Appendix C

Computing codes

This appendix provides codes for several models presented in this textbook. The codes are provided in SAS, R, and/or WinBUGS. The codes include the following.

Negative binomial model

SAS code

```
PROC GENMOD DATA=texasdata;
MODEL crashes=logaadt/DIST=NEGBIN OFFSET=loglength
LINK=LOG;
RUN;
```

R code

```
dat<-read.csv('texasdata.csv')
summary(dat)
glm.nb(crashes~logaadt+offset(log(length)),data=dat)
```

WinBUGS code

```
model
{
for(i in 1:N) {
crashes[i] ~ dnegbin(p[i],r)
   p[i] <- r/(r+mu[i])
mu[i] <-exp(beta0 +beta1*logaadt[i]+loglength[i])
}
beta0~dnorm(0.0,0.001)
beta1~dnorm(0.0,0.001)
r~dgamma(0.001,0.001)
}
INITIALIZATION of parameters
```

```
list(
beta0=-10,
beta1=0,
r=1,
)
Alternative specification:
model
{
for(i in 1:N) {
crashes[i] ~ dpois(mu[i])
mu[i] <- exp(beta0 +beta1*logaadt[i]+loglength[i])*e[i]
}
for (i in 1:N) { e[i] ~ dgamma(phi, phi) }
beta0~dnorm(0.0,0.001)
beta1~dnorm(0.0,0.001)
phi~dgamma(0.001,0.001)
}
INITIALIZATION of parameters
list(
beta0=-10,
beta1=0,
phi=1,
)
```

Negative binomial model with varying dispersion parameter

SAS code

```
PROC NLMIXED DATA=texasdata COVB;
PARMS beta0 1 beta1 1 rho 1;
mu = exp(loglength + beta0 + beta1*logaadt);
eta_p = rho+loglength;
phi = exp(eta_p);
alpha=1/phi;
loglike = (lgamma(crashes+(1/alpha)) - lgamma(crashes+1) -
lgamma(1/alpha) + crashes*log(alpha*mu) - (crashes+(1/alpha))
*log(1+alpha*mu));
MODEL crashes ~ general(loglike);
RUN;
```

WinBUGS code

```
model
{
for(i in 1:N) {
Crashes[i] ~ dpois(mu[i])
mu[i] <- exp(beta0 +beta1*logaadt[i]+loglength[i])*e[i]
}
for (i in 1:N) { e[i] ~ dgamma(phi[i], phi[i]) }
phi[i] <-exp(rho+loglength[i])
beta0~dnorm(0.0,0.001)
beta1~dnorm(0.0,0.001)
rho~dgamma(0.001,0.001)
}
INITIALIZATION for parameters
list(
beta0=-10,
beta1=0,
rho=1,
)
```

Radom effects negative binomial model

SAS code

```
PROC NLMIXED DATA=texasdata COVB;
PARMS beta0 1 beta1 1 alpha 1 s2u 0;
mu = exp(loglength + beta0 + beta1*logaadt + u);
loglike = (lgamma(crashes+(1/alpha)) - lgamma(crashes+1) -
lgamma(1/alpha) + crashes*log(alpha*mu) - (crashes+(1/alpha))
*log(1+alpha*mu));
MODEL crashes ~ general(loglike);
RANDOM u ~ normal(0,s2u) subject=subject;
RUN;
```

WinBUGS code

```
model
{
for(i in 1:N) {
Crashes[i] ~ dpois(mu[i])
mu[i] <- exp(beta0 +beta1*logaadt[i]+loglength[i]+u[i])*e[i]
```

```
}
for (i in 1:N) { e[i] ~ dgamma(phi, phi)
        u[i] ~ dnorm(u0, v0)}
beta0~dnorm(0.0,0.001)
beta1~dnorm(0.0,0.001)
u0~dnorm(0.0,0.001)
phi~dgamma(0.001,0.001)
v0~dgamma(0.001,0.001)
}
INITIALIZATION for parameters
list(
beta0=-10,
beta1=0,
u0=0,
phi=1,
v0=0
)
```

Random parameters negative binomial model

SAS code

```
PROC NLMIXED DATA=texasdata COVB;
PARMS beta0 1 beta1 1 alpha 1 s2u 0 s2v 0 cov 0;
mu = exp(loglength + beta0 + beta1*logaadt);
loglike = (lgamma(crashes+(1/alpha)) - lgamma(crashes+1) -
lgamma(1/alpha) +
      crashes*log(alpha*mu) - (crashes+(1/alpha))
*log(1+alpha*mu));
MODEL crashes ~ general(loglike);
RANDOM beta0 beta1 ~ normal([0,0],[s2u,cov,s2v]) subject=subject;
RUN;
```

WinBUGS code

```
model
{
for(i in 1:N) {
Crashes[i] ~ dpois(mu[i])
mu[i] <- exp(beta0[i] +beta1[i]*logaadt[i]+loglength[i])*e[i]
}
for (i in 1:N) {
        beta0[i] ~ dnorm(u0, v0)
```

```
            beta1[i] ~ dnorm(u1, v1)
            e[i] ~ dgamma(phi, phi)
            }
u0~dnorm(0.0,0.001)
u1~dnorm(0.0,0.001)
phi~dgamma(0.001,0.001)
v0~dgamma(0.001,0.001)
v1~dgamma(0.001,0.001)
}
INITIALIZATION for parameters
list(
u0=-10,
u1=0,
v0=1,
v1=1,
phi=1,
)
```

Poisson-lognormal model

WinBUGS code

```
model
{
for( i in 1 : N) {
crashes[i] ~ dpois(mu[i])
mu[i] <- exp(beta0 + beta1 * logaadt[i] + lambda[i])
lambda[i] ~ dnorm(0.0, tau)
}
alpha ~ dnorm(0.0,0.001)
beta ~ dnorm(0.0,0.001)
tau<-1/(sigma*sigma)
   sigma~ dunif(0,100)
}
```

Alternative specification:

```
model
{
for( i in 1 : N) {
crashes[i] ~ dpois(mu[i])
log(mu[i]) <- alpha + beta * x[i] + lambda[i]
lambda[i] ~ dnorm(0.0, tau)
}
alpha ~ dnorm(0.0,0.001)
```

```
beta ~ dnorm(0.0,0.001)
   tau <- 1/(sigma*sigma)
sigma ~ dgamma(0.001, 0.001)
}
```

Negative binomial-Lindley model

WinBUGS code

```
model
{
for(i in 1:N) {
crashes[i] ~ dnegbin(p[i],r)
p[i] <- r/(r+a[i]*mu[i])
mu[i] <-exp(beta0 +beta1*logaadt[i]+loglength[i])
a[i]~dgamma(f[i],t)
f[i]<-1+z[i]
z[i]~dbern(k)
}
r<-1/alpha
t<-(1-k)/k
beta0~dnorm(0.0,0.001)
beta1~dnorm(0.0,0.001)
alpha~dunif(0.1,10)
k~dbeta(N/3,N/2)
}
INITIALIZATION of parameters
list(
beta0=-10,
beta1=0,
alpha=0.1,
k=0.5
)
```

Conway–Maxwell–Poisson distribution

R code

The complete code is located at:
https://cran.r-project.org/web/packages/COMPoissonReg/COMPoissonReg.pdf

Multinomial logit model

SAS code

```
PROC NLMIXED DATA=crashdata;
PARMS k0=0 a0=0 b0=0 c0=0
      k1=0 a1=0 b1=0 c1=0
      k2=0 a2=0 b2=0 c2=0
      k3=0 a3=0 b3=0 c3=0
      k4=0 a4=0 b4=0 c4=0 ;
  k= exp(k0 + k1*variable1 + k2* variable2 + k3* variable3 + k4*
variable4 );
  a= exp(a0 + a1* variable1 + a2* variable2 + a3* variable3 + a4*
variable4 );
  b= exp(b0 + b1* variable1+ b2* variable2 + b3* variable3 + b4*
variable4 );
  c= exp(c0 + c1* variable1 + c2* variable2 + c3* variable3 + c4*
variable4 );
  denominator= 1 + k + a + b + c;
  probability1 = k / denominator;
  probability2 = a / denominator;
  probability3 = b / denominator;
  probability4 = c / denominator;
  probability5 = 1 / denominator;
  if severity=1 then loglike=log(probability1);
  if severity=2 then loglike=log(probability2);
  if severity=3 then loglike=log(probability3);
  if severity=4 then loglike=log(probability4);
  if severity=5 then loglike=log(probability5);
  model severity ~ general(loglike);
RUN;
```

Nested logit model

SAS code

This is a 2-level model. Level 1 at the bottom has alternatives that are nested at level 2 in two nests (first nest has three alternatives and second nest has two alternatives). The nests are joined at the top of the tree.

```
PROC MDC DATA=crashdata COVEST=hess;
MODEL decision = variable1 variable2 variable3 variable4/
        TYPE=nlogit
        CHOICE=(alt);
```

```
   ID id;
UTILITY u(1, ) = variable1 variable2 variable3,
         u(2, 1 2@1) = variable4;
NEST level(1) = (1 2 3@ 1, 4 5 @ 2),
      level(2) = (1 2 @ 1);
   RUN;
```