1 Time to Second Birth.

We obtain the following result shown in Figure 1 by running OpenBUGS code. The OpenBUGS code is attached in Appendix A.

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
b0	1009.0	25.27	2.008	958.0	1011.0	1055.0	1001	10000
b1	-1.192	1.086	0.08641	-3.161	-1.276	1.006	1001	10000
b2	-171.2	23.44	0.2182	-217.0	-171.3	-125.1	1001	10000
time.emma	799.6	430.5	4.604	-34.61	798.3	1659.0	1001	10000
time.helga	978.6	424.8	4.075	139.7	976.6	1799.0	1001	10000

Figure 1: OpenBUGS result for problem 1

(a) The mean of β_2 is -171.2 and its 95% credible set is [-217.0, -125.1]. Variable `death` is significant since the 95% credible set of β_2 does not contain 0.

(b) The mean of β_1 is -1.192 and its 95% credible set is [-3.161, 1.006]. Variable `mage` is not significant in influencing the response `time` since the 95% credible set of β_1 contains 0.

(c) The predicted time between the births of Helga is 978.6 days.

(d) The 95% credible set for the predicted time between births of Emma is [-34.61, 1659.0].

2 Tasmanian Clouds.

(a) According to the ANOVA analysis with main effects only, Spring, Summer, and Winter are significant. The result is shown in Figure 2.

(b) According to the ANOVA analysis with main effects and two interactions, Spring, Summer, Winter, S & Autumn and U & Autumn are significant. The result is shown in Figure 3.

3 Miller Lumber Company Customer Survey.

(a) We propose a Poisson model with `hunits`, `aveinc`, `aveage`, `distcomp`, and `diststore` as covariates and `customers` as response. The OpenBUGS code is shown in Appendix C. Result for the coefficients of the proposed Poisson model is shown in Figure 4.

Node statistics

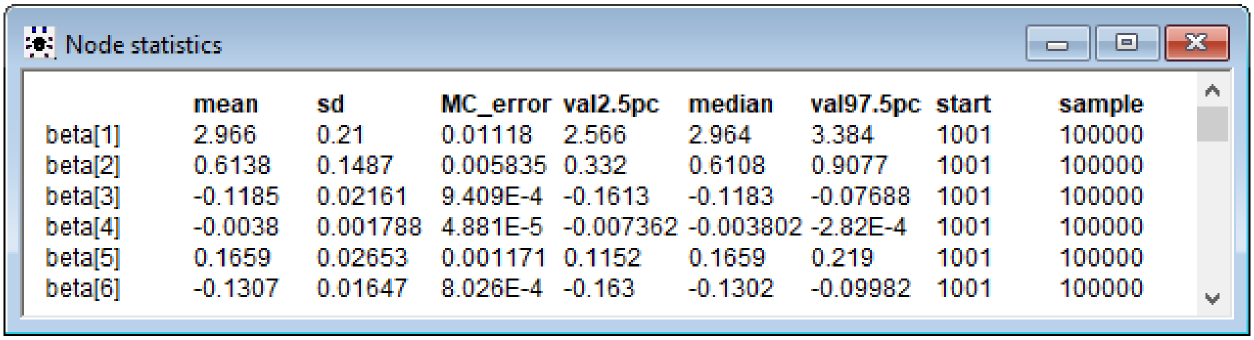
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha[1]	0.06868	0.04678	1.408E-4	-0.02326	0.0686	0.1606	1001	100000
alpha[2]	-0.06868	0.04678	1.408E-4	-0.1606	-0.0686	0.02328	1001	100000
beta[1]	0.1579	0.07651	1.625E-4	0.008656	0.1579	0.3088	1001	100000
beta[2]	0.1776	0.08406	3.933E-4	0.01295	0.1771	0.3447	1001	100000
beta[3]	0.0128	0.084	3.817E-4	-0.153	0.01292	0.1776	1001	100000
beta[4]	-0.3483	0.07995	3.512E-4	-0.5054	-0.3481	-0.1913	1001	100000
ca[1,2]	0.1374	0.09356	2.816E-4	-0.04653	0.1372	0.3212	1001	100000
cb[1,2]	-0.01971	0.1306	4.526E-4	-0.2766	-0.01936	0.2375	1001	100000
cb[1,3]	0.1451	0.1304	4.377E-4	-0.1106	0.1445	0.4028	1001	100000
cb[1,4]	0.5062	0.1254	3.992E-4	0.2613	0.5062	0.7529	1001	100000
cb[2,3]	0.1648	0.1396	6.787E-4	-0.1076	0.1641	0.4404	1001	100000
cb[2,4]	0.5259	0.1346	6.352E-4	0.2624	0.525	0.7916	1001	100000
cb[3,4]	0.3611	0.1346	6.171E-4	0.096	0.3613	0.6259	1001	100000

Figure 2: OpenBUGS result for 2(a)

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha[1]	0.07179	0.04631	1.498E-4	-0.01903	0.07184	0.1626	1001	100000
alpha[2]	-0.07179	0.04631	1.498E-4	-0.1626	-0.07184	0.01903	1001	100000
alpha.beta[1,1]	-0.07157	0.07532	1.684E-4	-0.2192	-0.07127	0.07637	1001	100000
alpha.beta[1,2]	-0.1513	0.08299	3.681E-4	-0.3148	-0.1511	0.01146	1001	100000
alpha.beta[1,3]	0.1636	0.08289	3.636E-4	4.666E-4	0.1637	0.3263	1001	100000
alpha.beta[1,4]	0.05922	0.07843	3.451E-4	-0.09549	0.05925	0.2127	1001	100000
alpha.beta[2,1]	0.07157	0.07532	1.684E-4	-0.07636	0.07127	0.2192	1001	100000
alpha.beta[2,2]	0.1513	0.08299	3.681E-4	-0.01146	0.1511	0.3148	1001	100000
alpha.beta[2,3]	-0.1636	0.08289	3.636E-4	-0.3263	-0.1637	-4.588E-4	1001	100000
alpha.beta[2,4]	-0.05922	0.07843	3.451E-4	-0.2127	-0.05925	0.0955	1001	100000
beta[1]	0.158	0.07539	1.627E-4	0.01087	0.1577	0.3066	1001	100000
beta[2]	0.1776	0.08317	3.696E-4	0.01469	0.1777	0.3415	1001	100000
beta[3]	0.01303	0.08277	3.707E-4	-0.1488	0.01282	0.1768	1001	100000
beta[4]	-0.3486	0.07877	3.283E-4	-0.5027	-0.3486	-0.1934	1001	100000
ca[1,2]	0.1436	0.09262	2.997E-4	-0.03806	0.1437	0.3252	1001	100000
cb[1,2]	-0.01958	0.129	4.334E-4	-0.2723	-0.01971	0.2323	1001	100000
cb[1,3]	0.145	0.1286	4.313E-4	-0.1087	0.1449	0.3965	1001	100000
cb[1,4]	0.5066	0.1234	3.746E-4	0.2649	0.5058	0.7499	1001	100000
cb[2,3]	0.1645	0.1378	6.479E-4	-0.1067	0.1649	0.4344	1001	100000
cb[2,4]	0.5261	0.1331	5.891E-4	0.2646	0.5263	0.7877	1001	100000
cb[3,4]	0.3616	0.1324	5.934E-4	0.1015	0.3614	0.623	1001	100000

Figure 3: OpenBUGS result for 2(b)



Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
beta[1]	2.966	0.21	0.01118	2.566	2.964	3.384	1001	100000
beta[2]	0.6138	0.1487	0.005835	0.332	0.6108	0.9077	1001	100000
beta[3]	-0.1185	0.02161	9.409E-4	-0.1613	-0.1183	-0.07688	1001	100000
beta[4]	-0.0038	0.001788	4.881E-5	-0.007362	-0.003802	-2.82E-4	1001	100000
beta[5]	0.1659	0.02653	0.001171	0.1152	0.1659	0.219	1001	100000
beta[6]	-0.1307	0.01647	8.026E-4	-0.163	-0.1302	-0.09982	1001	100000

Figure 4: OpenBUGS result for coefficients of Poisson model

(b) We use Laud-Ibrahim critierion to decide on the best two covariates. We obtain the following results shown in Figure 5. As we prefer one model that with lower Laud-Ibrahim value compared with that with higher Laud-Ibrahim value, we prefer the model 10, which means that we consider covariates `distcomp` and `diststore` as the best two covariates.

(c) By fixing `hunits=720`, `aveinc=70000`, `aveage=6`, `distcomp=4`, and `diststore=8`, we obtain the results shown in Figure We found that the mean response is 8.958 and its 95% credible set is [7.719,10.34]. The predictive response is 8.957 and its 95% credible set is [4.0, 15.0].

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Comp[1,2]	0.5032	0.5	0.001511	0.0	1.0	1.0	1001	100000
Comp[1,3]	4.0E-4	0.02	6.244E-5	0.0	0.0	0.0	1001	100000
Comp[1,4]	0.0	0.0	3.162E-13	0.0	0.0	0.0	1001	100000
Comp[1,5]	0.6426	0.4792	0.001635	0.0	1.0	1.0	1001	100000
Comp[1,6]	8.5E-4	0.02914	9.293E-5	0.0	0.0	0.0	1001	100000
Comp[1,7]	1.0E-5	0.003162	9.976E-6	0.0	0.0	0.0	1001	100000
Comp[1,8]	5.1E-4	0.02258	6.844E-5	0.0	0.0	0.0	1001	100000
Comp[1,9]	4.0E-5	0.006324	1.99E-5	0.0	0.0	0.0	1001	100000
Comp[1,10]	1.0E-5	0.003162	1.001E-5	0.0	0.0	0.0	1001	100000
Comp[2,3]	2.8E-4	0.01673	5.254E-5	0.0	0.0	0.0	1001	100000
Comp[2,4]	1.0E-5	0.003162	1.001E-5	0.0	0.0	0.0	1001	100000
Comp[2,5]	0.6397	0.4801	0.001715	0.0	1.0	1.0	1001	100000
Comp[2,6]	8.5E-4	0.02914	9.178E-5	0.0	0.0	0.0	1001	100000
Comp[2,7]	0.0	0.0	3.162E-13	0.0	0.0	0.0	1001	100000
Comp[2,8]	5.9E-4	0.02428	7.491E-5	0.0	0.0	0.0	1001	100000
Comp[2,9]	1.0E-5	0.003162	1.001E-5	0.0	0.0	0.0	1001	100000
Comp[2,10]	0.0	0.0	3.162E-13	0.0	0.0	0.0	1001	100000
Comp[3,4]	0.2194	0.4139	0.001639	0.0	0.0	1.0	1001	100000
Comp[3,5]	0.9999	0.008944	2.796E-5	1.0	1.0	1.0	1001	100000
Comp[3,6]	0.5639	0.4959	0.001971	0.0	1.0	1.0	1001	100000
Comp[3,7]	0.2078	0.4057	0.001667	0.0	0.0	1.0	1001	100000
Comp[3,8]	0.5505	0.4974	0.001663	0.0	1.0	1.0	1001	100000
Comp[3,9]	0.2317	0.4219	0.001567	0.0	0.0	1.0	1001	100000
Comp[3,10]	0.1519	0.3589	0.001519	0.0	0.0	1.0	1001	100000
Comp[4,5]	1.0	0.003162	9.976E-6	1.0	1.0	1.0	1001	100000
Comp[4,6]	0.8264	0.3787	0.001485	0.0	1.0	1.0	1001	100000
Comp[4,7]	0.4805	0.4996	0.002191	0.0	0.0	1.0	1001	100000
Comp[4,8]	0.8157	0.3877	0.001481	0.0	1.0	1.0	1001	100000
Comp[4,9]	0.5206	0.4996	0.002056	0.0	1.0	1.0	1001	100000
Comp[4,10]	0.3829	0.4861	0.00216	0.0	0.0	1.0	1001	100000
Comp[5,6]	1.6E-4	0.01265	4.151E-5	0.0	0.0	0.0	1001	100000
Comp[5,7]	0.0	0.0	3.162E-13	0.0	0.0	0.0	1001	100000
Comp[5,8]	1.7E-4	0.01304	4.255E-5	0.0	0.0	0.0	1001	100000
Comp[5,9]	0.0	0.0	3.162E-13	0.0	0.0	0.0	1001	100000
Comp[5,10]	0.0	0.0	3.162E-13	0.0	0.0	0.0	1001	100000
Comp[6,7]	0.1636	0.3699	0.001539	0.0	0.0	1.0	1001	100000
Comp[6,8]	0.4843	0.4998	0.002006	0.0	0.0	1.0	1001	100000
Comp[6,9]	0.1851	0.3884	0.001525	0.0	0.0	1.0	1001	100000
Comp[6,10]	0.118	0.3226	0.001329	0.0	0.0	1.0	1001	100000
Comp[7,8]	0.8285	0.377	0.001547	0.0	1.0	1.0	1001	100000
Comp[7,9]	0.5386	0.4985	0.002175	0.0	1.0	1.0	1001	100000
Comp[7,10]	0.4015	0.4902	0.002201	0.0	0.0	1.0	1001	100000
Comp[8,9]	0.1955	0.3966	0.001487	0.0	0.0	1.0	1001	100000
Comp[8,10]	0.1241	0.3297	0.001316	0.0	0.0	1.0	1001	100000
Comp[9,10]	0.3652	0.4815	0.00204	0.0	0.0	1.0	1001	100000

Figure 5: OpenBUGS result for model comparison

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
beta[1]	2.966	0.21	0.01118	2.566	2.964	3.384	1001	100000
beta[2]	0.6138	0.1487	0.005835	0.332	0.6108	0.9077	1001	100000
beta[3]	-0.1185	0.02161	9.409E-4	-0.1613	-0.1183	-0.07688	1001	100000
beta[4]	-0.0038	0.001788	4.881E-5	-0.007362	-0.003802	-2.82E-4	1001	100000
beta[5]	0.1659	0.02653	0.001171	0.1152	0.1659	0.219	1001	100000
beta[6]	-0.1307	0.01647	8.026E-4	-0.163	-0.1302	-0.09982	1001	100000
deviance	565.2	3.527	0.1101	560.3	564.5	573.8	1001	100000
lambdastar	8.958	0.6681	0.02432	7.719	8.929	10.34	1001	100000
ystar	8.957	3.064	0.02564	4.0	9.0	15.0	1001	100000

Figure 6: OpenBUGS result for mean and predictive response

A OpenBUGS Code for Problem 1

```
model{
for (i in 1:N){
 time[i] ~ dnorm(mu[i], tau)
 mu[i] <- b0 + b1* mage[i] + b2*death[i]
}
b0 ~ dnorm(0, 0.001)
b1 ~ dnorm(0, 0.001)
b2 ~ dnorm(0, 0.001)
tau ~ dgamma(0.001, 0.001)

# prediction for Helga
mage.helga <- 24
death.helga <- 0
mu.helga <- b0 + b1*mage.helga + b2*death.helga
time.helga ~ dnorm(mu.helga, tau)

# prediction for Emma
mage.emma <- 28
death.emma <- 1
mu.emma <- b0 + b1*mage.emma + b2*death.emma
time.emma ~ dnorm(mu.emma, tau)
}

DATA
list(N=16341)
```

```
DATA(mage, death, and time)

INITS
list(b0=1,  b1=0,  b2=0, tau=1)
```

B OpenBUGS Code for Problem 2

```
model{
   for(i in 1:n){
   DIFF[i]  ~ dnorm( mu[i], tau )
    mu[i] <-  mu0 + alpha[Seeded[i]]  + beta[Season[i]] + alpha.beta[ Seeded[i], Season[
    }
#CR (corner) constraints
  # alpha[1] <- 0;
  # beta[1]<- 0;
  # alpha.beta[1,1]<- 0;
  # for( a in 2:leva) {alpha.beta[a,1]<- 0}
  # for(b in 2:levb)  {alpha.beta[1,b]<- 0}

##STZ (sum-to-zero) constraints
   alpha[1] <- - sum(alpha[2:leva])
   beta[1] <- - sum(beta[2:levb])
 for(a in 1:leva) {alpha.beta[a,1] <- - sum(alpha.beta[a, 2:levb])}
   for(b in  2:levb) {alpha.beta[1,b] <-  - sum(alpha.beta[2:leva, b])}


#PRIORS
mu0 ~ dnorm(0, 0.0001)
for(a in 2:leva) {alpha[a] ~ dnorm(0, 0.0001)}
for(b in 2:levb) {beta[b] ~ dnorm(0, 0.0001)}
for(a in 2:leva) {for(b in 2:levb){
      alpha.beta[a,b] ~ dnorm(0, 0.0001) }}
tau ~ dgamma(0.001, 0.001)
s <- 1/sqrt(tau)

#PAIRWISE COMPARISONS

for(i in 1:1)  {for(j in i+1:2) {ca[i,j] <- alpha[i]-alpha[j]}}
for(i in 1:3)  {for(j in i+1:4) {cb[i,j] <- beta[i]-beta[j]}}

}
```

```
DATA 1
list(n =108, leva= 2, levb= 4, Seeded=c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
DIFF = c(0.45,0.182,0.66,0.053,-0.058,0.233,-0.008,-0.227,0.053,2.187,0.058,-0.523,0.83,
)

INITS
list(mu0=0, alpha=c(NA,0), beta=c(NA,0,0,0), alpha.beta = structure(.Data=c(NA, NA, NA,
.Dim=c(2,4)), tau = 1)
```

C OpenBUGS Code for Problem 3

OpenBUGS code for Poisson model.

```
    model{

for (i in 1:N) {
hunits0[i] <- hunits[i]/1000
aveinc0[i] <- aveinc[i]/10000

customers[i] ~ dpois(lambda[i])
lambda[i] <- exp(beta[1]+beta[2]*hunits0[i]+beta[3]*aveinc0[i]+beta[4]*aveage[i]
+beta[5]*distcomp[i]+beta[6]*diststore[i])
}

for (j in 1:6) {
beta[j] ~ dnorm(0, 0.0001)
}

hunits.star <- 720/1000
aveinc.star <- 70000/10000
aveage.star <- 6
distcomp.star <- 4.1
diststore.star <- 8

# mean response
lambdastar <- exp(beta[1]+beta[2]*hunits.star+beta[3]*aveinc.star+beta[4]*aveage.star
+beta[5]*distcomp.star+beta[6]*diststore.star)

# predictive response
ystar ~ dpois(lambdastar)
```

```
}
```

OpenBUGS code for finding the best two covariates.

```
    model{

for (i in 1:N) {
hunits0[i] <- hunits[i]/1000
aveinc0[i] <- aveinc[i]/10000

# ten competing models
lambda[1, i] <- exp(a[1]+a[2]*hunits0[i]+a[3]*aveinc0[i])
lambda[2, i] <- exp(b[1]+b[2]*hunits0[i]+b[3]*aveage[i])
lambda[3, i] <- exp(c[1]+c[2]*hunits0[i]+c[3]*distcomp[i])
lambda[4, i] <- exp(d[1]+d[2]*hunits0[i]+d[3]*diststore[i])
lambda[5, i] <- exp(e[1]+e[2]*aveinc0[i]+e[3]*aveage[i])
lambda[6, i] <- exp(f[1]+f[2]*aveinc0[i]+f[3]*distcomp[i])
lambda[7, i] <- exp(g[1]+g[2]*aveinc0[i]+g[3]*diststore[i])
lambda[8, i] <- exp(h[1]+h[2]*aveage[i]+h[3]*distcomp[i])
lambda[9, i] <- exp(k[1]+k[2]*aveage[i]+k[3]*diststore[i])
lambda[10, i] <- exp(m[1]+m[2]*distcomp[i]+m[3]*diststore[i])
}

# compare models
for (j in 1:10) {
L[j] <- sqrt(sum(D2[j, ])+pow(sd(Customer.new[j, ]), 2))

# datasets for different models
for (i in 1:N) {
Customer[j, i] <- customers[i]
Customer[j, i] ~ dpois(lambda[j, i])
D2[j, i] <- pow(customers[i]-Customer.new[j, i], 2)
Customer.new[j, i] ~ dpois(lambda[j, i])
}
}

for (i in 1:9) {
for (j in i+1:10) {
Comp[i, j] <- step(L[j]-L[i])
}
```

```
}

for (j in 1:3) {
a[j] ~ dnorm(0, 0.01)
b[j] ~ dnorm(0, 0.01)
c[j] ~ dnorm(0, 0.01)
d[j] ~ dnorm(0, 0.01)
e[j] ~ dnorm(0, 0.01)
f[j] ~ dnorm(0, 0.01)
g[j] ~ dnorm(0, 0.01)
h[j] ~ dnorm(0, 0.01)
k[j] ~ dnorm(0, 0.01)
m[j] ~ dnorm(0, 0.01)
}

}
```