Chapter 6

RESUMÉ

This thesis focuses on the analysis of insurance policy lifetimes in the form of **grouped data**, where the lifetime of a policy is measured from the inception date (entry month) up to the lapsing date (month in which policy lapsed) or a pre-determined cut-off date. Data from the insurance industry are extensive data sets with very large sample sizes. The focus in this thesis is on the estimation of lifetime distributions, based on a large sample of discrete lifetimes of policies that are grouped into intervals of lifetimes. The aim of the research is the statistical modelling of **parametric** survival distributions of **grouped survival data** of long- and shortterm policies in the insurance industry, by means of a method of maximum likelihood estimation **subject to constraints**. This scenario has become extremely important, not only for application in the actuarial context, but also in other fields.

In this thesis, the analysis takes account of the actual lifetime (duration) of the policy rather than just recording the fact that the policy lapsed or was still alive after (say) twelve months. In other words, the response variable, lifetime, is a continuous one, and the whole distribution of lifetimes can be used. A **general experimental design** admits that all the policies can be used in the analysis, even those policies with inception dates very close to the cut-off point.

Special attention has been given to **staggered entry** of policies, where policies written in different months or time-periods have different entry times.

The methodology of maximum likelihood estimation subject to constraints, used in this thesis, leads to **explicit expressions** for the estimates of the parameters, as well as for approximated variances and covariances of the estimates, which gives **exact** maximum likelihood estimates of the parameters. This makes direct extension to more complex designs feasible.

Once the parameters of the survival distributions have been estimated, estimated hazard and survivor functions, odds of a lapse, **odds ratios** and **hazard ratios** at time t can be directly calculated, as well as estimated percentiles for the fitted survival distributions. These estimates form the statistical foundation for scientific decisionmaking with respect to actuarial design, maintenance and marketing of insurance policies.

Parametric regression models are fitted and important indicators of the effect of the covariates are defined such as risk scores (hazard ratios) and indices (odds ratios). This is in contrast to the famous semiparametric Cox's proportional hazards model. David Oakes states in the chapter on Survival Analysis in [39] that "following the incorporation of software for fitting proportional hazards models into packages such as BMDP and SAS, this model of Cox, for better or worse, became standard for the analysis of survival data. But the assumption of proportional hazards has no compelling mathematical justification and is often found to be false in applications."

It is generally assumed in the actuarial industry that incorrect assumptions regarding lifetime distributions have severe implications with respect to lapse probabilities and estimated income. For this reason non-parametric and semi-parametric models are standard practice. The implications of a proper and sound parametric model are far-reaching for the characterization of lapse probabilities and income, which are the corner stones of the insurance industry. It directly determines lapse indices in terms of risk factors, including the period of lapsing under consideration. It also determines the behaviour of such indices over time. It can also be assumed that the use of grouped data for determining lifetime distributions is more robust with respect to wrong assumptions than continuous data.

In this way a contribution is made to the global handling of lapse indices and risk scores. For example, if a log-logistic distribution holds, the indices are constant, independent of the lapsing period. If a Weibull distribution holds, the risk scores are constant, independent of the lapsing period. In any event, if the lifetime distribution is known, the lapse indices for any time period are known.

A complete exposition of these structural relationships and practical implications of it must be investigated further.

Although the methodology in this thesis is developed specifically for the insurance industry, it may be applied in the normal context of research and scientific decisionmaking, that includes for example survival distributions for the **medical, biological, engineering, econometric and sociological sciences**.

The potential for extending the methodology to other realistic practical application is unlimited. This can be to the advance of the insurance industry in general. The models should reflect an interactive adaptability for direct application in practice by salesforce (the marketing people on ground level), as well as for actuarial planning.

Appendix A

COMPUTER PROGRAMS

The SAS/IML programs appear under the appropriate chapter heading.

A.1 Chapter 3: Maximum Likelihood Estimation

A.1.1 A Fixed Censoring Time Standard Program using PROC LIFEREG

Program for fitting a single survival distribution to grouped survival data

```
options nodate pagesize=500 pageno=1;
libname hsbc 'c:\hsbc1\sd2';

title1 'Fitting of a single survival model: the standard SAS method';
title2 'Fixed censoring time';

data fin;
input lower upper freq;
cards;
 .    12     66
12    17    158
17    24    254
24    28    157
28    34    250
34    37     35
37     .   1666
;

proc lifereg data=fin covout outest=ops;
model (lower, upper) =  / dist=weibull;       *default is Weibull;
model (lower, upper) =  / dist=llogistic;
model (lower, upper) =  / dist=lnormal;
weight freq;
output out=weib cdf=cdf predicted=months
quantiles = 0.02 to 0.98 by .02 control=c;
title 'Fit single Weibull curve (SAS method)';
title2 'Fixed censoring time';

data par;
set ops;
if _N_=1;
lambdaSAS=exp(-intercept/_scale_);
```

```
alphaSAS=1/_scale_;
oualphaSAS=-intercept/_scale_;

proc print data=par;
var lambdaSAS oualphaSAS alphaSAS;
run;
```

A.1.2 Staggered Entry of Policies Standard Program using PROC LIFEREG

Program for fitting one Weibull/log-logistic/lognormal distribution to the four histograms of the entry groups

```
options nodate pagesize=500 pageno=1;
libname hsbc 'c:\hsbc1\sd2';

title1 'STAGGERED ENTRY OF POLICIES AT FOUR ENTRY TIMES';
title2 'Fit one survival distribution to the four histograms of the entry groups';
title3 'Standard SAS method';

data fin;
input lower upper freq;
cards;
 .    12     66
12    17    158
17    24    254
24    28    157
28    34    250
34    37     35
37     .   1666
 .    12    118
12    17    166
17    24    229
24    28    200
28    34    172
34     .   1924
 .    12    154
12    17     99
17    24    242
24    28    117
28     .   1674
 .    12    175
12    17    166
17    24    207
24     .   1848
;

proc lifereg data=fin covout outest=ops;
model (lower, upper) =  / dist=weibull;       *default is Weibull;
model (lower, upper) =  / dist=llogistic;
model (lower, upper) =  / dist=lnormal;
weight freq;
output out=weib cdf=cdf predicted=months
quantiles = 0.02 to 0.98 by .02 control=c;
title 'Fit single Weibull curve (SAS method)';
title2 'Four entry dates';

data par;
set ops;
if _N_=1;
lambdaSAS=exp(-intercept/_scale_);
alphaSAS=1/_scale_;
oualphaSAS=-intercept/_scale_;

proc print data=par;
var lambdaSAS oualphaSAS alphaSAS;
run;
```

A.2 Chapter 3: M L Estimation subject to Constraints

A.2.1 A Fixed Censoring Time - IML Programs

1. **Program for fitting a Weibull distribution to grouped survival data**

```
proc iml worksize=60;
reset nolog;

**********Frequency vector;
f={66,158,254,157,250,35,1666};

**********Vector of upper boundaries;
x={12,17,24,28,34,37};

**********Relative frequency vector;
n=f[+];
k=nrow(f);
d=k-1;
p=f/n;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d,1,1)@cusum(J(1,k,1));
S2=J(1,k,1)@cusum(J(d,1,1));
S=S1<=S2;  print S;

X1=J(d,1,1)||log(x);         print X1;
C=I(d)-X1*inv(X1`*X1)*X1`;  print C;
projmatrix=X1*inv(X1`*X1)*X1`;  print projmatrix;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;  print p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;              *Weibull;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>1e-6);
     j=j+1;
     p1=p;
     ps=S*p;
     g=C*log(-log(1-ps));                                 *Weibull;
     Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;          *Weibull;
*********covariance matrix;
sig=(1/n)*(diag(m)-m*m`);
V=sig;
***************************;
     p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                        *Weibull;
     verskil=sqrt((p-p1)`*(p-p1));
     print i j p m;
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;     print m;
end;

**********Parameter vector for linear model;
par=inv(x1`*x1)*x1`*log(-log(1-ms));  print par;          *Weibull;

**********Parameters for Weibull model;
lambda=exp(par[1]);
alpha=par[2];

print 'Weibull parameters: MLE subject to constraints';
print 'lambda=' lambda  'alpha=' alpha;

**********Compute Wald statistic;
p=p0;
ps=S*p;
Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;              *Weibull;
g=C*log(-log(1-ps));                                      *Weibull;
******covariance matrix;
sig=(1/n)*(diag(p)-p*p`);
```

```
V=sig;
*************************;
wald=g`*ginv(Gp*V*Gp`)*g; nu=eigval(C); nu=nu[+];
discr=wald/n;
prob=1-probchi(wald,nu) ;
alpha=par[2];
Gini=1-0.5##(1/alpha);
print 'Measure of fit';
print 'Wald='wald 'Discrepancy=' discr;
print 'prob=' prob 'degrees of freedom=' nu 'Gini=' Gini;
```

2. **Program for fitting a log-logistic distribution to grouped survival data**

```
proc iml worksize=60;
reset nolog;

**********Frequency vector;
f={66,158,254,157,250,35,1666};

**********Vector of upper boundaries;
x={12,17,24,28,34,37};

**********Relative frequency vector;
n=f[+];
k=nrow(f);
d=k-1;
p=f/n;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d,1,1)@cusum(J(1,k,1));
S2=J(1,k,1)@cusum(J(d,1,1));
S=S1<=S2;  print S;

X1=J(d,1,1)||log(x);         print X1;
C=I(d)-X1*inv(X1`*X1)*X1`;  print C;
projmatrix=X1*inv(X1`*X1)*X1`;  print projmatrix;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;  print p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
Gm=C*(diag(1/ms)+diag(1/(1-ms)))*S;                          *loglogistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
     g=C*(log(ps)-log(1-ps));                                *loglogistic;
     Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                     *loglogistic;
*********covariance matrix;
sig=(1/n)*(diag(m)-m*m`);
V=sig;
***************************;
     p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                           *loglogistic;
     verskil=sqrt((p-p1)`*(p-p1));
     print i j p m;
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;     print m;
end;

**********Parameter vector for linear model;
par=inv(x1`*x1)*x1`*(log(ms)-log(1-ms));  print par;       *loglogistic;

**********Parameters for loglogistic model;
lambda=exp(par[1]);
alpha=par[2];

print 'Loglogistic parameters: MLE subject to constraints';
print 'lambda=' lambda  'alpha=' alpha;

**********Compute Wald statistic;
p=p0;
ps=S*p;
Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                          *loglogistic;
g=C*(log(ps)-log(1-ps));                                     *loglogistic;
******covariance matrix;
```

```
sig=(1/n)*(diag(p)-p*p');
V=sig;
*************************;
wald=g'*ginv(Gp*V*Gp')*g; nu=eigval(C); nu=nu[+];
discr=wald/n;
prob=1-probchi(wald,nu) ;
alpha=par[2];
Gini=1-0.5##(1/alpha);
print 'Measure of fit';
print 'Wald='wald 'Discrepancy=' discr;
print 'prob=' prob 'degrees of freedom=' nu 'Gini=' Gini;
```

3. Program for fitting a lognormal distribution to grouped survival data

```
proc iml worksize=60;
reset nolog;

**********Frequency vector;
f={66,158,254,157,250,35,1666};

**********Vector of upper boundaries;
x={12,17,24,28,34,37};
x=log(x);

**********Relative frequency vector;
n=f[+];
k=nrow(f);
d=k-1;
p=f/n;

pi=(gamma(0.5))##2;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d,1,1)@cusum(J(1,k,1));
S2=J(1,k,1)@cusum(J(d,1,1));
S=S1<=S2;  print S;

X1=J(d,1,1)||log(x);         print X1;
C=I(d)-X1*inv(X1'*X1)*X1';  print C;
projmatrix=X1*inv(X1'*X1)*X1';  print projmatrix;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;  print p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
arg1=2;
arg2=3;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);
mu=par[1];sigma=par[2];
Gm=C*diag(sqrt(2#pi)/(exp(-(x-mu)#(x-mu)/2/sigma/sigma)))*S; *lognormal;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>1e-6);
     j=j+1;
     p1=p;
     ps=S*p;
     g=C*probit(ps);                                      *lognormal;
     parp=inv( J(2,1,1)||probit(ps[arg1]//ps[arg2]) )*(x[arg1]//x[arg2]);
     mup=parp[1];sigmap=parp[2];
     Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S; *lognormal;
*****covariance matrix;
       sig=(1/n)*(diag(m)-m*m');
       V=sig;
**********************;
     p=p-(Gm*V)'*ginv(Gp*V*Gm')*g;                        *lognormal;
     verskil=sqrt((p-p1)'*(p-p1));
     end;
verskil1=sqrt((p-m)'*(p-m));
m=p;ms=S*m;     print m;
end;

**********Parameter vector for linear model;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);

**********Parameters for Lognormal model;
mu=par[1]; sigma=par[2];

print 'Lognormal parameters: MLE subject to constraints';
print 'mu=' mu  'sigma=' sigma;

**********Compute Wald statistic;
p=p0;
ps=S*p;
  Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S;
  g=C*probit(ps);
*****covariance matrix;
sig=(1/n)*(diag(p)-p*p`);
V=sig;
**********************;
wald=(g)`*ginv(Gp*V*Gp`)*g; nu=eigval(C); nu=nu[+];
discr=wald/n;
prob=1-probchi(wald,nu) ;
print 'Measure of fit';
print 'Wald='wald 'Discrepancy=' discr;
print 'prob=' prob 'degrees of freedom=' nu;
```

A.2.2 Staggered Entry of Policies - IML Programs

1. **Program for fitting one survival distribution to the four histograms**

Program for fitting one Weibull/log-logistic distribution to the four histograms of the entry groups

```
title1 'STAGGERED ENTRY OF POLICIES AT FOUR ENTRY TIMES';
title2 'Fit one survival distribution to the four histograms of the entry groups';
title3 'Constraints: specified model';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector;
f1={66,158,254,157,250,35,1666};

**********Vector of upper boundaries;
x1={12,17,24,28,34,37};

**********Frequency vector;
f2={118,166,229,200,172,1924};

**********Vector of upper boundaries;
x2={12,17,24,28,34};

**********Frequency vector;
f3={154,99,242,117,1674};

**********Vector of upper boundaries;
x3={12,17,24,28};

**********Frequency vector;
f4={175,166,207,1848};

**********Vector of upper boundaries;
x4={12,17,24};

**********Relative frequency vectors;
n1=f1[+]; n2=f2[+]; n3=f3[+]; n4=f4[+]; n=n1+n2+n3+n4;
k1=nrow(f1); d1=k1-1;
k2=nrow(f2); d2=k2-1;
k3=nrow(f3); d3=k3-1;
k4=nrow(f4); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;
p1=f1/n1; p2=f2/n2; p3=f3/n3; p4=f4/n4;
p=p1//p2//p3//p4;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);

lx1=J(d1,1,1)||log(x1);
lx2=J(d2,1,1)||log(x2);
lx3=J(d3,1,1)||log(x3);
lx4=J(d4,1,1)||log(x4);

xc=lx1//lx2//lx3//lx4;
C=I(d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;  print p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;             *Weibull;
*Gm=C*(diag(1/ms)+diag(1/(1-ms)))*S;                    *Loglogistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>1e-6);
     j=j+1;
     p1=p;
     ps=S*p;
     g=C*log(-log(1-ps));                                    *Weibull;
    *g=C*(log(ps)-log(1-ps));                                *Loglogistic;
     Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;             *Weibull;
    *Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                     *loglogistic;
**********covariance matrix;
m1=m[1:k1];
m2=m[k1+1:k1+k2];
m3=m[k1+k2+1:k1+k2+k3];
m4=m[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(m1)-m1*m1');
sig2=(1/n2)*(diag(m2)-m2*m2');
sig3=(1/n3)*(diag(m3)-m3*m3');
sig4=(1/n4)*(diag(m4)-m4*m4');
sig=block(sig1,sig2,sig3,sig4);
V=sig;
***************************;
     p=p-(Gm*V)'*ginv(Gp*V*Gm')*g;                           *Weibull;
    *p=p-(Gm*V)'*ginv(Gp*V*Gm')*g;                           *loglogistic;
     verskil=sqrt((p-p1)'*(p-p1));
     end;
verskil1=sqrt((p-m)'*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
par=inv(xc'*xc)*xc'*log(-log(1-ms));                          *Weibull;
*par=inv(xc'*xc)*xc'*(log(ms)-log(1-ms));                     *Loglogisties;
print par;

**********Parameters for Weibull(*Loglogistic) model;
oualpha=par[1];
lambda=exp(par[1]);
alpha=par[2];

print 'Weibull(*Loglogistic) parameters: MLE subject to constraints';
print 'lambda=' lambda oualpha 'alpha=' alpha;

**********Hazard and Survival function, Odds*********************************;
whaz12=(lambda*alpha*12**(alpha-1))/(1+lambda*12**alpha);
whaz24=(lambda*alpha*24**(alpha-1))/(1+lambda*24**alpha);
wsurv12=(1+lambda*12**alpha)**(-1);
wsurv24=(1+lambda*24**alpha)**(-1);
wodds12=(1-wsurv12)/wsurv12;
wodds24=(1-wsurv24)/wsurv24;

           /*
llhaz12=(lambda*alpha*12**(alpha-1))/(1+lambda*12**alpha);
llhaz24=(lambda*alpha*24**(alpha-1))/(1+lambda*24**alpha);
llsurv12=(1+lambda*12**alpha)**(-1);
llsurv24=(1+lambda*24**alpha)**(-1);
llodds12=(1-llsurv12)/llsurv12;
llodds24=(1-llsurv24)/llsurv24;
           */

print whaz12 whaz24 wsurv12 wsurv24 wodds12 wodds24;
*print llhaz12 llhaz24 llsurv12 llsurv24 llodds12 llodds24;

**********Percentiles*******************************************************;
wmedian=((1/lambda)#log(2))##(1/alpha);

 wperc5=((1/lambda)#log(100/(100- 5)))##(1/alpha);
wperc10=((1/lambda)#log(100/(100-10)))##(1/alpha);
wperc20=((1/lambda)#log(100/(100-20)))##(1/alpha);
wperc25=((1/lambda)#log(100/(100-25)))##(1/alpha);
wperc30=((1/lambda)#log(100/(100-30)))##(1/alpha);
wperc40=((1/lambda)#log(100/(100-40)))##(1/alpha);
wperc50=((1/lambda)#log(100/(100-50)))##(1/alpha);
wperc60=((1/lambda)#log(100/(100-60)))##(1/alpha);
wperc70=((1/lambda)#log(100/(100-70)))##(1/alpha);
wperc75=((1/lambda)#log(100/(100-75)))##(1/alpha);
wperc80=((1/lambda)#log(100/(100-80)))##(1/alpha);
wperc90=((1/lambda)#log(100/(100-90)))##(1/alpha);
wperc95=((1/lambda)#log(100/(100-95)))##(1/alpha);

                /*
lmedian=(1/lambda)##(1/alpha);

 llperc5=((1/lambda)#( 5/(100- 5)))##(1/alpha);
llperc10=((1/lambda)#(10/(100-10)))##(1/alpha);
llperc20=((1/lambda)#(20/(100-20)))##(1/alpha);
llperc25=((1/lambda)#(25/(100-25)))##(1/alpha);
llperc30=((1/lambda)#(30/(100-30)))##(1/alpha);
```

```
llperc40=((1/lambda)#(40/(100-40)))##(1/alpha);
llperc50=((1/lambda)#(50/(100-50)))##(1/alpha);
llperc60=((1/lambda)#(60/(100-60)))##(1/alpha);
llperc70=((1/lambda)#(70/(100-70)))##(1/alpha);
llperc75=((1/lambda)#(75/(100-75)))##(1/alpha);
llperc80=((1/lambda)#(80/(100-80)))##(1/alpha);
llperc90=((1/lambda)#(90/(100-90)))##(1/alpha);
llperc95=((1/lambda)#(95/(100-95)))##(1/alpha);

                 */

print wmedian;
print wperc5  wperc10 wperc20 wperc25 wperc30;
print wperc40 wperc50 wperc60 wperc70;
print wperc75 wperc80 wperc90 wperc95;
*print lmedian;
*print llperc5  llperc10 llperc20 llperc25 llperc30;
*print llperc40 llperc50 llperc60 llperc70;
*print llperc75 llperc80 llperc90 llperc95;

**********Compute Wald statistic*********************************************;
p=p0;
ps=S*p;

Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;                *Weibull;
*Gp=C*diag(1/ps+1/(1-ps))*S;                               *Loglogistic;
g=C*log(-log(1-ps));                                       *Weibull;
*g=C*(log(ps)-log(1-ps));                                  *Loglogistic;

******covariance matrix;
p1=p[1:k1];
p2=p[k1+1:k1+k2];
p3=p[k1+k2+1:k1+k2+k3];
p4=p[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(p1)-p1*p1`);
sig2=(1/n2)*(diag(p2)-p2*p2`);
sig3=(1/n3)*(diag(p3)-p3*p3`);
sig4=(1/n4)*(diag(p4)-p4*p4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
*************************;
wald=g`*ginv(Gp*V*Gp`)*g; nu=eigval(C); nu=nu[+];
discr=wald/n;
prob=1-probchi(wald,nu) ;
alpha=par[2];
Gini=1-0.5##(1/alpha);
print 'Measure of fit';
print 'Wald='wald 'Discrepancy=' discr;
print 'prob=' prob 'degrees of freedom=' nu 'Gini=' Gini;
```

Program for fitting one lognormal distribution to the four histograms of the entry groups

```
title1 'STAGGERED ENTRY OF POLICIES AT FOUR ENTRY TIMES';
title2 'Fit one lognormal distribution to the four entry groups';
title3 'Constraints: specified model';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector;
f1={66,158,254,157,250,35,1666};

**********Vector of upper boundaries;
x1={12,17,24,28,34,37};

**********Frequency vector;
f2={118,166,229,200,172,1924};

**********Vector of upper boundaries;
x2={12,17,24,28,34};

**********Frequency vector;
f3={154,99,242,117,1674};

**********Vector of upper boundaries;
x3={12,17,24,28};

**********Frequency vector;
f4={175,166,207,1848};

**********Vector of upper boundaries;
x4={12,17,24};

**********Relative frequency vectors;
n1=f1[+]; n2=f2[+]; n3=f3[+]; n4=f4[+]; n=n1+n2+n3+n4;
k1=nrow(f1); d1=k1-1;
```

```
k2=nrow(f2); d2=k2-1;
k3=nrow(f3); d3=k3-1;
k4=nrow(f4); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;
p1=f1/n1; p2=f2/n2; p3=f3/n3; p4=f4/n4;
p=p1//p2//p3//p4;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);   print S;

lx1=J(d1,1,1)||log(x1);
lx2=J(d2,1,1)||log(x2);
lx3=J(d3,1,1)||log(x3);
lx4=J(d4,1,1)||log(x4);

xc=lx1//lx2//lx3//lx4;      print xc;
C=I(d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;  print p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
arg1=2;
arg2=3;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);
mu=par[1];sigma=par[2];
Gm=C*diag(sqrt(2#pi)/(exp(-(x-mu)#(x-mu)/2/sigma/sigma)))*S;         *lognormal;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>1e-6);
     j=j+1;
     p1=p;
     ps=S*p;
     g=C*probit(ps);                                               *lognormal;
     parp=inv( J(2,1,1)||probit(ps[arg1]//ps[arg2]) )*(x[arg1]//x[arg2]);
     mup=parp[1];sigmap=parp[2];
     Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S; *lognormal;
**********covariance matrix;
m1=m[1:k1];
m2=m[k1+1:k1+k2];
m3=m[k1+k2+1:k1+k2+k3];
m4=m[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(m1)-m1*m1`);
sig2=(1/n2)*(diag(m2)-m2*m2`);
sig3=(1/n3)*(diag(m3)-m3*m3`);
sig4=(1/n4)*(diag(m4)-m4*m4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
****************************;
     p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                                  *lognormal;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);  *lognormal;

**********Parameters for Lognormal model;
mu=par[1]; sigma=par[2];

print 'Lognormal parameters: MLE subject to constraints';
print 'mu=' mu  'sigma=' sigma;

**********Compute Wald statistic;
p=p0;
ps=S*p;
```

```
Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S;    *lognormal;
g=C*probit(ps);                                                     *lognormal;
******covariance matrix;
p1=p[1:k1];
p2=p[k1+1:k1+k2];
p3=p[k1+k2+1:k1+k2+k3];
p4=p[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(p1)-p1*p1`);
sig2=(1/n2)*(diag(p2)-p2*p2`);
sig3=(1/n3)*(diag(p3)-p3*p3`);
sig4=(1/n4)*(diag(p4)-p4*p4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
*************************;
wald=g`*ginv(Gp*V*Gp`)*g; nu=eigval(C); nu=nu[+];
discr=wald/n;
prob=1-probchi(wald,nu) ;
alpha=par[2];
Gini=1-0.5##(1/alpha);
print 'Measure of fit';
print 'Wald='wald 'Discrepancy=' discr;
print 'prob=' prob 'degrees of freedom=' nu 'Gini=' Gini;
```

2. **Programs for fitting four survival distributions to the four histograms and then set the four sets of parameters equal**

Program for fitting four Weibull/log-logistic distributions to the four histograms and then set the lambda's equal and the alpha's equal

```
title1 'STAGGERED ENTRY OF POLICIES AT FOUR ENTRY TIMES';
title2 'Fit four survival distributions to the four entry groups';
title3 'Restrictions: specified model AND set lambda's equal and alpha's equal';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector;
f1={66,158,254,157,250,35,1666};

**********Vector of upper boundaries;
x1={12,17,24,28,34,37};

**********Frequency vector;
f2={118,166,229,200,172,1924};

**********Vector of upper boundaries;
x2={12,17,24,28,34};

**********Frequency vector;
f3={154,99,242,117,1674};

**********Vector of upper boundaries;
x3={12,17,24,28};

**********Frequency vector;
f4={175,166,207,1848};

**********Vector of upper boundaries;
x4={12,17,24};

**********Relative frequency vectors;
n1=f1[+]; n2=f2[+]; n3=f3[+]; n4=f4[+]; n=n1+n2+n3+n4;
k1=nrow(f1); d1=k1-1;
k2=nrow(f2); d2=k2-1;
k3=nrow(f3); d3=k3-1;
k4=nrow(f4); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;
p1=f1/n1; p2=f2/n2; p3=f3/n3; p4=f4/n4;
p=p1//p2//p3//p4;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);   print S;

lx1=J(d1,1,1)||log(x1);
lx2=J(d2,1,1)||log(x2);
lx3=J(d3,1,1)||log(x3);
```

```
lx4=J(d4,1,1)||log(x4);

xc=block(lx1,lx2,lx3,lx4);
C=I(d)-xc*inv(xc`*xc)*xc`;
CP=C;
C=C//({1 0 -1 0 0 0 0 0, 1 0 0 0 -1 0 0 0, 1 0 0 0 0 0 -1 0,
       0 1 0 -1 0 0 0 0, 0 1 0 0 0 -1 0 0, 0 1 0 0 0 0 0 -1} *inv(xc`*xc)*xc`);
print xc C;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;  print p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;                 *Weibull;
*Gm=C*(diag(1/ms)+diag(1/(1-ms)))*S;                        *Loglogistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>1e-6);
     j=j+1;
     p1=p;
     ps=S*p;
     g=C*log(-log(1-ps));                                   *Weibull;
    *g=C*(log(ps)-log(1-ps));                               *Loglogistic;
     Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;            *Weibull;
    *Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                    *Loglogistic;
**********covariance matrix;
m1=m[1:k1];
m2=m[k1+1:k1+k2];
m3=m[k1+k2+1:k1+k2+k3];
m4=m[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(m1)-m1*m1`);
sig2=(1/n2)*(diag(m2)-m2*m2`);
sig3=(1/n3)*(diag(m3)-m3*m3`);
sig4=(1/n4)*(diag(m4)-m4*m4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
****************************;
     p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                          *Weibull;
    *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                          *Loglogistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
par=inv(xc`*xc)*xc`*log(-log(1-ms));  print par;            *Weibull;
*par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms));                   *Loglogistic;

**********Parameters for Weibull(*Loglogistic) model;
oualpha=par[1];
lambda=exp(par[1]);
alpha=par[2];

print 'Weibull(*Loglogistic) parameters: MLE subject to constraints';
print 'lambda=' lambda oualpha 'alpha=' alpha;

**********Compute Wald statistic;
p=p0;
ps=S*p;

Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;                 *Weibull;
*Gp=C*diag(1/ps+1/(1-ps))*S;                                *Loglogistic;
g=C*log(-log(1-ps));                                        *Weibull;
*g=C*(log(ps)-log(1-ps));                                   *Loglogistic;

******covariance matrix;
p1=p[1:k1];
p2=p[k1+1:k1+k2];
p3=p[k1+k2+1:k1+k2+k3];
p4=p[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(p1)-p1*p1`);
sig2=(1/n2)*(diag(p2)-p2*p2`);
sig3=(1/n3)*(diag(p3)-p3*p3`);
sig4=(1/n4)*(diag(p4)-p4*p4`);
```

```
sig=block(sig1,sig2,sig3,sig4);
V=sig;
*************************;
wald=g`*ginv(Gp*V*Gp`)*g; nu=eigval(CP); nu=nu[+]+4;
discr=wald/n;
prob=1-probchi(wald,nu) ;
alpha=par[2];
Gini=1-0.5##(1/alpha);
print 'Measure of fit';
print 'Wald='wald 'Discrepancy=' discr;
print 'prob=' prob 'degrees of freedom=' nu 'Gini=' Gini;
```

Program for fitting four lognormal distributions to the four histograms and then set the mu's equal and the sigma's equal

```
title1 'STAGGERED ENTRY OF POLICIES AT FOUR ENTRY TIMES';
title2 'Fit four lognormal distributions to the four entry groups';
title3 'Restrictions: specified model AND set mu's equal and sigma's equal';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector;
f1={66,158,254,157,250,35,1666};

**********Vector of upper boundaries;
x1={12,17,24,28,34,37};

**********Frequency vector;
f2={118,166,229,200,172,1924};

**********Vector of upper boundaries;
x2={12,17,24,28,34};

**********Frequency vector;
f3={154,99,242,117,1674};

**********Vector of upper boundaries;
x3={12,17,24,28};

**********Frequency vector;
f4={175,166,207,1848};

**********Vector of upper boundaries;
x4={12,17,24};

**********Relative frequency vectors;
n1=f1[+]; n2=f2[+]; n3=f3[+]; n4=f4[+]; n=n1+n2+n3+n4;
k1=nrow(f1); d1=k1-1;
k2=nrow(f2); d2=k2-1;
k3=nrow(f3); d3=k3-1;
k4=nrow(f4); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;
p1=f1/n1; p2=f2/n2; p3=f3/n3; p4=f4/n4;
p=p1//p2//p3//p4;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);   print S;

lx1=J(d1,1,1)||log(x1);
lx2=J(d2,1,1)||log(x2);
lx3=J(d3,1,1)||log(x3);
lx4=J(d4,1,1)||log(x4);

xc=block(lx1,lx2,lx3,lx4);
C=I(d)-xc*inv(xc`*xc)*xc`;
CP=C;
C=C//({1 0 -1 0 0 0 0 0, 1 0 0 0 -1 0 0 0, 1 0 0 0 0 0 -1 0,
       0 1 0 -1 0 0 0 0, 0 1 0 0 0 -1 0 0, 0 1 0 0 0 0 0 -1} *inv(xc`*xc)*xc`);
print xc C;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;  print p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
```

```
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
arg1=2;
arg2=3;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);
mu=par[1];sigma=par[2];
Gm=C*diag(sqrt(2#pi)/(exp(-(x-mu)#(x-mu)/2/sigma/sigma)))*S;         *lognormal;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>1e-6);
     j=j+1;
     p1=p;
     ps=S*p;
     g=C*probit(ps);                                             *lognormal;
     parp=inv( J(2,1,1)||probit(ps[arg1]//ps[arg2]) )*(x[arg1]//x[arg2]);
     mup=parp[1];sigmap=parp[2];
     Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S; *lognormal;
**********covariance matrix;
m1=m[1:k1];
m2=m[k1+1:k1+k2];
m3=m[k1+k2+1:k1+k2+k3];
m4=m[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(m1)-m1*m1`);
sig2=(1/n2)*(diag(m2)-m2*m2`);
sig3=(1/n3)*(diag(m3)-m3*m3`);
sig4=(1/n4)*(diag(m4)-m4*m4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
****************************;
     p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                               *lognormal;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);  *lognormal;

**********Parameters for Lognormal model;
mu=par[1]; sigma=par[2];

print 'Lognormal parameters: MLE subject to constraints';
print 'mu=' mu  'sigma=' sigma;

**********Compute Wald statistic;
p=p0;
ps=S*p;

Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S;   *lognormal;
g=C*probit(ps);                                                    *lognormal;
******covariance matrix;
p1=p[1:k1];
p2=p[k1+1:k1+k2];
p3=p[k1+k2+1:k1+k2+k3];
p4=p[k1+k2+k3+1:k1+k2+k3+k4];
sig1=(1/n1)*(diag(p1)-p1*p1`);
sig2=(1/n2)*(diag(p2)-p2*p2`);
sig3=(1/n3)*(diag(p3)-p3*p3`);
sig4=(1/n4)*(diag(p4)-p4*p4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
*************************;
wald=g`*ginv(Gp*V*Gp`)*g; nu=eigval(CP); nu=nu[+]+4;
discr=wald/n;
prob=1-probchi(wald,nu) ;
alpha=par[2];
Gini=1-0.5##(1/alpha);
print 'Measure of fit';
print 'Wald='wald 'Discrepancy=' discr;
print 'prob=' prob 'degrees of freedom=' nu 'Gini=' Gini;
```

3. Program for fitting a joint histogram to the four histograms of the entry groups

```
title1 'STAGGERED ENTRY OF POLICIES AT FOUR ENTRY TIMES';
title2 'Fit a joint histogram to the four histograms of the entry groups';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector;
f1={66,158,254,157,250,35,1666};
```

```
**********Vector of upper boundaries;
x1={12,17,24,28,34,37};

**********Frequency vector;
f2={118,166,229,200,172,1924};

**********Vector of upper boundaries;
x2={12,17,24,28,34};

**********Frequency vector;
f3={154,99,242,117,1674};

**********Vector of upper boundaries;
x3={12,17,24,28};

**********Frequency vector;
f4={175,166,207,1848};

**********Vector of upper boundaries;
x4={12,17,24};

**********Relative frequency vectors;
n1=f1[+]; n2=f2[+]; n3=f3[+]; n4=f4[+]; n=n1+n2+n3+n4;
k1=nrow(f1); d1=k1-1;
k2=nrow(f2); d2=k2-1;
k3=nrow(f3); d3=k3-1;
k4=nrow(f4); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;
p1=f1/n1; p2=f2/n2; p3=f3/n3; p4=f4/n4;
p=p1//p2//p3//p4;
**********Constraints imposed by the experimental design;
Gm=
(I(d4)    ||J(d4,1,0)||J(d4,1,0)||J(d4,1,0)||J(d4,1,0)||-I(d4)    ||J(d4,1,0)||
(I(d4)    ||J(d4,1,0)||J(d4,1,0)||J(d4,1,0)||J(d4,1,0)||J(d4,d4,0)||J(d4,1,0)||
(I(d4)    ||J(d4,1,0)||J(d4,1,0)||J(d4,1,0)||J(d4,1,0)||J(d4,d4,0)||J(d4,1,0)||
(J(1,d4,0)||        1||        1||        1||        1||J(1,d4,0) ||       -1||
(J(1,d4,0)||        1||        1||        1||        1||J(1,d4,0) ||        0||
(J(1,d4,0)||        1||        1||        1||        1||J(1,d4,0) ||        0||
(J(1,d4,0)||        0||        1||        0||        0||J(1,d4,0) ||        0||
(J(1,d4,0)||        1||        0||        0||        0||J(1,d4,0) ||       -1||

J(d4,1,0)||J(d4,1,0)||J(d4,d4,0)||J(d4,1,0)||J(d4,1,0)||J(d4,d4,0)||J(d4,1,0))//
J(d4,1,0)||J(d4,1,0)||-I(d4)    ||J(d4,1,0)||J(d4,1,0)||J(d4,d4,0)||J(d4,1,0))//
J(d4,1,0)||J(d4,1,0)||J(d4,d4,0)||J(d4,1,0)||J(d4,1,0)||-I(d4)    ||J(d4,1,0))//
       -1||       -1||J(1,d4,0) ||        0||        0||J(1,d4,0) ||        0)//
        0||        0||J(1,d4,0) ||       -1||       -1||J(1,d4,0) ||        0)//
        0||        0||J(1,d4,0) ||        0||        0||J(1,d4,0) ||       -1)//
       -1||        0||J(1,d4,0) ||        0||        0||J(1,d4,0) ||        0)//
        0||        0||J(1,d4,0) ||        0||        0||J(1,d4,0) ||        0);

*print Gm;

**********starting value for m;
m=p; Gp=Gm;
p0=p;
**********iteration over m;
verskil=1;
i=0;
do while (verskil>1e-6);
i=i+1;
p=p0;
g=Gm*p;
**********covariance matrix;
m1=m[1:k1];
m2=m[k1+1:k1+k2];
m3=m[k1+k2+1:k1+k2+k3];
m4=m[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(m1)-m1*m1`);
sig2=(1/n2)*(diag(m2)-m2*m2`);
sig3=(1/n3)*(diag(m3)-m3*m3`);
sig4=(1/n4)*(diag(m4)-m4*m4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
****************************;
p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;

*****Define joint frequencies;
p1=p[1:k1];
p2=p[k1+1:k1+k2];
p3=p[k1+k2+1:k1+k2+k3];
p4=p[k1+k2+k3+1:k1+k2+k3+k4];
p=p1//p2//p3//p4;

np1=n1#p1; np2=n2#p2; np3=n3#p3; np4=n4#p4; np=n#p;
****************************;

*print i p m np1 np2 np3 np4 np;
```

```
verskil=sqrt((p-m)`*(p-m));
m=p; m1=p1; m2=p2; m3=p3; m4=p4;
end;

*****Print frequencies of joint histogram;
print 'Frequencies of joint histogram=' np;
print 'Relative Frequencies of joint histogram=' p;

**********Compute Wald statistic;
******covariance matrix;
p1=p[1:k1];
p2=p[k1+1:k1+k2];
p3=p[k1+k2+1:k1+k2+k3];
p4=p[k1+k2+k3+1:k1+k2+k3+k4];

sig1=(1/n1)*(diag(p1)-p1*p1`);
sig2=(1/n2)*(diag(p2)-p2*p2`);
sig3=(1/n3)*(diag(p3)-p3*p3`);
sig4=(1/n4)*(diag(p4)-p4*p4`);
sig=block(sig1,sig2,sig3,sig4);
V=sig;
************************;
Gp=Gm;
wald=g`*ginv(Gp*V*Gp`)*g;
discr=wald/n;

print 'Wald=' wald 'Discrepancy=' discr;
```

A.3 Chapter 3: Simulation Studies

A.3.1 Program to simulate continuous right-censored lifetime data

1. **Program to generate continuous right-censored lifetime data from the Weibull(Loglogistic) distribution and to run simulations with the technique of MLE under constraints as well as the standard technique of MLE (1000 samples of size 100 from Weib(Logl)(0.15;0.5) - censored at 50)**

```
proc iml worksize=6000 symsize=2000;
reset noname nocenter;

**********Contents of module*************************************************;

start mod_est(x,f) global(lambda,alpha) ;

*****Relative frequencies;
n=100;
k=nrow(f);
d=k-1;
p=f/n;

*****Design matrix and matrix orthogonal to design matrix;
S1=J(d,1,1)@cusum(J(1,k,1));
S2=J(1,k,1)@cusum(J(d,1,1));
S=S1<=S2;

x1=J(d,1,1)||log(x);
C=I(d)-x1*inv(x1`*x1)*x1`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
  m=p;
  ms=S*m; ps=ms;
  p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;                     *Weibull;
*Gm=C*(diag(1/ms)+diag(1/(1-ms)))*S;                            *Loglogistic;

*****iteration over p;
   verskil=1;
   j=0;
   do while (verskil>1e-6);
   j=j+1;
   p1=p;
   ps=S*p;
   g=C*log(-log(1-ps));                                         *Weibull;
  *g=C*(log(ps)-log(1-ps));                                     *Loglogistic;
   Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;                  *Weibull;
  *Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                          *Loglogistic;
*****covariance matrix;
sig=(1/n)*(diag(m)-m*m`);
V=sig;
***********************;
   p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                                *Weibull;
  *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                                *Loglogistic;
   verskil=sqrt((p-p1)`*(p-p1));
   end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;

******Parameter vector for linear model;
par=inv(x1`*x1)*x1`*log(-log(1-ms));                            *Weibull;
*par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms));                       *Loglogistic;

**********Parameters for Weibull(*Loglogistic) model;
oualpha=par[1];
lambda=exp(par[1]);
alpha=par[2];

finish mod_est;

*****************************************************************;
**********Simulate 1000 samples of size 100 from Weib(Logl)(0.15;0.5)
```

```
                         - censored at 50;

lambda=0.15;
alpha=0.5;
n=100;
z=1000;
DT=((1/lambda)#(-log(1-ranuni(J(n,z,0)))))##(1/alpha);     *Weibull;
*FT=ranuni(J(n,z,0));
*DT=(FT/(lambda#(1-FT)))##(1/alpha);                        *Loglogistic;
DT=DT><J(n,z,50);                                           *censored at 50;

**********Define class boundaries and a frequency vector for each sample
          of continuous values, then run the module and store the 1000 estimates
          of lambda and alpha in a file PARMS;

filename parms 'c:\sim\sd2\wiml100a.sd2';                   *Weibull;
*filename parms 'c:\sim\sd2\lliml100a.sd2';                 *Loglogistic;
file parms;

do w=1 to z;
T=DT[,w];
B=T;
T[rank(T),]=B;
spnr=J(n,1,1)#w;

Y=(T=J(n,1,50));
nc=Y[+];
if nc=0 then nc=1e-4;
nl=n-nc;
ox=T[1:nl];
nc=1e-4<>nc;
f=J(nl,1,1)//nc;
perccens=(nc/n)#100;
             nr=nrow(ox); nr1=nr-1;
             x1=ox[1:nr1];x2=ox[2:nr];
             x=(x1+x2)/2;
             x=x//50;

run mod_est(x,f);
put lambda +3 alpha +3 perccens;
end;

closefile parms;

**********Put simulated data in a format that can be inputted in SAS
                   as a thousand continuous data sets;
%macro subgr(a);
%do i=1 %to &a;
name={"spnr" "time" "cens"};
di=DT[,&i];
cens=(di<J(n,1,50));
spnr=J(n,1,&i);
di=spnr||di||cens;
create d&i from di [colname=name];
append from di;
%end;
%mend subgr;
%subgr(1000);

**********Repeat the LIFEREG procedure of SAS one thousand times to get
          estimates for the intercept and scale parameters;

%macro mac(stel);
%do i=1 %to &stel;

proc lifereg data=d&i noprint outest=out&i  (keep=intercept _scale_);
model time*cens(0)= ;                          *Weibull;
*model time*cens(0)= / d=llogistic;            *Loglogistic;
run;

%end;
%mend mac;
%mac(1000);

**********Append the thousand estimates;

%macro ind(stel);
 %do i=2 %to &stel;

proc append base=out1 data=out&i;
run;

%end;
%mend ind;
%ind(1000);

*****************************************************************;
**********Sampling distribution of parameter estimates*********;
```

```
*****IML estimates;
data spv_iml;
infile 'c:\sim\sd2\wiml100a.sd2';                              *Weibull;
*infile 'c:\sim\sd2\liml100a.sd2';                             *Loglogistic;
input lambda alpha;
title1 'Sampling distribution: 1000 samples of size 100 from
        Weib(Logl)(0.15;0.5) - censored at 50';
title2 'IML method';

proc univariate data=spv_iml normal plot;
var lambda alpha;
run;

*****SAS estimates;
data sim.wsas100a (keep=lambda alpha);                         *Weibull;
*data sim.lsas100a (keep=lambda alpha);                        *Loglogistic;
set out1;
lambda=exp(-intercept/_scale_);
alpha=1/_scale_;

title1 'Sampling distribution: 1000 samples of size 100 from
        Weib(Logl)(0.15;0.5) - censored at 50';
title2 'SAS method';

proc univariate data=sim.wsas100a normal plot;                 *Weibull;
*proc univariate data=sim.lsas100a normal plot;                *Loglogistic;
var lambda alpha;
run;
```

2. **Program to generate continuous right-censored lifetime data from the log-normal distribution and to run simulations with the technique of MLE under constraints as well as the standard technique of MLE (1000 samples of size 200 from Lognormal -normal(2;0.5) - censored at 8)**

```
proc iml worksize=60 symsize=2000;
reset noname nocenter;

**********Contents of module**************************************************;

start mod_est(x,f,nl) global(mu,sigma) ;

*****Relative frequencies;
x=log(x);
n=200;
k=nrow(f);
d=k-1;
pi=(gamma(0.5))**2;
p=f/n;

*****Design matrix and matrix orthogonal to design matrix;
S1=J(d,1,1)@cusum(J(1,k,1));
S2=J(1,k,1)@cusum(J(d,1,1));
S=S1<=S2;

x1=J(d,1,1)||x;
C=I(d)-x1*inv(x1`*x1)*x1`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
  m=p;
  ms=S*m; ps=ms;
  p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
arg1=15;
arg2=nl-15;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);
mu=par[1];sigma=par[2];
Gm=C*diag(sqrt(2#pi)/(exp(-(x-mu)#(x-mu)/2/sigma/sigma)))*S;

*****iteration over p;
   verskil=1;
   j=0;
   do while (verskil>1e-6);
   j=j+1;
   p1=p;
   ps=S*p;
   g=C*probit(ps);
   parp=inv( J(2,1,1)||probit(ps[arg1]//ps[arg2]) )*(x[arg1]//x[arg2]);
   mup=parp[1];sigmap=parp[2];
   Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S;
*****covariance matrix;
```

```
sig=(1/n)*(diag(m)-m*m`);
V=sig;
**********************;
   p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;
   verskil=sqrt((p-p1)`*(p-p1));
   end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;

******Parameter vector for linear model;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);

**********Parameters for Lognormal model;
mu=par[1];
sigma=par[2];

finish mod_est;

*****************************************************************;
**********Simulate 1000 samples of size 200 from Lognormal(2;(0.5)^2)
                          - censored at 8;

n=200;
z=1000;
mean=2;
stddev=0.5;
AA=mean+stddev#normal(J(n,z,0));
DT=exp(AA);
DT=DT><J(n,z,8);      *8=censor point;

**********Define class boundaries and a frequency vector for each sample
          of continuous values, then run the module and store the
          1000 estimates of lambda and alpha in a file PARMS;

filename parms 'c:\sim\sd2\lnik200a.sd2';
file parms;

do w=1 to z;
T=DT[,w];
B=T;
T[rank(T),]=B;
spnr=J(n,1,1)#w;

Y=(T=J(n,1,8));        *8=censor point;
nc=Y[+];
if nc=0 then nc=1e-4;
nl=n-nc;
ox=T[1:nl];
nc=1e-4<>nc;
f=J(nl,1,1)//nc;
perccens=(nc/n)#100;
             nr=nrow(ox); nr1=nr-1;
             x1=ox[1:nr1];x2=ox[2:nr];
             x=(x1+x2)/2;
             x=x//8;   *8=censor point;
*print f x nl;
run mod_est(x,f,nl);
put (mu) +3 (sigma) +3  (perccens) +3 (nl);
end;

closefile parms;

**********Put simulated data in a format that can be inputted in SAS
                   as a thousand continuous data sets;
%macro subgr(a);
%do i=1 %to &a;
name={"spnr" "time" "cens"};
di=DT[,&i];
cens=(di<J(n,1,8));     *8=censor point;
spnr=J(n,1,&i);
di=spnr||di||cens;
create d&i from di [colname=name];
append from di;
%end;
%mend subgr;
%subgr(1000);

**********Repeat the LIFEREG procedure of SAS one thousand times to get
          estimates for the intercept and scale parameters;

%macro mac(stel);
%do i=1 %to &stel;

proc lifereg data=d&i noprint outest=out&i  (keep=intercept _scale_);
model time*cens(0)= / d=lnormal;
run;

%end;
```

```
%mend mac;
%mac(1000);

**********Append the thousand estimates;

%macro ind(stel);
 %do i=2 %to &stel;

proc append base=out1 data=out&i;
run;

%end;
%mend ind;
%ind(1000);

*****************************************************************;
**********Sampling distribution of parameter estimates*********;

*****IML estimates;
data spv_iml;
infile 'c:\sim\sd2\lnik200a.sd2';
input mu sigma;
*ods html body='c:\sim\lnik200a.htm';
title1 'Sampling distribution: 1000 samples of size 200 from lognormal
        - normal(2;0.5) - censored at 8';
title2 'IML method';

proc univariate data=spv_iml normal plot;
var mu sigma;
run;

*****SAS estimates;

data sim.lnsk200a (keep=mu sigma);
set out1;
mu=intercept;
sigma=_scale_;
title1 'Sampling distribution: 1000 samples of size 200 from lognormal
        - normal(2;0.5) - censored at 8';
title2 'SAS method';

proc univariate data=sim.lnsk200a normal plot;
var mu sigma;
run;
```

A.3.2 Program to simulate grouped right-censored lifetime data

1. **Program to generate right-censored grouped lifetime data from the Weibull (loglogistic) distribution and to run simulations with the technique of MLE under constraints as well as the standard technique of MLE (1000 samples of size 2000 from Weib(30;1.8) - censored at 0.15 (grouped into 5 classes))**

```
proc iml worksize=6000 symsize=2000;
reset noname nocenter;

**********Contents of module**************************************************;

start mod_est(x,f) global(lambda,alpha) ;

*****Relative frequencies;
n=2000;
k=nrow(f);
d=k-1;
p=f/n;

*****Design matrix and matrix orthogonal to design matrix;
S1=J(d,1,1)@cusum(J(1,k,1));
S2=J(1,k,1)@cusum(J(d,1,1));
S=S1<=S2;

x1=J(d,1,1)||log(x);
C=I(d)-x1*inv(x1`*x1)*x1`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
  m=p;
  ms=S*m; ps=ms;
  p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
```

```
do while (verskil1>1e-6);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;                   *Weibull;
*Gm=C*(diag(1/ms)+diag(1/(1-ms)))*S;                          *Loglogistic;

*****iteration over p;
   verskil=1;
   j=0;
   do while (verskil>1e-6);
   j=j+1;
   p1=p;
   ps=S*p;
   g=C*log(-log(1-ps));                                       *Weibull;
  *g=C*(log(ps)-log(1-ps));                                   *Loglogistic;
   Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;                *Weibull;
  *Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                        *Loglogistic;
*****covariance matrix;
sig=(1/n)*(diag(m)-m*m`);
V=sig;
**********************;
   p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                              *Weibull;
  *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                              *Loglogistic;
   verskil=sqrt((p-p1)`*(p-p1));
   end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;

******Parameter vector for linear model;
par=inv(x1`*x1)*x1`*log(-log(1-ms));                          *Weibull;
*par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms));                     *Loglogistic;

**********Parameters for Weibull(*Loglogistic) model;
oualpha=par[1];
lambda=exp(par[1]);
alpha=par[2];

finish mod_est;

****************************************************************;
**********Simulate 1000 samples of size 2000 from Weib(Logl)(30;1.8) -
          censored at 0.15 and then define lower and upper class boundaries
          for 5 class intervals;

lambda=30;
alpha=1.8;
n=2000;
z=1000;
TT=((1/lambda)#(-log(1-ranuni(J(n,z,0)))))##(1/alpha);      *Weibull;
*FT=ranuni(J(n,z,0));
*TT=(FT/(lambda#(1-FT)))##(1/alpha);                          *Loglogistic;
x1={0,0.08,0.10,0.12,0.15};    *lower boundaries;            *censored at 0.15;
x2={0.08,0.10,0.12,0.15,100}; *upper boundaries;
k=5;
****************************************************************;
**********calculate the frequency vector for each sample, then run the
module and store the 1000 estimates of lambda and alpha in a file PARMS;

filename parms 'c:\sim\sd2\wigtwdc1.sd2';                     *Weibull;
*filename parms 'c:\sim\sd2\ligtwdc1.sd2';                    *Loglogistic;
file parms;

do w=1 to z;
T=TT[w,];
A=(J(k,1,1)*T)<=(x2*J(1,n,1));
B=(J(k,1,1)*T)>=(x1*J(1,n,1));
E=A=B;
f=E*J(n,1,1);
perccens=(f[5,1]/n)#100;
x=x2[1:4,];

run mod_est(x,f);
put lambda +3 alpha +3 perccens;
end;

closefile parms;

**********Put simulated data in a format that can be inputted in SAS
                   as a thousand grouped data sets;
%macro subgr(a);
%do i=1 %to &a;
name={"lower" "upper" "frek"};
lower=.//x1[2:5,];
upper=x2[1:4,]//.;

T=TT[&i,];
A=(J(k,1,1)*T)<=(x2*J(1,n,1));
B=(J(k,1,1)*T)>=(x1*J(1,n,1));
```

```
E=A=B;
frek=E*J(n,1,1);
di=lower||upper||frek;
create d&i from di [colname=name];
append from di;
%end;
%mend subgr;
%subgr(1000);

**********Repeat the LIFEREG procedure of SAS for grouped data one thousand
          times to get estimates for the intercept and scale parameters;

%macro mac(stel);
%do i=1 %to &stel;

proc lifereg data=d&i noprint outest=out&i  (keep=intercept _scale_);
model (lower,upper)= ;                          *Weibull;
*model (lower,upper)= / d=llogistic;            *Loglogistic;
weight frek;
run;

%end;
%mend mac;
%mac(1000);

**********Append the thousand estimates;

%macro ind(stel);
 %do i=2 %to &stel;

proc append base=out1 data=out&i;
run;

%end;
%mend ind;
%ind(1000);

*****************************************************************;
**********Sampling distribution of parameter estimates*********;

*****IML estimates;

data spv_iml;
infile 'c:\sim\sd2\wigtwdc1.sd2';                          *Weibull;
*infile 'c:\sim\sd2\ligtwdc1.sd2';                         *Loglogistic;
input lambda alpha;
title1 'Sampling distribution: 1000 samples of size 2000 from
        Weib(Logl)(30;1.8) - censored at 0.15 (5 classes)';
title2 'IML method';

proc univariate data=spv_iml normal plot;
var lambda alpha;
run;

*****SAS estimates;
data sim.wsgtwdc1 (keep=lambda alpha);                     *Weibull;
*data sim.lsgtwdc1 (keep=lambda alpha);                    *Loglogistic;
set out1;
lambda=exp(-intercept/_scale_);
alpha=1/_scale_;

title1 'Sampling distribution: 1000 samples of size 2000 from Weib(30;1.8)
        - censored at 0.15 (5 classes)';
title2 'SAS method';

proc univariate data=sim.wsgtwdc1 normal plot;             *Weibull;
*proc univariate data=sim.lsgtwdc1 normal plot;            *Loglogistic;
var lambda alpha;
run;
```

2. Program to generate right-censored grouped lifetime data from the lognormal distribution and to run simulations with the technique of MLE under constraints as well as the standard technique of MLE (1000 samples of size 2000 from Lognormal - normal(2;0.5) - censored at 8 (grouped into 5 classes))

```
proc iml worksize=60 symsize=2000;
reset noname nocenter;

**********Contents of module**************************************************;

start mod_est(x,f) global(mu,sigma) ;

*****Relative frequencies;
x=log(x);
n=2000;
k=nrow(f);
d=k-1;
pi=(gamma(0.5))**2;
```

```
p=f/n; ;

*****Design matrix and matrix orthogonal to design matrix;
S1=J(d,1,1)@cusum(J(1,k,1));
S2=J(1,k,1)@cusum(J(d,1,1));
S=S1<=S2;

x1=J(d,1,1)||x;
C=I(d)-x1*inv(x1`*x1)*x1`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
  m=p;
  ms=S*m; ps=ms;
  p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>1e-6);
i=i+1;
p=p0;
arg1=2;
arg2=3;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);
mu=par[1];sigma=par[2];
Gm=C*diag(sqrt(2#pi)/(exp(-(x-mu)#(x-mu)/2/sigma/sigma)))*S;

*****iteration over p;
   verskil=1;
   j=0;
   do while (verskil>1e-6);
   j=j+1;
   p1=p;
   ps=S*p;
   g=C*probit(ps);
   parp=inv( J(2,1,1)||probit(ps[arg1]//ps[arg2]) )*(x[arg1]//x[arg2]);
   mup=parp[1];sigmap=parp[2];
   Gp=C*diag(sqrt(2#pi)/(exp(-(x-mup)#(x-mup)/2/sigmap/sigmap)))*S;
*****covariance matrix;
sig=(1/n)*(diag(m)-m*m`);
V=sig;
***********************;
   p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;
   verskil=sqrt((p-p1)`*(p-p1));
   end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;

******Parameter vector for linear model;
par=inv( J(2,1,1)||probit(ms[arg1]//ms[arg2]) )*(x[arg1]//x[arg2]);

**********Parameters for Lognormal model;
mu=par[1];
sigma=par[2];

finish mod_est;

****************************************************************************;
**********Simulate 1000 samples of size 2000 from Lognormal(2;(0.5)^2) -
censored at 8 and then define lower and upper boundaries for 5 class intervals;

n=2000;
z=1000;
mean=2;
stddev=0.5;
AA=mean+stddev#normal(J(z,n,0));
DT=exp(AA);
DT=DT><J(z,n,8);      *8=censor point;
x1={0,4,6,7,8};        *lower boundaries;  *censored at 8;
x2={4,6,7,8,10000};    *upper boundaries;
k=5;

*******************************************************************;
**********calculate the frequency vector for each sample, then run
the module and store the 1000 estimates of lambda and alpha in a file PARMS;

filename parms 'c:\sim\sd2\lnigtwa1.sd2';
file parms;

do w=1 to z;
T=DT[w,];
A=(J(k,1,1)*T)< (x2*J(1,n,1));
B=(J(k,1,1)*T)>=(x1*J(1,n,1));
E=A=B;
f=E*J(n,1,1);
perccens=(f[5,1]/n)#100;
```

```
x=x2[1:4,];

run mod_est(x,f);
put mu +3 sigma +3  perccens;
end;

closefile parms;

**********Put simulated data in a format that can be inputted in SAS
                   as a thousand grouped data sets;
%macro subgr(a);
%do i=1 %to &a;
name={"lower" "upper" "frek"};
lower=.//x1[2:5,];
upper=x2[1:4,]//.;

T=DT[&i,];
A=(J(k,1,1)*T)< (x2*J(1,n,1));
B=(J(k,1,1)*T)>=(x1*J(1,n,1));
E=A=B;
frek=E*J(n,1,1);
di=lower||upper||frek;
create d&i from di [colname=name];
append from di;
%end;
%mend subgr;
%subgr(1000);

**********Repeat the LIFEREG procedure of SAS for grouped data one thousand
          times to get estimates for the intercept and scale parameters;

%macro mac(stel);
%do i=1 %to &stel;

proc lifereg data=d&i noprint outest=out&i  (keep=intercept _scale_);
model (lower,upper)= / d=lnormal;
weight frek;
run;

%end;
%mend mac;
%mac(1000);

**********Append the thousand estimates;

%macro ind(stel);
 %do i=2 %to &stel;

proc append base=out1 data=out&i;
run;

%end;
%mend ind;
%ind(1000);

****************************************************************;
**********Sampling distribution of parameter estimates*********;

*****IML estimates;

data spv_iml;
infile 'c:\sim\sd2\lnigtwa1.sd2';
input mu sigma;
title1 'Sampling distribution: 1000 samples of size 2000 from lognormal
        - normal(2;0.5) - censored at 8 (5 classes)';
title2 'IML method';

proc univariate data=spv_iml normal plot;
var mu sigma;
run;

*****SAS estimates;

data sim.lnsgtwa1 (keep=mu sigma);
set out1;
mu=intercept;
sigma=_scale_;
title1 'Sampling distribution: 1000 samples of size 2000 from lognormal
        - normal(2;0.5) - censored at 8 (5 classes)';
title2 'SAS method';

proc univariate data=sim.lnsgtwa1 normal plot;
var mu sigma;
run;
```

A.4 Chapter 4: Maximum Likelihood Estimation

A.4.1 Staggered Entry of Policies Standard Programs using PROC LIFEREG

1. **Program for fitting a log-logistic/Weibull regression model with one predictor to grouped survival data**

```
options nodate pagesize=500 pageno=1;
libname hsbc 'c:\hsbc1\sd2';

title 'AGE:three levels (define two dummies)';

data agegr;
input lower upper  agegr1 agegr2  freq;
*input lower upper  agegr $ freq;
*at class statement: use one column A B C A B C etc.;
cards;
 .    12  1  0   29
 .    12  0  1   21
 .    12 -1 -1   16
12    17  1  0   59
12    17  0  1   50
12    17 -1 -1   49
17    24  1  0   95
17    24  0  1   91
17    24 -1 -1   68
24    28  1  0   73
24    28  0  1   45
24    28 -1 -1   39
28    34  1  0  108
28    34  0  1   75
28    34 -1 -1   67
34    37  1  0   15
34    37  0  1   13
34    37 -1 -1    7
37     .  1  0  642
37     .  0  1  553
37     . -1 -1  471
 .    12  1  0   41
 .    12  0  1   49
 .    12 -1 -1   28
12    17  1  0   75
12    17  0  1   62
12    17 -1 -1   29
17    24  1  0  103
17    24  0  1   61
17    24 -1 -1   65
24    28  1  0   92
24    28  0  1   66
24    28 -1 -1   42
28    34  1  0   83
28    34  0  1   54
28    34 -1 -1   35
34     .  1  0  628
34     .  0  1  753
34     . -1 -1  543
 .    12  1  0   68
 .    12  0  1   40
 .    12 -1 -1   46
12    17  1  0   34
12    17  0  1   44
12    17 -1 -1   21
17    24  1  0   99
17    24  0  1   83
17    24 -1 -1   60
24    28  1  0   57
24    28  0  1   33
24    28 -1 -1   27
28     .  1  0  570
28     .  0  1  533
28     . -1 -1  571
 .    12  1  0   71
 .    12  0  1   54
 .    12 -1 -1   50
12    17  1  0   60
12    17  0  1   61
12    17 -1 -1   45
17    24  1  0   69
17    24  0  1   68
17    24 -1 -1   70
24     .  1  0  573
```

```
24    .  0  1  616
24    . -1 -1  659
;

data add;
c=1;
run;

data fin;
set agegr add;

proc lifereg data=fin covout;
*class agegr;
*model (lower, upper) = agegr / dist=llogistic;
model (lower, upper) = agegr1 agegr2 / dist=llogistic;
weight freq;
output out=llog cdf=cdf predicted=months
quantiles = 0.02 to 0.98 by .02
control=c;
title 'Fit Loglogistic curve (SAS method) to HSBC: MPC policies - predictor AGE';
title2 'Four entry dates';
run;

data par;
intercept=3.86266;
_scale_=0.48394;
theta1=-0.08757;
theta2= 0.01693;
theta3= -(theta1+theta2);
if _N_=1;

oualphaSASage1=-(intercept+theta1*1+theta2*0)/_scale_;
lambdaSASage1=exp(-(intercept+theta1*1+theta2*0)/_scale_);
oualphaSASage2=-(intercept+theta1*0+theta2*1)/_scale_;
lambdaSASage2=exp(-(intercept+theta1*0+theta2*1)/_scale_);
oualphaSASage3=-(intercept+theta1*(-1)+theta2*(-1))/_scale_;
lambdaSASage3=exp(-(intercept+theta1*(-1)+theta2*(-1))/_scale_);
alphaSAS=1/_scale_;
oubetaSAS=1/_scale_;
oualphaBASELINESAS=-(intercept/_scale_);
lambdaBASELINESAS=exp(-(intercept/_scale_));

proc print data=par;
var oualphaSASage1 oualphaSASage2 oualphaSASage3
    lambdaSASage1 lambdaSASage2 lambdaSASage3
    alphaSAS oubetaSAS
    oualphaBASELINESAS lambdaBASELINESAS;
run;
```

2. Program for fitting a log-logistic/Weibull regression model with two predictors to grouped survival data

```
options nodate pagesize=500 pageno=1;
libname hsbc 'c:\hsbc1\sd2';

title1 'AGE:three levels (define two dummies)';
title2 'SCORE:three levels (define two dummies)';

data agegr;
input lower upper  agegr1 agegr2  score1 score2 freq;
*input lower upper  agegr $ freq;
*at class statement: use one column of A B C A B C etc.;
cards;
 .   12  1  0  1  0 12
12   17  1  0  1  0 34
17   24  1  0  1  0 51
24   28  1  0  1  0 39
28   34  1  0  1  0 57
34   37  1  0  1  0 11
37    .  1  0  1  0 59

 .   12  1  0  0  1 10
12   17  1  0  0  1 12
17   24  1  0  0  1 22
24   28  1  0  0  1 19
28   34  1  0  0  1 32
34   37  1  0  0  1  4
37    .  1  0  0  1 418

 .   12  1  0 -1 -1  7
12   17  1  0 -1 -1 13
17   24  1  0 -1 -1 22
24   28  1  0 -1 -1 15
28   34  1  0 -1 -1 19
34   37  1  0 -1 -1  0
37    .  1  0 -1 -1 165

 .   12  0  1  1  0 13
12   17  0  1  1  0 14
```

```
17   24  0  1  1  0 45
24   28  0  1  1  0 27
28   34  0  1  1  0 33
34   37  0  1  1  0  4
37    .  0  1  1  0  66

 .   12  0  1  0  1  4
12   17  0  1  0  1 22
17   24  0  1  0  1 22
24   28  0  1  0  1  8
28   34  0  1  0  1 25
34   37  0  1  0  1  4
37    .  0  1  0  1 297

 .   12  0  1 -1 -1  4
12   17  0  1 -1 -1 14
17   24  0  1 -1 -1 24
24   28  0  1 -1 -1 10
28   34  0  1 -1 -1 17
34   37  0  1 -1 -1  5
37    .  0  1 -1 -1 190

 .   12 -1 -1  1  0 10
12   17 -1 -1  1  0 25
17   24 -1 -1  1  0 29
24   28 -1 -1  1  0 17
28   34 -1 -1  1  0 46
34   37 -1 -1  1  0  2
37    . -1 -1  1  0 116

 .   12 -1 -1  0  1  6
12   17 -1 -1  0  1 13
17   24 -1 -1  0  1 28
24   28 -1 -1  0  1 16
28   34 -1 -1  0  1 16
34   37 -1 -1  0  1  5
37    . -1 -1  0  1 273

 .   12 -1 -1 -1 -1  0
12   17 -1 -1 -1 -1 11
17   24 -1 -1 -1 -1 11
24   28 -1 -1 -1 -1  6
28   34 -1 -1 -1 -1  5
34   37 -1 -1 -1 -1  0
37    . -1 -1 -1 -1  82

 .   12  1  0  1  0 22
12   17  1  0  1  0 25
17   24  1  0  1  0 58
24   28  1  0  1  0 53
28   34  1  0  1  0 40
34    .  1  0  1  0  45

 .   12  1  0  0  1 10
12   17  1  0  0  1 26
17   24  1  0  0  1 32
24   28  1  0  0  1 20
28   34  1  0  0  1 29
34    .  1  0  0  1 379

 .   12  1  0 -1 -1  9
12   17  1  0 -1 -1 24
17   24  1  0 -1 -1 13
24   28  1  0 -1 -1 19
28   34  1  0 -1 -1 14
34    .  1  0 -1 -1 204

 .   12  0  1  1  0 24
12   17  0  1  1  0 24
17   24  0  1  1  0 28
24   28  0  1  1  0 30
28   34  0  1  1  0 25
34    .  0  1  1  0 106

 .   12  0  1  0  1 12
12   17  0  1  0  1 20
17   24  0  1  0  1 14
24   28  0  1  0  1 17
28   34  0  1  0  1 16
34    .  0  1  0  1 409

 .   12  0  1 -1 -1 13
12   17  0  1 -1 -1 18
17   24  0  1 -1 -1 19
24   28  0  1 -1 -1 19
28   34  0  1 -1 -1 13
34    .  0  1 -1 -1 238

 .   12 -1 -1  1  0 13
```

```
12   17 -1 -1  1  0 15
17   24 -1 -1  1  0 32
24   28 -1 -1  1  0 19
28   34 -1 -1  1  0 17
34    . -1 -1  1  0 107

 .   12 -1 -1  0  1 11
12   17 -1 -1  0  1 13
17   24 -1 -1  0  1 22
24   28 -1 -1  0  1 17
28   34 -1 -1  0  1 12
34    . -1 -1  0  1 319

 .   12 -1 -1 -1 -1  4
12   17 -1 -1 -1 -1  1
17   24 -1 -1 -1 -1 11
24   28 -1 -1 -1 -1  6
28   34 -1 -1 -1 -1  6
34    . -1 -1 -1 -1 117

 .   12  1  0  1  0 34
12   17  1  0  1  0 16
17   24  1  0  1  0 50
24   28  1  0  1  0 23
28    .  1  0  1  0  54

 .   12  1  0  0  1 19
12   17  1  0  0  1  2
17   24  1  0  0  1 32
24   28  1  0  0  1 24
28    .  1  0  0  1 317

 .   12  1  0 -1 -1 15
12   17  1  0 -1 -1 16
17   24  1  0 -1 -1 17
24   28  1  0 -1 -1 10
28    .  1  0 -1 -1 199

 .   12  0  1  1  0 19
12   17  0  1  1  0 18
17   24  0  1  1  0 38
24   28  0  1  1  0 16
28    .  0  1  1  0  75

 .   12  0  1  0  1 16
12   17  0  1  0  1 14
17   24  0  1  0  1 25
24   28  0  1  0  1 10
28    .  0  1  0  1 263

 .   12  0  1 -1 -1  5
12   17  0  1 -1 -1 12
17   24  0  1 -1 -1 20
24   28  0  1 -1 -1  7
28    .  0  1 -1 -1 195

 .   12 -1 -1  1  0 28
12   17 -1 -1  1  0 16
17   24 -1 -1  1  0 22
24   28 -1 -1  1  0 12
28    . -1 -1  1  0  98

 .   12 -1 -1  0  1 13
12   17 -1 -1  0  1  0
17   24 -1 -1  0  1 24
24   28 -1 -1  0  1  4
28    . -1 -1  0  1 323

 .   12 -1 -1 -1 -1  5
12   17 -1 -1 -1 -1  5
17   24 -1 -1 -1 -1 14
24   28 -1 -1 -1 -1 11
28    . -1 -1 -1 -1 150

 .   12  1  0  1  0 40
12   17  1  0  1  0 30
17   24  1  0  1  0 30
24    .  1  0  1  0  50

 .   12  1  0  0  1  9
12   17  1  0  0  1 14
17   24  1  0  0  1 27
24    .  1  0  0  1 301

 .   12  1  0 -1 -1 22
12   17  1  0 -1 -1 16
17   24  1  0 -1 -1 12
24    .  1  0 -1 -1 222
```

```
 .   12  0  1  1  0 24
12   17  0  1  1  0 30
17   24  0  1  1  0 29
24   .   0  1  1  0  81

 .   12  0  1  0  1 14
12   17  0  1  0  1 15
17   24  0  1  0  1 12
24   .   0  1  0  1 307

 .   12  0  1 -1 -1 16
12   17  0  1 -1 -1 16
17   24  0  1 -1 -1 27
24   .   0  1 -1 -1 228

 .   12 -1 -1  1  0 20
12   17 -1 -1  1  0 22
17   24 -1 -1  1  0 28
24   .  -1 -1  1  0 119

 .   12 -1 -1  0  1 19
12   17 -1 -1  0  1 12
17   24 -1 -1  0  1 26
24   .  -1 -1  0  1 369

 .   12 -1 -1 -1 -1 11
12   17 -1 -1 -1 -1 11
17   24 -1 -1 -1 -1 16
24   .  -1 -1 -1 -1 171
;

data add;
c=1;
run;

data fin;
set agegr add;

proc lifereg data=fin covout;
*class agegr;
*model (lower, upper) = agegr / dist=llogistic;
model (lower, upper) = agegr1 agegr2 score1 score2 / dist=llogistic;
weight freq;
output out=llog cdf=cdf predicted=months
quantiles = 0.02 to 0.98 by .02
control=c;
title 'Fit Loglogistic curve (SAS method) to HSBC: MPC policies - predictor AGE';
title2 'Four entry dates';
run;

data par;
intercept=3.80119;
_scale_=0.44454;
thetaA1=-0.09129;
thetaA2= 0.0052689;
thetaA3= -(thetaA1+thetaA2);
thetaS1=-0.46574;
thetaS2= 0.31782;
thetaS3= -(thetaS1+thetaS2);
if _N_=1;

oualphaSASa1s1=-(intercept+thetaA1*1+thetaA2*0+thetaS1*1+thetaS2*0)/_scale_;
lambdaSASa1s1=exp(-(intercept+thetaA1*1+thetaA2*0+thetaS1*1+thetaS2*0)/_scale_);
oualphaSASa1s2=-(intercept+thetaA1*1+thetaA2*0+thetaS1*0+thetaS2*1)/_scale_;
lambdaSASa1s2=exp(-(intercept+thetaA1*1+thetaA2*0+thetaS1*0+thetaS2*1)/_scale_);
oualphaSASa1s3=-(intercept+thetaA1*1+thetaA2*0+thetaS1*(-1)+thetaS2*(-1))/_scale_;
lambdaSASa1s3=exp(-(intercept+thetaA1*1+thetaA2*0+thetaS1*(-1)+thetaS2*(-1))/_scale_);
oualphaSASa2s1=-(intercept+thetaA1*0+thetaA2*1+thetaS1*1+thetaS2*0)/_scale_;
lambdaSASa2s1=exp(-(intercept+thetaA1*0+thetaA2*1+thetaS1*1+thetaS2*0)/_scale_);
oualphaSASa2s2=-(intercept+thetaA1*0+thetaA2*1+thetaS1*0+thetaS2*1)/_scale_;
lambdaSASa2s2=exp(-(intercept+thetaA1*0+thetaA2*1+thetaS1*0+thetaS2*1)/_scale_);
oualphaSASa2s3=-(intercept+thetaA1*0+thetaA2*1+thetaS1*(-1)+thetaS2*(-1))/_scale_;
lambdaSASa2s3=exp(-(intercept+thetaA1*0+thetaA2*1+thetaS1*(-1)+thetaS2*(-1))/_scale_);
oualphaSASa3s1=-(intercept+thetaA1*(-1)+thetaA2*(-1)+thetaS1*1+thetaS2*0)/_scale_;
lambdaSASa3s1=exp(-(intercept+thetaA1*(-1)+thetaA2*(-1)+thetaS1*1+thetaS2*0)/_scale_);
oualphaSASa3s2=-(intercept+thetaA1*(-1)+thetaA2*(-1)+thetaS1*0+thetaS2*1)/_scale_;
lambdaSASa3s2=exp(-(intercept+thetaA1*(-1)+thetaA2*(-1)+thetaS1*0+thetaS2*1)/_scale_);
oualphaSASa3s3=-(intercept+thetaA1*(-1)+thetaA2*(-1)+thetaS1*(-1)+thetaS2*(-1))/_scale_;
lambdaSASa3s3=exp(-(intercept+thetaA1*(-1)+thetaA2*(-1)+thetaS1*(-1)+thetaS2*(-1))/_scale_);
alphaSAS=1/_scale_;
oubetaSAS=1/_scale_;
oualphaBASELINESAS=-(intercept/_scale_);
lambdaBASELINESAS=exp(-(intercept/_scale_));

proc print data=par;
var oualphaSASa1s1 oualphaSASa1s2 oualphaSASa1s3
    lambdaSASa1s1 lambdaSASa1s2 lambdaSASa1s3;
proc print data=par;
var oualphaSASa2s1 oualphaSASa2s2 oualphaSASa2s3
```

```
    lambdaSASa2s1 lambdaSASa2s2 lambdaSASa2s3;
proc print data=par;
var oualphaSASa3s1 oualphaSASa3s2 oualphaSASa3s3
    lambdaSASa3s1 lambdaSASa3s2 lambdaSASa3s3;
proc print data=par;
var alphaSAS oubetaSAS oualphaBASELINESAS lambdaBASELINESAS;
run;
```

A.5 Chapter 4: M L Estimation subject to Constraints

A.5.1 Staggered Entry of Policies - IML Programs

1. **Program for fitting a log-logistic regression model (constant shape, one covariate)**

```
title1 'Fitting of regression model with one covariate';
title2 'Staggered entry: constant shape parameter';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector (first entry,3 agegroups);
                f11={29,59,95,73,108,15,642};
                f12={21,50,91,45,75,13,553};
                f13={16,49,68,39, 67, 7,471};

**********Vector of upper boundaries;
                x1={12,17,24,28,34,37};

**********Frequency vector (second entry,3 agegroups);
                f21={41,75,103,92,83,628};
                f22={49,62,61,66,54,753};
                f23={28,29,65,42,35,543};

**********Vector of upper boundaries;
                x2={12,17,24,28,34};

**********Frequency vector (third entry,3 agegroups);
                f31={68,34,99,57,570};
                f32={40,44,83,33,533};
                f33={46,21,60,27,571};

**********Vector of upper boundaries;
                x3={12,17,24,28};

**********Frequency vector (fourth entry,3 agegroups);
                f41={71,60,69,573};
                f42={54,61,68,616};
                f43={50,45,70,659};

**********Vector of upper boundaries;
                x4={12,17,24};

**********Relative frequency vectors;
n11=f11[+];    n21=f21[+];    n31=f31[+];    n41=f41[+];    n1=n11+n21+n31+n41;
n12=f12[+];    n22=f22[+];    n32=f32[+];    n42=f42[+];    n2=n12+n22+n32+n42;
n13=f13[+];    n23=f23[+];    n33=f33[+];    n43=f43[+];    n3=n13+n23+n33+n43;
                                                             n=n1+n2+n3;
k1=nrow(f11); d1=k1-1;
k2=nrow(f21); d2=k2-1;
k3=nrow(f31); d3=k3-1;
k4=nrow(f41); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;

p11=f11/n11;    p21=f21/n21;    p31=f31/n31;    p41=f41/n41;    p1=p11//p21//p31//p41;
p12=f12/n12;    p22=f22/n22;    p32=f32/n32;    p42=f42/n42;    p2=p12//p22//p32//p42;
p13=f13/n13;    p23=f23/n23;    p33=f33/n33;    p43=f43/n43;    p3=p13//p23//p33//p43;
                                                                p=p1//p2//p3;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
```

```
S=block(S1,S2,S3,S4);
S=I(3)@S;

AA=J(54,1,1);        *54=3 times((7-1)+(6-1)+(5-1)+(4-1))=3 times 18=54;
BB=J(3,1,1);
CC=(I(2)//J(1,2,-1));
DD=J(18,1,1);             *d=18;
EE=J(9,1,1);

lx=BB@(log(x1)//log(x2)//log(x3)//log(x4));

xc=AA||(CC@DD)||lx;  print xc;

C=I(3#d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
*Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;                *Weibull;
Gm=C*diag(1/ms+1/(1-ms))*S;                                 *log-logistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
    *g=C*log(-log(1-ps));                                   *Weibull;
    g=C*(log(ps)-log(1-ps));                                *log-logistic;
    *Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;            *Weibull;
    Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                     *log-logistic;
******************covariance matrix**********************************;
m11=m[1:k1]; m21=m[k1+1:k1+k2];
m31=m[k1+k2+1:k1+k2+k3]; m41=m[k1+k2+k3+1:k1+k2+k3+k4];

m12=m[k+1:k+k1]; m22=m[k+k1+1:k+k1+k2];
m32=m[k+k1+k2+1:k+k1+k2+k3]; m42=m[k+k1+k2+k3+1:k+k1+k2+k3+k4];

m13=m[2#k+1:2#k+k1]; m23=m[2#k+k1+1:2#k+k1+k2];
m33=m[2#k+k1+k2+1:2#k+k1+k2+k3]; m43=m[2#k+k1+k2+k3+1:2#k+k1+k2+k3+k4];

sig11=(1/n11)*(diag(m11)-m11*m11`);
sig21=(1/n21)*(diag(m21)-m21*m21`);
sig31=(1/n31)*(diag(m31)-m31*m31`);
sig41=(1/n41)*(diag(m41)-m41*m41`);
sig1=block(sig11,sig21,sig31,sig41);

sig12=(1/n12)*(diag(m12)-m12*m12`);
sig22=(1/n22)*(diag(m22)-m22*m22`);
sig32=(1/n32)*(diag(m32)-m32*m32`);
sig42=(1/n42)*(diag(m42)-m42*m42`);
sig2=block(sig12,sig22,sig32,sig42);

sig13=(1/n13)*(diag(m13)-m13*m13`);
sig23=(1/n23)*(diag(m23)-m23*m23`);
sig33=(1/n33)*(diag(m33)-m33*m33`);
sig43=(1/n43)*(diag(m43)-m43*m43`);
sig3=block(sig13,sig23,sig33,sig43);

sig=block(sig1,sig2,sig3);
V=sig;
*********************************************************************;
    *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                          *Weibull;
    p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                           *log-logistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
*par=inv(xc`*xc)*xc`*log(-log(1-ms));                       *Weibull;
par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms));                    *log-logistic;

print par[format=E20.];

**********Regression coefficients;
```

```
oualphaBASELINE=par[1];
betaA1=par[2];
betaA2=par[3];
betaA3=-(par[2]+par[3]);
lambdaBASELINE=exp(par[1]);
lambdaA1=exp(par[1]+par[2]);
lambdaA2=exp(par[1]+par[3]);
lambdaA3=exp(par[1]+betaA3);
alpha=par[4];

**********Indices, lambda's and constant alpha for age levels******************;
indexA1=exp(betaA1);        *constant shape parameter;
indexA2=exp(betaA2);
indexA3=exp(betaA3);

lambdaA1=lambdaBASELINE#indexA1;    *same as lambdaA1=exp(par[1]+par[2]);
lambdaA2=lambdaBASELINE#indexA2;    *same as lambdaA2=exp(par[1]+par[3]);
lambdaA3=lambdaBASELINE#indexA3;    *same as lambdaA3=exp(par[1]+betaA3);
oualphaA1=log(lambdaA1);            *same as oualphaA1=oualphaBASELINE+betaA1;
oualphaA2=log(lambdaA2);
oualphaA3=log(lambdaA3);

print ' Loglogistic parameters, beta effects and indices for age levels A1 A2 A3';

print 'lambdaBASELINE=' lambdaBASELINE[format=E20.]
'oualphaBASELINE=' oualphaBASELINE alpha;
print 'lambda(age=A1)=' lambdaA1[format=E20.]
'lambda(age=A2)=' lambdaA2[format=E20.]
'lambda(age=A3)=' lambdaA3[format=E20.];
print 'alpha=' alpha[format=E20.] oualphaA1 oualphaA2 oualphaA3;
print 'beta(age=A1)=' betaA1 'beta(age=A2)=' betaA2 'beta(age=A3)=' betaA3
'index(age=A1)=' indexA1[format=E20.] 'index(age=A2)=' indexA2[format=E20.]
'index(age=A3)=' indexA3[format=E20.];

**********Hazard ratio and Odds ratio*******************************************;
hazBASELINE12=(lambdaBASELINE*alpha*12**(alpha-1))/(1+lambdaBASELINE*12**alpha);
hazBASELINE24=(lambdaBASELINE*alpha*24**(alpha-1))/(1+lambdaBASELINE*24**alpha);
survBASELINE12=(1+lambdaBASELINE*12**alpha)**(-1);
survBASELINE24=(1+lambdaBASELINE*24**alpha)**(-1);
oddsBASELINE12=(1-survBASELINE12)/survBASELINE12;
oddsBASELINE24=(1-survBASELINE24)/survBASELINE24;

hazAGE_A1_12=(lambdaA1*alpha*12**(alpha-1))/(1+lambdaA1*12**alpha);
hazAGE_A1_24=(lambdaA1*alpha*24**(alpha-1))/(1+lambdaA1*24**alpha);
survAGE_A1_12=(1+lambdaA1*12**alpha)**(-1);
survAGE_A1_24=(1+lambdaA1*24**alpha)**(-1);
oddsAGE_A1_12=(1-survAGE_A1_12)/survAGE_A1_12;
oddsAGE_A1_24=(1-survAGE_A1_24)/survAGE_A1_24;

hazratioAGE_A1_12=hazAGE_A1_12/hazBASELINE12;
hazratioAGE_A1_24=hazAGE_A1_24/hazBASELINE24;
oddsratioAGE_A1_12=oddsAGE_A1_12/oddsBASELINE12;
oddsratioAGE_A1_24=oddsAGE_A1_24/oddsBASELINE24;

print hazBASELINE12 hazBASELINE24 survBASELINE12 survBASELINE24
      oddsBASELINE12 oddsBASELINE24;

print hazAGE_A1_12 hazAGE_A1_24;
print survAGE_A1_12 survAGE_A1_24;
print oddsAGE_A1_12 oddsAGE_A1_24;
print hazratioAGE_A1_12 hazratioAGE_A1_24 oddsratioAGE_A1_12 oddsratioAGE_A1_24;
*******************************************************************************;
hazAGE_A2_12=(lambdaA2*alpha*12**(alpha-1))/(1+lambdaA2*12**alpha);
hazAGE_A2_24=(lambdaA2*alpha*24**(alpha-1))/(1+lambdaA2*24**alpha);
survAGE_A2_12=(1+lambdaA2*12**alpha)**(-1);
survAGE_A2_24=(1+lambdaA2*24**alpha)**(-1);
oddsAGE_A2_12=(1-survAGE_A2_12)/survAGE_A2_12;
oddsAGE_A2_24=(1-survAGE_A2_24)/survAGE_A2_24;

hazratioAGE_A2_12=hazAGE_A2_12/hazBASELINE12;
hazratioAGE_A2_24=hazAGE_A2_24/hazBASELINE24;
oddsratioAGE_A2_12=oddsAGE_A2_12/oddsBASELINE12;
oddsratioAGE_A2_24=oddsAGE_A2_24/oddsBASELINE24;

print hazAGE_A2_12 hazAGE_A2_24;
print survAGE_A2_12 survAGE_A2_24;
print oddsAGE_A2_12 oddsAGE_A2_24;
print hazratioAGE_A2_12 hazratioAGE_A2_24 oddsratioAGE_A2_12 oddsratioAGE_A2_24;
*******************************************************************************;
hazAGE_A3_12=(lambdaA3*alpha*12**(alpha-1))/(1+lambdaA3*12**alpha);
hazAGE_A3_24=(lambdaA3*alpha*24**(alpha-1))/(1+lambdaA3*24**alpha);
survAGE_A3_12=(1+lambdaA3*12**alpha)**(-1);
survAGE_A3_24=(1+lambdaA3*24**alpha)**(-1);
oddsAGE_A3_12=(1-survAGE_A3_12)/survAGE_A3_12;
oddsAGE_A3_24=(1-survAGE_A3_24)/survAGE_A3_24;

hazratioAGE_A3_12=hazAGE_A3_12/hazBASELINE12;
hazratioAGE_A3_24=hazAGE_A3_24/hazBASELINE24;
oddsratioAGE_A3_12=oddsAGE_A3_12/oddsBASELINE12;
```

```
oddsratioAGE_A3_24=oddsAGE_A3_24/oddsBASELINE24;

print hazAGE_A3_12 hazAGE_A3_24;
print survAGE_A3_12 survAGE_A3_24;
print oddsAGE_A3_12 oddsAGE_A3_24;
print hazratioAGE_A3_12 hazratioAGE_A3_24 oddsratioAGE_A3_12 oddsratioAGE_A3_24;

**********Median Lifetime*****************************************************;
medianBASELINE=(1/lambdaBASELINE)##(1/alpha);
medianA1=(1/lambdaA1)##(1/alpha);
medianA2=(1/lambdaA2)##(1/alpha);
medianA3=(1/lambdaA3)##(1/alpha);

 perc5BASELINE=((1/lambdaBASELINE)#( 5/(100- 5)))##(1/alpha);
perc10BASELINE=((1/lambdaBASELINE)#(10/(100-10)))##(1/alpha);
perc20BASELINE=((1/lambdaBASELINE)#(20/(100-20)))##(1/alpha);
perc25BASELINE=((1/lambdaBASELINE)#(25/(100-25)))##(1/alpha);
perc30BASELINE=((1/lambdaBASELINE)#(30/(100-30)))##(1/alpha);
perc40BASELINE=((1/lambdaBASELINE)#(40/(100-40)))##(1/alpha);
perc50BASELINE=((1/lambdaBASELINE)#(50/(100-50)))##(1/alpha);
perc60BASELINE=((1/lambdaBASELINE)#(60/(100-60)))##(1/alpha);
perc70BASELINE=((1/lambdaBASELINE)#(70/(100-70)))##(1/alpha);
perc75BASELINE=((1/lambdaBASELINE)#(75/(100-75)))##(1/alpha);
perc80BASELINE=((1/lambdaBASELINE)#(80/(100-80)))##(1/alpha);
perc90BASELINE=((1/lambdaBASELINE)#(90/(100-90)))##(1/alpha);
perc95BASELINE=((1/lambdaBASELINE)#(95/(100-95)))##(1/alpha);

 perc5A1=((1/lambdaA1)#( 5/(100- 5)))##(1/alpha);
perc10A1=((1/lambdaA1)#(10/(100-10)))##(1/alpha);
perc20A1=((1/lambdaA1)#(20/(100-20)))##(1/alpha);
perc25A1=((1/lambdaA1)#(25/(100-25)))##(1/alpha);
perc30A1=((1/lambdaA1)#(30/(100-30)))##(1/alpha);
perc40A1=((1/lambdaA1)#(40/(100-40)))##(1/alpha);
perc50A1=((1/lambdaA1)#(50/(100-50)))##(1/alpha);
perc60A1=((1/lambdaA1)#(60/(100-60)))##(1/alpha);
perc70A1=((1/lambdaA1)#(70/(100-70)))##(1/alpha);
perc75A1=((1/lambdaA1)#(75/(100-75)))##(1/alpha);
perc80A1=((1/lambdaA1)#(80/(100-80)))##(1/alpha);
perc90A1=((1/lambdaA1)#(90/(100-90)))##(1/alpha);
perc95A1=((1/lambdaA1)#(95/(100-95)))##(1/alpha);

 perc5A2=((1/lambdaA2)#( 5/(100- 5)))##(1/alpha);
perc10A2=((1/lambdaA2)#(10/(100-10)))##(1/alpha);
perc20A2=((1/lambdaA2)#(20/(100-20)))##(1/alpha);
perc25A2=((1/lambdaA2)#(25/(100-25)))##(1/alpha);
perc30A2=((1/lambdaA2)#(30/(100-30)))##(1/alpha);
perc40A2=((1/lambdaA2)#(40/(100-40)))##(1/alpha);
perc50A2=((1/lambdaA2)#(50/(100-50)))##(1/alpha);
perc60A2=((1/lambdaA2)#(60/(100-60)))##(1/alpha);
perc70A2=((1/lambdaA2)#(70/(100-70)))##(1/alpha);
perc75A2=((1/lambdaA2)#(75/(100-75)))##(1/alpha);
perc80A2=((1/lambdaA2)#(80/(100-80)))##(1/alpha);
perc90A2=((1/lambdaA2)#(90/(100-90)))##(1/alpha);
perc95A2=((1/lambdaA2)#(95/(100-95)))##(1/alpha);

 perc5A3=((1/lambdaA3)#( 5/(100- 5)))##(1/alpha);
perc10A3=((1/lambdaA3)#(10/(100-10)))##(1/alpha);
perc20A3=((1/lambdaA3)#(20/(100-20)))##(1/alpha);
perc25A3=((1/lambdaA3)#(25/(100-25)))##(1/alpha);
perc30A3=((1/lambdaA3)#(30/(100-30)))##(1/alpha);
perc40A3=((1/lambdaA3)#(40/(100-40)))##(1/alpha);
perc50A3=((1/lambdaA3)#(50/(100-50)))##(1/alpha);
perc60A3=((1/lambdaA3)#(60/(100-60)))##(1/alpha);
perc70A3=((1/lambdaA3)#(70/(100-70)))##(1/alpha);
perc75A3=((1/lambdaA3)#(75/(100-75)))##(1/alpha);
perc80A3=((1/lambdaA3)#(80/(100-80)))##(1/alpha);
perc90A3=((1/lambdaA3)#(90/(100-90)))##(1/alpha);
perc95A3=((1/lambdaA3)#(95/(100-95)))##(1/alpha);

print perc5BASELINE perc10BASELINE perc20BASELINE perc25BASELINE perc30BASELINE;
print perc40BASELINE perc50BASELINE perc60BASELINE perc70BASELINE;
print perc75BASELINE perc80BASELINE perc90BASELINE perc95BASELINE;

print perc5A1 perc10A1 perc20A1 perc25A1 perc30A1;
print perc40A1 perc50A1 perc60A1 perc70A1;
print perc75A1 perc80A1 perc90A1 perc95A1;

print perc5A2 perc10A2 perc20A2 perc25A2 perc30A2;
print perc40A2 perc50A2 perc60A2 perc70A2;
print perc75A2 perc80A2 perc90A2 perc95A2;

print perc5A3 perc10A3 perc20A3 perc25A3 perc30A3;
print perc40A3 perc50A3 perc60A3 perc70A3;
print perc75A3 perc80A3 perc90A3 perc95A3;

print 'medianlifetimeBASELINE=' medianBASELINE;
print 'medianlifetime(age=A1)=' medianA1
      'medianlifetime(age=A2)=' medianA2
      'medianlifetime(age=A3)=' medianA3;
```

2. Program for fitting a log-logistic regression model (shape alters, one covariate)

```
title1 'Fitting of regression model with one covariate';
title2 'Staggered entry: shape parameter alters';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector (first entry,3 agegroups);
                f11={29,59,95,73,108,15,642};
                f12={21,50,91,45,75,13,553};
                f13={16,49,68,39, 67, 7,471};

**********Vector of upper boundaries;
                x1={12,17,24,28,34,37};

**********Frequency vector (second entry,3 agegroups);
                f21={41,75,103,92,83,628};
                f22={49,62,61,66,54,753};
                f23={28,29,65,42,35,543};

**********Vector of upper boundaries;
                x2={12,17,24,28,34};

**********Frequency vector (third entry,3 agegroups);
                f31={68,34,99,57,570};
                f32={40,44,83,33,533};
                f33={46,21,60,27,571};

**********Vector of upper boundaries;
                x3={12,17,24,28};

**********Frequency vector (fourth entry,3 agegroups);
                f41={71,60,69,573};
                f42={54,61,68,616};
                f43={50,45,70,659};

**********Vector of upper boundaries;
                x4={12,17,24};

**********Relative frequency vectors;
n11=f11[+];    n21=f21[+];    n31=f31[+];    n41=f41[+];    n1=n11+n21+n31+n41;
n12=f12[+];    n22=f22[+];    n32=f32[+];    n42=f42[+];    n2=n12+n22+n32+n42;
n13=f13[+];    n23=f23[+];    n33=f33[+];    n43=f43[+];    n3=n13+n23+n33+n43;
                                                            n=n1+n2+n3;
k1=nrow(f11); d1=k1-1;
k2=nrow(f21); d2=k2-1;
k3=nrow(f31); d3=k3-1;
k4=nrow(f41); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;

p11=f11/n11;    p21=f21/n21;    p31=f31/n31;    p41=f41/n41;    p1=p11//p21//p31//p41;
p12=f12/n12;    p22=f22/n22;    p32=f32/n32;    p42=f42/n42;    p2=p12//p22//p32//p42;
p13=f13/n13;    p23=f23/n23;    p33=f33/n33;    p43=f43/n43;    p3=p13//p23//p33//p43;
                                                                p=p1//p2//p3;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);
S=I(3)@S;

AA=J(54,1,1);       *54=3 times((7-1)+(6-1)+(5-1)+(4-1))=3 times 18=54;
BB=J(3,1,1);
CC=(I(2)//J(1,2,-1));
DD=J(18,1,1);            *d=18;
EE=J(9,1,1);

lx=log(x1)//log(x2)//log(x3)//log(x4);

xc=AA||(CC@DD)||I(3)@lx;  print xc;

C=I(3#d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
```

```
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
*Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;              *Weibull;
Gm=C*diag(1/ms+1/(1-ms))*S;                               *log-logistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
    *g=C*log(-log(1-ps));                                 *Weibull;
    g=C*(log(ps)-log(1-ps));                              *log-logistic;
    *Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;          *Weibull;
    Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                   *log-logistic;
*****************covariance matrix***********************************;
m11=m[1:k1]; m21=m[k1+1:k1+k2];
m31=m[k1+k2+1:k1+k2+k3]; m41=m[k1+k2+k3+1:k1+k2+k3+k4];

m12=m[k+1:k+k1]; m22=m[k+k1+1:k+k1+k2];
m32=m[k+k1+k2+1:k+k1+k2+k3]; m42=m[k+k1+k2+k3+1:k+k1+k2+k3+k4];

m13=m[2#k+1:2#k+k1]; m23=m[2#k+k1+1:2#k+k1+k2];
m33=m[2#k+k1+k2+1:2#k+k1+k2+k3]; m43=m[2#k+k1+k2+k3+1:2#k+k1+k2+k3+k4];

sig11=(1/n11)*(diag(m11)-m11*m11`);
sig21=(1/n21)*(diag(m21)-m21*m21`);
sig31=(1/n31)*(diag(m31)-m31*m31`);
sig41=(1/n41)*(diag(m41)-m41*m41`);
sig1=block(sig11,sig21,sig31,sig41);

sig12=(1/n12)*(diag(m12)-m12*m12`);
sig22=(1/n22)*(diag(m22)-m22*m22`);
sig32=(1/n32)*(diag(m32)-m32*m32`);
sig42=(1/n42)*(diag(m42)-m42*m42`);
sig2=block(sig12,sig22,sig32,sig42);

sig13=(1/n13)*(diag(m13)-m13*m13`);
sig23=(1/n23)*(diag(m23)-m23*m23`);
sig33=(1/n33)*(diag(m33)-m33*m33`);
sig43=(1/n43)*(diag(m43)-m43*m43`);
sig3=block(sig13,sig23,sig33,sig43);

sig=block(sig1,sig2,sig3);
V=sig;
*********************************************************************;
    *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                        *Weibull;
    p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                         *log-logistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
*par=inv(xc`*xc)*xc`*log(-log(1-ms));                       *Weibull;
par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms)); print par;         *log-logistic;

print par[format=E20.];

**********Regression coefficients;
oualphaBASELINE=par[1];
lambdaBASELINE=exp(par[1]);
betaA1=par[2];
betaA2=par[3];
betaA3=-(par[2]+par[3]);
lambdaA1=exp(par[1]+par[2]);
lambdaA2=exp(par[1]+par[3]);
lambdaA3=exp(par[1]+betaA3);
alphaA1=par[4];
alphaA2=par[5];
alphaA3=par[6];
alphaBASELINE=(n1#alphaA1+n2#alphaA2+n3#alphaA3)/n;

**********Lambda's and alpha's for age levels*****************;
lambdaA1=exp(par[1]+par[2]);
lambdaA2=exp(par[1]+par[3]);
lambdaA3=exp(par[1]+betaA3);
oualphaA1=log(lambdaA1);
oualphaA2=log(lambdaA2);
oualphaA3=log(lambdaA3);

print ' Loglogistic parameters and beta effects for age levels A1 A2 A3';
print 'lambdaBASELINE=' lambdaBASELINE[format=E20.]
```

```
      'oualphaBASELINE=' oualphaBASELINE alphaBASELINE;
print 'lambda(age=A1)=' lambdaA1[format=E20.]
      'lambda(age=A2)=' lambdaA2[format=E20.]
      'lambda(age=A3)=' lambdaA3[format=E20.];
print oualphaA1 oualphaA2 oualphaA3;
print 'alpha(age=A1)=' alphaA1[format=E20.]
      'alpha(age=A2)=' alphaA2[format=E20.]
      'alpha(age=A3)=' alphaA3[format=E20.];

print 'beta(age=A1)=' betaA1 'beta(age=A2)=' betaA2 'beta(age=A3)=' betaA3;

**********Hazard ratio and Odds ratio******************************************;
hazBASELINE12=(lambdaBASELINE*alphaBASELINE*12**(alphaBASELINE-1))/(1+lambdaBASELINE*12**alphaBASELINE);
hazBASELINE24=(lambdaBASELINE*alphaBASELINE*24**(alphaBASELINE-1))/(1+lambdaBASELINE*24**alphaBASELINE);
survBASELINE12=(1+lambdaBASELINE*12**alphaBASELINE)**(-1);
survBASELINE24=(1+lambdaBASELINE*24**alphaBASELINE)**(-1);
oddsBASELINE12=(1-survBASELINE12)/survBASELINE12;
oddsBASELINE24=(1-survBASELINE24)/survBASELINE24;

hazAGE_A1_12=(lambdaA1*alphaA1*12**(alphaA1-1))/(1+lambdaA1*12**alphaA1);
hazAGE_A1_24=(lambdaA1*alphaA1*24**(alphaA1-1))/(1+lambdaA1*24**alphaA1);
survAGE_A1_12=(1+lambdaA1*12**alphaA1)**(-1);
survAGE_A1_24=(1+lambdaA1*24**alphaA1)**(-1);
oddsAGE_A1_12=(1-survAGE_A1_12)/survAGE_A1_12;
oddsAGE_A1_24=(1-survAGE_A1_24)/survAGE_A1_24;

hazratioAGE_A1_12=hazAGE_A1_12/hazBASELINE12;
hazratioAGE_A1_24=hazAGE_A1_24/hazBASELINE24;
oddsratioAGE_A1_12=oddsAGE_A1_12/oddsBASELINE12;
oddsratioAGE_A1_24=oddsAGE_A1_24/oddsBASELINE24;

print hazBASELINE12 hazBASELINE24 survBASELINE12 survBASELINE24
      oddsBASELINE12 oddsBASELINE24;

print hazAGE_A1_12 hazAGE_A1_24;
print survAGE_A1_12 survAGE_A1_24;
print oddsAGE_A1_12 oddsAGE_A1_24;
print hazratioAGE_A1_12 hazratioAGE_A1_24 oddsratioAGE_A1_12 oddsratioAGE_A1_24;
******************************************************************************;
hazAGE_A2_12=(lambdaA2*alphaA2*12**(alphaA2-1))/(1+lambdaA2*12**alphaA2);
hazAGE_A2_24=(lambdaA2*alphaA2*24**(alphaA2-1))/(1+lambdaA2*24**alphaA2);
survAGE_A2_12=(1+lambdaA2*12**alphaA2)**(-1);
survAGE_A2_24=(1+lambdaA2*24**alphaA2)**(-1);
oddsAGE_A2_12=(1-survAGE_A2_12)/survAGE_A2_12;
oddsAGE_A2_24=(1-survAGE_A2_24)/survAGE_A2_24;

hazratioAGE_A2_12=hazAGE_A2_12/hazBASELINE12;
hazratioAGE_A2_24=hazAGE_A2_24/hazBASELINE24;
oddsratioAGE_A2_12=oddsAGE_A2_12/oddsBASELINE12;
oddsratioAGE_A2_24=oddsAGE_A2_24/oddsBASELINE24;

print hazAGE_A2_12 hazAGE_A2_24;
print survAGE_A2_12 survAGE_A2_24;
print oddsAGE_A2_12 oddsAGE_A2_24;
print hazratioAGE_A2_12 hazratioAGE_A2_24 oddsratioAGE_A2_12 oddsratioAGE_A2_24;
******************************************************************************;
hazAGE_A3_12=(lambdaA3*alphaA3*12**(alphaA3-1))/(1+lambdaA3*12**alphaA3);
hazAGE_A3_24=(lambdaA3*alphaA3*24**(alphaA3-1))/(1+lambdaA3*24**alphaA3);
survAGE_A3_12=(1+lambdaA3*12**alphaA3)**(-1);
survAGE_A3_24=(1+lambdaA3*24**alphaA3)**(-1);
oddsAGE_A3_12=(1-survAGE_A3_12)/survAGE_A3_12;
oddsAGE_A3_24=(1-survAGE_A3_24)/survAGE_A3_24;

hazratioAGE_A3_12=hazAGE_A3_12/hazBASELINE12;
hazratioAGE_A3_24=hazAGE_A3_24/hazBASELINE24;
oddsratioAGE_A3_12=oddsAGE_A3_12/oddsBASELINE12;
oddsratioAGE_A3_24=oddsAGE_A3_24/oddsBASELINE24;

print hazAGE_A3_12 hazAGE_A3_24;
print survAGE_A3_12 survAGE_A3_24;
print oddsAGE_A3_12 oddsAGE_A3_24;
print hazratioAGE_A3_12 hazratioAGE_A3_24 oddsratioAGE_A3_12 oddsratioAGE_A3_24;

**********Median Lifetime*****************************************************;
medianBASELINE=(1/lambdaBASELINE)##(1/alphaBASELINE);
medianA1=(1/lambdaA1)##(1/alphaA1);
medianA2=(1/lambdaA2)##(1/alphaA2);
medianA3=(1/lambdaA3)##(1/alphaA3);

 perc5BASELINE=((1/lambdaBASELINE)#( 5/(100- 5))##(1/alphaBASELINE);
perc10BASELINE=((1/lambdaBASELINE)#(10/(100-10)))##(1/alphaBASELINE);
perc20BASELINE=((1/lambdaBASELINE)#(20/(100-20)))##(1/alphaBASELINE);
perc25BASELINE=((1/lambdaBASELINE)#(25/(100-25)))##(1/alphaBASELINE);
perc30BASELINE=((1/lambdaBASELINE)#(30/(100-30)))##(1/alphaBASELINE);
perc40BASELINE=((1/lambdaBASELINE)#(40/(100-40)))##(1/alphaBASELINE);
perc50BASELINE=((1/lambdaBASELINE)#(50/(100-50)))##(1/alphaBASELINE);
perc60BASELINE=((1/lambdaBASELINE)#(60/(100-60)))##(1/alphaBASELINE);
perc70BASELINE=((1/lambdaBASELINE)#(70/(100-70)))##(1/alphaBASELINE);
perc75BASELINE=((1/lambdaBASELINE)#(75/(100-75)))##(1/alphaBASELINE);
```

```
perc80BASELINE=((1/lambdaBASELINE)#(80/(100-80)))##(1/alphaBASELINE);
perc90BASELINE=((1/lambdaBASELINE)#(90/(100-90)))##(1/alphaBASELINE);
perc95BASELINE=((1/lambdaBASELINE)#(95/(100-95)))##(1/alphaBASELINE);

 perc5A1=((1/lambdaA1)#( 5/(100- 5)))##(1/alphaA1);
perc10A1=((1/lambdaA1)#(10/(100-10)))##(1/alphaA1);
perc20A1=((1/lambdaA1)#(20/(100-20)))##(1/alphaA1);
perc25A1=((1/lambdaA1)#(25/(100-25)))##(1/alphaA1);
perc30A1=((1/lambdaA1)#(30/(100-30)))##(1/alphaA1);
perc40A1=((1/lambdaA1)#(40/(100-40)))##(1/alphaA1);
perc50A1=((1/lambdaA1)#(50/(100-50)))##(1/alphaA1);
perc60A1=((1/lambdaA1)#(60/(100-60)))##(1/alphaA1);
perc70A1=((1/lambdaA1)#(70/(100-70)))##(1/alphaA1);
perc75A1=((1/lambdaA1)#(75/(100-75)))##(1/alphaA1);
perc80A1=((1/lambdaA1)#(80/(100-80)))##(1/alphaA1);
perc90A1=((1/lambdaA1)#(90/(100-90)))##(1/alphaA1);
perc95A1=((1/lambdaA1)#(95/(100-95)))##(1/alphaA1);

 perc5A2=((1/lambdaA2)#( 5/(100- 5)))##(1/alphaA2);
perc10A2=((1/lambdaA2)#(10/(100-10)))##(1/alphaA2);
perc20A2=((1/lambdaA2)#(20/(100-20)))##(1/alphaA2);
perc25A2=((1/lambdaA2)#(25/(100-25)))##(1/alphaA2);
perc30A2=((1/lambdaA2)#(30/(100-30)))##(1/alphaA2);
perc40A2=((1/lambdaA2)#(40/(100-40)))##(1/alphaA2);
perc50A2=((1/lambdaA2)#(50/(100-50)))##(1/alphaA2);
perc60A2=((1/lambdaA2)#(60/(100-60)))##(1/alphaA2);
perc70A2=((1/lambdaA2)#(70/(100-70)))##(1/alphaA2);
perc75A2=((1/lambdaA2)#(75/(100-75)))##(1/alphaA2);
perc80A2=((1/lambdaA2)#(80/(100-80)))##(1/alphaA2);
perc90A2=((1/lambdaA2)#(90/(100-90)))##(1/alphaA2);
perc95A2=((1/lambdaA2)#(95/(100-95)))##(1/alphaA2);

 perc5A3=((1/lambdaA3)#( 5/(100- 5)))##(1/alphaA3);
perc10A3=((1/lambdaA3)#(10/(100-10)))##(1/alphaA3);
perc20A3=((1/lambdaA3)#(20/(100-20)))##(1/alphaA3);
perc25A3=((1/lambdaA3)#(25/(100-25)))##(1/alphaA3);
perc30A3=((1/lambdaA3)#(30/(100-30)))##(1/alphaA3);
perc40A3=((1/lambdaA3)#(40/(100-40)))##(1/alphaA3);
perc50A3=((1/lambdaA3)#(50/(100-50)))##(1/alphaA3);
perc60A3=((1/lambdaA3)#(60/(100-60)))##(1/alphaA3);
perc70A3=((1/lambdaA3)#(70/(100-70)))##(1/alphaA3);
perc75A3=((1/lambdaA3)#(75/(100-75)))##(1/alphaA3);
perc80A3=((1/lambdaA3)#(80/(100-80)))##(1/alphaA3);
perc90A3=((1/lambdaA3)#(90/(100-90)))##(1/alphaA3);
perc95A3=((1/lambdaA3)#(95/(100-95)))##(1/alphaA3);

print perc5BASELINE perc10BASELINE perc20BASELINE perc25BASELINE perc30BASELINE;
print perc40BASELINE perc50BASELINE perc60BASELINE perc70BASELINE;
print perc75BASELINE perc80BASELINE perc90BASELINE perc95BASELINE;

print perc5A1 perc10A1 perc20A1 perc25A1 perc30A1;
print perc40A1 perc50A1 perc60A1 perc70A1;
print perc75A1 perc80A1 perc90A1 perc95A1;

print perc5A2 perc10A2 perc20A2 perc25A2 perc30A2;
print perc40A2 perc50A2 perc60A2 perc70A2;
print perc75A2 perc80A2 perc90A2 perc95A2;

print perc5A3 perc10A3 perc20A3 perc25A3 perc30A3;
print perc40A3 perc50A3 perc60A3 perc70A3;
print perc75A3 perc80A3 perc90A3 perc95A3;

print 'medianlifetimeBASELINE=' medianBASELINE;
print 'medianlifetime(age=A1)=' medianA1
      'medianlifetime(age=A2)=' medianA2
      'medianlifetime(age=A3)=' medianA3;
```

3. Program for fitting a Weibull regression model (constant shape, one covariate)

```
title1 'Fitting of regression model with one covariate';
title2 'Staggered entry: constant shape parameter';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector (first entry,3 agegroups);
                f11={29,59,95,73,108,15,642};
                f12={21,50,91,45,75,13,553};
                f13={16,49,68,39, 67, 7,471};

**********Vector of upper boundaries;
                x1={12,17,24,28,34,37};

**********Frequency vector (second entry,3 agegroups);
                f21={41,75,103,92,83,628};
                f22={49,62,61,66,54,753};
                f23={28,29,65,42,35,543};
```

```
**********Vector of upper boundaries;
                x2={12,17,24,28,34};

**********Frequency vector (third entry,3 agegroups);
                f31={68,34,99,57,570};
                f32={40,44,83,33,533};
                f33={46,21,60,27,571};

**********Vector of upper boundaries;
                x3={12,17,24,28};

**********Frequency vector (fourth entry,3 agegroups);
                f41={71,60,69,573};
                f42={54,61,68,616};
                f43={50,45,70,659};

**********Vector of upper boundaries;
                x4={12,17,24};

**********Relative frequency vectors;
n11=f11[+];    n21=f21[+];    n31=f31[+];    n41=f41[+];    n1=n11+n21+n31+n41;
n12=f12[+];    n22=f22[+];    n32=f32[+];    n42=f42[+];    n2=n12+n22+n32+n42;
n13=f13[+];    n23=f23[+];    n33=f33[+];    n43=f43[+];    n3=n13+n23+n33+n43;
                                                             n=n1+n2+n3;
k1=nrow(f11); d1=k1-1;
k2=nrow(f21); d2=k2-1;
k3=nrow(f31); d3=k3-1;
k4=nrow(f41); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;

p11=f11/n11;    p21=f21/n21;    p31=f31/n31;    p41=f41/n41;    p1=p11//p21//p31//p41;
p12=f12/n12;    p22=f22/n22;    p32=f32/n32;    p42=f42/n42;    p2=p12//p22//p32//p42;
p13=f13/n13;    p23=f23/n23;    p33=f33/n33;    p43=f43/n43;    p3=p13//p23//p33//p43;
                                                                p=p1//p2//p3;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);
S=I(3)@S;

AA=J(54,1,1);        *54=3 times((7-1)+(6-1)+(5-1)+(4-1))=3 times 18=54;
BB=J(3,1,1);
CC=(I(2)//J(1,2,-1));
DD=J(18,1,1);              *d=18;
EE=J(9,1,1);

lx=BB@(log(x1)//log(x2)//log(x3)//log(x4));

xc=AA||(CC@DD)||lx;  print xc;

C=I(3#d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;              *Weibull;
*Gm=C*diag(1/ms+1/(1-ms))*S;                             *log-logistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
    g=C*log(-log(1-ps));                                 *Weibull;
    *g=C*(log(ps)-log(1-ps));                            *Loglogistic;
    Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;          *Weibull;
    *Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                 *loglogistic;
******************covariance matrix***********************************;
m11=m[1:k1]; m21=m[k1+1:k1+k2];
m31=m[k1+k2+1:k1+k2+k3]; m41=m[k1+k2+k3+1:k1+k2+k3+k4];
```

```
m12=m[k+1:k+k1]; m22=m[k+k1+1:k+k1+k2];
m32=m[k+k1+k2+1:k+k1+k2+k3]; m42=m[k+k1+k2+k3+1:k+k1+k2+k3+k4];

m13=m[2#k+1:2#k+k1]; m23=m[2#k+k1+1:2#k+k1+k2];
m33=m[2#k+k1+k2+1:2#k+k1+k2+k3]; m43=m[2#k+k1+k2+k3+1:2#k+k1+k2+k3+k4];

sig11=(1/n11)*(diag(m11)-m11*m11`);
sig21=(1/n21)*(diag(m21)-m21*m21`);
sig31=(1/n31)*(diag(m31)-m31*m31`);
sig41=(1/n41)*(diag(m41)-m41*m41`);
sig1=block(sig11,sig21,sig31,sig41);

sig12=(1/n12)*(diag(m12)-m12*m12`);
sig22=(1/n22)*(diag(m22)-m22*m22`);
sig32=(1/n32)*(diag(m32)-m32*m32`);
sig42=(1/n42)*(diag(m42)-m42*m42`);
sig2=block(sig12,sig22,sig32,sig42);

sig13=(1/n13)*(diag(m13)-m13*m13`);
sig23=(1/n23)*(diag(m23)-m23*m23`);
sig33=(1/n33)*(diag(m33)-m33*m33`);
sig43=(1/n43)*(diag(m43)-m43*m43`);
sig3=block(sig13,sig23,sig33,sig43);

sig=block(sig1,sig2,sig3);
V=sig;
***********************************************************************;
    p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                           *Weibull;
    *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                          *loglogistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
par=inv(xc`*xc)*xc`*log(-log(1-ms));                          *Weibull;
*par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms)); print par;          *log-logistic;

print par[format=E20.];

**********Regression coefficients;
oualphaBASELINE=par[1];
betaA1=par[2];
betaA2=par[3];
betaA3=-(par[2]+par[3]);
lambdaBASELINE=exp(par[1]);
lambdaA1=exp(par[1]+par[2]);
lambdaA2=exp(par[1]+par[3]);
lambdaA3=exp(par[1]+betaA3);
alpha=par[4];

**********Risk scores, lambda's and constant alpha for age levels******************;
riskscoreA1=exp(betaA1);      *constant shape parameter;
riskscoreA2=exp(betaA2);
riskscoreA3=exp(betaA3);

lambdaA1=lambdaBASELINE#riskscoreA1;
lambdaA2=lambdaBASELINE#riskscoreA2;
lambdaA3=lambdaBASELINE#riskscoreA3;
oualphaA1=log(lambdaA1);
oualphaA2=log(lambdaA2);
oualphaA3=log(lambdaA3);

print ' Weibull parameters, beta effects and risk scores for age levels A1 A2 A3';
print 'lambdaBASELINE=' lambdaBASELINE[format=E20.]
'oualphaBASELINE=' oualphaBASELINE
'lambda(age=A1)=' lambdaA1[format=E20.]
'lambda(age=A2)=' lambdaA2[format=E20.]
'lambda(age=A3)=' lambdaA3[format=E20.]
'alpha=' alpha[format=E20.] oualphaA1 oualphaA2 oualphaA3
'beta(age=A1)=' betaA1 'beta(age=A2)=' betaA2 'beta(age=A3)=' betaA3
'riskscore(age=A1)=' riskscoreA1[format=E20.] 'riskscore(age=A2)=' riskscoreA2[format=E20.]
'riskscore(age=A3)=' riskscoreA3[format=E20.];

**********Hazard ratio and Odds ratio*******************************************;
hazBASELINE12=lambdaBASELINE*alpha*12**(alpha-1);
hazBASELINE24=lambdaBASELINE*alpha*24**(alpha-1);
survBASELINE12=exp(-lambdaBASELINE*12**alpha);
survBASELINE24=exp(-lambdaBASELINE*24**alpha);
oddsBASELINE12=(1-survBASELINE12)/survBASELINE12;
oddsBASELINE24=(1-survBASELINE24)/survBASELINE24;

hazAGE_A1_12=lambdaA1*alpha*12**(alpha-1);
hazAGE_A1_24=lambdaA1*alpha*24**(alpha-1);
survAGE_A1_12=exp(-lambdaA1*12**alpha);
survAGE_A1_24=exp(-lambdaA1*24**alpha);
```

```
oddsAGE_A1_12=(1-survAGE_A1_12)/survAGE_A1_12;
oddsAGE_A1_24=(1-survAGE_A1_24)/survAGE_A1_24;

hazratioAGE_A1_12=hazAGE_A1_12/hazBASELINE12;
hazratioAGE_A1_24=hazAGE_A1_24/hazBASELINE24;
oddsratioAGE_A1_12=oddsAGE_A1_12/oddsBASELINE12;
oddsratioAGE_A1_24=oddsAGE_A1_24/oddsBASELINE24;

print hazBASELINE12 hazBASELINE24 survBASELINE12 survBASELINE24
      oddsBASELINE12 oddsBASELINE24;

print hazAGE_A1_12 hazAGE_A1_24;
print survAGE_A1_12 survAGE_A1_24;
print oddsAGE_A1_12 oddsAGE_A1_24;
print hazratioAGE_A1_12 hazratioAGE_A1_24 oddsratioAGE_A1_12 oddsratioAGE_A1_24;
********************************************************************************;
hazAGE_A2_12=lambdaA2*alpha*12**(alpha-1);
hazAGE_A2_24=lambdaA2*alpha*24**(alpha-1);
survAGE_A2_12=exp(-lambdaA2*12**alpha);
survAGE_A2_24=exp(-lambdaA2*24**alpha);
oddsAGE_A2_12=(1-survAGE_A2_12)/survAGE_A2_12;
oddsAGE_A2_24=(1-survAGE_A2_24)/survAGE_A2_24;

hazratioAGE_A2_12=hazAGE_A2_12/hazBASELINE12;
hazratioAGE_A2_24=hazAGE_A2_24/hazBASELINE24;
oddsratioAGE_A2_12=oddsAGE_A2_12/oddsBASELINE12;
oddsratioAGE_A2_24=oddsAGE_A2_24/oddsBASELINE24;

print hazAGE_A2_12 hazAGE_A2_24;
print survAGE_A2_12 survAGE_A2_24;
print oddsAGE_A2_12 oddsAGE_A2_24;
print hazratioAGE_A2_12 hazratioAGE_A2_24 oddsratioAGE_A2_12 oddsratioAGE_A2_24;
********************************************************************************;
hazAGE_A3_12=lambdaA3*alpha*12**(alpha-1);
hazAGE_A3_24=lambdaA3*alpha*24**(alpha-1);
survAGE_A3_12=exp(-lambdaA3*12**alpha);
survAGE_A3_24=exp(-lambdaA3*24**alpha);
oddsAGE_A3_12=(1-survAGE_A3_12)/survAGE_A3_12;
oddsAGE_A3_24=(1-survAGE_A3_24)/survAGE_A3_24;

hazratioAGE_A3_12=hazAGE_A3_12/hazBASELINE12;
hazratioAGE_A3_24=hazAGE_A3_24/hazBASELINE24;
oddsratioAGE_A3_12=oddsAGE_A3_12/oddsBASELINE12;
oddsratioAGE_A3_24=oddsAGE_A3_24/oddsBASELINE24;

print hazAGE_A3_12 hazAGE_A3_24;
print survAGE_A3_12 survAGE_A3_24;
print oddsAGE_A3_12 oddsAGE_A3_24;
print hazratioAGE_A3_12 hazratioAGE_A3_24 oddsratioAGE_A3_12 oddsratioAGE_A3_24;

**********Median Lifetime*******************************************************;
medianBASELINE=((1/lambdaBASELINE)#log(2))##(1/alpha);
medianA1=((1/lambdaA1)#log(2))##(1/alpha);
medianA2=((1/lambdaA2)#log(2))##(1/alpha);
medianA3=((1/lambdaA3)#log(2))##(1/alpha);

 perc5BASELINE=((1/lambdaBASELINE)#log(100/(100- 5)))##(1/alpha);
perc10BASELINE=((1/lambdaBASELINE)#log(100/(100-10)))##(1/alpha);
perc20BASELINE=((1/lambdaBASELINE)#log(100/(100-20)))##(1/alpha);
perc25BASELINE=((1/lambdaBASELINE)#log(100/(100-25)))##(1/alpha);
perc30BASELINE=((1/lambdaBASELINE)#log(100/(100-30)))##(1/alpha);
perc40BASELINE=((1/lambdaBASELINE)#log(100/(100-40)))##(1/alpha);
perc50BASELINE=((1/lambdaBASELINE)#log(100/(100-50)))##(1/alpha);
perc60BASELINE=((1/lambdaBASELINE)#log(100/(100-60)))##(1/alpha);
perc70BASELINE=((1/lambdaBASELINE)#log(100/(100-70)))##(1/alpha);
perc75BASELINE=((1/lambdaBASELINE)#log(100/(100-75)))##(1/alpha);
perc80BASELINE=((1/lambdaBASELINE)#log(100/(100-80)))##(1/alpha);
perc90BASELINE=((1/lambdaBASELINE)#log(100/(100-90)))##(1/alpha);
perc95BASELINE=((1/lambdaBASELINE)#log(100/(100-95)))##(1/alpha);

 perc5A1=((1/lambdaA1)#log(100/(100- 5)))##(1/alpha);
perc10A1=((1/lambdaA1)#log(100/(100-10)))##(1/alpha);
perc20A1=((1/lambdaA1)#log(100/(100-20)))##(1/alpha);
perc25A1=((1/lambdaA1)#log(100/(100-25)))##(1/alpha);
perc30A1=((1/lambdaA1)#log(100/(100-30)))##(1/alpha);
perc40A1=((1/lambdaA1)#log(100/(100-40)))##(1/alpha);
perc50A1=((1/lambdaA1)#log(100/(100-50)))##(1/alpha);
perc60A1=((1/lambdaA1)#log(100/(100-60)))##(1/alpha);
perc70A1=((1/lambdaA1)#log(100/(100-70)))##(1/alpha);
perc75A1=((1/lambdaA1)#log(100/(100-75)))##(1/alpha);
perc80A1=((1/lambdaA1)#log(100/(100-80)))##(1/alpha);
perc90A1=((1/lambdaA1)#log(100/(100-90)))##(1/alpha);
perc95A1=((1/lambdaA1)#log(100/(100-95)))##(1/alpha);

 perc5A2=((1/lambdaA2)#log(100/(100- 5)))##(1/alpha);
perc10A2=((1/lambdaA2)#log(100/(100-10)))##(1/alpha);
perc20A2=((1/lambdaA2)#log(100/(100-20)))##(1/alpha);
perc25A2=((1/lambdaA2)#log(100/(100-25)))##(1/alpha);
perc30A2=((1/lambdaA2)#log(100/(100-30)))##(1/alpha);
```

```
perc40A2=((1/lambdaA2)#log(100/(100-40)))##(1/alpha);
perc50A2=((1/lambdaA2)#log(100/(100-50)))##(1/alpha);
perc60A2=((1/lambdaA2)#log(100/(100-60)))##(1/alpha);
perc70A2=((1/lambdaA2)#log(100/(100-70)))##(1/alpha);
perc75A2=((1/lambdaA2)#log(100/(100-75)))##(1/alpha);
perc80A2=((1/lambdaA2)#log(100/(100-80)))##(1/alpha);
perc90A2=((1/lambdaA2)#log(100/(100-90)))##(1/alpha);
perc95A2=((1/lambdaA2)#log(100/(100-95)))##(1/alpha);

 perc5A3=((1/lambdaA3)#log(100/(100- 5)))##(1/alpha);
perc10A3=((1/lambdaA3)#log(100/(100-10)))##(1/alpha);
perc20A3=((1/lambdaA3)#log(100/(100-20)))##(1/alpha);
perc25A3=((1/lambdaA3)#log(100/(100-25)))##(1/alpha);
perc30A3=((1/lambdaA3)#log(100/(100-30)))##(1/alpha);
perc40A3=((1/lambdaA3)#log(100/(100-40)))##(1/alpha);
perc50A3=((1/lambdaA3)#log(100/(100-50)))##(1/alpha);
perc60A3=((1/lambdaA3)#log(100/(100-60)))##(1/alpha);
perc70A3=((1/lambdaA3)#log(100/(100-70)))##(1/alpha);
perc75A3=((1/lambdaA3)#log(100/(100-75)))##(1/alpha);
perc80A3=((1/lambdaA3)#log(100/(100-80)))##(1/alpha);
perc90A3=((1/lambdaA3)#log(100/(100-90)))##(1/alpha);
perc95A3=((1/lambdaA3)#log(100/(100-95)))##(1/alpha);

print perc5BASELINE perc10BASELINE perc20BASELINE perc25BASELINE perc30BASELINE;
print perc40BASELINE perc50BASELINE perc60BASELINE perc70BASELINE;
print perc75BASELINE perc80BASELINE perc90BASELINE perc95BASELINE;

print perc5A1 perc10A1 perc20A1 perc25A1 perc30A1;
print perc40A1 perc50A1 perc60A1 perc70A1;
print perc75A1 perc80A1 perc90A1 perc95A1;

print perc5A2 perc10A2 perc20A2 perc25A2 perc30A2;
print perc40A2 perc50A2 perc60A2 perc70A2;
print perc75A2 perc80A2 perc90A2 perc95A2;

print perc5A3 perc10A3 perc20A3 perc25A3 perc30A3;
print perc40A3 perc50A3 perc60A3 perc70A3;
print perc75A3 perc80A3 perc90A3 perc95A3;

print 'medianlifetimeBASELINE=' medianBASELINE;
print 'medianlifetime(age=A1)=' medianA1
      'medianlifetime(age=A2)=' medianA2
      'medianlifetime(age=A3)=' medianA3;
```

4. Program for fitting a Weibull regression model (shape alters, one covariate)

```
title1 'Fitting of regression model with one covariate';
title2 'Staggered entry: shape parameter alters';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector (first entry,3 agegroups);
                f11={29,59,95,73,108,15,642};
                f12={21,50,91,45,75,13,553};
                f13={16,49,68,39, 67, 7,471};

**********Vector of upper boundaries;
                x1={12,17,24,28,34,37};

**********Frequency vector (second entry,3 agegroups);
                f21={41,75,103,92,83,628};
                f22={49,62,61,66,54,753};
                f23={28,29,65,42,35,543};

**********Vector of upper boundaries;
                x2={12,17,24,28,34};

**********Frequency vector (third entry,3 agegroups);
                f31={68,34,99,57,570};
                f32={40,44,83,33,533};
                f33={46,21,60,27,571};

**********Vector of upper boundaries;
                x3={12,17,24,28};

**********Frequency vector (fourth entry,3 agegroups);
                f41={71,60,69,573};
                f42={54,61,68,616};
                f43={50,45,70,659};

**********Vector of upper boundaries;
                x4={12,17,24};

**********Relative frequency vectors;
n11=f11[+];    n21=f21[+];    n31=f31[+];    n41=f41[+];    n1=n11+n21+n31+n41;
n12=f12[+];    n22=f22[+];    n32=f32[+];    n42=f42[+];    n2=n12+n22+n32+n42;
n13=f13[+];    n23=f23[+];    n33=f33[+];    n43=f43[+];    n3=n13+n23+n33+n43;
```

```
                                                                        n=n1+n2+n3;
k1=nrow(f11); d1=k1-1;
k2=nrow(f21); d2=k2-1;
k3=nrow(f31); d3=k3-1;
k4=nrow(f41); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;

p11=f11/n11;    p21=f21/n21;    p31=f31/n31;    p41=f41/n41;    p1=p11//p21//p31//p41;
p12=f12/n12;    p22=f22/n22;    p32=f32/n32;    p42=f42/n42;    p2=p12//p22//p32//p42;
p13=f13/n13;    p23=f23/n23;    p33=f33/n33;    p43=f43/n43;    p3=p13//p23//p33//p43;
                                                                p=p1//p2//p3;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);
S=I(3)@S;

AA=J(54,1,1);       *54=3 times((7-1)+(6-1)+(5-1)+(4-1))=3 times 18=54;
BB=J(3,1,1);
CC=(I(2)//J(1,2,-1));
DD=J(18,1,1);           *d=18;
EE=J(9,1,1);

lx=log(x1)//log(x2)//log(x3)//log(x4);

xc=AA||(CC@DD)||I(3)@lx;  print xc;

C=I(3#d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;              *Weibull;
*Gm=C*diag(1/ms+1/(1-ms))*S;                               *log-logistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
    g=C*log(-log(1-ps));                                 *Weibull;
    *g=C*(log(ps)-log(1-ps));                             *Loglogistic;
    Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;          *Weibull;
    *Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                  *loglogistic;
******************covariance matrix**********************************;
m11=m[1:k1]; m21=m[k1+1:k1+k2];
m31=m[k1+k2+1:k1+k2+k3]; m41=m[k1+k2+k3+1:k1+k2+k3+k4];

m12=m[k+1:k+k1]; m22=m[k+k1+1:k+k1+k2];
m32=m[k+k1+k2+1:k+k1+k2+k3]; m42=m[k+k1+k2+k3+1:k+k1+k2+k3+k4];

m13=m[2#k+1:2#k+k1]; m23=m[2#k+k1+1:2#k+k1+k2];
m33=m[2#k+k1+k2+1:2#k+k1+k2+k3]; m43=m[2#k+k1+k2+k3+1:2#k+k1+k2+k3+k4];

sig11=(1/n11)*(diag(m11)-m11*m11`);
sig21=(1/n21)*(diag(m21)-m21*m21`);
sig31=(1/n31)*(diag(m31)-m31*m31`);
sig41=(1/n41)*(diag(m41)-m41*m41`);
sig1=block(sig11,sig21,sig31,sig41);

sig12=(1/n12)*(diag(m12)-m12*m12`);
sig22=(1/n22)*(diag(m22)-m22*m22`);
sig32=(1/n32)*(diag(m32)-m32*m32`);
sig42=(1/n42)*(diag(m42)-m42*m42`);
sig2=block(sig12,sig22,sig32,sig42);

sig13=(1/n13)*(diag(m13)-m13*m13`);
sig23=(1/n23)*(diag(m23)-m23*m23`);
sig33=(1/n33)*(diag(m33)-m33*m33`);
sig43=(1/n43)*(diag(m43)-m43*m43`);
```

```
sig3=block(sig13,sig23,sig33,sig43);

sig=block(sig1,sig2,sig3);
V=sig;
**********************************************************************;
   p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                         *Weibull;
   * p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                         *loglogistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
par=inv(xc`*xc)*xc`*log(-log(1-ms));                        *Weibull;
*par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms)); print par;          *log-logistic;

print par[format=E20.];

**********Regression coefficients;
oualphaBASELINE=par[1];
lambdaBASELINE=exp(par[1]);
betaA1=par[2];
betaA2=par[3];
betaA3=-(par[2]+par[3]);
lambdaA1=exp(par[1]+par[2]);
lambdaA2=exp(par[1]+par[3]);
lambdaA3=exp(par[1]+betaA3);
alphaA1=par[4];
alphaA2=par[5];
alphaA3=par[6];
alphaBASELINE=(n1#alphaA1+n2#alphaA2+n3#alphaA3)/n;

**********lambda's and alpha's for age levels*****************;
lambdaA1=lambdaBASELINE#exp(par[2]);
lambdaA2=lambdaBASELINE#exp(par[3]);
lambdaA3=lambdaBASELINE#exp(betaA3);
oualphaA1=log(lambdaA1);
oualphaA2=log(lambdaA2);
oualphaA3=log(lambdaA3);

print ' Weibull parameters, beta effects for age levels A1 A2 A3';
print 'lambdaBASELINE=' lambdaBASELINE[format=E20.]
      'oualphaBASELINE=' oualphaBASELINE alphaBASELINE;
print 'lambda(age=A1)=' lambdaA1[format=E20.]
      'lambda(age=A2)=' lambdaA2[format=E20.]
      'lambda(age=A3)=' lambdaA3[format=E20.];
print oualphaA1 oualphaA2 oualphaA3;
print 'alpha(age=A1)=' alphaA1[format=E20.]
      'alpha(age=A2)=' alphaA2[format=E20.]
      'alpha(age=A3)=' alphaA3[format=E20.];

print 'beta(age=A1)=' betaA1 'beta(age=A2)=' betaA2 'beta(age=A3)=' betaA3;

**********Hazard ratio and Odds ratio*****************************************;
hazBASELINE12=lambdaBASELINE*alphaBASELINE*12**(alphaBASELINE-1);
hazBASELINE24=lambdaBASELINE*alphaBASELINE*24**(alphaBASELINE-1);
survBASELINE12=exp(-lambdaBASELINE*12**alphaBASELINE);
survBASELINE24=exp(-lambdaBASELINE*24**alphaBASELINE);
oddsBASELINE12=(1-survBASELINE12)/survBASELINE12;
oddsBASELINE24=(1-survBASELINE24)/survBASELINE24;

hazAGE_A1_12=lambdaA1*alphaA1*12**(alphaA1-1);
hazAGE_A1_24=lambdaA1*alphaA1*24**(alphaA1-1);
survAGE_A1_12=exp(-lambdaA1*12**alphaA1);
survAGE_A1_24=exp(-lambdaA1*24**alphaA1);
oddsAGE_A1_12=(1-survAGE_A1_12)/survAGE_A1_12;
oddsAGE_A1_24=(1-survAGE_A1_24)/survAGE_A1_24;

hazratioAGE_A1_12=hazAGE_A1_12/hazBASELINE12;
hazratioAGE_A1_24=hazAGE_A1_24/hazBASELINE24;
oddsratioAGE_A1_12=oddsAGE_A1_12/oddsBASELINE12;
oddsratioAGE_A1_24=oddsAGE_A1_24/oddsBASELINE24;

print hazBASELINE12 hazBASELINE24 survBASELINE12 survBASELINE24
      oddsBASELINE12 oddsBASELINE24;

print hazAGE_A1_12 hazAGE_A1_24;
print survAGE_A1_12 survAGE_A1_24;
print oddsAGE_A1_12 oddsAGE_A1_24;
print hazratioAGE_A1_24 hazratioAGE_A1_24 oddsratioAGE_A1_12 oddsratioAGE_A1_24;
********************************************************************************;
hazAGE_A2_12=lambdaA2*alphaA2*12**(alphaA2-1);
hazAGE_A2_24=lambdaA2*alphaA2*24**(alphaA2-1);
survAGE_A2_12=exp(-lambdaA2*12**alphaA2);
survAGE_A2_24=exp(-lambdaA2*24**alphaA2);
oddsAGE_A2_12=(1-survAGE_A2_12)/survAGE_A2_12;
oddsAGE_A2_24=(1-survAGE_A2_24)/survAGE_A2_24;
```

```
hazratioAGE_A2_12=hazAGE_A2_12/hazBASELINE12;
hazratioAGE_A2_24=hazAGE_A2_24/hazBASELINE24;
oddsratioAGE_A2_12=oddsAGE_A2_12/oddsBASELINE12;
oddsratioAGE_A2_24=oddsAGE_A2_24/oddsBASELINE24;

print hazAGE_A2_12 hazAGE_A2_24;
print survAGE_A2_12 survAGE_A2_24;
print oddsAGE_A2_12 oddsAGE_A2_24;
print hazratioAGE_A2_12 hazratioAGE_A2_24 oddsratioAGE_A2_12 oddsratioAGE_A2_24;
*******************************************************************************;
hazAGE_A3_12=lambdaA3*alphaA3*12**(alphaA3-1);
hazAGE_A3_24=lambdaA3*alphaA3*24**(alphaA3-1);
survAGE_A3_12=exp(-lambdaA3*12**alphaA3);
survAGE_A3_24=exp(-lambdaA3*24**alphaA3);
oddsAGE_A3_12=(1-survAGE_A3_12)/survAGE_A3_12;
oddsAGE_A3_24=(1-survAGE_A3_24)/survAGE_A3_24;

hazratioAGE_A3_12=hazAGE_A3_12/hazBASELINE12;
hazratioAGE_A3_24=hazAGE_A3_24/hazBASELINE24;
oddsratioAGE_A3_12=oddsAGE_A3_12/oddsBASELINE12;
oddsratioAGE_A3_24=oddsAGE_A3_24/oddsBASELINE24;

print hazAGE_A3_12 hazAGE_A3_24;
print survAGE_A3_12 survAGE_A3_24;
print oddsAGE_A3_12 oddsAGE_A3_24;
print hazratioAGE_A3_12 hazratioAGE_A3_24 oddsratioAGE_A3_12 oddsratioAGE_A3_24;

**********Median Lifetime*****************************************************;
medianBASELINE=((1/lambdaBASELINE)#log(2))##(1/alphaBASELINE);
medianA1=((1/lambdaA1)#log(2))##(1/alphaA1);
medianA2=((1/lambdaA2)#log(2))##(1/alphaA2);
medianA3=((1/lambdaA3)#log(2))##(1/alphaA3);

 perc5BASELINE=((1/lambdaBASELINE)#log(100/(100- 5)))##(1/alphaBASELINE);
perc10BASELINE=((1/lambdaBASELINE)#log(100/(100-10)))##(1/alphaBASELINE);
perc20BASELINE=((1/lambdaBASELINE)#log(100/(100-20)))##(1/alphaBASELINE);
perc25BASELINE=((1/lambdaBASELINE)#log(100/(100-25)))##(1/alphaBASELINE);
perc30BASELINE=((1/lambdaBASELINE)#log(100/(100-30)))##(1/alphaBASELINE);
perc40BASELINE=((1/lambdaBASELINE)#log(100/(100-40)))##(1/alphaBASELINE);
perc50BASELINE=((1/lambdaBASELINE)#log(100/(100-50)))##(1/alphaBASELINE);
perc60BASELINE=((1/lambdaBASELINE)#log(100/(100-60)))##(1/alphaBASELINE);
perc70BASELINE=((1/lambdaBASELINE)#log(100/(100-70)))##(1/alphaBASELINE);
perc75BASELINE=((1/lambdaBASELINE)#log(100/(100-75)))##(1/alphaBASELINE);
perc80BASELINE=((1/lambdaBASELINE)#log(100/(100-80)))##(1/alphaBASELINE);
perc90BASELINE=((1/lambdaBASELINE)#log(100/(100-90)))##(1/alphaBASELINE);
perc95BASELINE=((1/lambdaBASELINE)#log(100/(100-95)))##(1/alphaBASELINE);

 perc5A1=((1/lambdaA1)#log(100/(100- 5)))##(1/alphaA1);
perc10A1=((1/lambdaA1)#log(100/(100-10)))##(1/alphaA1);
perc20A1=((1/lambdaA1)#log(100/(100-20)))##(1/alphaA1);
perc25A1=((1/lambdaA1)#log(100/(100-25)))##(1/alphaA1);
perc30A1=((1/lambdaA1)#log(100/(100-30)))##(1/alphaA1);
perc40A1=((1/lambdaA1)#log(100/(100-40)))##(1/alphaA1);
perc50A1=((1/lambdaA1)#log(100/(100-50)))##(1/alphaA1);
perc60A1=((1/lambdaA1)#log(100/(100-60)))##(1/alphaA1);
perc70A1=((1/lambdaA1)#log(100/(100-70)))##(1/alphaA1);
perc75A1=((1/lambdaA1)#log(100/(100-75)))##(1/alphaA1);
perc80A1=((1/lambdaA1)#log(100/(100-80)))##(1/alphaA1);
perc90A1=((1/lambdaA1)#log(100/(100-90)))##(1/alphaA1);
perc95A1=((1/lambdaA1)#log(100/(100-95)))##(1/alphaA1);

 perc5A2=((1/lambdaA2)#log(100/(100- 5)))##(1/alphaA2);
perc10A2=((1/lambdaA2)#log(100/(100-10)))##(1/alphaA2);
perc20A2=((1/lambdaA2)#log(100/(100-20)))##(1/alphaA2);
perc25A2=((1/lambdaA2)#log(100/(100-25)))##(1/alphaA2);
perc30A2=((1/lambdaA2)#log(100/(100-30)))##(1/alphaA2);
perc40A2=((1/lambdaA2)#log(100/(100-40)))##(1/alphaA2);
perc50A2=((1/lambdaA2)#log(100/(100-50)))##(1/alphaA2);
perc60A2=((1/lambdaA2)#log(100/(100-60)))##(1/alphaA2);
perc70A2=((1/lambdaA2)#log(100/(100-70)))##(1/alphaA2);
perc75A2=((1/lambdaA2)#log(100/(100-75)))##(1/alphaA2);
perc80A2=((1/lambdaA2)#log(100/(100-80)))##(1/alphaA2);
perc90A2=((1/lambdaA2)#log(100/(100-90)))##(1/alphaA2);
perc95A2=((1/lambdaA2)#log(100/(100-95)))##(1/alphaA2);

 perc5A3=((1/lambdaA3)#log(100/(100- 5)))##(1/alphaA3);
perc10A3=((1/lambdaA3)#log(100/(100-10)))##(1/alphaA3);
perc20A3=((1/lambdaA3)#log(100/(100-20)))##(1/alphaA3);
perc25A3=((1/lambdaA3)#log(100/(100-25)))##(1/alphaA3);
perc30A3=((1/lambdaA3)#log(100/(100-30)))##(1/alphaA3);
perc40A3=((1/lambdaA3)#log(100/(100-40)))##(1/alphaA3);
perc50A3=((1/lambdaA3)#log(100/(100-50)))##(1/alphaA3);
perc60A3=((1/lambdaA3)#log(100/(100-60)))##(1/alphaA3);
perc70A3=((1/lambdaA3)#log(100/(100-70)))##(1/alphaA3);
perc75A3=((1/lambdaA3)#log(100/(100-75)))##(1/alphaA3);
perc80A3=((1/lambdaA3)#log(100/(100-80)))##(1/alphaA3);
perc90A3=((1/lambdaA3)#log(100/(100-90)))##(1/alphaA3);
perc95A3=((1/lambdaA3)#log(100/(100-95)))##(1/alphaA3);
```

```
print perc5BASELINE perc10BASELINE perc20BASELINE perc25BASELINE perc30BASELINE;
print perc40BASELINE perc50BASELINE perc60BASELINE perc70BASELINE;
print perc75BASELINE perc80BASELINE perc90BASELINE perc95BASELINE;

print perc5A1 perc10A1 perc20A1 perc25A1 perc30A1;
print perc40A1 perc50A1 perc60A1 perc70A1;
print perc75A1 perc80A1 perc90A1 perc95A1;

print perc5A2 perc10A2 perc20A2 perc25A2 perc30A2;
print perc40A2 perc50A2 perc60A2 perc70A2;
print perc75A2 perc80A2 perc90A2 perc95A2;

print perc5A3 perc10A3 perc20A3 perc25A3 perc30A3;
print perc40A3 perc50A3 perc60A3 perc70A3;
print perc75A3 perc80A3 perc90A3 perc95A3;

print 'medianlifetimeBASELINE=' medianBASELINE;
print 'medianlifetime(age=A1)=' medianA1
      'medianlifetime(age=A2)=' medianA2
      'medianlifetime(age=A3)=' medianA3;
```

5. Program for fitting a log-logistic/Weibull regression model with a continuous predictor

```
title1 'Fitting of regression model with one ordinal predictor';
title2 'Staggered entry: constant shape parameter';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector (first entry,3 agegroups);
                f11={29,59,95,73,108,15,642};
                f12={21,50,91,45,75,13,553};
                f13={16,49,68,39, 67, 7,471};

**********Vector of upper boundaries;
                x1={12,17,24,28,34,37};

**********Frequency vector (second entry,3 agegroups);
                f21={41,75,103,92,83,628};
                f22={49,62,61,66,54,753};
                f23={28,29,65,42,35,543};

**********Vector of upper boundaries;
                x2={12,17,24,28,34};

**********Frequency vector (third entry,3 agegroups);
                f31={68,34,99,57,570};
                f32={40,44,83,33,533};
                f33={46,21,60,27,571};

**********Vector of upper boundaries;
                x3={12,17,24,28};

**********Frequency vector (fourth entry,3 agegroups);
                f41={71,60,69,573};
                f42={54,61,68,616};
                f43={50,45,70,659};

**********Vector of upper boundaries;
                x4={12,17,24};

**********Relative frequency vectors;
n11=f11[+];     n21=f21[+];     n31=f31[+];     n41=f41[+];     n1=n11+n21+n31+n41;
n12=f12[+];     n22=f22[+];     n32=f32[+];     n42=f42[+];     n2=n12+n22+n32+n42;
n13=f13[+];     n23=f23[+];     n33=f33[+];     n43=f43[+];     n3=n13+n23+n33+n43;
                                                                n=n1+n2+n3;
k1=nrow(f11); d1=k1-1;
k2=nrow(f21); d2=k2-1;
k3=nrow(f31); d3=k3-1;
k4=nrow(f41); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;

p11=f11/n11;     p21=f21/n21;     p31=f31/n31;     p41=f41/n41;     p1=p11//p21//p31//p41;
p12=f12/n12;     p22=f22/n22;     p32=f32/n32;     p42=f42/n42;     p2=p12//p22//p32//p42;
p13=f13/n13;     p23=f23/n23;     p33=f33/n33;     p43=f43/n43;     p3=p13//p23//p33//p43;
                                                                    p=p1//p2//p3;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
```

```
S=block(S1,S2,S3,S4);
S=I(3)@S;

AA=J(54,1,1);        *54=3 times((7-1)+(6-1)+(5-1)+(4-1))=3 times 18=54;
BB=J(3,1,1);
DD=J(18,1,1);        *d=18;

lx=log(x1)//log(x2)//log(x3)//log(x4);

z={1,2,3};
*z={26,39.5,52};     *midpoints of age intervals;

xc=(AA||BB@lx||(z@DD));  print xc;

C=I(3#d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
*Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;                *Weibull;
Gm=C*diag(1/ms+1/(1-ms))*S;                                 *log-logistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
    *g=C*log(-log(1-ps));                                   *Weibull;
    g=C*(log(ps)-log(1-ps));                                *log-logistic;
    *Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;            *Weibull;
    Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                     *log-logistic;
******************covariance matrix***********************************;
m11=m[1:k1]; m21=m[k1+1:k1+k2];
m31=m[k1+k2+1:k1+k2+k3]; m41=m[k1+k2+k3+1:k1+k2+k3+k4];

m12=m[k+1:k+k1]; m22=m[k+k1+1:k+k1+k2];
m32=m[k+k1+k2+1:k+k1+k2+k3]; m42=m[k+k1+k2+k3+1:k+k1+k2+k3+k4];

m13=m[2#k+1:2#k+k1]; m23=m[2#k+k1+1:2#k+k1+k2];
m33=m[2#k+k1+k2+1:2#k+k1+k2+k3]; m43=m[2#k+k1+k2+k3+1:2#k+k1+k2+k3+k4];

sig11=(1/n11)*(diag(m11)-m11*m11`);
sig21=(1/n21)*(diag(m21)-m21*m21`);
sig31=(1/n31)*(diag(m31)-m31*m31`);
sig41=(1/n41)*(diag(m41)-m41*m41`);
sig1=block(sig11,sig21,sig31,sig41);

sig12=(1/n12)*(diag(m12)-m12*m12`);
sig22=(1/n22)*(diag(m22)-m22*m22`);
sig32=(1/n32)*(diag(m32)-m32*m32`);
sig42=(1/n42)*(diag(m42)-m42*m42`);
sig2=block(sig12,sig22,sig32,sig42);

sig13=(1/n13)*(diag(m13)-m13*m13`);
sig23=(1/n23)*(diag(m23)-m23*m23`);
sig33=(1/n33)*(diag(m33)-m33*m33`);
sig43=(1/n43)*(diag(m43)-m43*m43`);
sig3=block(sig13,sig23,sig33,sig43);

sig=block(sig1,sig2,sig3);
V=sig;
**********************************************************************;
    *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                          *Weibull;
    p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                           *log-logistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
*par=inv(xc`*xc)*xc`*log(-log(1-ms));                       *Weibull;
par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms)); print par;         *log-logistic;

print par[format=E20.];
```

```
**********Regression coefficients;
oualphaBASELINE=par[1];
lambdaBASELINE=exp(par[1]);

alpha=par[2];

beta=par[3]; *AGE ordinal;

print ' Loglogistic parameters and beta effect for age ';
*print ' Weibull parameters and beta effect for age ';

print 'lambdaBASELINE=' lambdaBASELINE[format=E20.]
'oualphaBASELINE=' oualphaBASELINE alpha;
print 'alpha=' alpha[format=E20.];
print 'beta=' beta[format=E20.];

**********lambda for each agegroup (z=1,2,3)*********************;
lambdaAGE1=exp(par[1]+(par[3]*1));
lambdaAGE2=exp(par[1]+(par[3]*2));
lambdaAGE3=exp(par[1]+(par[3]*3));

oualphaAGE1=log(lambdaAGE1);
oualphaAGE2=log(lambdaAGE2);
oualphaAGE3=log(lambdaAGE3);

print 'lambda(age=1)=' lambdaAGE1[format=E20.]
'lambda(age=2)=' lambdaAGE2[format=E20.]
'lambda(age=3)=' lambdaAGE3[format=E20.];
print oualphaAGE1 oualphaAGE2 oualphaAGE3;
```

6. Program for fitting a log-logistic regression model with two predictors

```
title1 'Fitting of regression model with two covariates';
title2 'Staggered entry: constant shape parameter';

proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector (first entry,3 agegroups,3 scoregroups);
                f111={12,34,51,39,57,11, 59};   f111=f111<>1e-04;
                f112={10,12,22,19,32, 4,418};   f112=f112<>1e-04;
                f113={ 7,13,22,15,19, 0,165};   f113=f113<>1e-04;
                f121={13,14,45,27,33, 4, 66};   f121=f121<>1e-04;
                f122={ 4,22,22, 8,25, 4,297};   f122=f122<>1e-04;
                f123={ 4,14,24,10,17, 5,190};   f123=f123<>1e-04;
                f131={10,25,29,17,46, 2,116};   f131=f131<>1e-04;
                f132={ 6,13,28,16,16, 5,273};   f132=f132<>1e-04;
                f133={ 0,11,11, 6, 5, 0, 82};   f133=f133<>1e-04;

**********Vector of upper boundaries;
                x1={12,17,24,28,34,37};

**********Frequency vector (second entry,3 agegroups,3 scoregroups);
                f211={22,25,58,53,40, 45};      f211=f211<>1e-04;
                f212={10,26,32,20,29,379};      f212=f212<>1e-04;
                f213={ 9,24,13,19,14,204};      f213=f213<>1e-04;
                f221={24,24,28,30,25,106};      f221=f221<>1e-04;
                f222={12,20,14,17,16,409};      f222=f222<>1e-04;
                f223={13,18,19,19,13,238};      f223=f223<>1e-04;
                f231={13,15,32,19,17,107};      f231=f231<>1e-04;
                f232={11,13,22,17,12,319};      f232=f232<>1e-04;
                f233={ 4, 1,11, 6, 6,117};      f233=f233<>1e-04;

**********Vector of upper boundaries;
                x2={12,17,24,28,34};

**********Frequency vector (third entry,3 agegroups,3 scoregroups);
                f311={34,16,50,23, 54};         f311=f311<>1e-04;
                f312={19, 2,32,24,317};         f312=f312<>1e-04;
                f313={15,16,17,10,199};         f313=f313<>1e-04;
                f321={19,18,38,16, 75};         f321=f321<>1e-04;
                f322={16,14,25,10,263};         f322=f322<>1e-04;
                f323={ 5,12,20, 7,195};         f323=f323<>1e-04;
                f331={28,16,22,12, 98};         f331=f331<>1e-04;
                f332={13, 0,24, 4,323};         f332=f332<>1e-04;
                f333={ 5, 5,14,11,150};         f333=f333<>1e-04;

**********Vector of upper boundaries;
                x3={12,17,24,28};

**********Frequency vector (fourth entry,3 agegroups,3 scoregroups);
                f411={40,30,30, 50};            f411=f411<>1e-04;
                f412={ 9,14,27,301};            f412=f412<>1e-04;
                f413={22,16,12,222};            f413=f413<>1e-04;
                f421={24,30,29, 81};            f421=f421<>1e-04;
                f422={14,15,12,307};            f422=f422<>1e-04;
                f423={16,16,27,228};            f423=f423<>1e-04;
                f431={20,22,28,119};            f431=f431<>1e-04;
                f432={19,12,26,369};            f432=f432<>1e-04;
                f433={11,11,16,171};            f433=f433<>1e-04;

**********Vector of upper boundaries;
                x4={12,17,24};

**********Relative frequency vectors;
n111=f111[+]; n211=f211[+]; n311=f311[+]; n411=f411[+]; n11=n111+n211+n311+n411;
n112=f112[+]; n212=f212[+]; n312=f312[+]; n412=f412[+]; n12=n112+n212+n312+n412;
n113=f113[+]; n213=f213[+]; n313=f313[+]; n413=f413[+]; n13=n113+n213+n313+n413;
n121=f121[+]; n221=f221[+]; n321=f321[+]; n421=f421[+]; n21=n121+n221+n321+n421;
n122=f122[+]; n222=f222[+]; n322=f322[+]; n422=f422[+]; n22=n122+n222+n322+n422;
n123=f123[+]; n223=f223[+]; n323=f323[+]; n423=f423[+]; n23=n123+n223+n323+n423;
n131=f131[+]; n231=f231[+]; n331=f331[+]; n431=f431[+]; n31=n131+n231+n331+n431;
n132=f132[+]; n232=f232[+]; n332=f332[+]; n432=f432[+]; n32=n132+n232+n332+n432;
n133=f133[+]; n233=f233[+]; n333=f333[+]; n433=f433[+]; n33=n133+n233+n333+n433;

                                                        n1=n11+n12+n13;
                                                        n2=n21+n22+n23;
                                                        n3=n31+n32+n33;
                                                        n=n1+n2+n3;

k1=nrow(f111); d1=k1-1;
k2=nrow(f211); d2=k2-1;
k3=nrow(f311); d3=k3-1;
k4=nrow(f411); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;

p111=f111/n111; p211=f211/n211; p311=f311/n311; p411=f411/n411;
p112=f112/n112; p212=f212/n212; p312=f312/n312; p412=f412/n412;
p113=f113/n113; p213=f213/n213; p313=f313/n313; p413=f413/n413;
p121=f121/n121; p221=f221/n221; p321=f321/n321; p421=f421/n421;
p122=f122/n122; p222=f222/n222; p322=f322/n322; p422=f422/n422;
p123=f123/n123; p223=f223/n223; p323=f323/n323; p423=f423/n423;
p131=f131/n131; p231=f231/n231; p331=f331/n331; p431=f431/n431;
p132=f132/n132; p232=f232/n232; p332=f332/n332; p432=f432/n432;
p133=f133/n133; p233=f233/n233; p333=f333/n333; p433=f433/n433;

p11=p111//p211//p311//p411;
p12=p112//p212//p312//p412;
p13=p113//p213//p313//p413;
p1=p11//p12//p13;

p21=p121//p221//p321//p421;
p22=p122//p222//p322//p422;
p23=p123//p223//p323//p423;
p2=p21//p22//p23;

p31=p131//p231//p331//p431;
p32=p132//p232//p332//p432;
p33=p133//p233//p333//p433;
p3=p31//p32//p33;

p=p1//p2//p3;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);
S=I(3)@I(3)@S;

AA=J(162,1,1);          *162=9 times((7-1)+(6-1)+(5-1)+(4-1))=9 times 18=162;
BB=J(3,1,1);
CC=(I(2)//J(1,2,-1));
DD=J(18,1,1);           *3 times d1=3(6)=18;
EE=J(15,1,1);           *3 times d2=3(5)=15;
FF=J(12,1,1);           *3 times d3=3(4)=12;
GG=J(9,1,1);            *3 times d4=3(3)=9;

KK=J(9,1,1);            *9=3 times 3;
LL=J(54,1,1);           *54=18 times 3;

lx=BB@BB@(log(x1)//log(x2)//log(x3)//log(x4));

HH=BB@(CC@DD);

xc=AA||CC@LL||HH||lx;  print xc;

C=I(9#d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;
ms=S*m; ps=ms;
```

```
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
*Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;                *Weibull;
Gm=C*diag(1/ms+1/(1-ms))*S;                                 *log-logistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
    *g=C*log(-log(1-ps));                                   *Weibull;
    g=C*(log(ps)-log(1-ps));                                *log-logistic;
    *Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;            *Weibull;
    Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                     *log-logistic;

******************covariance matrix**********************************;
m111=m[1:k1];
m211=m[k1+1:k1+k2];
m311=m[k1+k2+1:k1+k2+k3];
m411=m[k1+k2+k3+1:k1+k2+k3+k4];

m112=m[k+1:k+k1];
m212=m[k+k1+1:k+k1+k2];
m312=m[k+k1+k2+1:k+k1+k2+k3];
m412=m[k+k1+k2+k3+1:k+k1+k2+k3+k4];

m113=m[2#k+1:2#k+k1];
m213=m[2#k+k1+1:2#k+k1+k2];
m313=m[2#k+k1+k2+1:2#k+k1+k2+k3];
m413=m[2#k+k1+k2+k3+1:2#k+k1+k2+k3+k4];

m121=m[3#k+1:3#k+k1];
m221=m[3#k+k1+1:3#k+k1+k2];
m321=m[3#k+k1+k2+1:3#k+k1+k2+k3];
m421=m[3#k+k1+k2+k3+1:3#k+k1+k2+k3+k4];

m122=m[4#k+1:4#k+k1];
m222=m[4#k+k1+1:4#k+k1+k2];
m322=m[4#k+k1+k2+1:4#k+k1+k2+k3];
m422=m[4#k+k1+k2+k3+1:4#k+k1+k2+k3+k4];

m123=m[5#k+1:5#k+k1];
m223=m[5#k+k1+1:5#k+k1+k2];
m323=m[5#k+k1+k2+1:5#k+k1+k2+k3];
m423=m[5#k+k1+k2+k3+1:5#k+k1+k2+k3+k4];

m131=m[6#k+1:6#k+k1];
m231=m[6#k+k1+1:6#k+k1+k2];
m331=m[6#k+k1+k2+1:6#k+k1+k2+k3];
m431=m[6#k+k1+k2+k3+1:6#k+k1+k2+k3+k4];

m132=m[7#k+1:7#k+k1];
m232=m[7#k+k1+1:7#k+k1+k2];
m332=m[7#k+k1+k2+1:7#k+k1+k2+k3];
m432=m[7#k+k1+k2+k3+1:7#k+k1+k2+k3+k4];

m133=m[8#k+1:8#k+k1];
m233=m[8#k+k1+1:8#k+k1+k2];
m333=m[8#k+k1+k2+1:8#k+k1+k2+k3];
m433=m[8#k+k1+k2+k3+1:9#k];

sig111=(1/n111)*(diag(m111)-m111*m111`);
sig112=(1/n112)*(diag(m112)-m112*m112`);
sig113=(1/n113)*(diag(m113)-m113*m113`);
sig211=(1/n211)*(diag(m211)-m211*m211`);
sig212=(1/n212)*(diag(m212)-m212*m212`);
sig213=(1/n213)*(diag(m213)-m213*m213`);
sig311=(1/n311)*(diag(m311)-m311*m311`);
sig312=(1/n312)*(diag(m312)-m312*m312`);
sig313=(1/n313)*(diag(m313)-m313*m313`);
sig411=(1/n411)*(diag(m411)-m411*m411`);
sig412=(1/n412)*(diag(m412)-m412*m412`);
sig413=(1/n413)*(diag(m413)-m413*m413`);

sig11=block(sig111,sig211,sig311,sig411);
sig12=block(sig112,sig212,sig312,sig412);
sig13=block(sig113,sig213,sig313,sig413);


sig121=(1/n121)*(diag(m121)-m121*m121`);
```

```
sig122=(1/n122)*(diag(m122)-m122*m122`);
sig123=(1/n123)*(diag(m123)-m123*m123`);
sig221=(1/n221)*(diag(m221)-m221*m221`);
sig222=(1/n222)*(diag(m222)-m222*m222`);
sig223=(1/n223)*(diag(m223)-m223*m223`);
sig321=(1/n321)*(diag(m321)-m321*m321`);
sig322=(1/n322)*(diag(m322)-m322*m322`);
sig323=(1/n323)*(diag(m323)-m323*m323`);
sig421=(1/n421)*(diag(m421)-m421*m421`);
sig422=(1/n422)*(diag(m422)-m422*m422`);
sig423=(1/n423)*(diag(m423)-m423*m423`);

sig21=block(sig121,sig221,sig321,sig421);
sig22=block(sig122,sig222,sig322,sig422);
sig23=block(sig123,sig223,sig323,sig423);


sig131=(1/n131)*(diag(m131)-m131*m131`);
sig132=(1/n132)*(diag(m132)-m132*m132`);
sig133=(1/n133)*(diag(m133)-m133*m133`);
sig231=(1/n231)*(diag(m231)-m231*m231`);
sig232=(1/n232)*(diag(m232)-m232*m232`);
sig233=(1/n233)*(diag(m233)-m233*m233`);
sig331=(1/n331)*(diag(m331)-m331*m331`);
sig332=(1/n332)*(diag(m332)-m332*m332`);
sig333=(1/n333)*(diag(m333)-m333*m333`);
sig431=(1/n431)*(diag(m431)-m431*m431`);
sig432=(1/n432)*(diag(m432)-m432*m432`);
sig433=(1/n433)*(diag(m433)-m433*m433`);

sig31=block(sig131,sig231,sig331,sig431);
sig32=block(sig132,sig232,sig332,sig432);
sig33=block(sig133,sig233,sig333,sig433);

sig=block(sig11,sig12,sig13,sig21,sig22,sig23,sig31,sig32,sig33);
V=sig;
*******************************************************************;
    *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                        *Weibull;
    p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                         *log-logistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
*par=inv(xc`*xc)*xc`*log(-log(1-ms));                      *Weibull;
par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms));                   *log-logistic;

print par[format=E20.];

**********Regression coefficients;
oualphaBASELINE=par[1];
betaA1=par[2];
betaA2=par[3];
betaA3=-(par[2]+par[3]);
betaS1=par[4];
betaS2=par[5];
betaS3=-(par[4]+par[5]);
lambdaBASELINE=exp(par[1]);
alpha=par[6];

**********Indices, lambda's and constant alpha for age levels*****************;
indexA1=exp(betaA1);          *constant shape parameter;
indexA2=exp(betaA2);
indexA3=exp(betaA3);
indexS1=exp(betaS1);
indexS2=exp(betaS2);
indexS3=exp(betaS3);

lambdaA1S1=lambdaBASELINE#indexA1#indexS1;
lambdaA1S2=lambdaBASELINE#indexA1#indexS2;
lambdaA1S3=lambdaBASELINE#indexA1#indexS3;
lambdaA2S1=lambdaBASELINE#indexA2#indexS1;
lambdaA2S2=lambdaBASELINE#indexA2#indexS2;
lambdaA2S3=lambdaBASELINE#indexA2#indexS3;
lambdaA3S1=lambdaBASELINE#indexA3#indexS1;
lambdaA3S2=lambdaBASELINE#indexA3#indexS2;
lambdaA3S3=lambdaBASELINE#indexA3#indexS3;

oualphaA1S1=oualphaBASELINE+betaA1+betaS1; *same as oualphaA1S1=log(lambdaA1S1);
oualphaA1S2=oualphaBASELINE+betaA1+betaS2;
oualphaA1S3=oualphaBASELINE+betaA1+betaS3;
oualphaA2S1=oualphaBASELINE+betaA2+betaS1;
oualphaA2S2=oualphaBASELINE+betaA2+betaS2;
oualphaA2S3=oualphaBASELINE+betaA2+betaS3;
oualphaA3S1=oualphaBASELINE+betaA3+betaS1;
oualphaA3S2=oualphaBASELINE+betaA3+betaS2;
```

```
oualphaA3S3=oualphaBASELINE+betaA3+betaS3;

print ' Loglogistic parameters, beta effects and indices: MLE subject to constraints';

print 'lambdaBASELINE=' lambdaBASELINE[format=E20.]  'oualphaBASELINE=' oualphaBASELINE;
print 'alpha=' alpha[format=E20.];

print 'lambda(age=A1,score=S1)=' lambdaA1S1[format=E20.]
      'lambda(age=A1,score=S2)=' lambdaA1S2[format=E20.]
      'lambda(age=A1,score=S3)=' lambdaA1S3[format=E20.];
print oualphaA1S1 oualphaA1S2 oualphaA1S3;

print 'lambda(age=A2,score=S1)=' lambdaA2S1[format=E20.]
      'lambda(age=A2,score=S2)=' lambdaA2S2[format=E20.]
      'lambda(age=A2,score=S3)=' lambdaA2S3[format=E20.];
print oualphaA2S1 oualphaA2S2 oualphaA2S3;

print 'lambda(age=A3,score=S1)=' lambdaA3S1[format=E20.]
      'lambda(age=A3,score=S2)=' lambdaA3S2[format=E20.]
      'lambda(age=A3,score=S3)=' lambdaA3S3[format=E20.];
print oualphaA3S1 oualphaA3S2 oualphaA3S3;

print 'beta(age=A1)=' betaA1 'beta(age=A2)=' betaA2 'beta(age=A3)=' betaA3;
print 'beta(score=S1)=' betaS1 'beta(score=S2)=' betaS2 'beta(score=S3)=' betaS3;
print 'index(age=A1)=' indexA1 'index(age=A2)=' indexA2 'index(age=A3)=' indexA3;
print 'index(score=S1)=' indexS1 'index(score=S2)=' indexS2 'index(score=S3)=' indexS3;

**********Hazard ratio and Odds ratio*****************************************;
hazBASELINE12=(lambdaBASELINE*alpha*12**(alpha-1))/(1+lambdaBASELINE*12**alpha);
hazBASELINE24=(lambdaBASELINE*alpha*24**(alpha-1))/(1+lambdaBASELINE*24**alpha);
survBASELINE12=(1+lambdaBASELINE*12**alpha)**(-1);
survBASELINE24=(1+lambdaBASELINE*24**alpha)**(-1);
oddsBASELINE12=(1-survBASELINE12)/survBASELINE12;
oddsBASELINE24=(1-survBASELINE24)/survBASELINE24;

hazA1S1_12=(lambdaA1S1*alpha*12**(alpha-1))/(1+lambdaA1S1*12**alpha);
hazA1S1_24=(lambdaA1S1*alpha*24**(alpha-1))/(1+lambdaA1S1*24**alpha);
hazA1S2_12=(lambdaA1S2*alpha*12**(alpha-1))/(1+lambdaA1S2*12**alpha);
hazA1S2_24=(lambdaA1S2*alpha*24**(alpha-1))/(1+lambdaA1S2*24**alpha);
hazA1S3_12=(lambdaA1S3*alpha*12**(alpha-1))/(1+lambdaA1S3*12**alpha);
hazA1S3_24=(lambdaA1S3*alpha*24**(alpha-1))/(1+lambdaA1S3*24**alpha);
hazA2S1_12=(lambdaA2S1*alpha*12**(alpha-1))/(1+lambdaA2S1*12**alpha);
hazA2S1_24=(lambdaA2S1*alpha*24**(alpha-1))/(1+lambdaA2S1*24**alpha);
hazA2S2_12=(lambdaA2S2*alpha*12**(alpha-1))/(1+lambdaA2S2*12**alpha);
hazA2S2_24=(lambdaA2S2*alpha*24**(alpha-1))/(1+lambdaA2S2*24**alpha);
hazA2S3_12=(lambdaA2S3*alpha*12**(alpha-1))/(1+lambdaA2S3*12**alpha);
hazA2S3_24=(lambdaA2S3*alpha*24**(alpha-1))/(1+lambdaA2S3*24**alpha);
hazA3S1_12=(lambdaA3S1*alpha*12**(alpha-1))/(1+lambdaA3S1*12**alpha);
hazA3S1_24=(lambdaA3S1*alpha*24**(alpha-1))/(1+lambdaA3S1*24**alpha);
hazA3S2_12=(lambdaA3S2*alpha*12**(alpha-1))/(1+lambdaA3S2*12**alpha);
hazA3S2_24=(lambdaA3S2*alpha*24**(alpha-1))/(1+lambdaA3S2*24**alpha);
hazA3S3_12=(lambdaA3S3*alpha*12**(alpha-1))/(1+lambdaA3S3*12**alpha);
hazA3S3_24=(lambdaA3S3*alpha*24**(alpha-1))/(1+lambdaA3S3*24**alpha);

survA1S1_12=(1+lambdaA1S1*12**alpha)**(-1);
survA1S1_24=(1+lambdaA1S1*24**alpha)**(-1);
survA1S2_12=(1+lambdaA1S2*12**alpha)**(-1);
survA1S2_24=(1+lambdaA1S2*24**alpha)**(-1);
survA1S3_12=(1+lambdaA1S3*12**alpha)**(-1);
survA1S3_24=(1+lambdaA1S3*24**alpha)**(-1);
survA2S1_12=(1+lambdaA2S1*12**alpha)**(-1);
survA2S1_24=(1+lambdaA2S1*24**alpha)**(-1);
survA2S2_12=(1+lambdaA2S2*12**alpha)**(-1);
survA2S2_24=(1+lambdaA2S2*24**alpha)**(-1);
survA2S3_12=(1+lambdaA2S3*12**alpha)**(-1);
survA2S3_24=(1+lambdaA2S3*24**alpha)**(-1);
survA3S1_12=(1+lambdaA3S1*12**alpha)**(-1);
survA3S1_24=(1+lambdaA3S1*24**alpha)**(-1);
survA3S2_12=(1+lambdaA3S2*12**alpha)**(-1);
survA3S2_24=(1+lambdaA3S2*24**alpha)**(-1);
survA3S3_12=(1+lambdaA3S3*12**alpha)**(-1);
survA3S3_24=(1+lambdaA3S3*24**alpha)**(-1);

oddsA1S1_12=(1-survA1S1_12)/survA1S1_12;
oddsA1S1_24=(1-survA1S1_24)/survA1S1_24;
oddsA1S2_12=(1-survA1S2_12)/survA1S2_12;
oddsA1S2_24=(1-survA1S2_24)/survA1S2_24;
oddsA1S3_12=(1-survA1S3_12)/survA1S3_12;
oddsA1S3_24=(1-survA1S3_24)/survA1S3_24;
oddsA2S1_12=(1-survA2S1_12)/survA2S1_12;
oddsA2S1_24=(1-survA2S1_24)/survA2S1_24;
oddsA2S2_12=(1-survA2S2_12)/survA2S2_12;
oddsA2S2_24=(1-survA2S2_24)/survA2S2_24;
oddsA2S3_12=(1-survA2S3_12)/survA2S3_12;
oddsA2S3_24=(1-survA2S3_24)/survA2S3_24;
oddsA3S1_12=(1-survA3S1_12)/survA3S1_12;
oddsA3S1_24=(1-survA3S1_24)/survA3S1_24;
oddsA3S2_12=(1-survA3S2_12)/survA3S2_12;
oddsA3S2_24=(1-survA3S2_24)/survA3S2_24;
```

```
oddsA3S3_12=(1-survA3S3_12)/survA3S3_12;
oddsA3S3_24=(1-survA3S3_24)/survA3S3_24;

hazratioA1S1_12=hazA1S1_12/hazBASELINE12;
hazratioA1S1_24=hazA1S1_24/hazBASELINE24;
hazratioA1S2_12=hazA1S2_12/hazBASELINE12;
hazratioA1S2_24=hazA1S2_24/hazBASELINE24;
hazratioA1S3_12=hazA1S3_12/hazBASELINE12;
hazratioA1S3_24=hazA1S3_24/hazBASELINE24;
hazratioA2S1_12=hazA2S1_12/hazBASELINE12;
hazratioA2S1_24=hazA2S1_24/hazBASELINE24;
hazratioA2S2_12=hazA2S2_12/hazBASELINE12;
hazratioA2S2_24=hazA2S2_24/hazBASELINE24;
hazratioA2S3_12=hazA2S3_12/hazBASELINE12;
hazratioA2S3_24=hazA2S3_24/hazBASELINE24;
hazratioA3S1_12=hazA3S1_12/hazBASELINE12;
hazratioA3S1_24=hazA3S1_24/hazBASELINE24;
hazratioA3S2_12=hazA3S2_12/hazBASELINE12;
hazratioA3S2_24=hazA3S2_24/hazBASELINE24;
hazratioA3S3_12=hazA3S3_12/hazBASELINE12;
hazratioA3S3_24=hazA3S3_24/hazBASELINE24;

oddsratioA1S1_12=oddsA1S1_12/oddsBASELINE12;
oddsratioA1S1_24=oddsA1S1_24/oddsBASELINE24;
oddsratioA1S2_12=oddsA1S2_12/oddsBASELINE12;
oddsratioA1S2_24=oddsA1S2_24/oddsBASELINE24;
oddsratioA1S3_12=oddsA1S3_12/oddsBASELINE12;
oddsratioA1S3_24=oddsA1S3_24/oddsBASELINE24;
oddsratioA2S1_12=oddsA2S1_12/oddsBASELINE12;
oddsratioA2S1_24=oddsA2S1_24/oddsBASELINE24;
oddsratioA2S2_12=oddsA2S2_12/oddsBASELINE12;
oddsratioA2S2_24=oddsA2S2_24/oddsBASELINE24;
oddsratioA2S3_12=oddsA2S3_12/oddsBASELINE12;
oddsratioA2S3_24=oddsA2S3_24/oddsBASELINE24;
oddsratioA3S1_12=oddsA3S1_12/oddsBASELINE12;
oddsratioA3S1_24=oddsA3S1_24/oddsBASELINE24;
oddsratioA3S2_12=oddsA3S2_12/oddsBASELINE12;
oddsratioA3S2_24=oddsA3S2_24/oddsBASELINE24;
oddsratioA3S3_12=oddsA3S3_12/oddsBASELINE12;
oddsratioA3S3_24=oddsA3S3_24/oddsBASELINE24;

print hazBASELINE12 hazBASELINE24 survBASELINE12 survBASELINE24 oddsBASELINE12 oddsBASELINE24;

print hazA1S1_12 hazA1S1_24 hazA1S2_12 hazA1S2_24 hazA1S3_12 hazA1S3_24;
print hazA2S1_12 hazA2S1_24 hazA2S2_12 hazA2S2_24 hazA2S3_12 hazA2S3_24;
print hazA3S1_12 hazA3S1_24 hazA3S2_12 hazA3S2_24 hazA3S3_12 hazA3S3_24;

print survA1S1_12 survA1S1_24 survA1S2_12 survA1S2_24 survA1S3_12 survA1S3_24;
print survA2S1_12 survA2S1_24 survA2S2_12 survA2S2_24 survA2S3_12 survA2S3_24;
print survA3S1_12 survA3S1_24 survA3S2_12 survA3S2_24 survA3S3_12 survA3S3_24;

print oddsA1S1_12 oddsA1S1_24 oddsA1S2_12 oddsA1S2_24 oddsA1S3_12 oddsA1S3_24;
print oddsA2S1_12 oddsA2S1_24 oddsA2S2_12 oddsA2S2_24 oddsA2S3_12 oddsA2S3_24;
print oddsA3S1_12 oddsA3S1_24 oddsA3S2_12 oddsA3S2_24 oddsA3S3_12 oddsA3S3_24;

print hazratioA1S1_12 hazratioA1S1_24 hazratioA1S2_12 hazratioA1S2_24 hazratioA1S3_12 hazratioA1S3_24;
print hazratioA2S1_12 hazratioA2S1_24 hazratioA2S2_12 hazratioA2S2_24 hazratioA2S3_12 hazratioA2S3_24;
print hazratioA3S1_12 hazratioA3S1_24 hazratioA3S2_12 hazratioA3S2_24 hazratioA3S3_12 hazratioA3S3_24;

print oddsratioA1S1_12 oddsratioA1S1_24 oddsratioA1S2_12 oddsratioA1S2_24 oddsratioA1S3_12 oddsratioA1S3_24;
print oddsratioA2S1_12 oddsratioA2S1_24 oddsratioA2S2_12 oddsratioA2S2_24 oddsratioA2S3_12 oddsratioA2S3_24;
print oddsratioA3S1_12 oddsratioA3S1_24 oddsratioA3S2_12 oddsratioA3S2_24 oddsratioA3S3_12 oddsratioA3S3_24;

**********Median Lifetime*****************************************************;
medianlifetime=(1/lambdaBASELINE)##(1/alpha);
medianA1S1=(1/lambdaA1S1)##(1/alpha);
medianA1S2=(1/lambdaA1S2)##(1/alpha);
medianA1S3=(1/lambdaA1S3)##(1/alpha);
medianA2S1=(1/lambdaA2S1)##(1/alpha);
medianA2S2=(1/lambdaA2S2)##(1/alpha);
medianA2S3=(1/lambdaA2S3)##(1/alpha);
medianA3S1=(1/lambdaA3S1)##(1/alpha);
medianA3S2=(1/lambdaA3S2)##(1/alpha);
medianA3S3=(1/lambdaA3S3)##(1/alpha);

print 'medianlifetime=' medianlifetime;
print 'medianlifetime(age=A1,score=S1)=' medianA1S1
      'medianlifetime(age=A1,score=S2)=' medianA1S2
      'medianlifetime(age=A1,score=S3)=' medianA1S3;
print 'medianlifetime(age=A2,score=S1)=' medianA2S1
      'medianlifetime(age=A2,score=S2)=' medianA2S2
      'medianlifetime(age=A2,score=S3)=' medianA2S3;
print 'medianlifetime(age=A3,score=S1)=' medianA3S1
      'medianlifetime(age=A3,score=S2)=' medianA3S2
      'medianlifetime(age=A3,score=S3)=' medianA3S3;
```

7. Program for fitting a Weibull regression model with two predictors

```
title1 'Fitting of regression model with two covariates';
title2 'Staggered entry: constant shape parameter';
```

```
proc iml worksize= 60;
reset nolog;
options pagesize=500;

**********Frequency vector (first entry,3 agegroups,3 scoregroups);
                f111={12,34,51,39,57,11, 59};   f111=f111<>1e-04;
                f112={10,12,22,19,32, 4,418};   f112=f112<>1e-04;
                f113={ 7,13,22,15,19, 0,165};   f113=f113<>1e-04;
                f121={13,14,45,27,33, 4, 66};   f121=f121<>1e-04;
                f122={ 4,22,22, 8,25, 4,297};   f122=f122<>1e-04;
                f123={ 4,14,24,10,17, 5,190};   f123=f123<>1e-04;
                f131={10,25,29,17,46, 2,116};   f131=f131<>1e-04;
                f132={ 6,13,28,16,16, 5,273};   f132=f132<>1e-04;
                f133={ 0,11,11, 6, 5, 0, 82};   f133=f133<>1e-04;

**********Vector of upper boundaries;
                x1={12,17,24,28,34,37};

**********Frequency vector (second entry,3 agegroups,3 scoregroups);
                f211={22,25,58,53,40, 45};      f211=f211<>1e-04;
                f212={10,26,32,20,29,379};      f212=f212<>1e-04;
                f213={ 9,24,13,19,14,204};      f213=f213<>1e-04;
                f221={24,24,28,30,25,106};      f221=f221<>1e-04;
                f222={12,20,14,17,16,409};      f222=f222<>1e-04;
                f223={13,18,19,19,13,238};      f223=f223<>1e-04;
                f231={13,15,32,19,17,107};      f231=f231<>1e-04;
                f232={11,13,22,17,12,319};      f232=f232<>1e-04;
                f233={ 4, 1,11, 6, 6,117};      f233=f233<>1e-04;

**********Vector of upper boundaries;
                x2={12,17,24,28,34};

**********Frequency vector (third entry,3 agegroups,3 scoregroups);
                f311={34,16,50,23, 54};         f311=f311<>1e-04;
                f312={19, 2,32,24,317};         f312=f312<>1e-04;
                f313={15,16,17,10,199};         f313=f313<>1e-04;
                f321={19,18,38,16, 75};         f321=f321<>1e-04;
                f322={16,14,25,10,263};         f322=f322<>1e-04;
                f323={ 5,12,20, 7,195};         f323=f323<>1e-04;
                f331={28,16,22,12, 98};         f331=f331<>1e-04;
                f332={13, 0,24, 4,323};         f332=f332<>1e-04;
                f333={ 5, 5,14,11,150};         f333=f333<>1e-04;

**********Vector of upper boundaries;
                x3={12,17,24,28};

**********Frequency vector (fourth entry,3 agegroups,3 scoregroups);
                f411={40,30,30, 50};            f411=f411<>1e-04;
                f412={ 9,14,27,301};            f412=f412<>1e-04;
                f413={22,16,12,222};            f413=f413<>1e-04;
                f421={24,30,29, 81};            f421=f421<>1e-04;
                f422={14,15,12,307};            f422=f422<>1e-04;
                f423={16,16,27,228};            f423=f423<>1e-04;
                f431={20,22,28,119};            f431=f431<>1e-04;
                f432={19,12,26,369};            f432=f432<>1e-04;
                f433={11,11,16,171};            f433=f433<>1e-04;

**********Vector of upper boundaries;
                x4={12,17,24};

**********Relative frequency vectors;
n111=f111[+]; n211=f211[+]; n311=f311[+]; n411=f411[+]; n11=n111+n211+n311+n411;
n112=f112[+]; n212=f212[+]; n312=f312[+]; n412=f412[+]; n12=n112+n212+n312+n412;
n113=f113[+]; n213=f213[+]; n313=f313[+]; n413=f413[+]; n13=n113+n213+n313+n413;
n121=f121[+]; n221=f221[+]; n321=f321[+]; n421=f421[+]; n21=n121+n221+n321+n421;
n122=f122[+]; n222=f222[+]; n322=f322[+]; n422=f422[+]; n22=n122+n222+n322+n422;
n123=f123[+]; n223=f223[+]; n323=f323[+]; n423=f423[+]; n23=n123+n223+n323+n423;
n131=f131[+]; n231=f231[+]; n331=f331[+]; n431=f431[+]; n31=n131+n231+n331+n431;
n132=f132[+]; n232=f232[+]; n332=f332[+]; n432=f432[+]; n32=n132+n232+n332+n432;
n133=f133[+]; n233=f233[+]; n333=f333[+]; n433=f433[+]; n33=n133+n233+n333+n433;

                                                        n1=n11+n12+n13;
                                                        n2=n21+n22+n23;
                                                        n3=n31+n32+n33;
                                                        n=n1+n2+n3;

k1=nrow(f111); d1=k1-1;
k2=nrow(f211); d2=k2-1;
k3=nrow(f311); d3=k3-1;
k4=nrow(f411); d4=k4-1;
k=k1+k2+k3+k4;
d=d1+d2+d3+d4;

p111=f111/n111; p211=f211/n211; p311=f311/n311; p411=f411/n411;
p112=f112/n112; p212=f212/n212; p312=f312/n312; p412=f412/n412;
p113=f113/n113; p213=f213/n213; p313=f313/n313; p413=f413/n413;
p121=f121/n121; p221=f221/n221; p321=f321/n321; p421=f421/n421;
p122=f122/n122; p222=f222/n222; p322=f322/n322; p422=f422/n422;
```

```
p123=f123/n123; p223=f223/n223; p323=f323/n323; p423=f423/n423;
p131=f131/n131; p231=f231/n231; p331=f331/n331; p431=f431/n431;
p132=f132/n132; p232=f232/n232; p332=f332/n332; p432=f432/n432;
p133=f133/n133; p233=f233/n233; p333=f333/n333; p433=f433/n433;

p11=p111//p211//p311//p411;
p12=p112//p212//p312//p412;
p13=p113//p213//p313//p413;
p1=p11//p12//p13;

p21=p121//p221//p321//p421;
p22=p122//p222//p322//p422;
p23=p123//p223//p323//p423;
p2=p21//p22//p23;

p31=p131//p231//p331//p431;
p32=p132//p232//p332//p432;
p33=p133//p233//p333//p433;
p3=p31//p32//p33;

p=p1//p2//p3;

**********Design matrix and matrix orthogonal to design matrix;

S1=J(d1,1,1)@cusum(J(1,k1,1));
S2=J(1,k1,1)@cusum(J(d1,1,1));
S1=S1<=S2;
S2=S1[1:d2,1:d1];
S3=S1[1:d3,1:d2];
S4=S1[1:d4,1:d3];
S=block(S1,S2,S3,S4);
S=I(3)@I(3)@S;

AA=J(162,1,1);           *162=9 times((7-1)+(6-1)+(5-1)+(4-1))=9 times 18=162;
BB=J(3,1,1);
CC=(I(2)//J(1,2,-1));
DD=J(18,1,1);            *3 times d1=3(6)=18;
EE=J(15,1,1);            *3 times d2=3(5)=15;
FF=J(12,1,1);            *3 times d3=3(4)=12;
GG=J(9,1,1);             *3 times d4=3(3)=9;

KK=J(9,1,1);             *9=3 times 3;
LL=J(54,1,1);            *54=18 times 3;

lx=BB@BB@(log(x1)//log(x2)//log(x3)//log(x4));

HH=BB@(CC@DD);

xc=AA||CC@LL||HH||lx;  print xc;

C=I(9#d)-xc*inv(xc`*xc)*xc`;

**********ITERATIVE PROCEDURE (double iterations over m and p);
*****starting value for m;
m=p;
ms=S*m; ps=ms;
p0=p;

*****iteration over m;
itr=0;
verskil1=1;
i=0;
do while (verskil1>0.00000001);
i=i+1;
p=p0;
Gm=-C*diag(1/(log(1-ms)))*diag(1/(1-ms))*S;             *Weibull;
*Gm=C*diag(1/ms+1/(1-ms))*S;                            *log-logistic;

**********iteration over p;
     verskil=1;
     j=0;
     do while (verskil>0.00000001);
     j=j+1;
     p1=p;
     ps=S*p;
    g=C*log(-log(1-ps));                                 *Weibull;
    *g=C*(log(ps)-log(1-ps));                            *log-logistic;
    Gp=-C*diag(1/(log(1-ps)))*diag(1/(1-ps))*S;          *Weibull;
    *Gp=C*(diag(1/ps)+diag(1/(1-ps)))*S;                 *log-logistic;

******************covariance matrix***********************************;
m111=m[1:k1];
m211=m[k1+1:k1+k2];
m311=m[k1+k2+1:k1+k2+k3];
m411=m[k1+k2+k3+1:k1+k2+k3+k4];

m112=m[k+1:k+k1];
m212=m[k+k1+1:k+k1+k2];
m312=m[k+k1+k2+1:k+k1+k2+k3];
```

```
m412=m[k+k1+k2+k3+1:k+k1+k2+k3+k4];

m113=m[2#k+1:2#k+k1];
m213=m[2#k+k1+1:2#k+k1+k2];
m313=m[2#k+k1+k2+1:2#k+k1+k2+k3];
m413=m[2#k+k1+k2+k3+1:2#k+k1+k2+k3+k4];

m121=m[3#k+1:3#k+k1];
m221=m[3#k+k1+1:3#k+k1+k2];
m321=m[3#k+k1+k2+1:3#k+k1+k2+k3];
m421=m[3#k+k1+k2+k3+1:3#k+k1+k2+k3+k4];

m122=m[4#k+1:4#k+k1];
m222=m[4#k+k1+1:4#k+k1+k2];
m322=m[4#k+k1+k2+1:4#k+k1+k2+k3];
m422=m[4#k+k1+k2+k3+1:4#k+k1+k2+k3+k4];

m123=m[5#k+1:5#k+k1];
m223=m[5#k+k1+1:5#k+k1+k2];
m323=m[5#k+k1+k2+1:5#k+k1+k2+k3];
m423=m[5#k+k1+k2+k3+1:5#k+k1+k2+k3+k4];

m131=m[6#k+1:6#k+k1];
m231=m[6#k+k1+1:6#k+k1+k2];
m331=m[6#k+k1+k2+1:6#k+k1+k2+k3];
m431=m[6#k+k1+k2+k3+1:6#k+k1+k2+k3+k4];

m132=m[7#k+1:7#k+k1];
m232=m[7#k+k1+1:7#k+k1+k2];
m332=m[7#k+k1+k2+1:7#k+k1+k2+k3];
m432=m[7#k+k1+k2+k3+1:7#k+k1+k2+k3+k4];

m133=m[8#k+1:8#k+k1];
m233=m[8#k+k1+1:8#k+k1+k2];
m333=m[8#k+k1+k2+1:8#k+k1+k2+k3];
m433=m[8#k+k1+k2+k3+1:9#k];

sig111=(1/n111)*(diag(m111)-m111*m111`);
sig112=(1/n112)*(diag(m112)-m112*m112`);
sig113=(1/n113)*(diag(m113)-m113*m113`);
sig211=(1/n211)*(diag(m211)-m211*m211`);
sig212=(1/n212)*(diag(m212)-m212*m212`);
sig213=(1/n213)*(diag(m213)-m213*m213`);
sig311=(1/n311)*(diag(m311)-m311*m311`);
sig312=(1/n312)*(diag(m312)-m312*m312`);
sig313=(1/n313)*(diag(m313)-m313*m313`);
sig411=(1/n411)*(diag(m411)-m411*m411`);
sig412=(1/n412)*(diag(m412)-m412*m412`);
sig413=(1/n413)*(diag(m413)-m413*m413`);

sig11=block(sig111,sig211,sig311,sig411);
sig12=block(sig112,sig212,sig312,sig412);
sig13=block(sig113,sig213,sig313,sig413);


sig121=(1/n121)*(diag(m121)-m121*m121`);
sig122=(1/n122)*(diag(m122)-m122*m122`);
sig123=(1/n123)*(diag(m123)-m123*m123`);
sig221=(1/n221)*(diag(m221)-m221*m221`);
sig222=(1/n222)*(diag(m222)-m222*m222`);
sig223=(1/n223)*(diag(m223)-m223*m223`);
sig321=(1/n321)*(diag(m321)-m321*m321`);
sig322=(1/n322)*(diag(m322)-m322*m322`);
sig323=(1/n323)*(diag(m323)-m323*m323`);
sig421=(1/n421)*(diag(m421)-m421*m421`);
sig422=(1/n422)*(diag(m422)-m422*m422`);
sig423=(1/n423)*(diag(m423)-m423*m423`);

sig21=block(sig121,sig221,sig321,sig421);
sig22=block(sig122,sig222,sig322,sig422);
sig23=block(sig123,sig223,sig323,sig423);


sig131=(1/n131)*(diag(m131)-m131*m131`);
sig132=(1/n132)*(diag(m132)-m132*m132`);
sig133=(1/n133)*(diag(m133)-m133*m133`);
sig231=(1/n231)*(diag(m231)-m231*m231`);
sig232=(1/n232)*(diag(m232)-m232*m232`);
sig233=(1/n233)*(diag(m233)-m233*m233`);
sig331=(1/n331)*(diag(m331)-m331*m331`);
sig332=(1/n332)*(diag(m332)-m332*m332`);
sig333=(1/n333)*(diag(m333)-m333*m333`);
sig431=(1/n431)*(diag(m431)-m431*m431`);
sig432=(1/n432)*(diag(m432)-m432*m432`);
sig433=(1/n433)*(diag(m433)-m433*m433`);

sig31=block(sig131,sig231,sig331,sig431);
sig32=block(sig132,sig232,sig332,sig432);
sig33=block(sig133,sig233,sig333,sig433);
```

```
sig=block(sig11,sig12,sig13,sig21,sig22,sig23,sig31,sig32,sig33);
V=sig;
*******************************************************************;
    p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                        *Weibull;
    *p=p-(Gm*V)`*ginv(Gp*V*Gm`)*g;                       *log-logistic;
     verskil=sqrt((p-p1)`*(p-p1));
     end;
verskil1=sqrt((p-m)`*(p-m));
m=p;ms=S*m;
end;
print m; print i j;

**********Parameter vector for linear model;
par=inv(xc`*xc)*xc`*log(-log(1-ms));                      *Weibull;
*par=inv(xc`*xc)*xc`*(log(ms)-log(1-ms));                 *log-logistic;

print par[format=E20.];

**********Regression coefficients;
oualphaBASELINE=par[1];
betaA1=par[2];
betaA2=par[3];
betaA3=-(par[2]+par[3]);
betaS1=par[4];
betaS2=par[5];
betaS3=-(par[4]+par[5]);
lambdaBASELINE=exp(par[1]);
alpha=par[6];

**********Indices, lambda's and constant alpha for age levels******************;
indexA1=exp(betaA1);          *constant shape parameter;
indexA2=exp(betaA2);
indexA3=exp(betaA3);
indexS1=exp(betaS1);
indexS2=exp(betaS2);
indexS3=exp(betaS3);

lambdaA1S1=lambdaBASELINE#indexA1#indexS1;
lambdaA1S2=lambdaBASELINE#indexA1#indexS2;
lambdaA1S3=lambdaBASELINE#indexA1#indexS3;
lambdaA2S1=lambdaBASELINE#indexA2#indexS1;
lambdaA2S2=lambdaBASELINE#indexA2#indexS2;
lambdaA2S3=lambdaBASELINE#indexA2#indexS3;
lambdaA3S1=lambdaBASELINE#indexA3#indexS1;
lambdaA3S2=lambdaBASELINE#indexA3#indexS2;
lambdaA3S3=lambdaBASELINE#indexA3#indexS3;

oualphaA1S1=oualphaBASELINE+betaA1+betaS1; *same as oualphaA1S1=log(lambdaA1S1);
oualphaA1S2=oualphaBASELINE+betaA1+betaS2;
oualphaA1S3=oualphaBASELINE+betaA1+betaS3;
oualphaA2S1=oualphaBASELINE+betaA2+betaS1;
oualphaA2S2=oualphaBASELINE+betaA2+betaS2;
oualphaA2S3=oualphaBASELINE+betaA2+betaS3;
oualphaA3S1=oualphaBASELINE+betaA3+betaS1;
oualphaA3S2=oualphaBASELINE+betaA3+betaS2;
oualphaA3S3=oualphaBASELINE+betaA3+betaS3;

print ' Weibull parameters, beta effects and indices: MLE subject to constraints';

print 'lambdaBASELINE=' lambdaBASELINE[format=E20.]  'oualphaBASELINE=' oualphaBASELINE;
print 'alpha=' alpha[format=E20.];

print 'lambda(age=A1,score=S1)=' lambdaA1S1[format=E20.]
      'lambda(age=A1,score=S2)=' lambdaA1S2[format=E20.]
      'lambda(age=A1,score=S3)=' lambdaA1S3[format=E20.];
print oualphaA1S1 oualphaA1S2 oualphaA1S3;

print 'lambda(age=A2,score=S1)=' lambdaA2S1[format=E20.]
      'lambda(age=A2,score=S2)=' lambdaA2S2[format=E20.]
      'lambda(age=A2,score=S3)=' lambdaA2S3[format=E20.];
print oualphaA2S1 oualphaA2S2 oualphaA2S3;

print 'lambda(age=A3,score=S1)=' lambdaA3S1[format=E20.]
      'lambda(age=A3,score=S2)=' lambdaA3S2[format=E20.]
      'lambda(age=A3,score=S3)=' lambdaA3S3[format=E20.];
print oualphaA3S1 oualphaA3S2 oualphaA3S3;

print 'beta(age=A1)=' betaA1 'beta(age=A2)=' betaA2 'beta(age=A3)=' betaA3;
print 'beta(score=S1)=' betaS1 'beta(score=S2)=' betaS2 'beta(score=S3)=' betaS3;
print 'index(age=A1)=' indexA1 'index(age=A2)=' indexA2 'index(age=A3)=' indexA3;
print 'index(score=S1)=' indexS1 'index(score=S2)=' indexS2 'index(score=S3)=' indexS3;

**********Hazard ratio and Odds ratio******************************************;
hazBASELINE12=lambdaBASELINE*alpha*12**(alpha-1);
hazBASELINE24=lambdaBASELINE*alpha*24**(alpha-1);
survBASELINE12=exp(-lambdaBASELINE*12**alpha);
survBASELINE24=exp(-lambdaBASELINE*24**alpha);
oddsBASELINE12=(1-survBASELINE12)/survBASELINE12;
```

```
oddsBASELINE24=(1-survBASELINE24)/survBASELINE24;

hazA1S1_12=lambdaA1S1*alpha*12**(alpha-1);
hazA1S1_24=lambdaA1S1*alpha*24**(alpha-1);
hazA1S2_12=lambdaA1S2*alpha*12**(alpha-1);
hazA1S2_24=lambdaA1S2*alpha*24**(alpha-1);
hazA1S3_12=lambdaA1S3*alpha*12**(alpha-1);
hazA1S3_24=lambdaA1S3*alpha*24**(alpha-1);
hazA2S1_12=lambdaA2S1*alpha*12**(alpha-1);
hazA2S1_24=lambdaA2S1*alpha*24**(alpha-1);
hazA2S2_12=lambdaA2S2*alpha*12**(alpha-1);
hazA2S2_24=lambdaA2S2*alpha*24**(alpha-1);
hazA2S3_12=lambdaA2S3*alpha*12**(alpha-1);
hazA2S3_24=lambdaA2S3*alpha*24**(alpha-1);
hazA3S1_12=lambdaA3S1*alpha*12**(alpha-1);
hazA3S1_24=lambdaA3S1*alpha*24**(alpha-1);
hazA3S2_12=lambdaA3S2*alpha*12**(alpha-1);
hazA3S2_24=lambdaA3S2*alpha*24**(alpha-1);
hazA3S3_12=lambdaA3S3*alpha*12**(alpha-1);
hazA3S3_24=lambdaA3S3*alpha*24**(alpha-1);

survA1S1_12=exp(-lambdaA1S1*12**alpha);
survA1S1_24=exp(-lambdaA1S1*24**alpha);
survA1S2_12=exp(-lambdaA1S2*12**alpha);
survA1S2_24=exp(-lambdaA1S2*24**alpha);
survA1S3_12=exp(-lambdaA1S3*12**alpha);
survA1S3_24=exp(-lambdaA1S3*24**alpha);
survA2S1_12=exp(-lambdaA2S1*12**alpha);
survA2S1_24=exp(-lambdaA2S1*24**alpha);
survA2S2_12=exp(-lambdaA2S2*12**alpha);
survA2S2_24=exp(-lambdaA2S2*24**alpha);
survA2S3_12=exp(-lambdaA2S3*12**alpha);
survA2S3_24=exp(-lambdaA2S3*24**alpha);
survA3S1_12=exp(-lambdaA3S1*12**alpha);
survA3S1_24=exp(-lambdaA3S1*24**alpha);
survA3S2_12=exp(-lambdaA3S2*12**alpha);
survA3S2_24=exp(-lambdaA3S2*24**alpha);
survA3S3_12=exp(-lambdaA3S3*12**alpha);
survA3S3_24=exp(-lambdaA3S3*24**alpha);

oddsA1S1_12=(1-survA1S1_12)/survA1S1_12;
oddsA1S1_24=(1-survA1S1_24)/survA1S1_24;
oddsA1S2_12=(1-survA1S2_12)/survA1S2_12;
oddsA1S2_24=(1-survA1S2_24)/survA1S2_24;
oddsA1S3_12=(1-survA1S3_12)/survA1S3_12;
oddsA1S3_24=(1-survA1S3_24)/survA1S3_24;
oddsA2S1_12=(1-survA2S1_12)/survA2S1_12;
oddsA2S1_24=(1-survA2S1_24)/survA2S1_24;
oddsA2S2_12=(1-survA2S2_12)/survA2S2_12;
oddsA2S2_24=(1-survA2S2_24)/survA2S2_24;
oddsA2S3_12=(1-survA2S3_12)/survA2S3_12;
oddsA2S3_24=(1-survA2S3_24)/survA2S3_24;
oddsA3S1_12=(1-survA3S1_12)/survA3S1_12;
oddsA3S1_24=(1-survA3S1_24)/survA3S1_24;
oddsA3S2_12=(1-survA3S2_12)/survA3S2_12;
oddsA3S2_24=(1-survA3S2_24)/survA3S2_24;
oddsA3S3_12=(1-survA3S3_12)/survA3S3_12;
oddsA3S3_24=(1-survA3S3_24)/survA3S3_24;

hazratioA1S1_12=hazA1S1_12/hazBASELINE12;
hazratioA1S1_24=hazA1S1_24/hazBASELINE24;
hazratioA1S2_12=hazA1S2_12/hazBASELINE12;
hazratioA1S2_24=hazA1S2_24/hazBASELINE24;
hazratioA1S3_12=hazA1S3_12/hazBASELINE12;
hazratioA1S3_24=hazA1S3_24/hazBASELINE24;
hazratioA2S1_12=hazA2S1_12/hazBASELINE12;
hazratioA2S1_24=hazA2S1_24/hazBASELINE24;
hazratioA2S2_12=hazA2S2_12/hazBASELINE12;
hazratioA2S2_24=hazA2S2_24/hazBASELINE24;
hazratioA2S3_12=hazA2S3_12/hazBASELINE12;
hazratioA2S3_24=hazA2S3_24/hazBASELINE24;
hazratioA3S1_12=hazA3S1_12/hazBASELINE12;
hazratioA3S1_24=hazA3S1_24/hazBASELINE24;
hazratioA3S2_12=hazA3S2_12/hazBASELINE12;
hazratioA3S2_24=hazA3S2_24/hazBASELINE24;
hazratioA3S3_12=hazA3S3_12/hazBASELINE12;
hazratioA3S3_24=hazA3S3_24/hazBASELINE24;

oddsratioA1S1_12=oddsA1S1_12/oddsBASELINE12;
oddsratioA1S1_24=oddsA1S1_24/oddsBASELINE24;
oddsratioA1S2_12=oddsA1S2_12/oddsBASELINE12;
oddsratioA1S2_24=oddsA1S2_24/oddsBASELINE24;
oddsratioA1S3_12=oddsA1S3_12/oddsBASELINE12;
oddsratioA1S3_24=oddsA1S3_24/oddsBASELINE24;
oddsratioA2S1_12=oddsA2S1_12/oddsBASELINE12;
oddsratioA2S1_24=oddsA2S1_24/oddsBASELINE24;
oddsratioA2S2_12=oddsA2S2_12/oddsBASELINE12;
oddsratioA2S2_24=oddsA2S2_24/oddsBASELINE24;
oddsratioA2S3_12=oddsA2S3_12/oddsBASELINE12;
```

```
oddsratioA2S3_24=oddsA2S3_24/oddsBASELINE24;
oddsratioA3S1_12=oddsA3S1_12/oddsBASELINE12;
oddsratioA3S1_24=oddsA3S1_24/oddsBASELINE24;
oddsratioA3S2_12=oddsA3S2_12/oddsBASELINE12;
oddsratioA3S2_24=oddsA3S2_24/oddsBASELINE24;
oddsratioA3S3_12=oddsA3S3_12/oddsBASELINE12;
oddsratioA3S3_24=oddsA3S3_24/oddsBASELINE24;

print hazBASELINE12 hazBASELINE24 survBASELINE12 survBASELINE24 oddsBASELINE12 oddsBASELINE24;

print hazA1S1_12 hazA1S1_24 hazA1S2_12 hazA1S2_24 hazA1S3_12 hazA1S3_24;
print hazA2S1_12 hazA2S1_24 hazA2S2_12 hazA2S2_24 hazA2S3_12 hazA2S3_24;
print hazA3S1_12 hazA3S1_24 hazA3S2_12 hazA3S2_24 hazA3S3_12 hazA3S3_24;

print survA1S1_12 survA1S1_24 survA1S2_12 survA1S2_24 survA1S3_12 survA1S3_24;
print survA2S1_12 survA2S1_24 survA2S2_12 survA2S2_24 survA2S3_12 survA2S3_24;
print survA3S1_12 survA3S1_24 survA3S2_12 survA3S2_24 survA3S3_12 survA3S3_24;

print oddsA1S1_12 oddsA1S1_24 oddsA1S2_12 oddsA1S2_24 oddsA1S3_12 oddsA1S3_24;
print oddsA2S1_12 oddsA2S1_24 oddsA2S2_12 oddsA2S2_24 oddsA2S3_12 oddsA2S3_24;
print oddsA3S1_12 oddsA3S1_24 oddsA3S2_12 oddsA3S2_24 oddsA3S3_12 oddsA3S3_24;

print hazratioA1S1_12 hazratioA1S1_24 hazratioA1S2_12 hazratioA1S2_24 hazratioA1S3_12 hazratioA1S3_24;
print hazratioA2S1_12 hazratioA2S1_24 hazratioA2S2_12 hazratioA2S2_24 hazratioA2S3_12 hazratioA2S3_24;
print hazratioA3S1_12 hazratioA3S1_24 hazratioA3S2_12 hazratioA3S2_24 hazratioA3S3_12 hazratioA3S3_24;

print oddsratioA1S1_12 oddsratioA1S1_24 oddsratioA1S2_12 oddsratioA1S2_24 oddsratioA1S3_12 oddsratioA1S3_24;
print oddsratioA2S1_12 oddsratioA2S1_24 oddsratioA2S2_12 oddsratioA2S2_24 oddsratioA2S3_12 oddsratioA2S3_24;
print oddsratioA3S1_12 oddsratioA3S1_24 oddsratioA3S2_12 oddsratioA3S2_24 oddsratioA3S3_12 oddsratioA3S3_24;

**********Median Lifetime*******************************************************;
medianBASELINE=((1/lambdaBASELINE)#log(2))##(1/alpha);
medianA1S1=((1/lambdaA1S1)#log(2))##(1/alpha);
medianA1S2=((1/lambdaA1S2)#log(2))##(1/alpha);
medianA1S3=((1/lambdaA1S3)#log(2))##(1/alpha);
medianA2S1=((1/lambdaA2S1)#log(2))##(1/alpha);
medianA2S2=((1/lambdaA2S2)#log(2))##(1/alpha);
medianA2S3=((1/lambdaA2S3)#log(2))##(1/alpha);
medianA3S1=((1/lambdaA3S1)#log(2))##(1/alpha);
medianA3S2=((1/lambdaA3S2)#log(2))##(1/alpha);
medianA3S3=((1/lambdaA3S3)#log(2))##(1/alpha);

print 'medianlifetimeBASELINE=' medianBASELINE;
print 'medianlifetime(age=A1,score=S1)=' medianA1S1
      'medianlifetime(age=A1,score=S2)=' medianA1S2
      'medianlifetime(age=A1,score=S3)=' medianA1S3;
print 'medianlifetime(age=A2,score=S1)=' medianA2S1
      'medianlifetime(age=A2,score=S2)=' medianA2S2
      'medianlifetime(age=A2,score=S3)=' medianA2S3;
print 'medianlifetime(age=A3,score=S1)=' medianA3S1
      'medianlifetime(age=A3,score=S2)=' medianA3S2
      'medianlifetime(age=A3,score=S3)=' medianA3S3;
```

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Bibliography

[1] AGRESTI, A. (1990). *Categorical data analysis.* New York: Wiley.

[2] BAIN, L.J. and ENGELHARDT, M. (1991). *Statistical analysis of reliability and life-testing models:theory and methods.* New York: Marcel Dekker.

[3] BENNETT, S. (1983). Analysis of survival data by the proportional odds model. Ph.D. Thesis, University of Reading, United Kingdom.

[4] CANTOR, A. (1998). *Extending Sas survival analysis techniques for medical research.* USA: SAS Institute Inc.

[5] COLLETT, D. (1994). *Modelling survival data in medical research.* London: Chapman Hall.

[6] COX, D.R. (1972). Regression models and life-tables. *Journal of the royal statistical society,* **34**: *187-220.*

[7] COX, D.R. and OAKES, D. (1984). *Analysis of survival data.* London: Chapman Hall.

[8] COX, D.R. (1995). Some remarks on the analysis of survival data. *Proceedings of the First Seattle Symposium in Biostatistics: Survival Analysis 123-169.*

[9] CROWDER, M.J., KIMBER, A.C., SMITH, R.L. and SWEETING, T.J. (1991) *Statistical analysis of reliability data.* London: Chapman Hall.

[10] CROWTHER, N.A.S. and MATTHEWS, G.B. (unpublished paper). Fitting distributions to grouped data.

[11] CROWTHER, N.A.S. and MATTHEWS, G.B. (1995). A Maximum likelihood estimation procedure when modelling in terms of constraints. *S.A. Statistical Journal,* **29(1)**, *29-51.*

[12] CROWTHER, N.A.S. and SHAW, T.M. (1989). An estimation procedure in categorical data analysis. *S.A. Statistical Journal,* **23(1)**: *63-83.*

[13] ELANDT-JOHNSON, R.C. and JOHNSON, N.L. (1990). *Survival models and data analysis.* New York: Wiley.

[14] FARRINGTON, C.P. (1996). Interval censored survival data: a generalized linear modeling approach. *Statistics in Medicine,* **15***: 283-292.*

[15] FINKELSTEIN, D.M. (1986). A proportional hazards model for interval censored failure time data. *Biometrics,* **42***: 845-854.*

[16] GROSS, A.J. and CLARK, V.A. (1975). *Survival distributions: reliability applications in the Biomedical Sciences.* New York: Wiley.

[17] HEITJAN, D.F. (1989). Inference from grouped continuous data: a review. *Ststistical Science,* **4***: 164-183.*

[18] HOSMER, D.W. and LEMESHOW, S. (1999). *Applied survival analysis: regression modeling of time to event data.* New York: Wiley.

[19] HUANG, J. and WELLNER, J.A. (1995). Interval censored survival data: a review of recent progress. *Proceedings of the First Seattle Symposium in Biostatistics: Survival Analysis 123-169.*

[20] JOUBERT, H.M. (1991). Fitting of distributions in the case of survey data. M.Sc. dissertation, University of South Africa, South Africa.

[21] KALBFLEISCH, J.D. and PRENTICE, R.L. (1980). *The statistical analysis of failure time data.* New York: Wiley.

[22] KLEIN, J.P. and MOESCHBERGER, M.L. (1997). *Survival analysis techniques for censored and truncated data.* New York: Springer-Verlag.

[23] KLEINBAUM, D.G. (1997). *Survival analysis: a self-learning text.* New York: Springer-Verlag.

[24] LAWLESS, J.F. (1982). *Statistical models and methods for lifetime data.* New York: Wiley.

[25] LE, C.T. (1997). *Applied survival analysis.* New York: Wiley.

[26] LEE, E.T. (1992). *Statistical methods for survival data analysis.* New York: Wiley.

[27] LINDSEY, J.K. (1998). A study of interval censoring in parametric regression models. *Lifetime Data Analysis,* **4***: 329-354.*

[28] LOUZADA-NETO, F. (1997). Extended hazard regression model for reliability and survival analysis. *Lifetime Data Analysis,* **3***: 367-381.*

[29] MALLER, R. and ZHOU, X. (1996). *Survival analysis with long-term survivors.* New York: Wiley.

[30] MATTHEWS, G.B. (1995). Maximum likelihood estimation when modelling in terms of constraints. Ph.D. Thesis, University of Pretoria, South Africa.

[31] MATTHEWS, G.B. and CROWTHER, N.A.S. (1998). A Maximum likelihood estimation procedure for the generalized linear model with restrictions. *S.A. Statistical Journal,* **32(2)**, *119-144.*

[32] McCullagh, P. (1980). Regression models for ordinal data (with discussion). *Journal of the Royal Statistical Society,* **42**: *109-142.*

[33] McKEAGUE, I.W. and ZHANG, M. (1996). Fitting Cox's proportional hazards model using grouped survival data. *Lifetime Data: Models in Reliability and Survival Analysis, 227-232.*

[34] MILLER, R.G. (1998). *Survival analysis.* New York: Wiley.

[35] NELSON, W. (1990). *Accelerated testing, statistical models, test plans, and data analysis.* New York: Wiley.

[36] ODELL, P.M., ANDERSON, K.M. and D'AGOSTINO, R.B. (1992). Maximum likelihood estimation for interval-censored data using a Weibull-based accelerated failure time model. *Biometrics,* **48**: *951-959.*

[37] PANJER, H.H. and WILLMOT, G.E. (1992). *Insurance risk models.* USA: Society of Actuaries.

[38] PARMAR, M.K.B. and MACHIN, D. (1995). *Survival analysis: a practical approach.* Chichester: Wiley.

[39] RAFTERY, A.E., TANNER, M.A. and WELLS, M.T. (2001). *Statistics in the 21st Century.* London: Chapman and Hall.

[40] SAS Institute Inc. (2000). *SAS/IML^{TM} user's guide for personal computers, Version 8.1* Cary, NC: SAS Institute Inc.

[41] WEI, L.J. (1992). The accelerated failure time model: a useful alternative to the Cox regression model in survival analysis. *Statistics in Medicine,* **11**: *1871-1879.*

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Abstract

Fitting of survival functions for grouped data

on insurance policies

by

Elizabeth Magrietha Louw

Supervisor: **Professor N.A.S. Crowther**
Degree: **Ph D**

The aim of the research is the statistical modelling of parametric survival distributions of grouped survival data of long- and shortterm policies in the insurance industry, by means of a method of maximum likelihood estimation subject to constraints.

This methodology leads to explicit expressions for the estimates of the parameters, as well as for approximated variances and covariances of the estimates, which gives exact maximum likelihood estimates of the parameters. This makes direct extension to more complex designs feasible.

The statistical modelling offers parametric models for survival distributions, in contrast with non-parametric models that are used commonly in the actuarial profession. When the parametric models provide a good fit to data, they tend to give more precise estimates of the quantities of interest such as odds ratios, hazard ratios or median lifetimes. These estimates form the statistical foundation for scientific decisionmaking with respect to actuarial design, maintenance and marketing of insurance policies.

Although the methodology in this thesis is developed specifically for the insurance industry, it may be applied in the normal context of research and scientific decisionmaking, that includes for example survival distributions for the medical, biological, engineering, econometric and sociological sciences.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Ekserp

Fitting of survival functions for grouped data on insurance policies

deur

Elizabeth Magrietha Louw

Promotor:	**Professor N.A.S. Crowther**
Graad:	**Ph D**

Die doelwit van die navorsing is die statistiese modellering van parametriese oorlewingsverdelings van gegroepeerde oorlewingsdata van lang- en korttermyn polisse in die versekeringsbedryf, deur middel van 'n metode van maksimum aanneemlikheidsberaming onderworpe aan beperkings.

Hierdie metode lei tot eksplisiete uitdrukkings vir die beramings van die parameters, asook vir benaderde variansies en kovariansies van die beramers, wat eksakte maksimum aanneemlikheidsberamings van die parameters gee. Dit maak direkte uitbreiding na meer komplekse ontwerpe moontlik.

Die statistiese modellering bied parametriese modelle vir oorlewingsverdelings, in teenstelling met nie-parametriese modelle wat algemeen in die aktuariële professie gebruik word. Indien die parametriese modelle 'n goeie passing by die data gee, sal hulle meer akkurate beramings van die hoeveelhede van belang, soos kruisprodukverhoudings, risikoverhoudings of mediaan leeftye gee. Hierdie beramings vorm die statistiese grondslag vir wetenskaplike besluitneming met betrekking tot aktuariële ontwerp, onderhoud en bemarking van versekeringspolisse.

Alhoewel die metodologie in hierdie tesis spesifiek vir die versekeringsbedryf ontwikkel is, kan dit in die normale konteks van navorsing en wetenskaplike besluitneming, wat byvoorbeeld oorlewingsverdelings vir die mediese, biologiese, ingenieurswese, ekonometriese en sosiologiese wetenskappe insluit, toegepas word.