta:=  Uniform(bounds["bet","low"],bounds["bet","up"]);
        let sigma:= Uniform(bounds["sig","low"],bounds["sig","up"]);
        let{j in 1..M[k]} omega[j]:= Uniform(0,1);
        let sum_ome:= sum{j in 1..M[k]} omega[j];
        let{j in 1..M[k]} omega[j]:=omega[j]/sum_ome;
    }
    if Result[k] == ’solved’ then{
       let feasol := feasol + 1;
       let thetall["mu",k]:= mu;
       let thetall["bet",k]:=  beta;
       let thetall["sig",k]:= sigma;
       let thetall["rr",k]:= exp(-beta/sigma);
       let thetall["med20",k]:= exp(mu+beta*log(20))*log(2)^sigma;
       let thetall["med50",k]:= exp(mu+beta*log(50))*log(2)^sigma;
```

```
       for{q in Q}{
        let qj:= 2;
        for{j in qj..M[k]}
         if xq[q] <= omega[1] then let quantall[q,k]:= 1;
         else
           if sum{l in 1..j-1}omega[l] < xq[q] <= sum{l in 1..j}omega[l] then{
              let quant[q]:= s[j];
              let qj := j;
              let quantall[q,k]:=quant[q];
              break;
           }
        }
     }
     else
    {
     let RRfeas:= RRfeas-1;
     let med20feas:= med20feas-1;
     let med50feas:= med50feas-1;
     printf "ERROR while processing data set number %1d:\n", k ;
    }
}

option omit_zero_rows 1;
printf "\nSIMULATION RESULTS ASSUMING THE WEIBULL DISTRIBUTION\n";
printf "Number of used data sets: %1d\n", D;
printf "Number of observations in each data set: %1d\n", N;

printf{K in 1..D} " Results data set number %1d (number of attempts: %1d):
              (mu, beta, sigma)= (%5.4f, %5.4f, %5.4f).\n", K,
               num_run[K], thetall["mu",K], thetall["bet",K], thetall["sig",K];

let{v in V} estresul[v,"mean"] := 1/feasol*sum{i in 1..D} thetall[v,i];
let{v in V} estresul[v,"std"]  := sqrt(1/(feasol-1)*sum{i in 1..D}
                                  (thetall[v,i]-estresul[v,"mean"])^2);
let{v in V} estresul[v,"bias"] := estresul[v,"mean"]-thetorg[v];
let{v in V} estresul[v,"rbias"]:= estresul[v,"bias"]/thetorg[v];
let{v in V} estresul[v,"mse"]  := estresul[v,"std"]^2 + estresul[v,"bias"]^2;

let{q in Q} estresulq[q,"mean"] := 1/feasol*sum{i in 1..D} quantall[q,i];
let{q in Q} estresulq[q,"std"]  := sqrt(1/(feasol-1)*sum{i in 1..D}
                                  (quantall[q,i]-estresulq[q,"mean"])^2);
let{q in Q} estresulq[q,"bias"] := estresulq[q,"mean"]-quantorg[q];
let{q in Q} estresulq[q,"rbias"]:= estresulq[q,"bias"]/quantorg[q];
let{q in Q} estresulq[q,"mse"]  := estresulq[q,"std"]^2 + estresulq[q,"bias"]^2;
```

```
printf "\n The means, standard deviations, biases, relative biases
                                nd MSE's of the parameters are:\n" ;
printf "    mu: %4.3f,%4.3f,%4.3f,%4.3f,%4.3f.\n", {e in EST} estresul["mu",e];
printf "  beta: %4.3f,%4.3f,%4.3f,%4.3f,%4.3f.\n", {e in EST} estresul["bet",e];
printf " sigma: %4.3f,%4.3f,%4.3f,%4.3f,%4.3f.\n\n", {e in EST} estresul["sig",e];

printf " The same values for relative risk and conditional median are:\n";
printf " RRisk: %4.3f,%4.3f,%4.3f,%4.3f,%4.3f.\n", {e in EST} estresul["rr",e];
printf " Med20: %4.3f,%4.3f,%4.3f,%4.3f,%4.3f.\n", {e in EST} estresul["med20",e];
printf " Med50: %4.3f,%4.3f,%4.3f,%4.3f,%4.3f.\n", {e in EST} estresul["med50",e];

printf "\n The means, standard deviations, biases, relative biases,
                                  and MSE's of the quantiles are:\n";
printf{q in Q} " %3.2f-quantile: %4.2f, %4.2f, %4.2f, %4.2f, %4.2f.\n",
                                       xq[q], {e in EST} estresulq[q,e];
```

C.2 MAPLE programme for calculation of confidence intervals

The following programme calculates the variances and the confidence intervals of the parameters estimates of Model (6.1). It is the programme for the Weibull regression model for the general case without other covariates.

```
restart:with(linalg):

# 1st Step: Invocation of AMPL's parameters' estimation results
M:=15: m:=M-3:
res:=matrix(M,2,readdata(‘G:\\Disss\\AMPL\\Output\\result.dat‘,[integer,float])):
evalm(res):
muhat:=res[1,2]: betahat:=res[2,2]: sigmahat:=res[3,2]:
zet:=vector(m): omegahat:=vector(m):
for i from 1 to m do zet[i]:=res[i+3,1]: omegahat[i]:=res[i+3,2]: od:

# 2nd Step: Definition of the log likelihood function
#  Definition of the conditional density and survival function
f:=(yps,es)->exp((yps-mu-beta*ln(es))/sigma-exp((yps-mu-beta*ln(es))/sigma)):
S:=(yps,es)->exp(-exp((yps-mu-beta*ln(es))/sigma)):

#  Invocation of observations and definition of indicator matrix
N:=361:
obs:=matrix(N,6,readdata(‘G:\\Disss\\Maple\\input\\data.txt‘,integer,6)):
hivn:=col(obs,1): hivp:=col(obs,2): y:=col(obs,3):
eps:=col(obs,4): xi1:=col(obs,5): xi2:=col(obs,6):
alfa:=matrix(N,m):
for i from 1 to N do for j from 1 to m
    do if hivn[i] <= zet[j] and zet[j] <=hivp[i]
```

```
       then alfa[i,j]:=1 else alfa[i,j]:= 0 fi; od: od:
lnsum:=matrix(N,m):
for i from 1 to N do for j from 1 to m do if alfa[i,j]=1 and eps[i]=1
then lnsum[i,j]:= ln(y[i]+hivp[i]-zet[j])
else lnsum[i,j]:= 0 fi; od: od:

#  Construction of the log likelihood function
i:='i': j:='j': k:='k':
l:=(mu,beta,sigma,omega)-> sum(
  'eps[i]*xi1[i]*ln(sum('alfa[i,j]/sigma*f(lnsum[i,j],zet[j])*omega[j]','j'=1..m-1)
   +alfa[i,m]/sigma*f(lnsum[i,m],zet[m])*(1-sum(omega[k],k=1..m-1)))
 + eps[i]*xi2[i]*ln(sum('alfa[i,j]*S(lnsum[i,j],zet[j])*omega[j]','j'=1..m-1)
   + alfa[i,m]*S(lnsum[i,m],zet[m])*(1-sum(omega[k],k=1..m-1)))
 + eps[i]*(1-xi1[i])*(1-xi2[i])*ln(sum('alfa[i,j]*
          (1-S(lnsum[i,j],zet[j]))*omega[j]','j'=1..m-1)
   + alfa[i,m]*(1-S(lnsum[i,m],zet[m]))*(1-sum(omega[k],k=1..m-1)))
 + (1-eps[i])*ln(sum('alfa[i,j]*omega[j]','j'=1..m-1)+alfa[i,m]*
          (1-sum(omega[k],k=1..m-1)))','i'=1..N):
eval(l(mu,beta,sigma,omega)):

# 3rd Step: Calculation of the Fisher information matrix
#  Determination of the Hessian matrix
matt:=hessian(l(mu,beta,sigma,omega),[mu,beta,sigma,omega[i] $ i=1..m-1]):

#  Calculation of the observed Fisher information matrix
mu:=muhat; beta:=betahat; sigma:=sigmahat;
for i from 1 to m-1 do omega[i]:=omegahat[i]: od:
hesse:=matrix(m-1+3,m-1+3):
for i from 1 to m-1+3 do for j from 1 to m-1+3 do
hesse[i,j]:= evalf(matt[i,j]):
od: od:
fishmat:=evalm(-1*hesse):

# 4th Step: Determination of 95% confidence intervals
varmat:=inverse(fishmat):
for i from 1 to m-1+3 do if varmat[i,i] < 0
then print (`Warning: there are negative variances!`)
fi; od:  # to make sure that variances are positive

#  Confidence intervals for mu,beta, and sigma;
mu:='mu':beta:='beta':sigma:='sigma':omega:='omega':
CI(mu):=[muhat-1.96*sqrt(varmat[1,1])/sqrt(361),
         muhat+1.96*sqrt(varmat[1,1])/sqrt(361)];
CI(beta):=[betahat-1.96*sqrt(varmat[2,2])/sqrt(361),
           betahat+1.96*sqrt(varmat[2,2])/sqrt(361)];
CI(sigma):=[sigmahat-1.96*sqrt(varmat[3,3])/sqrt(361),
```

```
            sigmahat+1.96*sqrt(varmat[3,3])/sqrt(361)];

#  Confidence intervals for omega;
for i from 1 to m-1 do CI(omega[i]):=
     [omegahat[i]-1.96*sqrt(varmat[i+3,i+3])/sqrt(361),
      omegahat[i]+1.96*sqrt(varmat[i+3,i+3])/sqrt(361)]; od:

#  Confidence interval for Accelerating Factor
AF:= exp(betahat);
CI('AF'):=[exp(betahat-1.96*sqrt(varmat[2,2])/sqrt(361)),
           exp(betahat+1.96*sqrt(varmat[2,2])/sqrt(361))];

#  Confidence interval for the Relative Risk
RR:=exp(-betahat/sigmahat);
Var('RR'):=RR^2/sigmahat*
   (varmat[2,2]-2*betahat/sigmahat*varmat[2,3]+(betahat/sigmahat)^2*varmat[3,3]);
CI('RR'):=[RR-1.96*sqrt(Var('RR'))/sqrt(361),RR+1.96*sqrt(Var('RR'))/sqrt(361)];
```

C.3 S-Plus programmes and functions

C.3.1 S-Plus functions

Generation of data sets: function `simgendat`

The S-Plus function `simgendat` generates a total of `nds` data sets according to model (7.1) for a data setting defined by the function arguments `nobs, distz, disty, bet, sig,` and `zens`. In case of the response variable, on average, 45% of exact observations, 45% of right-censored, and 10% of left-censored values are generated. Note that in lines 10 and 21, we rest one from `zl`. This is because S-Plus interprets any interval (in the function `kaplanMeier`) as semi-open.

```
1   "simgendat" <- function(nds, nobs, disty, distz,
                            mu, bet, sig, zens, dataout, seed) {
      set.seed(seed)
```

1. Generation of Z: 1–normal, 2–Weibull; but Z is measured on a discrete scale

```
      if (distz==1)
5          z<-round(pmax(1,rnorm(nobs*nds,50,10)))
      else z<-round(pmin(pmax(1,rweibull(nobs*nds,2,50)),100))
```

2. Generation of (zl-1,zr]: 1–avl = 16, 2–current status data

```
      if (zens==1){
        zmenos <-round(runif(nobs*nds,1,15))
        zmas   <-round(runif(nobs*nds,1,15))
```

```
10      zl <- matrix(pmax(1,z-zmenos)-1,nobs)
        zr <- matrix(z+zmas,nobs)
      }
      else{   # zens==2
        zcens <-round(runif(nobs*nds,1,15))
15      pvar  <-rbinom(nobs*nds,1,0.65)
          zcsd <- ifelse(pvar==1,z+zcens,pmax(1,z-zcens))
          sm <- rep(NA,nds)
          for(i in 1:nds)
              sm[i] <- max(zcsd[((i-1)*nobs+1):(i*nobs)])
20        sm <- rep(sm,rep(nobs,length(sm)))
          zl <- matrix(ifelse(pvar==1,1,zcsd)-1,nobs)
          zr <- matrix(ifelse(pvar==1,zcsd,sm),nobs)
        }
```

3. Generation of Y: 1–log logistic, 2–Weibull

```
      if (disty==1)
25          y <- round(exp(mu+bet*log(z)+sig*rlogis(nobs*nds)),2)
      else  y <- round(rweibull(nobs*nds,1/sig,exp(mu+bet*log(z))),2)
```

4. Generation of U: on average 45% exact obs., 45% r-censored, 10% l-censored

```
      del <- matrix(runif(nobs*nds),nobs)
      del <- ifelse(del<=0.45,1,ifelse(del <= 0.9,0,2))
      ycens <- round(runif(nobs*nds,1,15),2)
30    u <- matrix(ifelse(del==0,pmax(1,y-ycens),ifelse(del==1,y,y+ycens)),nobs)
```

5. Creation of data frame

```
      simdata <- data.frame(matrix(rbind(u,del,zl,zr),nobs))
      dimnames(simdata)[[2]] <- 1:(nds*4)  # best for immediate use in AMPL
      write.table(simdata,dataout,sep="\t",dimnames.write=T)
      simdata
35  }
```

S-Plus Function `turnb`

This S-PLUS function computes the values of the density function (on semi-closed intervals) of a survival time T given $\hat{S}(t)$ obtained with the use of the S-PLUS function `kaplanMeier`.

```
"turnb" <- function(mat)  #mat: kaplanMeier(censor(..)~..)$fits[[1]]
{
  mr <- dim(mat)[1]
  n <- ifelse(mat[1, 1] < 0 && mat[1, 2] == 0, mr, mr + 1)
  low <- rep(NA, n)
```

```
  upp <- rep(NA, n)
  prob <- rep(NA, n)
  KMlows <- mat[, 1]
  KMupps <- c(0, mat[, 2])
  KMprobs <-  - diff(c(1, mat[, 3]))
  if(mat[1, 1] < 0) {                       # Distinguishing 1st line mat
    if(mat[1, 2] != 0) {                    # case: (-\infty,a>0]
      low[1:(n - 1)] <- KMupps[1:(n - 1)]
      upp[1:(n - 1)] <- c(mat[1, 2], KMlows[2:(n - 1)])
      prob[1:(n - 1)] <- KMprobs
    }
    else {                                  # case: (-\infty,0]
      low[1:(n - 1)] <- KMupps[2:n]
      upp[1:(n - 1)] <- KMlows[2:n]
      prob[1:(n - 1)] <- KMprobs[2:n]
    }
  }
  else {                                    # case: (a,b]
    low[1:(n - 1)] <- KMupps[1:(n - 1)]
    upp[1:(n - 1)] <- KMlows
    prob[1:(n - 1)] <- KMprobs
  }
  if(mat[mr, 2] > 1e+020) {
    low[n] <- mat[mr, 1]
    upp[n] <- Inf
    prob[n] <- mat[mr, 3]
  }
  else {
    low[n] <- mat[mr, 2]
    upp[n] <- Inf
    prob[n] <- 1 - sum(prob[1:n - 1])
  }
  table <- data.frame(low, upp, prob)
  dimnames(table)[[2]] <- c("Lower", "Upper", "Probability")
  print("Turnbull Intervals")
  table
}
```

S-Plus function `expzcont`

The function `expzcont` calculates the expected values of an interval-censored variable given the intervals and the NPMLE of the density function of T.

```
"expzcont" <- function(xl, xr, mat)
{
# (xl,xr]: observed interval; mat: turnb() output
```

```
    M <- dim(mat)[1]
    midpoin <- rowMeans(mat[, 1:2])
    tbprobs <- mat[, 3]
    probsum <- midpoin * tbprobs
    M <- length(probsum[!is.na(probsum)])
    probsum <- probsum[1:M]
    tbprobs <- tbprobs[1:M]
    seqmat1 <- rep(mat[,1],each=length(xl))
    seqmat2 <- rep(mat[,2],each=length(xl))
    indiq <- matrix((xl <= seqmat1) * (xr >= seqmat2), length(xl))[, 1:M]
    zexp <- as.vector((indiq %*% probsum)/(indiq %*% tbprobs))
}
```

C.3.2 S-Plus programme for simulation study

This S-Plus programme carries out the parameter estimation of an accelerated failure time modelwith an interval-censored covariate using both midpoint and conditional mean imputation for Z for a total of D generated data sets. Besides, it estimates the quantiles of the covariate's distribution function by means of the function `qkaplanMeier`. It also uses the functions `turnb` (in line 93) and `expzcont` (line 97).

```
1   disty <- readline("Distribution of Y (1: Log logistic, 2: Weibull)? ")
    distz <- readline("Distribution of Z (1: Normal, 2: Weibull)? ")
    D <- as.numeric(readline("Number of data sets to be generated? "))

5   muorg   <- as.numeric(readline("True value of mu? "))
    betaorg <- as.numeric(readline("True value of beta? "))
    sigmaorg<- as.numeric(readline("True value of sigma? "))
    RRorg   <- round(exp(-betaorg/sigmaorg),3)
    Med20org<- round(ifelse(disty==1,exp(muorg+betaorg*log(20)),
10                  exp(muorg+betaorg*log(20))*log(2)^sigmaorg),3)
    Med50org<- round(ifelse(disty==1,exp(muorg+betaorg*log(50)),
                    exp(muorg+betaorg*log(50))*log(2)^sigmaorg),3)
    orgvals <- c(muorg,betaorg,sigmaorg,RRorg,Med20org,Med50org)

15  seed50.1 <- as.numeric(readline("Ranseed 50.1? "))
    seed50.2 <- as.numeric(readline("Ranseed 50.2? "))
    seed150.1 <-as.numeric(readline("Ranseed 150.1? "))
    seed150.2 <-as.numeric(readline("Ranseed 150.2? "))
    seed300.1 <-as.numeric(readline("Ranseed 300.1? "))
20  seed300.2 <-as.numeric(readline("Ranseed 300.2? "))
    seedlist <- list(c(seed50.1,seed50.2),c(seed150.1,seed150.2),
                                          c(seed300.1,seed300.2))

    distmod <- readline("Which distribution do you assume for Y? ")
25  results.out.cm <- readline("Output file for condmean results? ")
```

```
results.out.mp <- readline("Output file for midpoint results? ")
results.out.tb <- readline("Output file for quantile estimates? ")

data.out50.1 <- readline("Output file data50.1? ")
data.out50.2 <- readline("Output file data50.2? ")
data.out150.1 <-readline("Output file data150.1? ")
data.out150.2 <-readline("Output file data150.2? ")
data.out300.1 <-readline("Output file data300.1? ")
data.out300.2 <-readline("Output file data300.2? ")
data.outfiles <- list(c(data.out50.1,data.out50.2),
    c(data.out150.1,data.out150.2),c(data.out300.1,data.out300.2))

xqlist <- c(0.1,0.25,0.5,0.75,0.9)
quantorgs <- ifelse(rep(distz==1,5),
             qnorm(c(0.1,0.25,0.5,0.75,0.9),50,10),
             qweibull(c(0.1,0.25,0.5,0.75,0.9),2,50))

six.results.cm <- rep(list(vector("list",2)),3)
names(six.results.cm)<-c("50 observations","150 observations",
                                           "300 observations")
six.results.mp <- rep(list(vector("list",2)),3)
names(six.results.mp)<-c("50 observations","150 observations",
                                           "300 observations")
six.results.tb <- rep(list(vector("list",2)),3)
names(six.results.tb)<-c("50 observations","150 observations",
                                           "300 observations")
nobs <- c(50,150,300)
zens <- 1:2

for(k in 1:3){
names(six.results.cm[[k]]) <-c("Narrow symmetric intervals",
                               "Current status data intervals")
names(six.results.mp[[k]]) <-c("Narrow symmetric intervals",
                               "Current status data intervals")
names(six.results.tb[[k]]) <-c("Narrow symmetric intervals",
                               "Current status data intervals")
n <- nobs[k]

for(l in 1:2){
loglindat <- simgendat(D,n,disty,distz,orgvals[1],orgvals[2],orgvals[3],
                       zens[l],data.outfiles[[k]][l],seedlist[[k]][l])
attach(loglindat)

thetalist.cm <- rep(list(numeric(D)),8)
names(thetalist.cm)<-c("mu","beta","sigma","Relrisk",
                       "Med20","Med50","std.mu","std.bet")
```

```
thetalist.mp <- rep(list(numeric(D)),8)
names(thetalist.mp)<-c("mu","beta","sigma","Relrisk",
                       "Med20","Med50","std.mu","std.bet")
quantlist <- rep(list(numeric(D)),5)
names(quantlist)<-c("10%","25%","50%","75%","90%")

# Reading the data
u.all  <- as.matrix(loglindat[,seq(1,ncol(loglindat),4)])
del.all<- as.matrix(loglindat[,seq(2,ncol(loglindat),4)])
zl.all <- as.matrix(loglindat[,seq(3,ncol(loglindat),4)])
zr.all <- as.matrix(loglindat[,seq(4,ncol(loglindat),4)])

for (d in 1:D){
u  <-  u.all[,d]
del<-del.all[,d]
zl <- zl.all[,d]
zr <- zr.all[,d]

# Calculus of the Turnbull estimator $\hat{F}_Z$
kap <- kaplanMeier(censor(zl,zr,rep(3,n))~1,
                   control=censorReg.control(1000,10000,0.001))
turnbprob <- round(cbind(turnb(kap$fits[[1]]),
                   cumsum(turnb(kap$fits[[1]])[,3])),4)

# Calculus of the conditional expected value of Z given $\hat{F}_Z$
zexp <- expzcont(zl,zr,turnbprob[,1:3])

# Adjusting the log linear model with censorReg()
censreg.cm<- censorReg(censor(u,u,del)~log(zexp),
                       dist=distmod,na.action=na.exclude)

mu.cm     <- censreg.cm$coeff[1]
beta.cm   <- censreg.cm$coeff[2]
sigma.cm  <- censreg.cm$scale
std.mu.cm <- sqrt(censreg.cm$var[1,1])
std.bet.cm<- sqrt(censreg.cm$var[2,2])
RR.cm     <- ifelse(exp(-beta.cm/sigma.cm)<75,exp(-beta.cm/sigma.cm),NA)
Med20.cm  <- ifelse(disty==1,exp(mu.cm+beta.cm*log(20)),
                             exp(mu.cm+beta.cm*log(20))*log(2)^sigma.cm)
Med50.cm  <- ifelse(disty==1,exp(mu.cm+beta.cm*log(50)),
                             exp(mu.cm+beta.cm*log(50))*log(2)^sigma.cm)
Med20.cm  <- ifelse(Med20.cm<2000,Med20.cm,NA)
Med50.cm  <- ifelse(Med50.cm<2000,Med50.cm,NA)

# Writing $\hat{\theta}$ into a list
thetalist.cm$mu[d]     <-mu.cm
```

```
thetalist.cm$beta[d]   <-beta.cm
thetalist.cm$sigma[d]  <-sigma.cm
thetalist.cm$Relrisk[d]<-RR.cm
thetalist.cm$Med20[d]  <-Med20.cm
thetalist.cm$Med50[d]  <-Med50.cm
thetalist.cm$std.mu[d] <-std.mu.cm
thetalist.cm$std.bet[d]<-std.bet.cm

# Calculus of the midpoints of (zl,zr]
zmidp <- rowMeans(cbind(zl,zr))

# Adjusting the log linear model with censorReg()
censreg.mp<- censorReg(censor(u,u,del)~log(zmidp),dist=distmod,
                                          na.action=na.exclude)

mu.mp     <- censreg.mp$coeff[1]
beta.mp   <- censreg.mp$coeff[2]
sigma.mp  <- censreg.mp$scale
std.mu.mp <- sqrt(censreg.mp$var[1,1])
std.bet.mp<- sqrt(censreg.mp$var[2,2])
RR.mp     <- ifelse(exp(-beta.mp/sigma.mp)<75,exp(-beta.mp/sigma.mp),NA)
Med20.mp  <- ifelse(disty==1,exp(mu.mp+beta.mp*log(20)),
                             exp(mu.mp+beta.mp*log(20))*log(2)^sigma.mp)
Med50.mp  <- ifelse(disty==1,exp(mu.mp+beta.mp*log(50)),
                             exp(mu.mp+beta.mp*log(50))*log(2)^sigma.mp)
Med20.mp  <- ifelse(Med20.mp<2000,Med20.mp,NA)
Med50.mp  <- ifelse(Med50.mp<2000,Med50.mp,NA)


# Writing $\hat{\theta}$ into a list
thetalist.mp$mu[d]     <-mu.mp
thetalist.mp$beta[d]   <-beta.mp
thetalist.mp$sigma[d]  <-sigma.mp
thetalist.mp$Relrisk[d]<-RR.mp
thetalist.mp$Med20[d]  <-Med20.mp
thetalist.mp$Med50[d]  <-Med50.mp
thetalist.mp$std.mu[d] <-std.mu.mp
thetalist.mp$std.bet[d]<-std.bet.mp

# Calculus of the five quantiles given $\hat{F}_Z$
quant <- qkaplanMeier(kaplanMeier(censor(zl,zr,rep(3,n))~1,
                      control=censorReg.control(1000,10000,0.001)),xqlist)
quant <- ifelse(quant==Inf,NA,quant)

# Writing the estimated quantiles into a list
quantlist$"10%"[d] <-quant[1]
```

```
   quantlist$"25%"[d] <-quant[2]
   quantlist$"50%"[d] <-quant[3]
   quantlist$"75%"[d] <-quant[4]
   quantlist$"90%"[d] <-quant[5]

   rm(u,del,zl,zr)
   rm(kap,turnbprob,zexp,censreg.cm)
   rm(mu.cm,beta.cm,sigma.cm,std.mu.cm,std.bet.cm,RR.cm,Med20.cm,Med50.cm)
   rm(zmidp,censreg.mp)
   rm(mu.mp,beta.mp,sigma.mp,std.mu.mp,std.bet.mp,RR.mp,Med20.mp,Med50.mp)
   rm(quant)
   }

   # Calculus of $\overline{\hat{\theta}}$ and its variance
   results.cm <- list(rep(NA,5),rep(NA,5),rep(NA,5),rep(NA,6),rep(NA,6),
                                                        rep(NA,6),NA,NA)
   results.mp <- list(rep(NA,5),rep(NA,5),rep(NA,5),rep(NA,6),rep(NA,6),
                                                        rep(NA,6),NA,NA)
   results.tb <- list(rep(NA,6),rep(NA,6),rep(NA,6),rep(NA,6),rep(NA,6))

   for(i in 1:8){
      results.cm[[i]][1] <- round(mean(thetalist.cm[[i]],na.rm=T),3)
      results.mp[[i]][1] <- round(mean(thetalist.mp[[i]],na.rm=T),3)
   }

   for(i in 1:6){
      results.cm[[i]][2] <- round(stdev(thetalist.cm[[i]],na.rm=T),3)
      results.cm[[i]][3] <- results.cm[[i]][1]-orgvals[i]
      results.cm[[i]][4] <- round(results.cm[[i]][3]/orgvals[i],3)
      results.cm[[i]][5] <- round(results.cm[[i]][2]^2+results.cm[[i]][3]^2,3)

      results.mp[[i]][2] <- round(stdev(thetalist.mp[[i]],na.rm=T),3)
      results.mp[[i]][3] <- results.mp[[i]][1]-orgvals[i]
      results.mp[[i]][4] <- round(results.mp[[i]][3]/orgvals[i],3)
      results.mp[[i]][5] <- round(results.mp[[i]][2]^2+results.mp[[i]][3]^2,3)
   }

   for(i in 4:6){
      results.cm[[i]][6] <- sum(!is.na(thetalist.cm[[i]]))
      results.mp[[i]][6] <- sum(!is.na(thetalist.mp[[i]]))
   }

   for(i in 1:5){
      results.tb[[i]][1] <- round(mean(quantlist[[i]],na.rm=T),2)
      results.tb[[i]][2] <- round(stdev(quantlist[[i]],na.rm=T),2)
      results.tb[[i]][3] <- round(results.tb[[i]][1]-quantorgs[i],2)
```

```
    results.tb[[i]][4] <- round(results.tb[[i]][3]/quantorgs[i],2)
    results.tb[[i]][5] <- round(results.tb[[i]][2]^2+results.tb[[i]][3]^2,2)
    results.tb[[i]][6] <- sum(!is.na(quantlist[[i]]))
  }

  names(results.cm)<-
      c( "Mu: Mean, Stdev., Bias, rel.Bias, MSE",
       "Beta: Mean, Stdev., Bias, rel.Bias, MSE",
      "Sigma: Mean, Stdev., Bias, rel.Bias, MSE",
      "RRisk: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
      "Med20: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
      "Med50: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
      "Mean of Stdev.'s of mu",
      "Mean of Stdev.'s of beta")
  names(results.mp)<-
      c( "Mu: Mean, Stdev., Bias, rel.Bias, MSE",
       "Beta: Mean, Stdev., Bias, rel.Bias, MSE",
      "Sigma: Mean, Stdev., Bias, rel.Bias, MSE",
      "RRisk: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
      "Med20: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
      "Med50: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
      "Mean of Stdev.'s of mu",
      "Mean of Stdev.'s of beta")
  names(results.tb)<-
      c("10% quantile: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
        "25% quantile: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
        "50% quantile: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
        "75% quantile: Mean, Stdev., Bias, rel.Bias, MSE, Valids",
        "90% quantile: Mean, Stdev., Bias, rel.Bias, MSE, Valids")

  six.results.cm[[k]][[l]]<-results.cm
  six.results.mp[[k]][[l]]<-results.mp
  six.results.tb[[k]][[l]]<-results.tb
  rm(loglindat,results.cm,results.mp,results.tb,thetalist.cm,
                                  thetalist.mp,quantlist,nquant)
  rm(u.all,del.all,zl.all,zr.all)
  }
  }
  rm(seedlist,seed50.1,seed50.2,seed150.1,seed150.2,seed300.1,seed300.2)
  rm(orgvals,muorg,betaorg,sigmaorg,RRorg,
      Med20org,Med50org,quantorgs,xqlist,D)
  rm(data.outfiles,data.out50.1,data.out50.2,data.out150.1,
     data.out150.2,data.out300.1,data.out300.2)
  rm(nobs,n,zens,disty,distz,distmod)

  dput(six.results.cm,results.out.cm)
```

```
dput(six.results.mp,results.out.mp)
dput(six.results.tb,results.out.tb)
finresult<-list(six.results.cm,six.results.mp,six.results.tb)
names(finresult)<-c("Conditional Mean Imputation","Midpoint Imputation",
                    "Turnbull's Quantile Estimation")

rm(results.out.cm,results.out.mp,results.out.tb)
rm(six.results.cm,six.results.mp,six.results.tb)
finresult
```