1 Cancer of Tongue.

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

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
beta0	-4.58	0.6303	0.03011	-5.868	-4.56	-3.389	1001	100000
beta1	0.5317	0.2887	0.01166	-0.03808	0.5291	1.101	1001	100000

Figure 1: OpenBUGS result for problem 1

The mean value of β_1 is 0.5317 and the 95% credible set is [-0.03808, 1.101].

2 Airfreight Breakage with Missing Data.

We obtained the following results shown in Figure 2 by running OpenBUGS code. The OpenBUGS code is attached in Appendix B.

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
beta0	2.171	0.1414	0.002581	1.885	2.175	2.439	1001	100000
beta1	0.2884	0.07351	0.001364	0.1449	0.2879	0.4339	1001	100000
deviance	109.6	3.276	0.03274	105.5	108.7	117.7	1001	100000
lambdastar	28.33	5.563	0.08469	18.86	27.82	40.5	1001	100000
x[5]	1.272	0.7807	0.003044	0.0	1.0	3.0	1001	100000
x[14]	2.549	0.8278	0.003874	1.0	3.0	4.0	1001	100000
x[15]	0.1965	0.4331	0.002702	0.0	0.0	1.0	1001	100000
y[1]	15.67	4.179	0.01547	8.0	15.0	24.0	1001	100000
y[9]	11.76	3.59	0.01873	5.0	12.0	19.0	1001	100000
ystar	28.3	7.701	0.08733	15.0	28.0	45.0	1001	100000

Figure 2: OpenBUGS result for problem 2

(a) We fits the model with coefficients $\beta_0 = 2.171$ and $\beta_1 = 0.2884$. The deviance of the fitted model is 109.6.

(b) Given $X = 4$, the average number of packages that are expected to be broken is 28.33, with 95% credible set as [18.86, 40.5].

(c) Given $X = 4$, the predicted number of broken packages is 28.3, with 95% credible set as [15.0, 45.0]. The mean value of expected response and predicted response is very close, while predicted response has higher variance then that of expected response. This can also be seen from the fact that expected response has wider 95% credible set than predicted response.

(d) The estimates for unobserved X_5, X_{14} and X_{15} are 1.272, 2.549, and 0.1965, respectively. The estimates for unobserved Y_1 and Y_9 are 15.67 and 11.76, respectively.

A Code for Problem 1

```
model {

for (i in 1:N) {

time[i] ~ dweib(v, lambda[i])I(censor[i], )
lambda[i] <- exp(beta0+beta1*DNA[i])

}
beta0 ~ dnorm(0, 0.0001)
beta1 ~ dnorm(0, 0.0001)
v ~ dexp(0.001)

}

DATA
list(N = 80)


DNA[] time[] censor[]
1 1 0
1 3 0
1 3 0
1 4 0
1 10 0
1 13 0
1 13 0
1 16 0
1 16 0
1 24 0
1 26 0
1 27 0
1 28 0
1 30 0
1 30 0
1 32 0
1 41 0
1 51 0
1 65 0
1 67 0
1 70 0
```

```
1 72 0
1 73 0
1 77 0
1 91 0
1 93 0
1 96 0
1 100 0
1 104 0
1 157 0
1 167 0
1 NA 61
1 NA 74
1 NA 79
1 NA 80
1 NA 81
1 NA 87
1 NA 87
1 NA 88
1 NA 89
1 NA 93
1 NA 97
1 NA 101
1 NA 104
1 NA 108
1 NA 109
1 NA 120
1 NA 131
1 NA 150
1 NA 231
1 NA 240
1 NA 400
2 1 0
2 3 0
2 4 0
2 5 0
2 5 0
2 8 0
2 12 0
2 13 0
2 18 0
2 23 0
2 26 0
```

```
2 27 0
2 30 0
2 42 0
2 56 0
2 62 0
2 69 0
2 104 0
2 104 0
2 112 0
2 129 0
2 181 0
2 NA 8
2 NA 67
2 NA 76
2 NA 104
2 NA 176
2 NA 231

END


INITS
list(v=1, beta0=0, beta1=0)
```

B Code for Problem 2

```
model {

for (i in 1:n) {
y[i] ~ dpois(lambda[i])
lambda[i] <- exp(beta0 + beta1*x[i])
x[i] ~ dpois(2)
}

beta0 ~ dnorm(0, 0.0001)
beta1 ~ dnorm(0, 0.0001)

xstar <- 4
# average broken
lambdastar <- exp(beta0 + beta1*xstar)
# predicted broken
```

```
ystar ~ dpois(lambdastar)
}

DATA
list(n=15)

x[] y[]
2 NA
1 16
0 9
2 17
NA 12
3 22
1 13
0 8
1 NA
2 19
3 17
0 11
1 10
NA 20
NA 2
END

INITS
list(beta0=0, beta1=0)
```