According to one version of Moore's law, the number of transistors on a state-of-the-art computer microprocessor roughly doubles every two years:

$$C \approx \gamma 2^{A/2}$$

where C is the transistor count of a microprocessor, A is the year number in which it was introduced, and γ is a positive constant.

(a) Mathematically manipulate Moore's law to show that $logC$ should roughly follow a simple linear regression on A. The coefficient of A is $\frac{log2}{2}$.

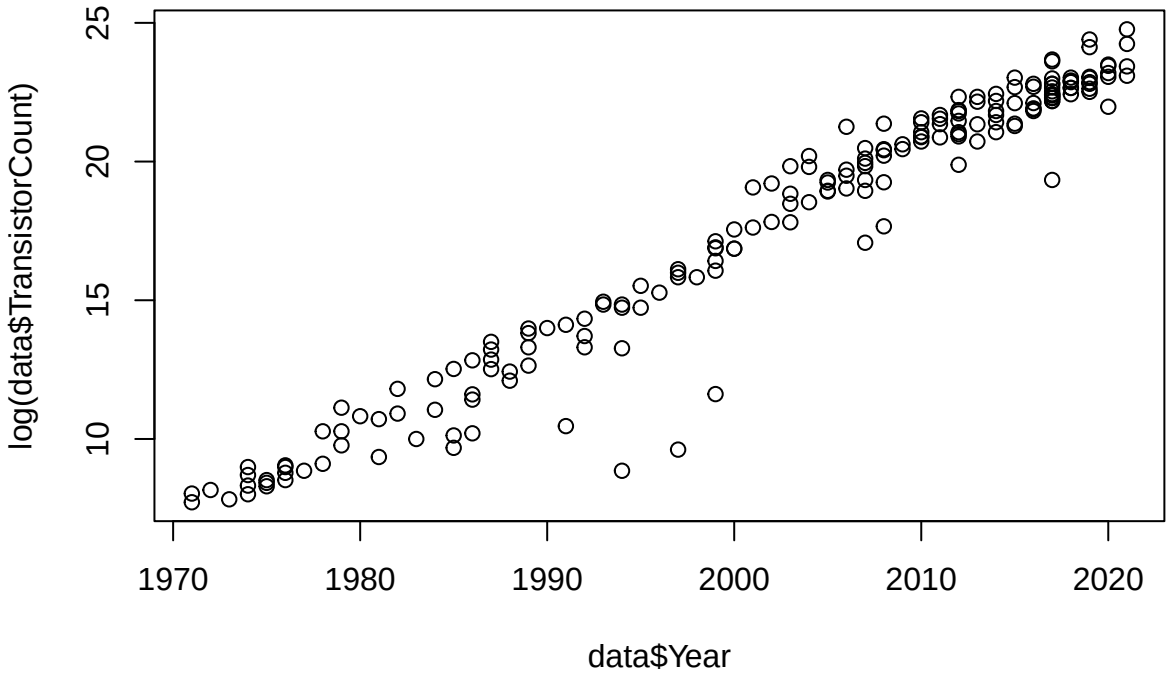
$$\begin{aligned} log(C) &\approx log(\gamma 2^{A/2}) \\ log(C) &\approx log(2^{A/2}) + log(\gamma) \\ log(C) &\approx \frac{log2}{2} A + log(\gamma) \end{aligned}$$

(b) Plot the data points as log transistor count versus year.

```
data<- read.csv("./mooreslawdata.csv", header=TRUE)
head(data, 10)
```

```
##              Processor Year TransistorCount
## 1           Intel 4004 1971            2250
## 2             TMX 1795 1971            3078
## 3           Intel 8008 1972            3500
## 4           NEC  COM-4 1973            2500
## 5           Intel 4040 1974            3000
## 6        Motorola 6800 1974            4100
## 7           Intel 8080 1974            6000
## 8             TMS 1000 1974            8000
## 9  MOS Technology 6502 1975            4528
## 10     Intersil IM6100 1975            4000
```

```
plot(data$Year, log(data$TransistorCount))
```



(c) For the given data, consider a normal-theory simple linear regression model of log transistor count on centered year of the form

$$\log C_i|\beta,\sigma^2,A_i \sim \text{indep. } N(\beta_1+\beta_2(A_i-\bar{A}),\sigma^2) \quad i=1,\ldots,190$$

where $\bar{A}$ is the average of A_i over all observations. Of course, Moore's law specifies a particular value for β_2, but your model will not assume this. Use independent priors

$$\beta_1,\beta_2 \sim \text{iid } N(0,1000^2)$$

$$\sigma^2 \sim \text{Inv-gamma}(0.001,0.001)$$

(i) List an appropriate JAGS model, and run the model by using multiple chains with overdispersed starting points, check convergence, and monitor β_1, β_2 and σ^2 for at least 2000 iterations (per chain) after burn-in. Based on the trace plot and Gelman-Rubin statistics, β_1, β_2 and σ^2 reached convergence.

```
'
model
{
  for (j in 1: length(TransistorCount)) {
    TransistorCount[j] ~ dnorm(beta1 + beta2*CenteredYear[j], sigmasqinv)}
  beta1 ~ dnorm(0, 0.000001)
  beta2 ~ dnorm(0, 0.000001)
  sigmasqinv ~ dgamma(0.001, 0.001)
  sigmasq <- 1/sigmasqinv
}
'
```

```
## [1] "\nmodel\n{\n  for (j in 1: length(TransistorCount)) {\n    TransistorCount[j] ~ dnorm(beta1 + b
```

```
d1 <- list(TransistorCount = log(data$TransistorCount),
           CenteredYear = data$Year - mean(data$Year))
#prepare chains for the model too
inits1 <- list(list(beta1=1000, beta2=1000, sigmasqinv=0.1),
```

```
              list(beta1=1000, beta2=1000, sigmasqinv=0.00001),
              list(beta1=1000, beta2=-1000, sigmasqinv=0.1),
              list(beta1=1000, beta2=-1000, sigmasqinv=0.00001))
library(rjags)
```

```
## Loading required package: coda
```

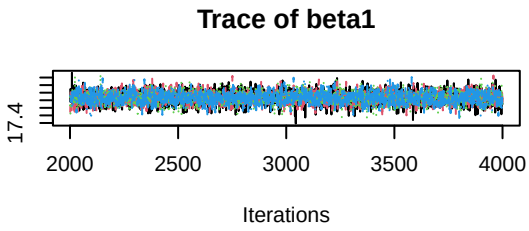
```
## Linked to JAGS 4.3.2
```

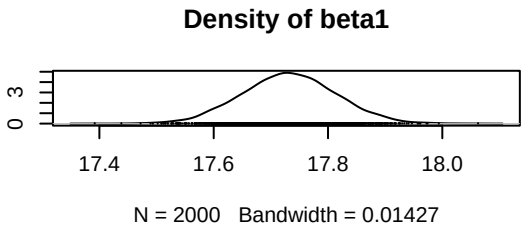
```
## Loaded modules: basemod,bugs
```

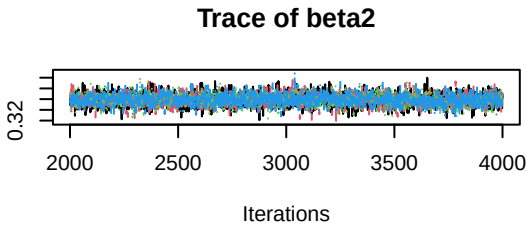
```
m1 <- jags.model("./moore.bug", d1, inits1, n.chains=4)
```

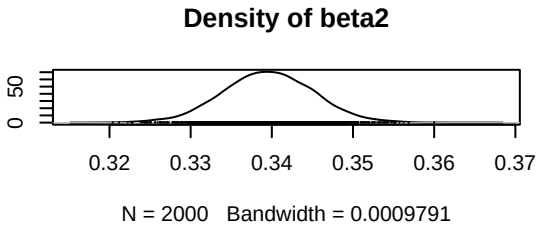
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 190
##    Unobserved stochastic nodes: 3
##    Total graph size: 490
##
## Initializing model
```

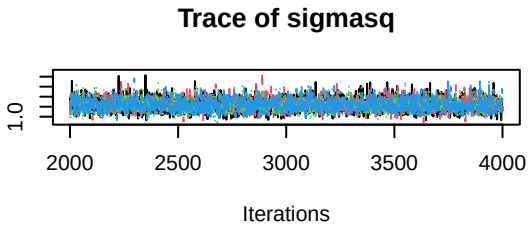
```
update(m1, 2000) #burn-in
x1 <- coda.samples(m1, c("beta1", "beta2", "sigmasq"), n.iter=2000)
plot(x1, smooth=FALSE)
```

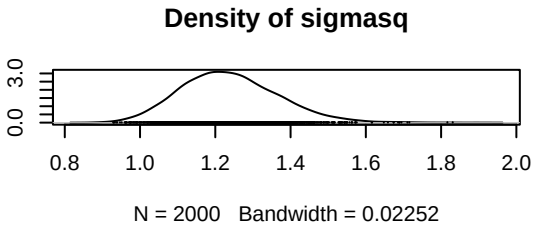










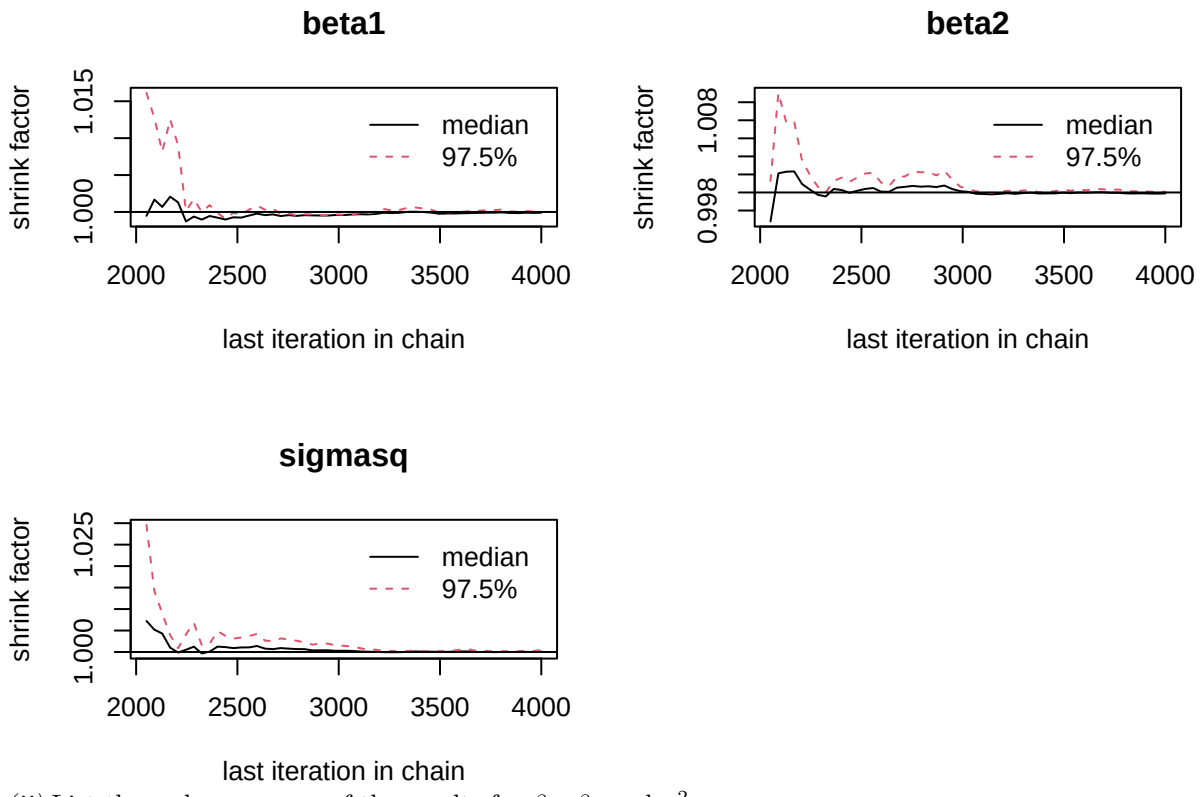



```
#Compute Gelman-Rubin statistics
gelman.diag(x1, autoburnin=FALSE)
```

```
## Potential scale reduction factors:
```

```
##
##         Point est. Upper C.I.
## beta1            1          1
## beta2            1          1
## sigmasq          1          1
##
## Multivariate psrf
##
## 1
```

```
gelman.plot(x1, autoburnin=FALSE)
```



(ii) List the coda summary of the results for β_1, β_2 and σ^2.

```
summary(x1)
```

```
##
## Iterations = 2001:4000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean       SD  Naive SE Time-series SE
## beta1   17.7319 0.082111 9.180e-04      9.529e-04
## beta2    0.3395 0.005595 6.256e-05      6.282e-05
## sigmasq  1.2378 0.128195 1.433e-03      1.431e-03
##
## 2. Quantiles for each variable:
```

```
##
##             2.5%     25%     50%     75%   97.5%
## beta1   17.5751 17.6771 17.7314 17.7860 17.8930
## beta2    0.3286  0.3358  0.3395  0.3433  0.3505
## sigmasq  1.0112  1.1462  1.2294  1.3190  1.5106
```

(iii) Give the approximate posterior mean for the slope is 0.3395, and 95% central posterior interval is (0.3286, 0.3505). The value determined in part(a) is $\frac{log(2)}{2} \approx 0.3466$. Thus the value is in accordance with Moore's law.

(iv) The approximate posterior mean for the intercept is 17.7319, and and 95% central posterior interval is (17.5751, 17.8930).

(d) Consider the model of the previous part. We will use it to predict the transistor count for a microprocessor introduced in 2024, and also to see if it extrapolates back to the invention of the transistor.
(i)

```
(PredCentered = 2024 - mean(data$Year))
```

```
## [1] 21.61579
```

```
(MeanYear = mean(data$Year))
```

```
## [1] 2002.384
```

```
'
model
{
  for (j in 1: length(TransistorCount)) {
    TransistorCount[j] ~ dnorm(beta1 + beta2*CenteredYear[j], sigmasqinv)}
  beta1 ~ dnorm(0, 0.000001)
  beta2 ~ dnorm(0, 0.000001)
  sigmasqinv ~ dgamma(0.001, 0.001)

  PredCentered <- 21.61579
  MeanYear <- 2002.384

  PredTransistorCount ~ dnorm(beta1 + beta2*PredCentered, sigmasqinv)
  TransistorInventionYear <- MeanYear - (beta1/beta2)

  sigmasq <- 1/sigmasqinv
}
'
```

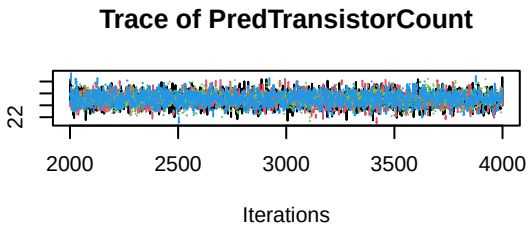
```
## [1] "\nmodel\n{\n  for (j in 1: length(TransistorCount)) {\n    TransistorCount[j] ~ dnorm(beta1 + b
```

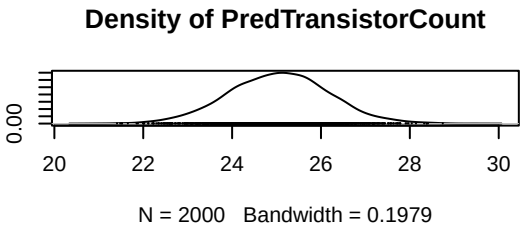
```
library(rjags)
m2 <- jags.model("./moore_1.bug", d1, inits1, n.chains=4)
```

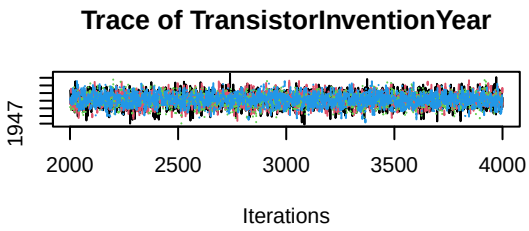
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 190
##    Unobserved stochastic nodes: 4
##    Total graph size: 497
```

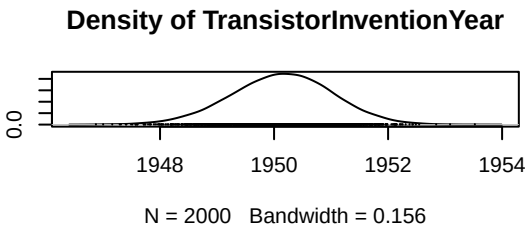
```
##
## Initializing model
```

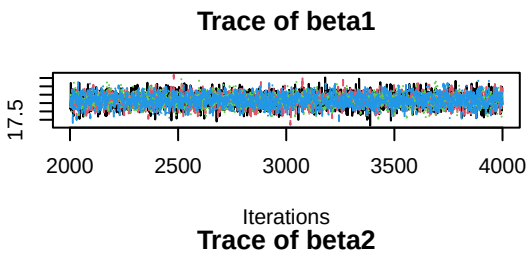
```
update(m2, 2000) #burn-in
x2 <- coda.samples(m2, c("beta1", "beta2", "sigmasq", "PredTransistorCount", "TransistorInventionYear")
plot(x2, smooth=FALSE)
```

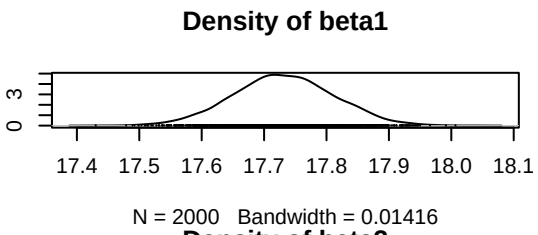


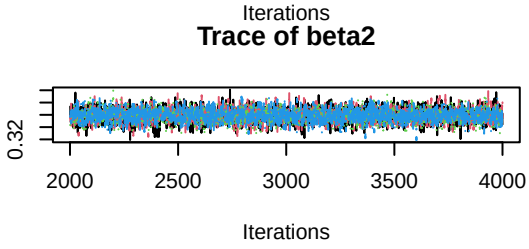


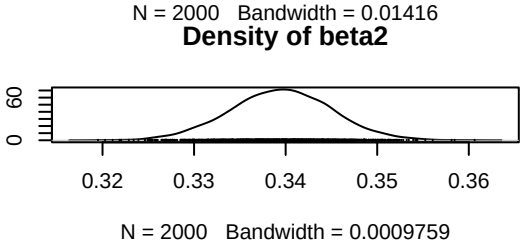


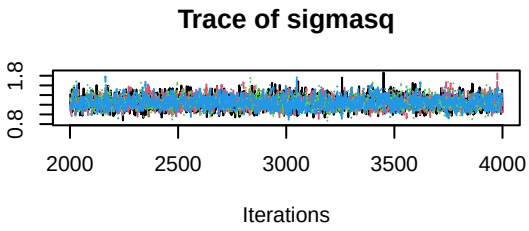


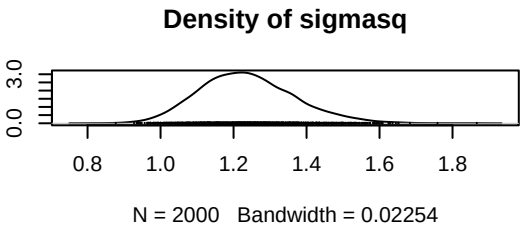










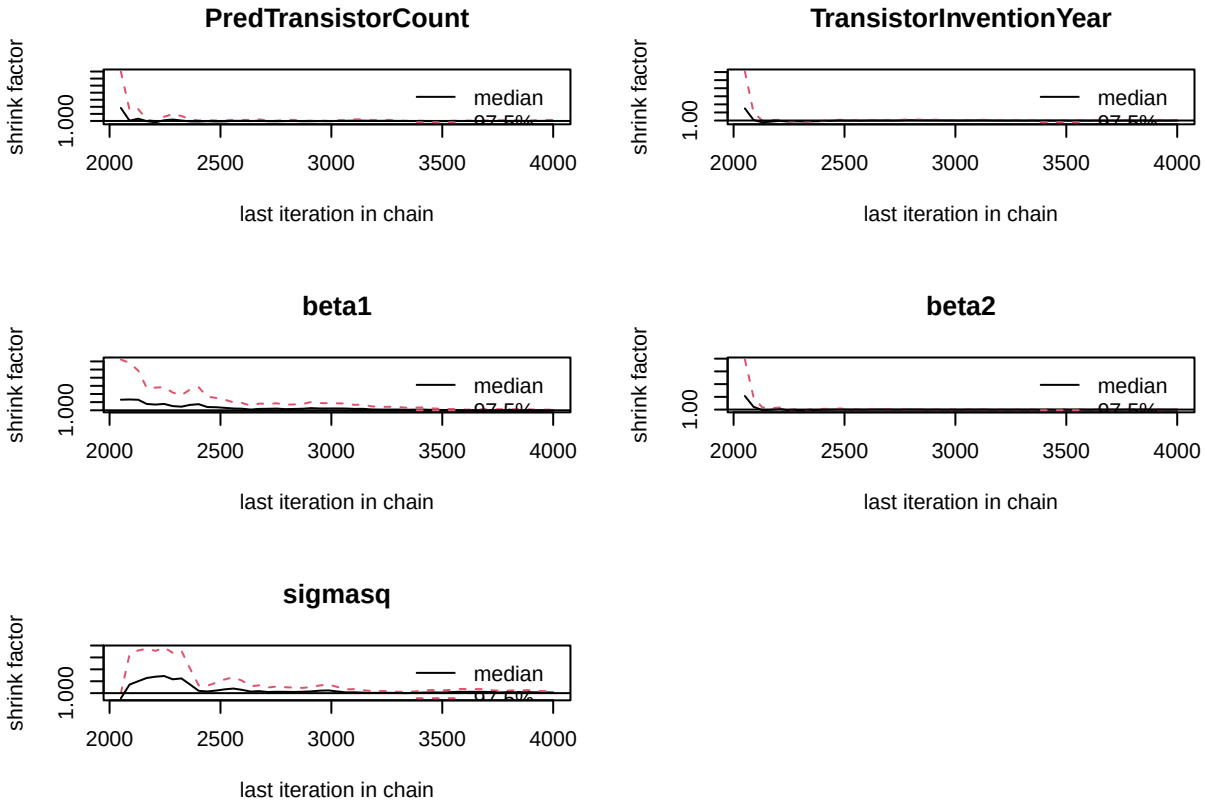



```
#Compute Gelman-Rubin statistics
gelman.diag(x2, autoburnin=FALSE)
```

```
## Potential scale reduction factors:
##
##                          Point est. Upper C.I.
## PredTransistorCount               1          1
```

```
## TransistorInventionYear          1          1
## beta1                            1          1
## beta2                            1          1
## sigmasq                          1          1
##
## Multivariate psrf
##
## 1
```

```
gelman.plot(x2, autoburnin=FALSE)
```



Based on the trace plot and Gelman-Rubin statistics, β_1, β_2, σ^2, PredTransistorCount and TransistorInventionYear reached convergence.

(ii)

```
summary(x2)
```

```
##
## Iterations = 2001:4000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                              Mean       SD  Naive SE Time-series SE
## PredTransistorCount       25.0519 1.126389 1.259e-02      1.248e-02
## TransistorInventionYear 1950.1382 0.888097 9.929e-03      1.002e-02
## beta1                     17.7314 0.080635 9.015e-04      8.914e-04
```

```
## beta2                       0.3395 0.005557 6.213e-05      6.289e-05
## sigmasq                     1.2374 0.129965 1.453e-03      1.453e-03
##
## 2. Quantiles for each variable:
##
##                                2.5%       25%       50%       75%     97.5%
## PredTransistorCount         22.8169   24.2847   25.0671   25.7973   27.2901
## TransistorInventionYear 1948.3531 1949.5507 1950.1533 1950.7474 1951.8442
## beta1                      17.5735   17.6772   17.7305   17.7853   17.8882
## beta2                       0.3286    0.3358    0.3395    0.3432    0.3504
## sigmasq                     1.0086    1.1467    1.2293    1.3186    1.5182
```

(iii) Given an approximate 95% central posterior predictive interval for the transistor count in billions for a microprocessor introduced in the year 2024. Based on the above summary, the interval is $(\frac{e^{22.82}}{10^9}, \frac{e^{27.29}}{10^9})$, which is (8.30, 732.70).

(iv) Explain mathematically why the regression model suggests that the transistor was invented in the year

$$\bar{A} - \beta_1/\beta_2.$$

Since

$$logC = \beta_1 + \beta_2(A_i - \bar{A})$$

when

$$C = 1, logC = 0, 0 = \beta_1 + \beta_2(A_i - \bar{A})$$

then

$$-\beta_1 = \beta_2(A_i - \bar{A})$$
$$A_i = \bar{A} - \beta_1/\beta_2$$

And the 95% central posterior interval for this quantity is (1948.3463, 1951.8453), and the actual year it was invented was 1947.

(e) One way to check for evidence of outliers in a regression is a posterior predictive p-value based on test quantity

$$T(y, X, \theta) = \max_i \left|\frac{\epsilon_i}{\sigma}\right|$$

where ϵ_i is the error for observation i. The larger T is (for the actual data), the more we should suspect the existence of at least one outlier.

(i) Show R code for computing simulated standardized error vectors $\frac{\epsilon}{\sigma}$ (as rows of a matrix).

```r
library(MASS)

mod <- lm(TransistorCount ~ CenteredYear, data=d1)
x <- model.matrix(mod)

# Gather posterior simulation
Nsim <- 2000
```

```
post.sigma.2.sim <- as.matrix(x1)[,"sigmasq"]
post.beta1.sim <- as.matrix(x1)[,"beta1"]
post.beta2.sim <- as.matrix(x1)[,"beta2"]

# Create the simulated standardized error vectors
error.std.sim <- matrix(NA, Nsim, nrow(data))

for (s in 1:Nsim) {
  error.std.sim[s,] <- (log(data$TransistorCount) - x %*% rbind(post.beta1.sim[s], post.beta2.sim[s]))
}
```

(ii) Show R code for computing simulated replicate standardized error vectors $\frac{\epsilon^{rep}}{\sigma}$ (as rows of a matrix), which are the standardized error vectors for the replicate response vectors y^{rep}.

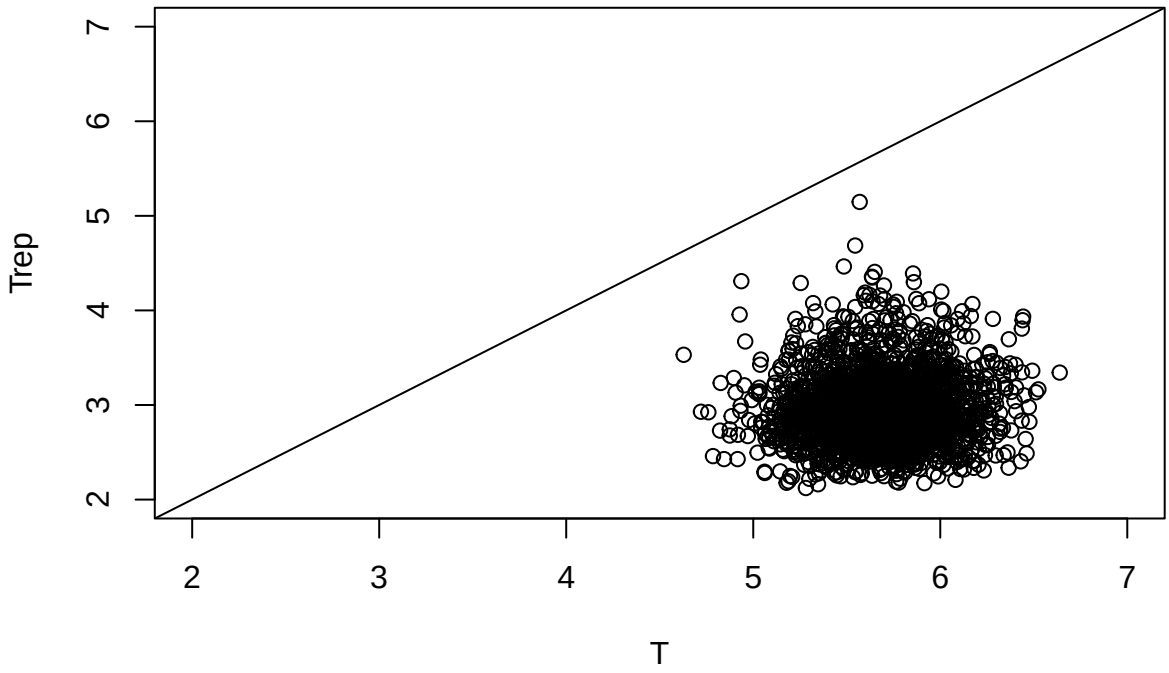
```
yreps <- matrix(NA, 2000, nrow(data))
for (s in 1:2000)
  yreps[s,] <- rnorm(nrow(data), x %*% rbind(post.beta1.sim[s], post.beta2.sim[s]),
                     sqrt(post.sigma.2.sim[s]))

error.std.sim.rep <- matrix(NA, Nsim, nrow(data))

for (s in 1:Nsim)
  error.std.sim.rep[s,] <- (yreps[s,] - x %*% rbind(post.beta1.sim[s],
                                                    post.beta2.sim[s]))/sqrt(post.sigma.2.sim[s])
```

(iii) Show R code for computing the simulated values of $T(y, X, \theta)$ and the simulated values of $T(y^{rep}, X, \theta)$.

```
T <- apply (abs(error.std.sim), 1, max)
Trep <- apply (abs(error.std.sim.rep), 1, max)
```

(iv) Plot the simulated values of $T(y^{rep}, X, \theta)$ versus those of $T(y, X, \theta)$, with a reference line indicating where $T(y^{rep}, X, \theta) = T(y, X, \theta)$.

```
plot(T,Trep, xlim=c(2,7), ylim=c(2,7))
abline(coef=c(0,1))
```



(v) Compute the approximate posterior predictive p-value, and we can see it is close to 1, which means the evidence of outliers.

```
mean(apply(yreps,1,max)>=max(d1$TransistorCount))
```

```
## [1] 0.974
```

(vi) F21 is the microprocessor that appears to be the most extreme outlier (for the log-scale counts).

```
diffs<-abs(apply(yreps,2,mean)-d1$TransistorCount)
data$Processor[which.max(diffs)]
```

```
## [1] "F21"
```