Project

Kevin Ros

2/22/2020

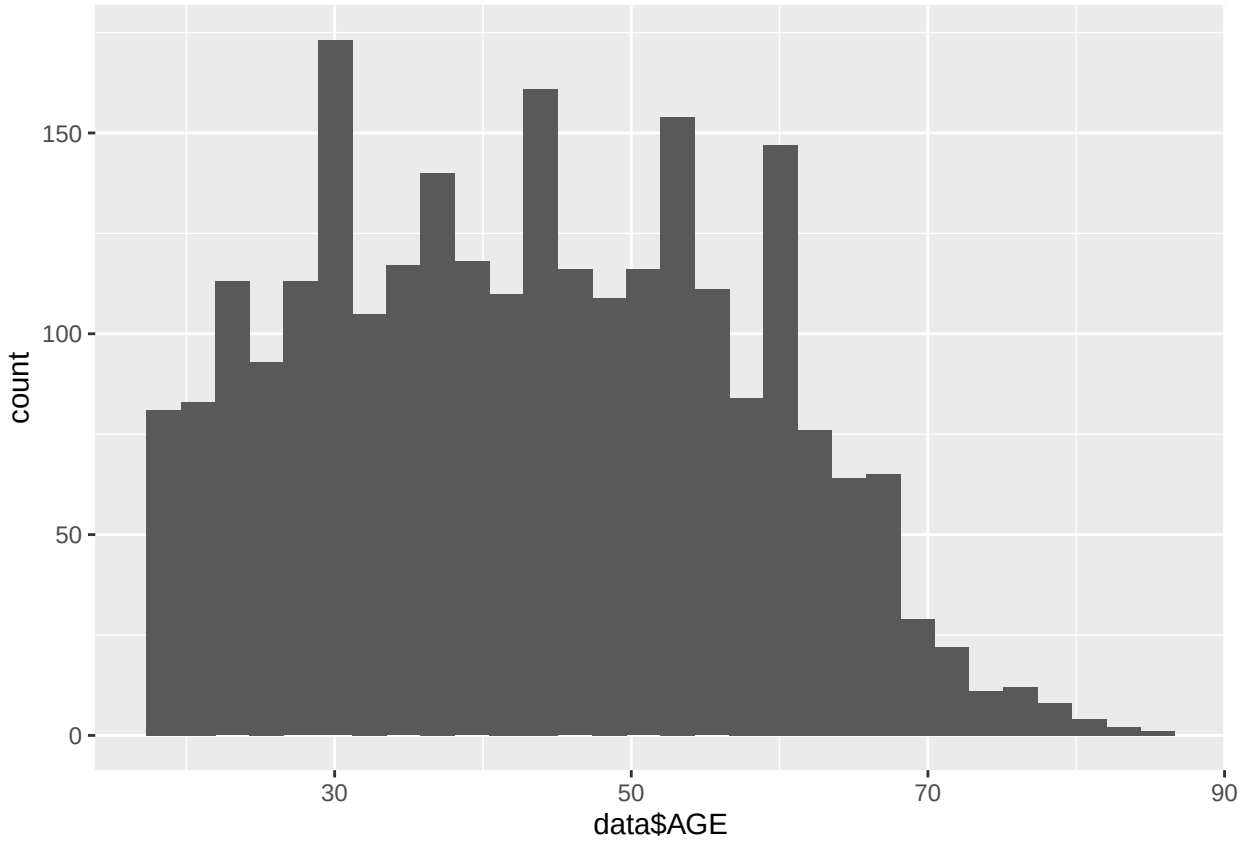
```
library(ggplot2)
library(coda)
library(runjags)
library(fastDummies)
data_full = data.frame(read.csv("nhis_00001.csv",header=TRUE))

data = data_full[0:5000,]
data = data[!data$EARNIMPOINT1 == 0,]
```

Exploring the dataset

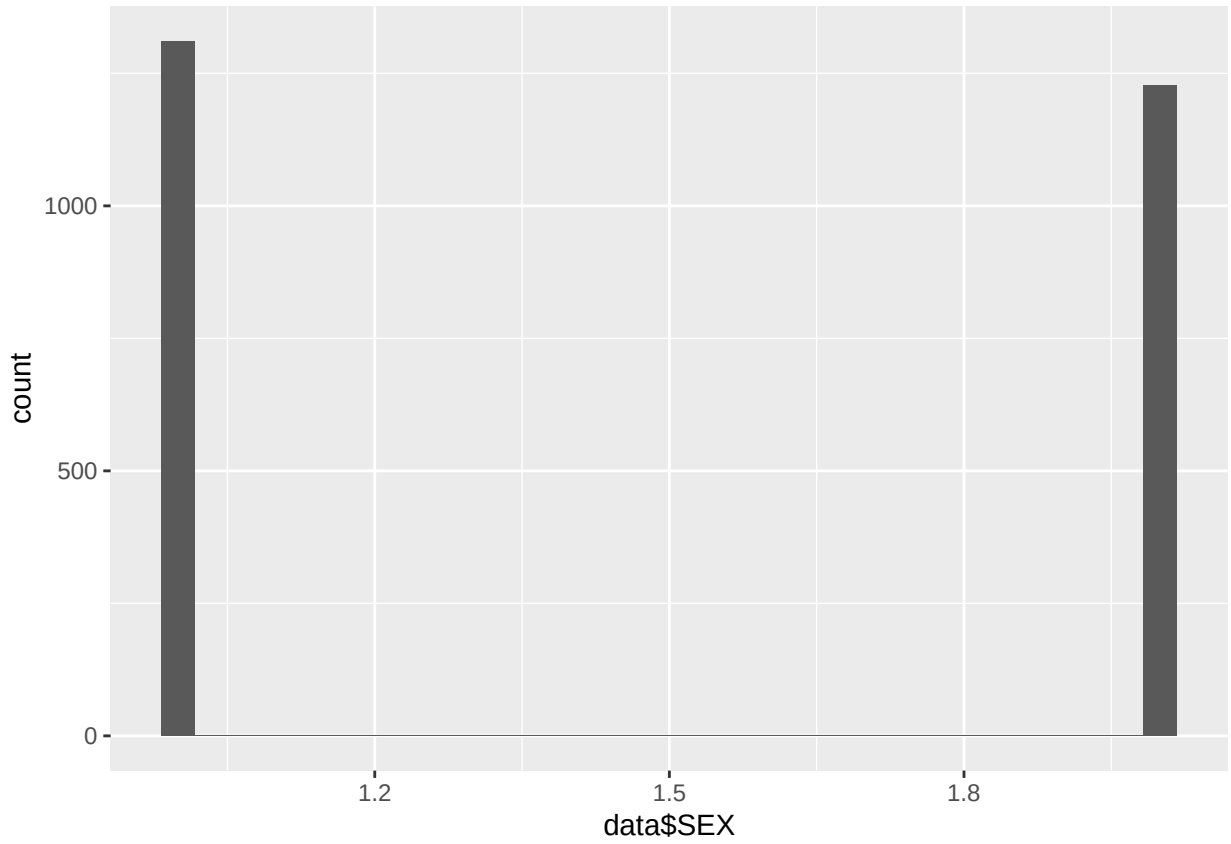
```
ggplot(data, aes(x = data$AGE)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



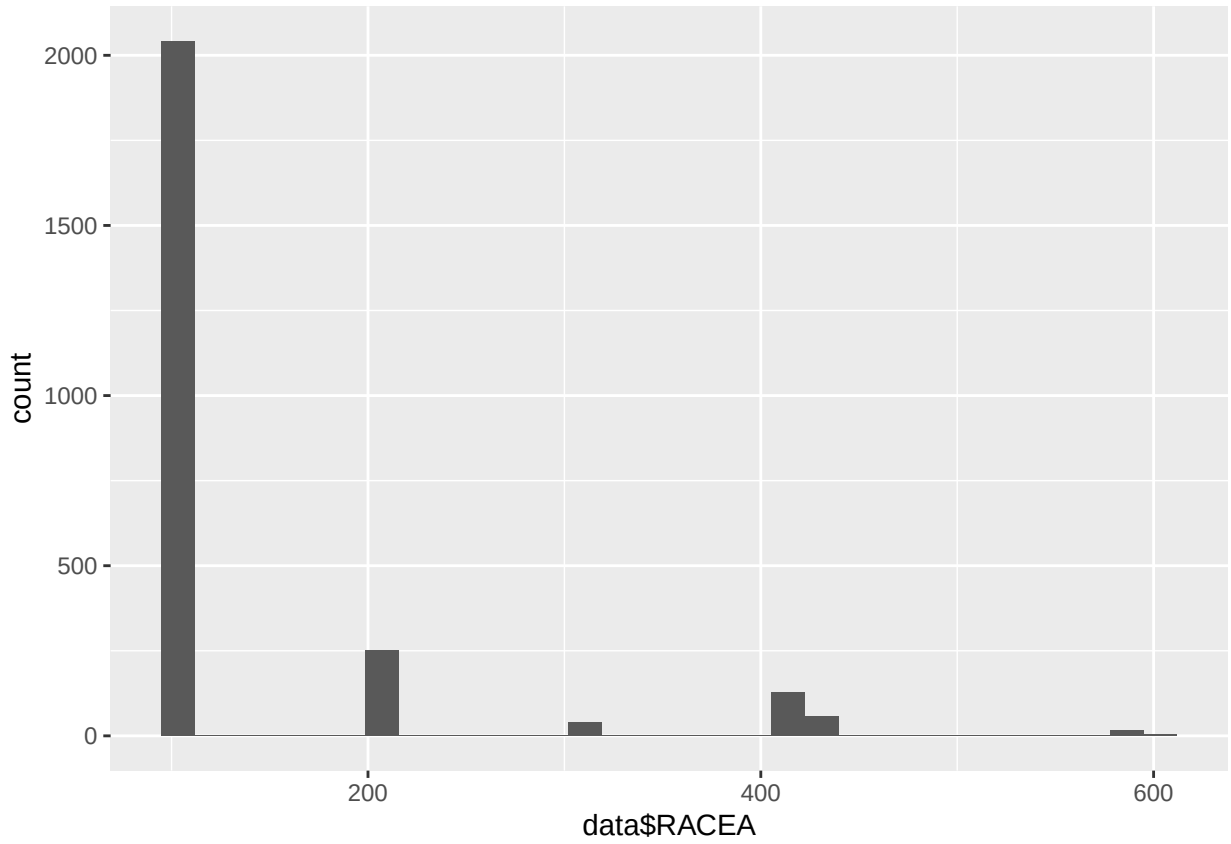

```
ggplot(data, aes(x = data$SEX)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



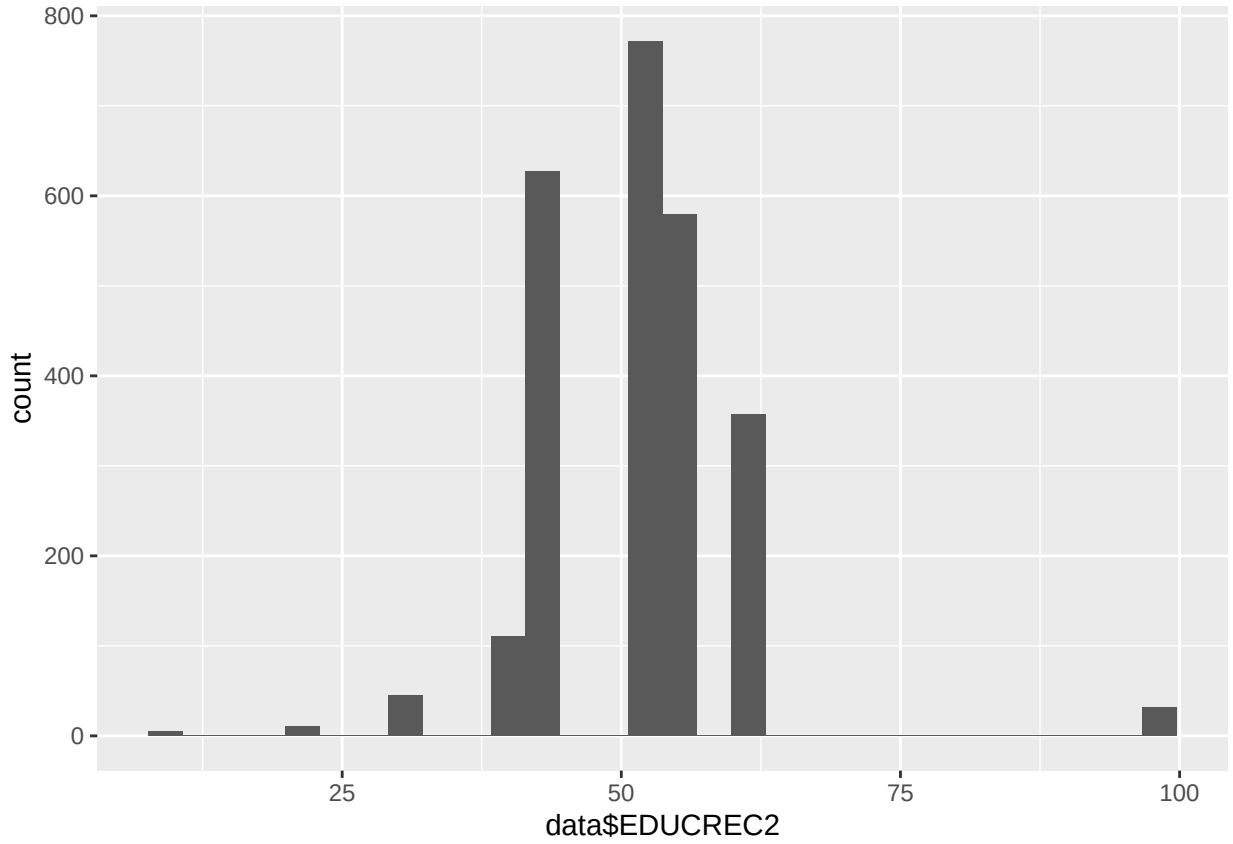

```
ggplot(data, aes(x = data$RACEA)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



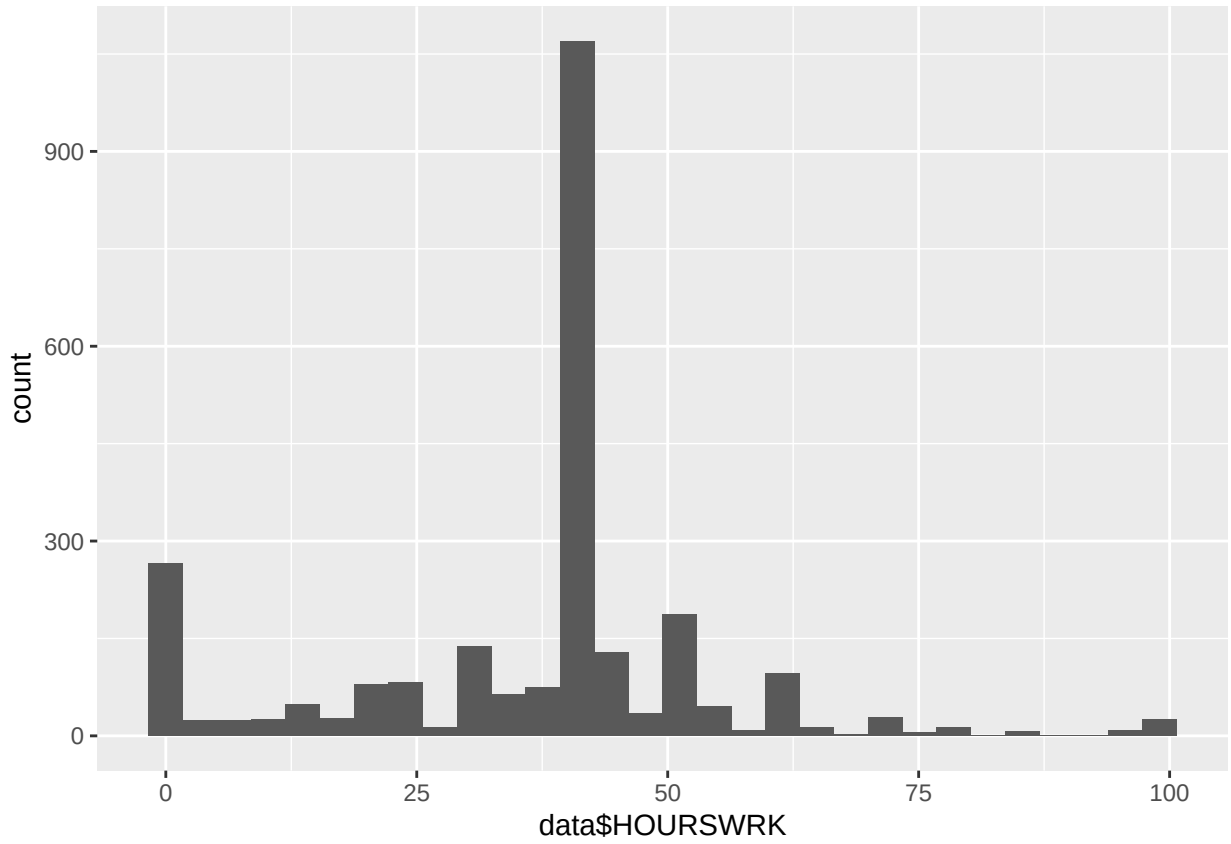

```
ggplot(data, aes(x = data$EDUCREC2)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



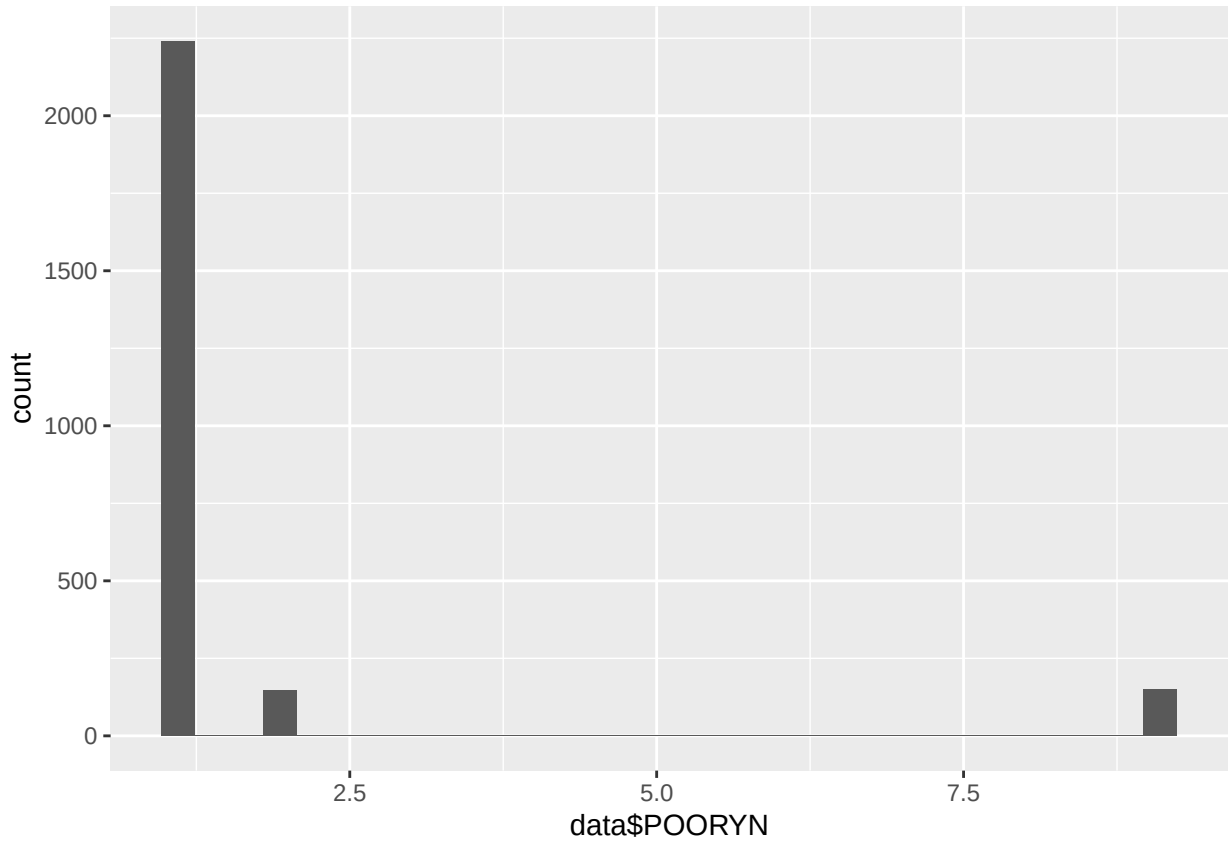

```
ggplot(data, aes(x = data$HOURSWRK)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



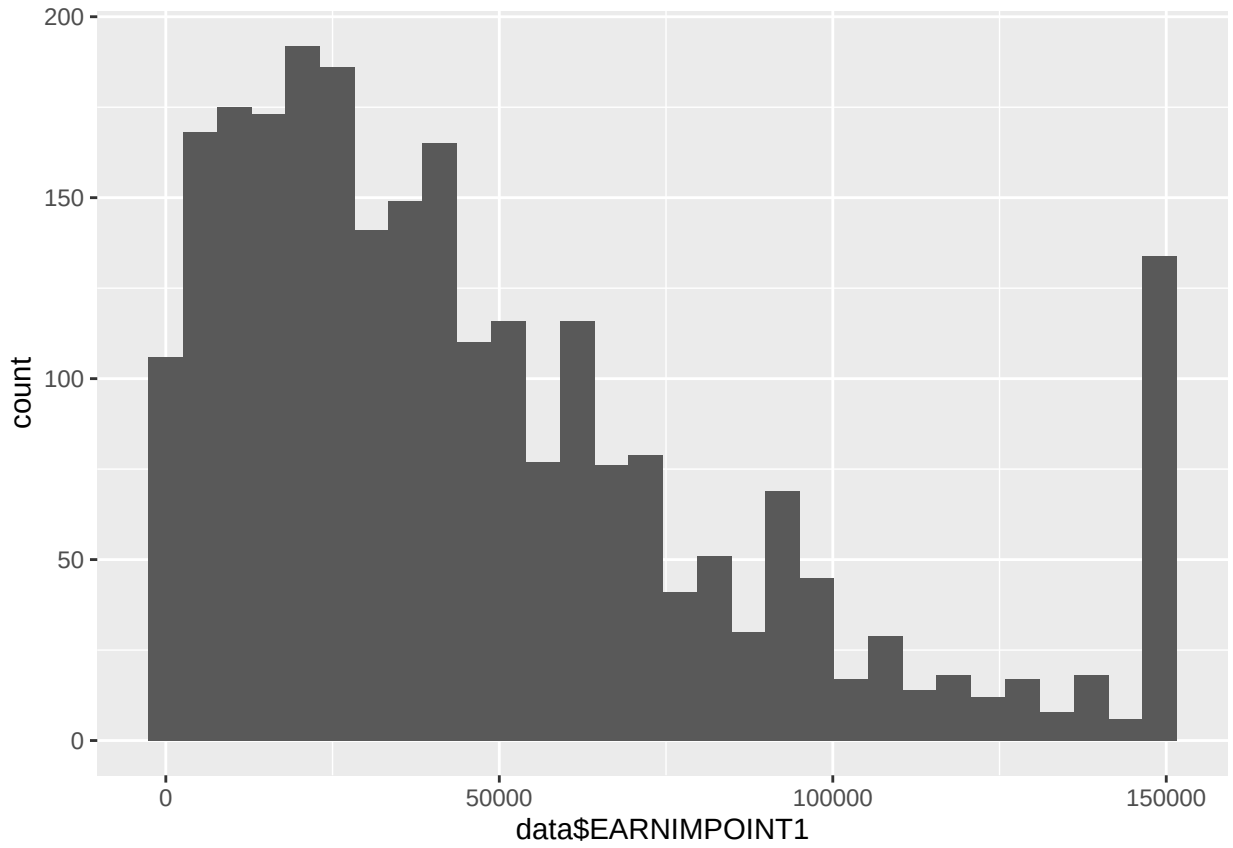

```
ggplot(data, aes(x = data$POORYN)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



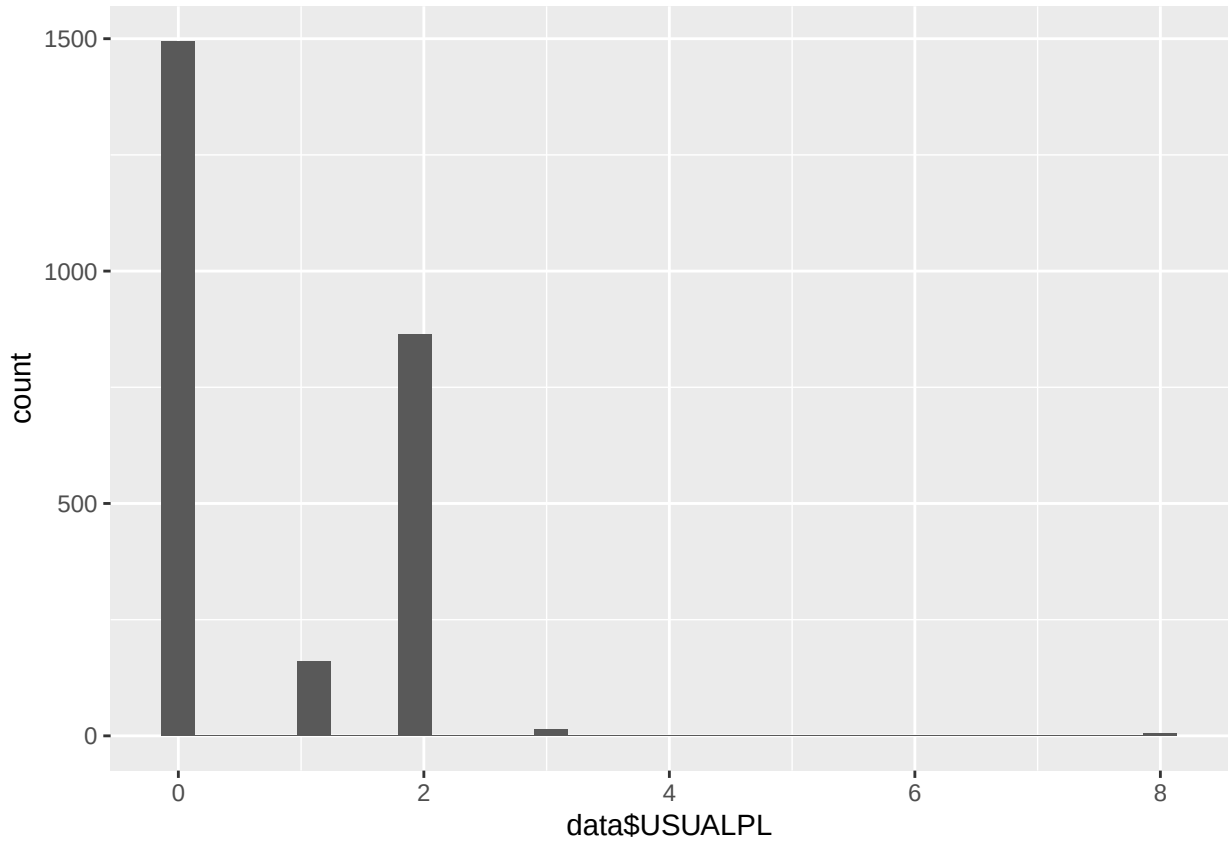

```
ggplot(data, aes(x = data$EARNIMPOINT1)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



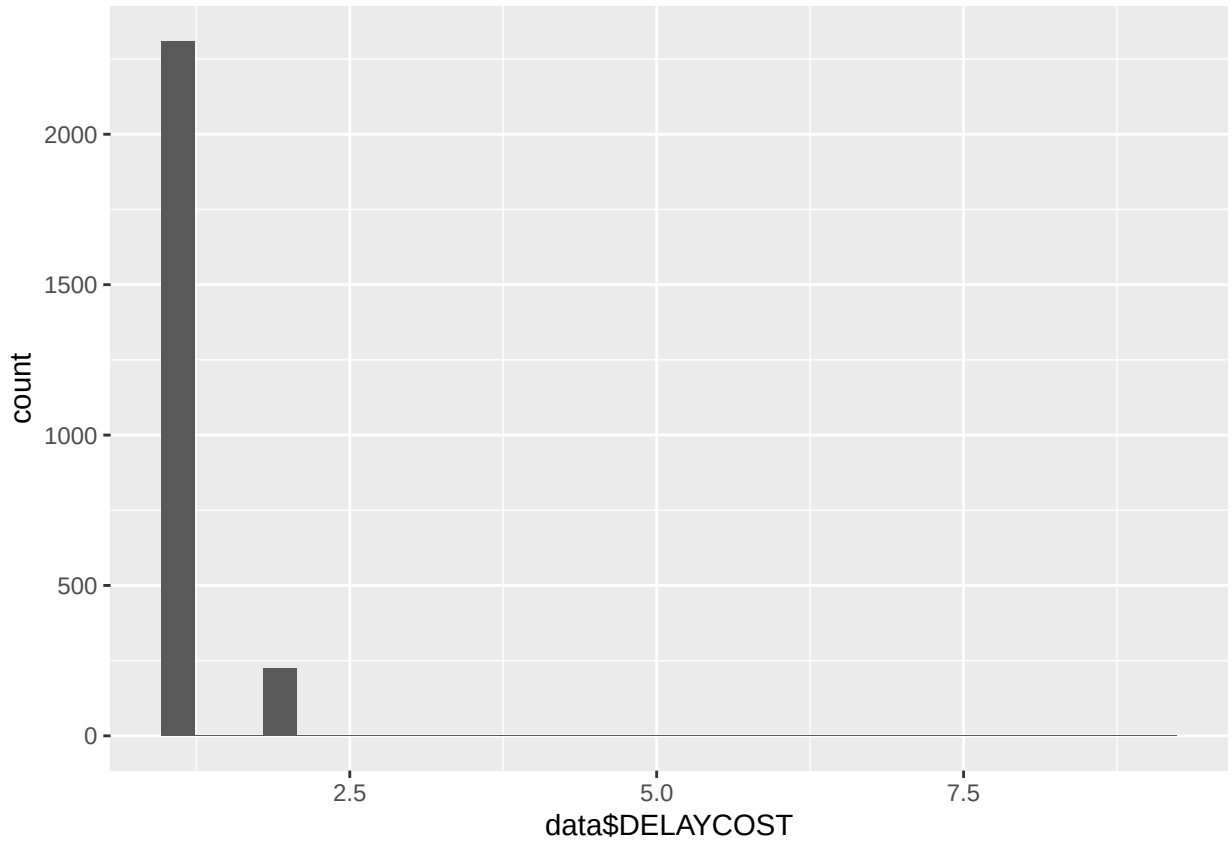

```
ggplot(data, aes(x = data$USUALPL)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



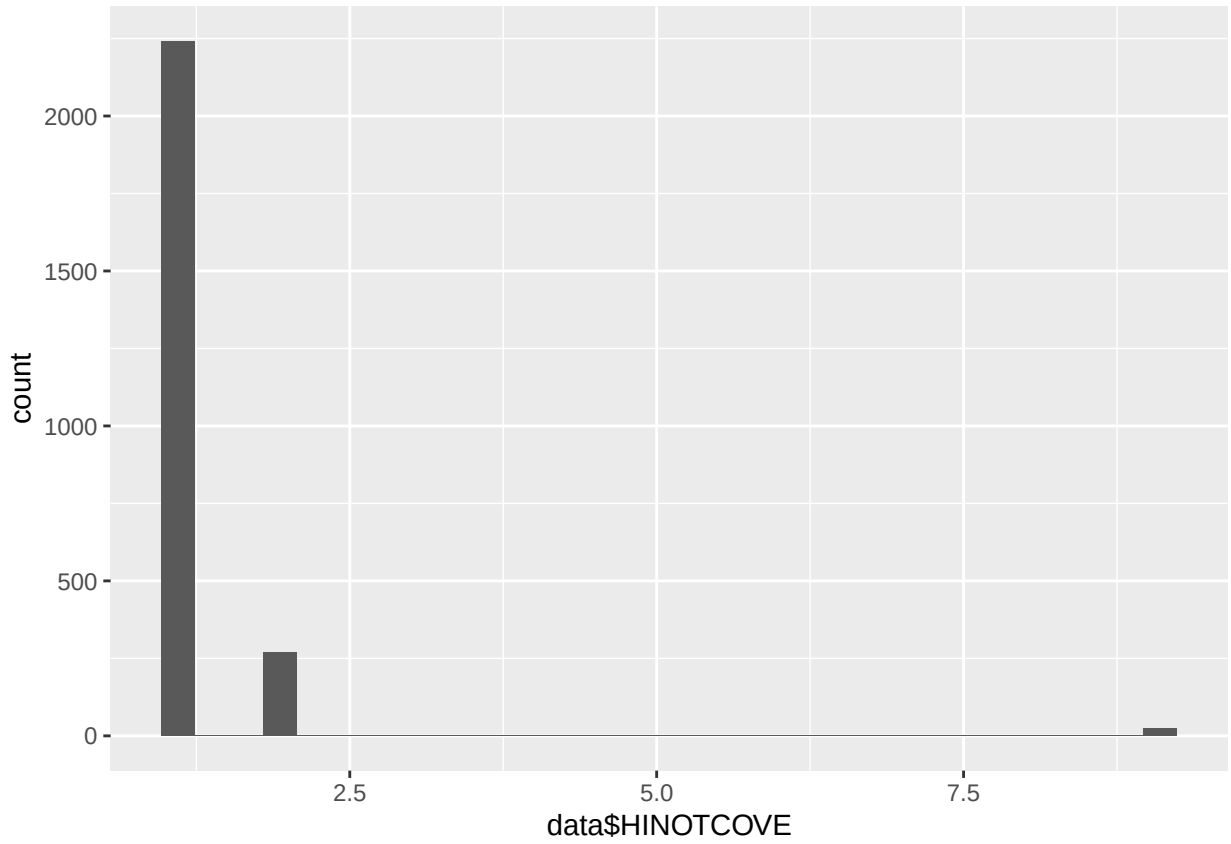

```
ggplot(data, aes(x = data$DELAYCOST)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



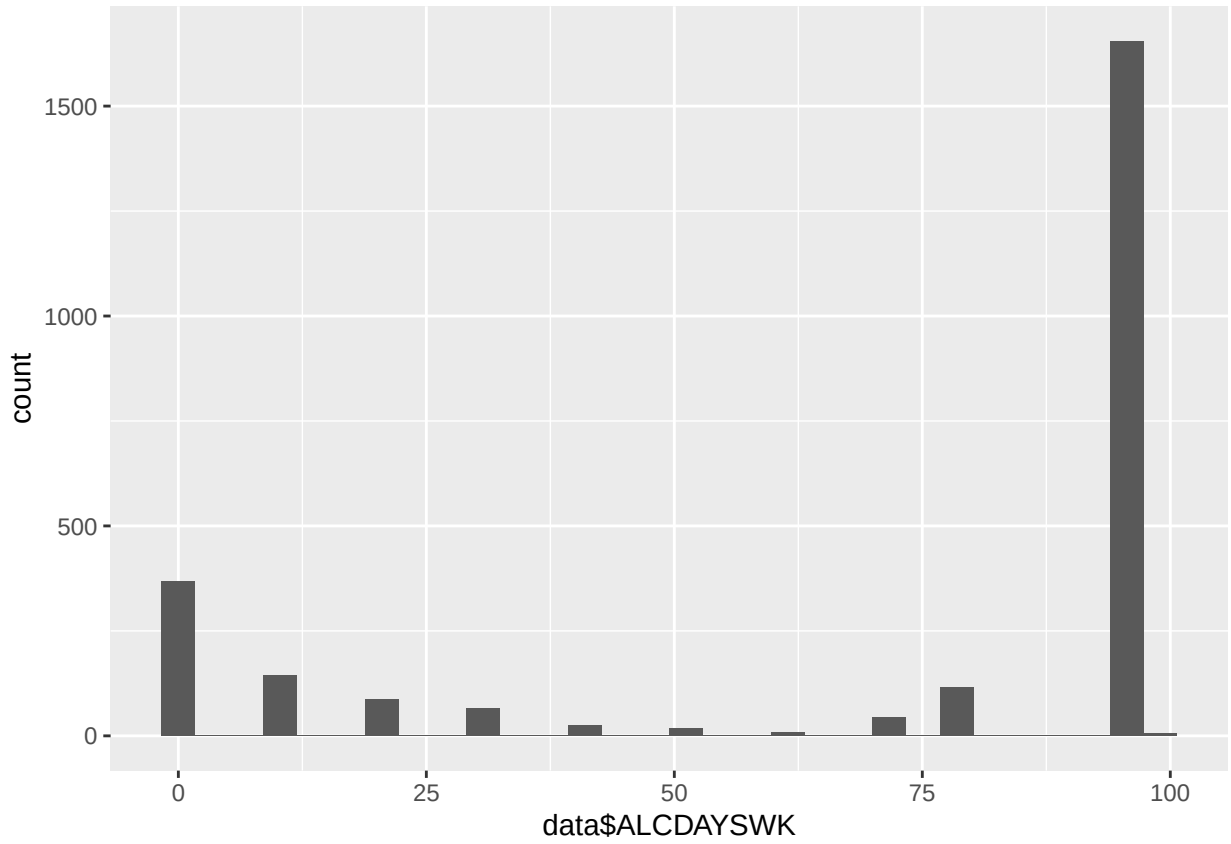

```
ggplot(data, aes(x = data$HINOTCOVE)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



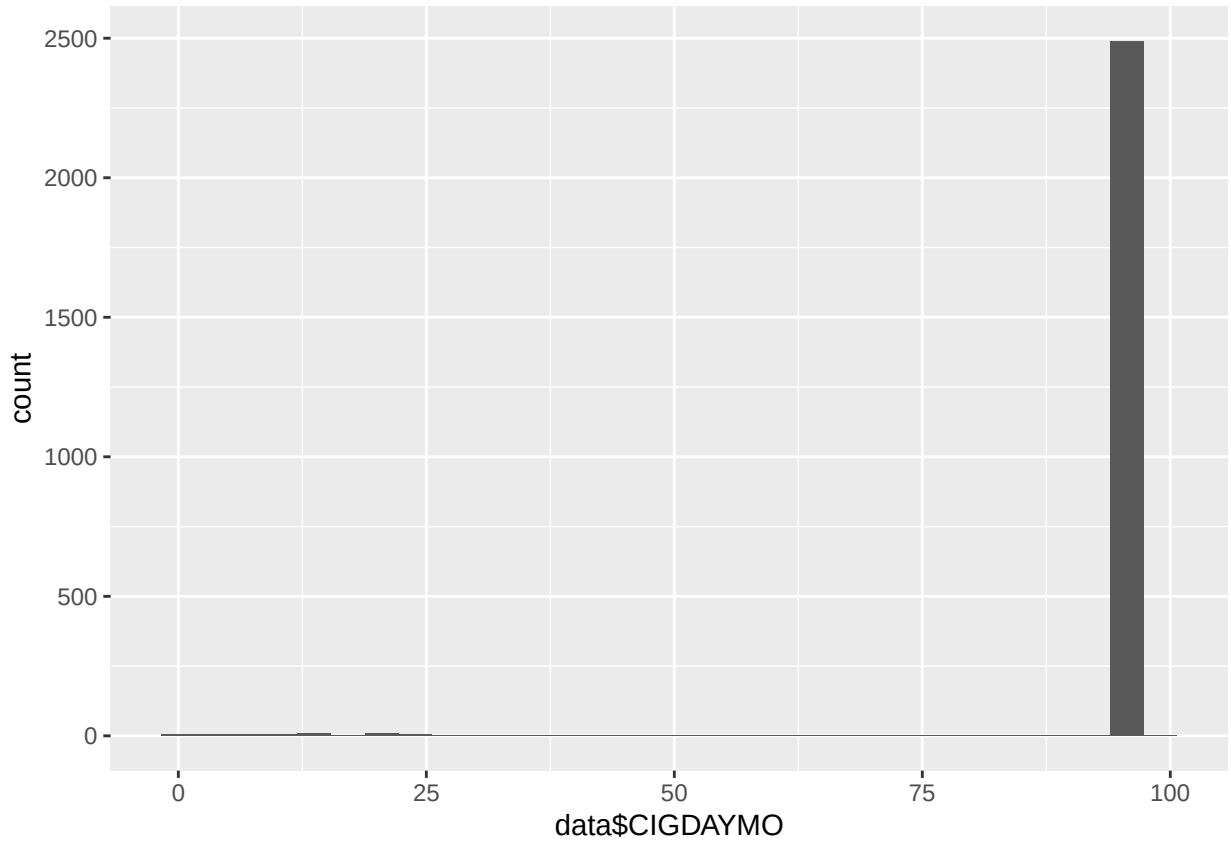

```
ggplot(data, aes(x = data$ALCDAYSWK)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



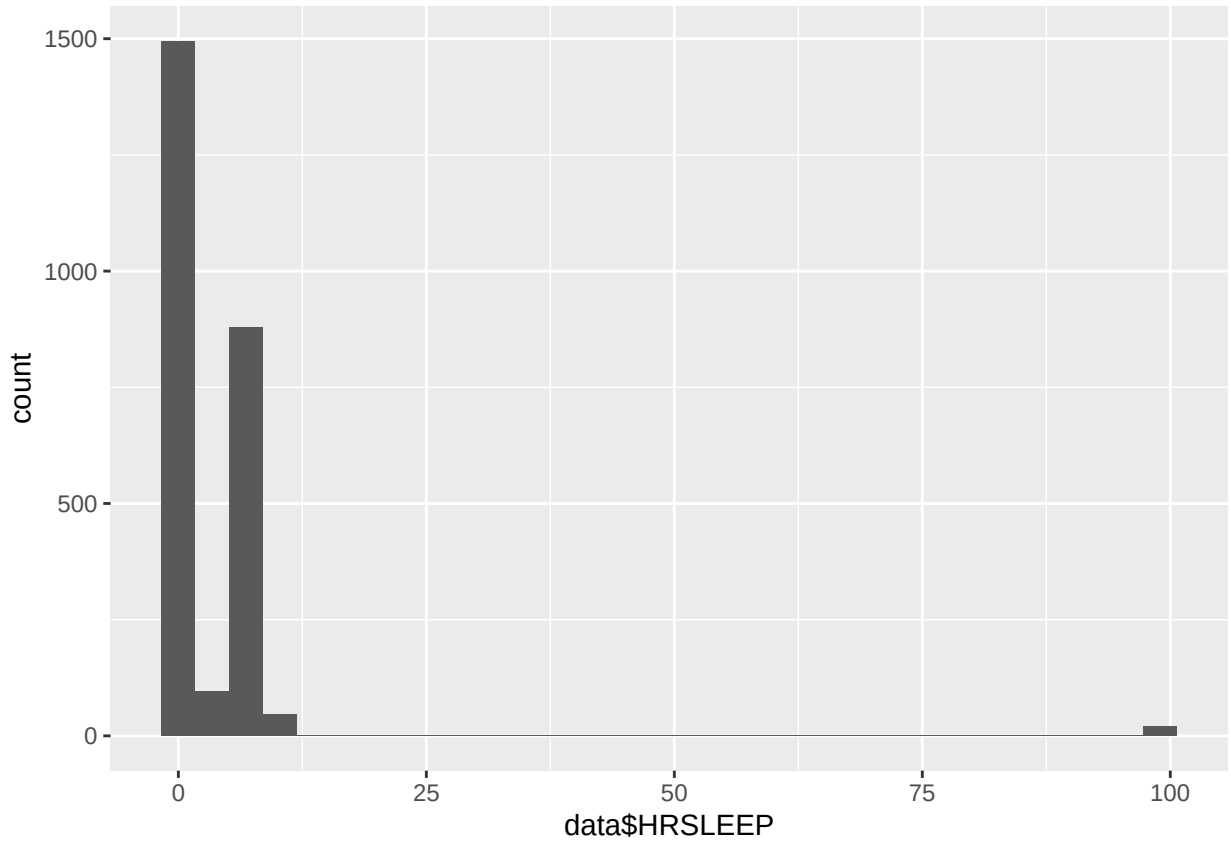

```
ggplot(data, aes(x = data$CIGDAYMO)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



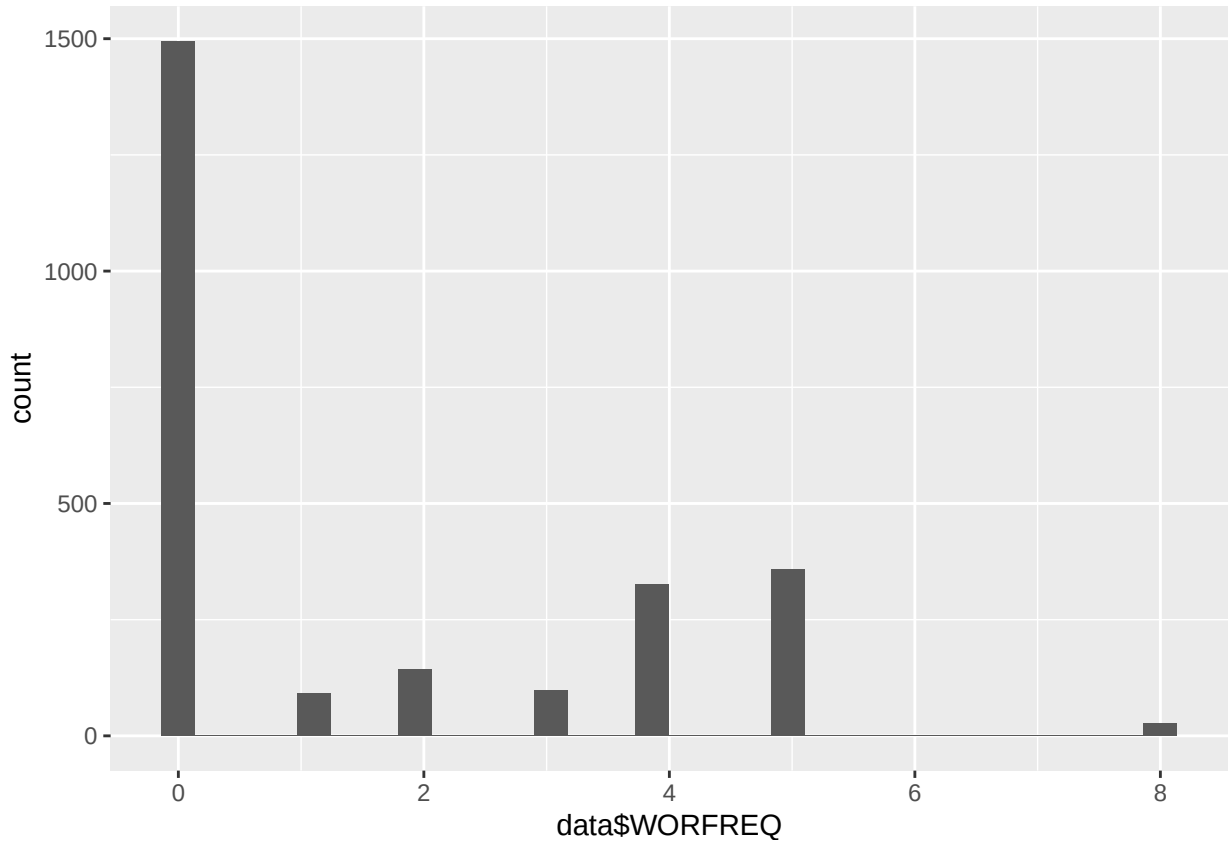

```
ggplot(data, aes(x = data$HRSLEEP)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



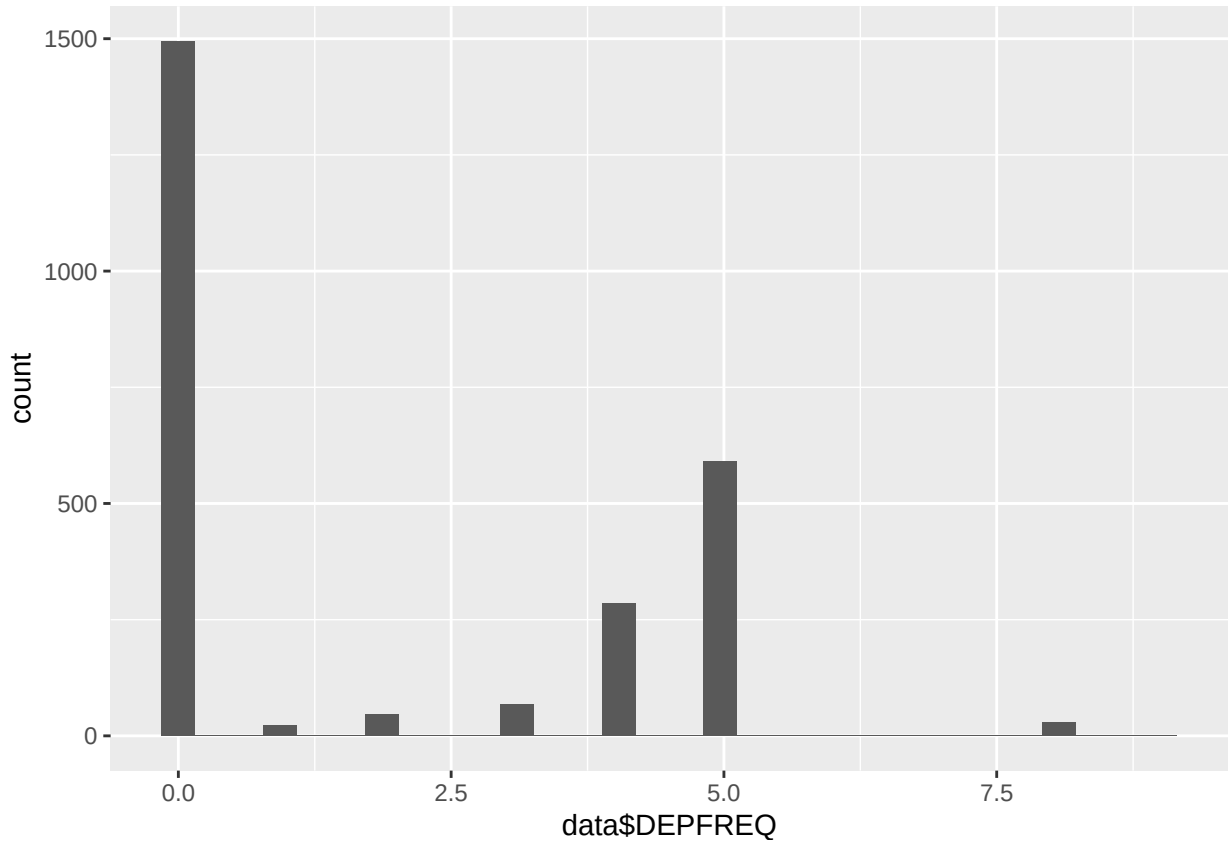

```
ggplot(data, aes(x = data$WORFREQ)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data, aes(x = data$DEPFREQ)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Synthesizing Earnings

```
# Using AGE, HOURSWRK, and EDUCREC2 to predict income
# Binary columns for EDUCREC2
data$EDU = fastDummies::dummy_cols(data$EDUCREC2)

modelString <-"
model {
## sampling
for (i in 1:N){
  y[i] ~ dnorm(beta0 + beta1*x_age[i] + beta2*x_hours[i] +
                                       beta3*x_edu_20[i] + beta4*x_edu_31[i] +
                                       beta5*x_edu_41[i] + beta6*x_edu_51[i] +
                                       beta7*x_edu_54[i] + beta8*x_edu_60[i], invsigma2)
}
## priors
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
beta2 ~ dnorm(mu2, g2)
beta3 ~ dnorm(mu3, g3)
beta4 ~ dnorm(mu4, g4)
beta5 ~ dnorm(mu5, g5)
beta6 ~ dnorm(mu6, g6)
beta7 ~ dnorm(mu7, g7)
beta8 ~ dnorm(mu8, g8)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
```

```
"

y = log(as.vector(data$EARNIMPOINT1))
#`y[!is.finite(y)] <- 0
x_age = as.vector(data$AGE)
x_hours = as.vector(data$HOURSWRK)
x_edu_20 = as.vector(data$EDU$.data_20)
x_edu_31 = as.vector(data$EDU$.data_31)
x_edu_41 = as.vector(data$EDU$.data_41)
x_edu_51 = as.vector(data$EDU$.data_51)
x_edu_54 = as.vector(data$EDU$.data_54)
x_edu_60 = as.vector(data$EDU$.data_60)

N = length(y)

the_data <- list("y" = y, "x_age" = x_age,
                 "x_hours" = x_hours, "x_edu_20" = x_edu_20,
                 "x_edu_31" = x_edu_31, "x_edu_41" = x_edu_41,
                 "x_edu_51" = x_edu_51, "x_edu_54" = x_edu_54,
                 "x_edu_60" = x_edu_60,
                 "N" = N,
                 "mu0" = 0, "g0" = 1, "mu1" = 0, "g1" = 1,
                 "mu2" = 0, "g2" = 1, "mu3" = 0, "g3" = 1,
                 "mu4" = 0, "g4" = 1, "mu5" = 0, "g5" = 1,
                 "mu6" = 0, "g6" = 1, "mu7" = 0, "g7" = 1,
                 "mu8" = 0, "g8" = 1,
                 "a" = 1, "b" = 1)

initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
                 "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}

posterior_MLR <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("beta0", "beta1", "beta2",
                                  "beta3", "beta4", "beta5",
                                  "beta6", "beta7", "beta8", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 20,
                      inits = initsfunction)
```

```
## Loading required namespace: rjags
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
## Burning in the model for 5000 iterations...
```

```
## Running the model for 100000 iterations...
## Simulation complete
## Calculating summary statistics...
```

```
## Warning: Convergence cannot be assessed with only 1 chain
```

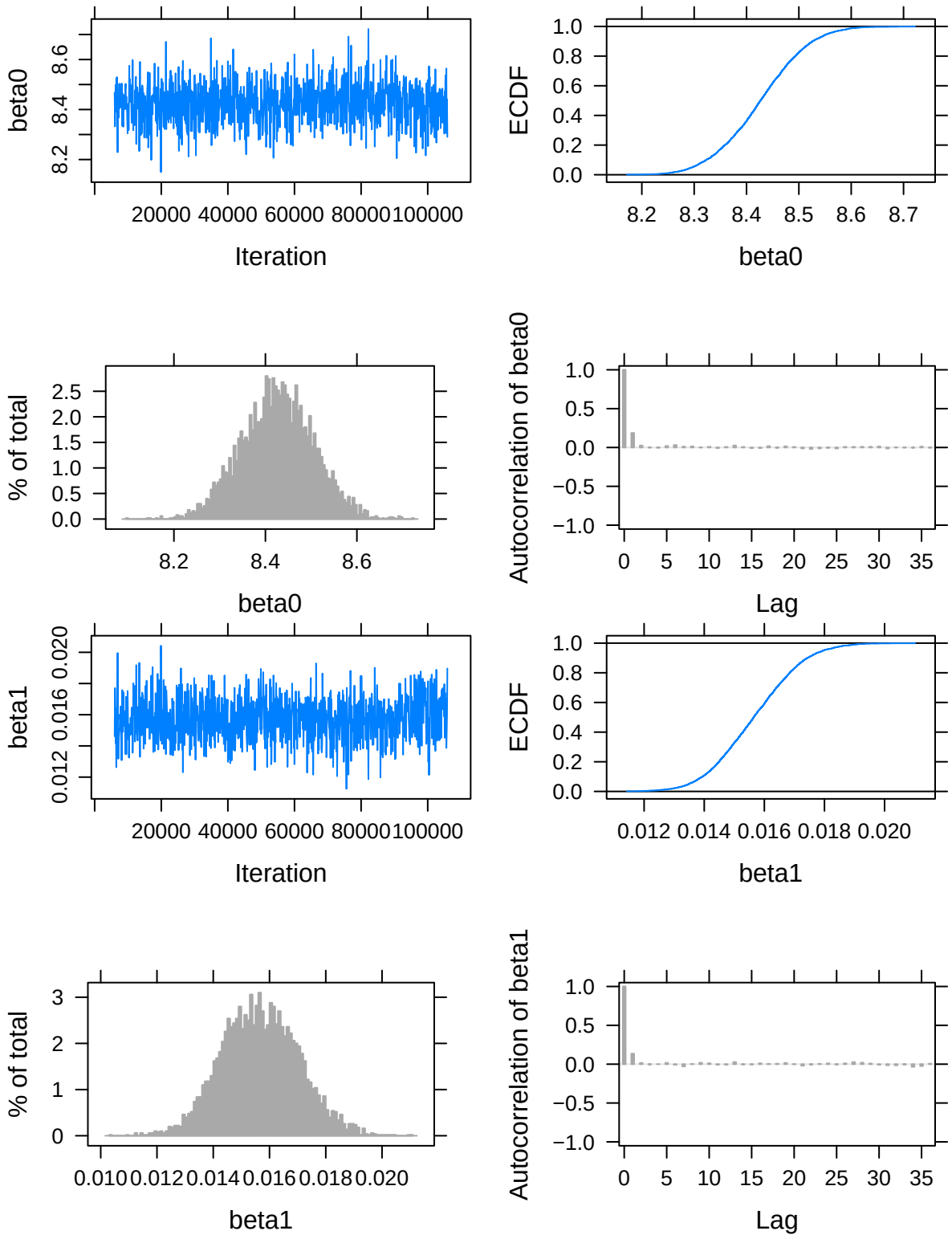
```
## Finished running the simulation
```

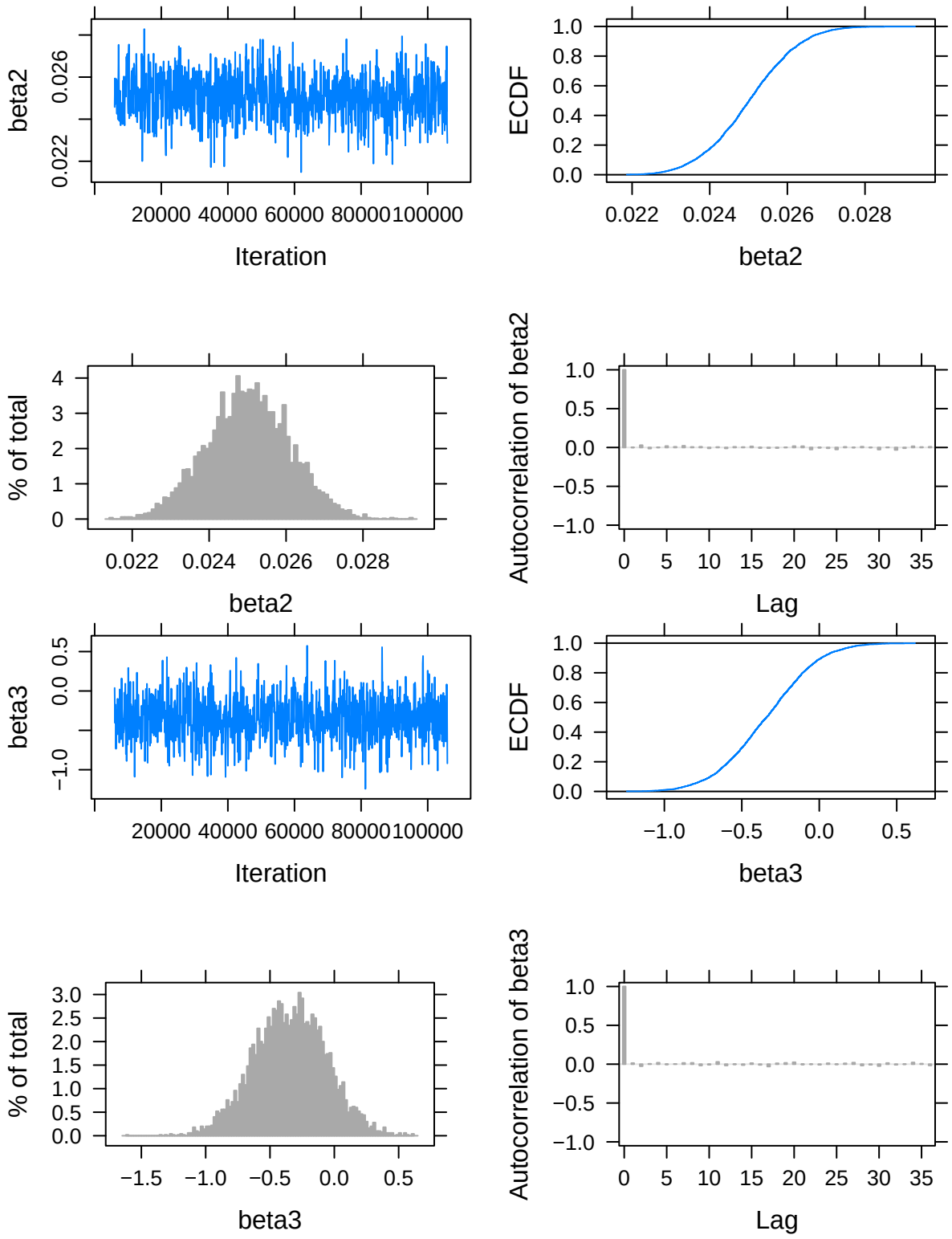
```
summary(posterior_MLR)
```

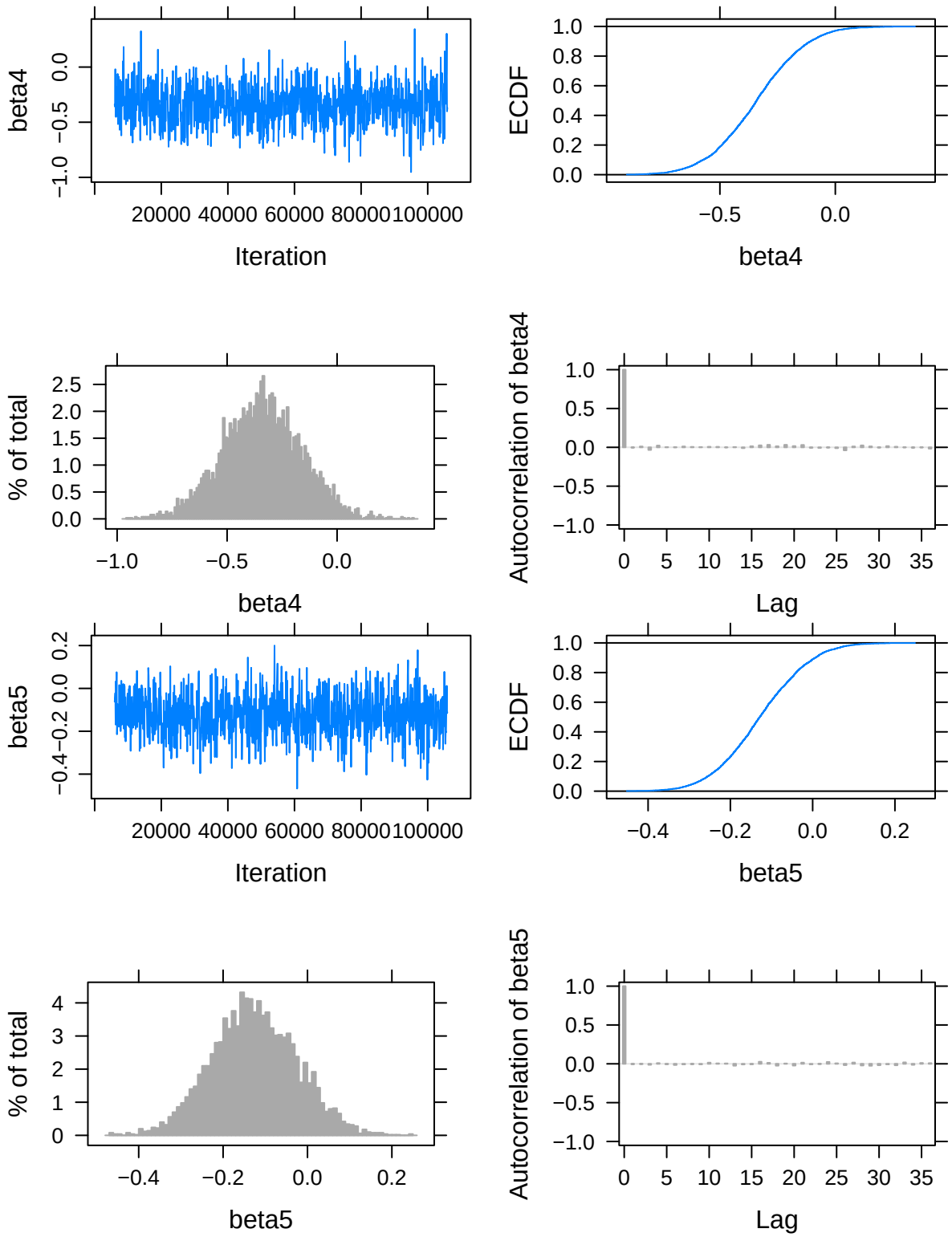
```
##           Lower95      Median    Upper95        Mean          SD Mode
## beta0  8.26793731  8.42636875 8.57535200  8.42621064 0.079094524   NA
## beta1  0.01302770  0.01564369 0.01838840  0.01567090 0.001382498   NA
## beta2  0.02291705  0.02501460 0.02714595  0.02501751 0.001086701   NA
## beta3 -0.89072362 -0.34109908 0.21545354 -0.34415869 0.283190378   NA
## beta4 -0.69206720 -0.34100255 0.01273726 -0.34056856 0.181827783   NA
## beta5 -0.32154015 -0.12853411 0.07112199 -0.12631231 0.101316739   NA
## beta6  0.09116082  0.18834399 0.29580805  0.18774513 0.052242682   NA
## beta7  0.50998680  0.61597142 0.72570563  0.61615202 0.055051031   NA
## beta8  0.71440464  0.83388006 0.95785454  0.83414518 0.063239144   NA
## sigma  0.95637865  0.98229043 1.01000744  0.98273173 0.013886186   NA
##              MCerr MC%ofSD SSeff       AC.200 psrf
## beta0 1.351274e-03     1.7  3426  0.008535824   NA
## beta1 2.240466e-05     1.6  3808  0.011545856   NA
## beta2 1.584273e-05     1.5  4705 -0.010593814   NA
## beta3 3.953067e-03     1.4  5132 -0.006063797   NA
## beta4 2.562471e-03     1.4  5035  0.007103974   NA
## beta5 1.432835e-03     1.4  5000  0.015160682   NA
## beta6 7.812940e-04     1.5  4471 -0.008571245   NA
## beta7 7.785392e-04     1.4  5000 -0.014563888   NA
## beta8 8.943365e-04     1.4  5000 -0.009614118   NA
## sigma 1.963803e-04     1.4  5000 -0.024263543   NA
```

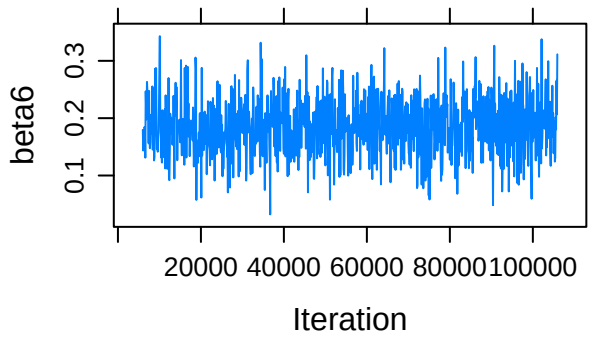
```
plot(posterior_MLR)
```

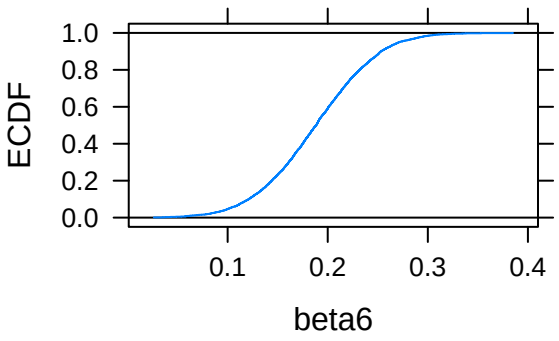
```
## Generating plots...
```

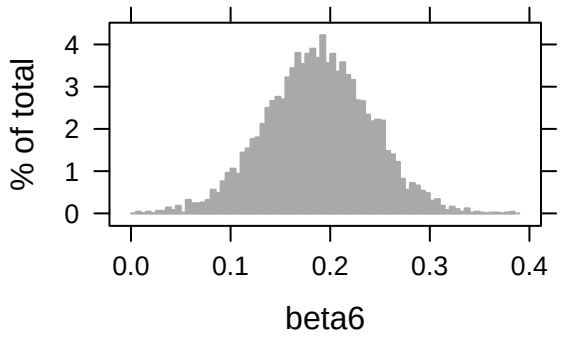

beta0
8.2
8.4
8.6
20000 40000 60000 80000 100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
8.2 8.3 8.4 8.5 8.6 8.7
beta0
% of total
2.5
2.0
1.5
1.0
0.5
0.0
8.2 8.4 8.6
beta0
Autocorrelation of beta0
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag
beta1
0.012 0.016 0.020
20000 40000 60000 80000 100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
0.012 0.014 0.016 0.018 0.020
beta1
% of total
3
2
1
0
0.010 0.012 0.014 0.016 0.018 0.020
beta1
Autocorrelation of beta1
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag

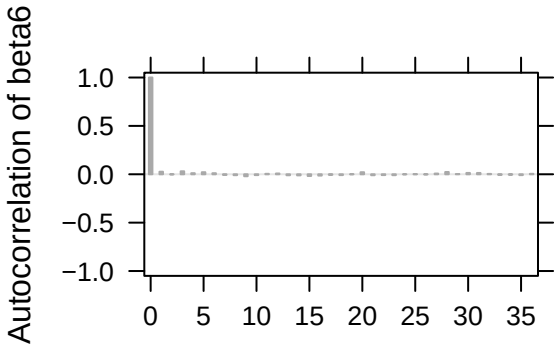

beta2
0.022
0.026
20000 40000 60000 80000 100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
0.022 0.024 0.026 0.028
beta2
% of total
4
3
2
1
0
0.022 0.024 0.026 0.028
beta2
Autocorrelation of beta2
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag
beta3
0.5
0.0
−1.0
20000 40000 60000 80000 100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
−1.0 −0.5 0.0 0.5
beta3
% of total
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−1.5 −1.0 −0.5 0.0 0.5
beta3
Autocorrelation of beta3
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag

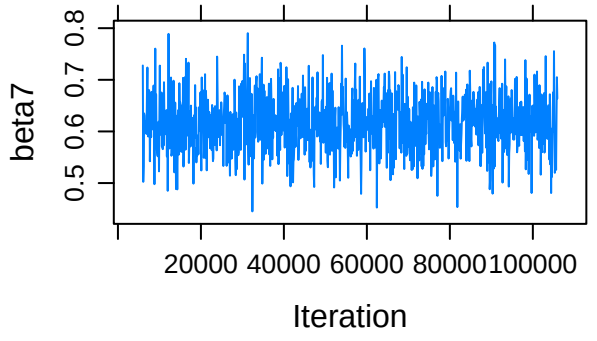

beta4
Iteration
ECDF
beta4
% of total
beta4
Autocorrelation of beta4
Lag
beta5
Iteration
ECDF
beta5
% of total
beta5
Autocorrelation of beta5
Lag

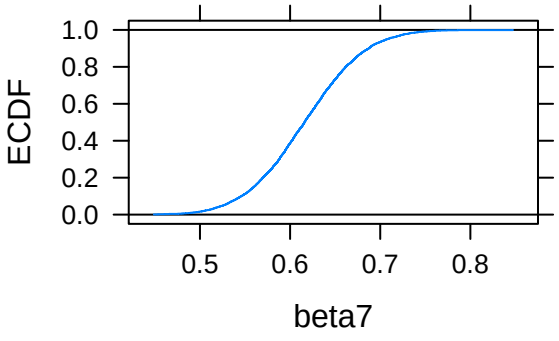

beta6
0.1 0.2 0.3
20000 40000 60000 80000 100000
Iteration

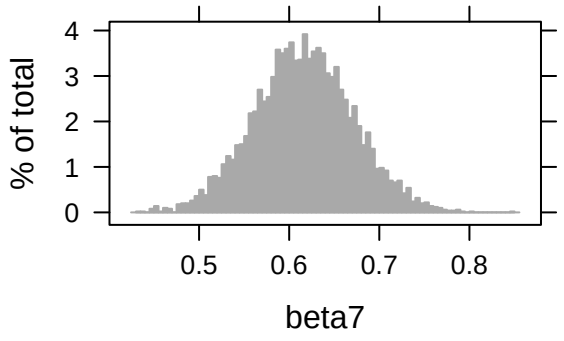

ECDF
0.0 0.2 0.4 0.6 0.8 1.0
0.1 0.2 0.3 0.4
beta6

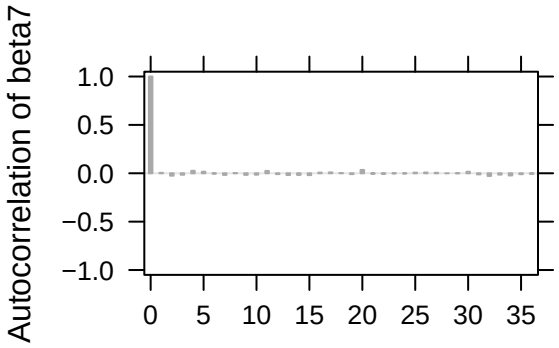

% of total
0 1 2 3 4
0.0 0.1 0.2 0.3 0.4
beta6

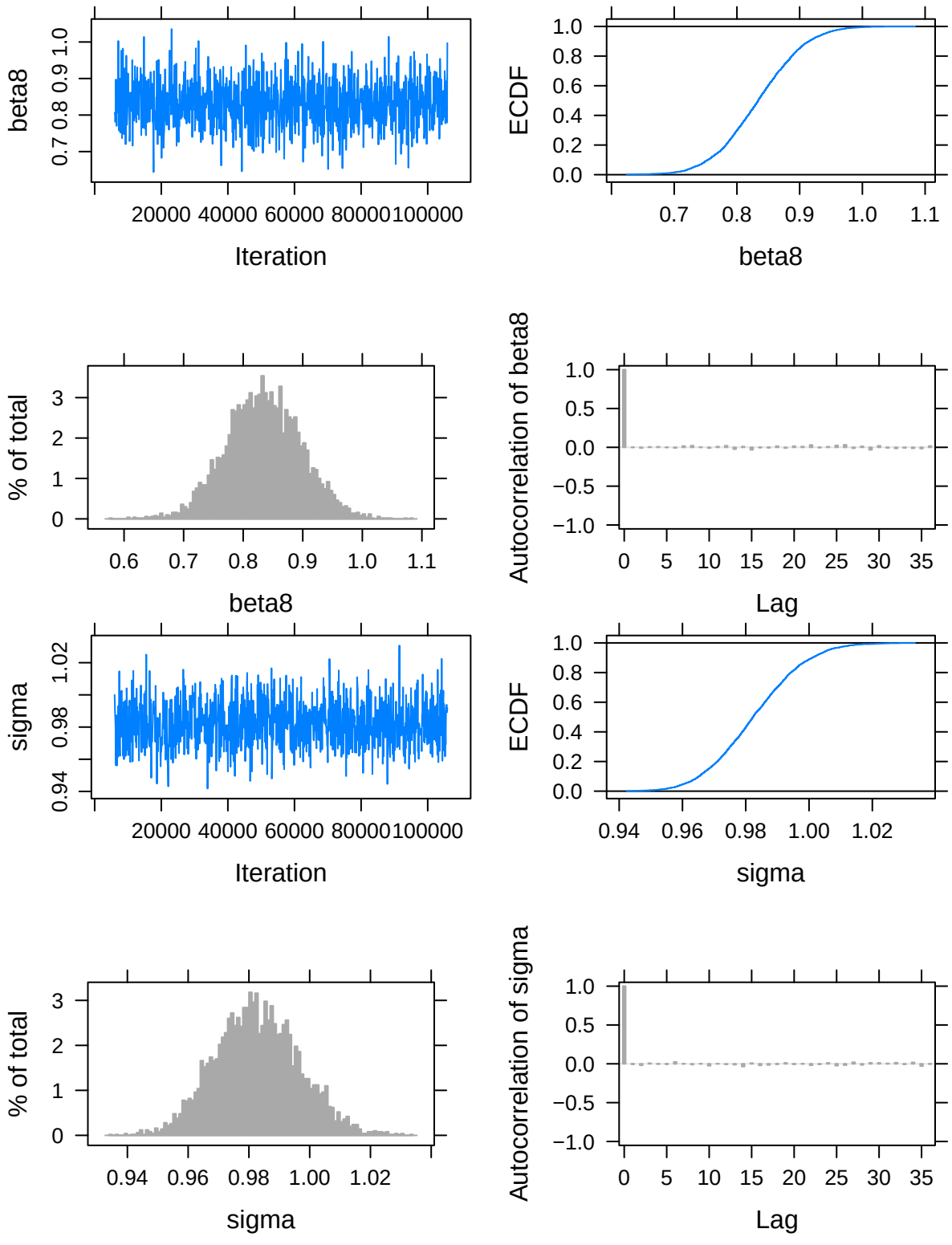

Autocorrelation of beta6
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag

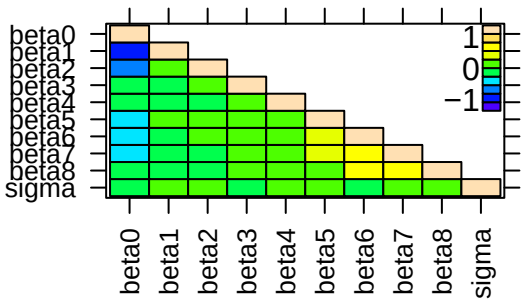

beta7
0.5 0.6 0.7 0.8
20000 40000 60000 80000 100000
Iteration


ECDF
0.0 0.2 0.4 0.6 0.8 1.0
0.5 0.6 0.7 0.8
beta7


% of total
0 1 2 3 4
0.5 0.6 0.7 0.8
beta7


Autocorrelation of beta7
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag


beta8
0.7 0.8 0.9 1.0
20000 40000 60000 80000 100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
0.7 0.8 0.9 1.0 1.1
beta8
% of total
3
2
1
0
0.6 0.7 0.8 0.9 1.0 1.1
beta8
Autocorrelation of beta8
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag
sigma
0.94 0.98 1.02
20000 40000 60000 80000 100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
0.94 0.96 0.98 1.00 1.02
sigma
% of total
3
2
1
0
0.94 0.96 0.98 1.00 1.02
sigma
Autocorrelation of sigma
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag




```
post <- as.mcmc(posterior_MLR)

synthesize <- function(X, index, n){
  mean_Y <- post[index, "beta0"] + X$x_age * post[index, "beta1"] + X$x_hours * post[index, "beta2"] + X
  synthetic_Y <- rnorm(n,mean_Y, post[index,"sigma"])
  data.frame(X$y, synthetic_Y)
}
n <- dim(data)[1]
params <- data.frame(y, x_age, x_hours, x_edu_20, x_edu_31, x_edu_41, x_edu_51, x_edu_54, x_edu_60)
synthetic_one <- synthesize(params,1,n)
names(synthetic_one) <- c("OriginalIncome", "SynIncome")
```

```
plot(synthetic_one$OriginalIncome, synthetic_one$SynIncome)
```