Lab 7

Isaac Kleisle-Murphy

April 6, 2020

Ingest and Setup

```r
library(runjags)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(coda)
library(stringr)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:runjags':
##
##     extract
```

```r
library(fastDummies)
library(ggplot2)


setwd("~/Documents/Swat_2020/Data_Privacy/Data-Confidentiality/datasets")
CEdata = read.csv("CEdata.csv")%>%
  mutate(logIncome = log(Income),
         logExp = log(Expenditure))%>%
  arrange(Race)%>%
  fastDummies::dummy_cols(., c("Race"), remove_first_dummy = T)%>%
  mutate(UrbanRural = UrbanRural - 1)
```

Rebuild Synthesis Model in JAGS

```r
predictors = c("logExp", paste0("Race_", 2:6), "UrbanRural")
```

```r
the_data = list(

  Y = CEdata$logIncome,
  N = nrow(CEdata),
  X = CEdata[, predictors]%>%
    as.matrix()%>%
    unname()%>%
    cbind(1, .),
  X.dim = length(predictors) + 1,
  I = diag(100),
  Z = rep(0, 100)

)



model_str = "

model{

for (ii in 1:N){
Y[ii] ~ dnorm(Mu[ii], invsigma2)
Mu[ii] <- inprod(Beta, X[ii, 1:X.dim])
}

Beta ~ dmnorm(Z[1:X.dim], I[1:X.dim, 1:X.dim])
sigma <- 1/sqrt(invsigma2)
invsigma2 ~ dgamma(1,1)

}


"


fit = run.jags(model_str, data = the_data, n.chains = 2, burnin = 10000, sample = 1000, adapt = 1000, t
               monitor = c("Beta", "sigma"))
```

```
## Warning: No initial values were provided - JAGS will use the same initial values
## for all chains

## Calling the simulation...
## Welcome to JAGS 4.3.0 on Mon Apr  6 21:52:11 2020
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 994
##    Unobserved stochastic nodes: 2
##    Total graph size: 21043
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 10000
## -------------------------------------------------| 10000
## ************************************************** 100%
## . . . Updating 10000
## -------------------------------------------------| 10000
## ************************************************** 100%
## . . . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 9 variables....
## Finished running the simulation
```

```
posterior_df = data.frame(as.mcmc(fit))
```

```
## Warning in as.mcmc.runjags(fit): Combining the 2 mcmc chains together
```

```
attr(posterior_df, "predictors") = predictors

syn_logInc_helper <- function(df, posterior_df, S = 23){

  set.seed(S)
  sl = sample(1:nrow(posterior_df), 1)
  X = df[, attr(posterior_df, "predictors")]%>%
    cbind(1, .)%>%
    as.matrix()
  B = posterior_df%>%
    dplyr::select(-sigma)%>%
    .[sl, ]%>%
    as.matrix()
  Mu = as.numeric(X%*%t(B))
  sigma = posterior_df$sigma[sl]

  Y_syn = rnorm(length(Mu), Mu, sigma)

  return(Y_syn)

}
```

Helper Functions for Risk Analysis

```
compute_logsumexp <- function(log_vector){
  log_vector_max <- max(log_vector)
  exp_vector <- exp(log_vector - log_vector_max)
  sum_exp <- sum(exp_vector)
  log_sum_exp <- log(sum_exp) + log_vector_max
  return(log_sum_exp)
}
```

```r
importance_sample <- function(i, H = 50){

  y_i <- CEdata_org$LogIncome[i]
  y_i_guesses <- seq((y_i - 2.5), (y_i + 2.5), 0.5)
  X_i <- CEdata_syn[, c("LogExpenditure", paste0("Race_", 2:6), "UrbanRural")]%>%
    cbind(1, .)%>%
    .[i,]
  G <- length(y_i_guesses)

  beta_draws <- posterior_df[1:H, grepl("Beta", colnames(posterior_df))]%>%
    as.matrix()
  sigma_draws <- posterior_df[1:H, "sigma"]




  CU_i_logZ_all <- rep(NA, G)
  for (g in 1:G){
    q_sum_H <- sum((dnorm(y_i_guesses[g],
                          mean = (beta_draws%*%t(X_i)),
                          sd = sigma_draws)) /
                     (dnorm(y_i, mean = (beta_draws%*%t(X_i)),
                            sd = sigma_draws)))
    log_pq_h_all <- rep(NA, H)
    for (h in 1:H){
      log_p_h <- sum(log(dnorm(CEdata_syn$LogIncome,
                               mean = (beta_draws[h,]%*%t(X_i)),
                               sd = sigma_draws[h])))
      log_q_h <- log(((dnorm(y_i_guesses[g],
                             mean = (beta_draws[h,]%*%t(X_i)),
                             sd = sigma_draws[h])) /
                        (dnorm(y_i, mean = (beta_draws[h,]%*%t(X_i)),
                               sd = sigma_draws[h]))) / q_sum_H)
      log_pq_h_all[h] <- log_p_h + log_q_h
    }
    CU_i_logZ_all[g] <- compute_logsumexp(log_pq_h_all)
  }


  prob <- exp(CU_i_logZ_all - max(CU_i_logZ_all)) /
    sum(exp(CU_i_logZ_all - max(CU_i_logZ_all)))
  outcome <- as.data.frame(cbind(y_i_guesses, prob))
  names(outcome) <- c("guess", "probability")
  result = outcome[order(outcome$probability, decreasing = TRUE), ]

  rank = which(result$guess == y_i)
  probability = result$probability[rank]

  return(c(rank, probability))
}
```

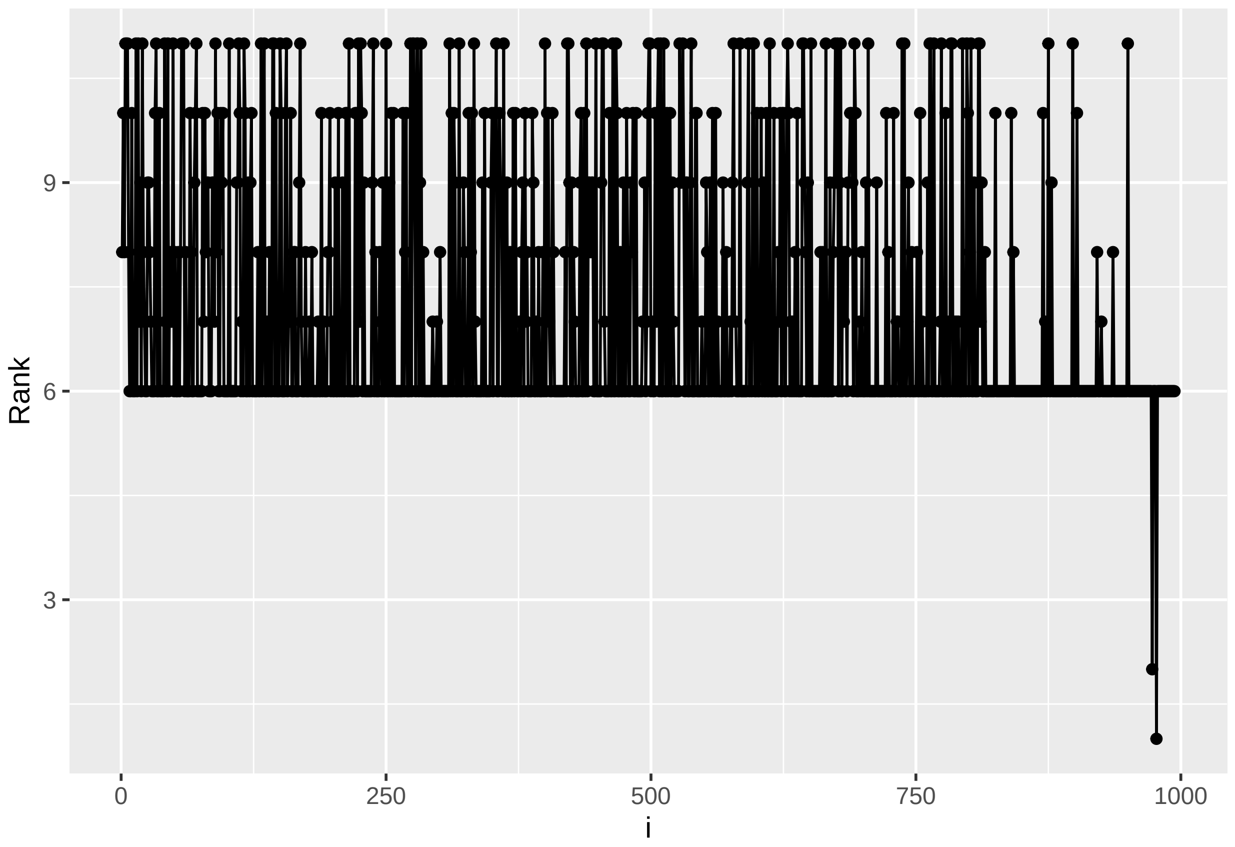
Main

Note that samples are in `posterior_df`, in the rebuild JAGs section above.

```r
synthesis = syn_logInc_helper(CEdata, posterior_df, S = 25)

CEdata_org = CEdata; CEdata_syn = CEdata
CEdata_syn$logIncome = synthesis; CEdata_syn$Income = exp(synthesis)



CEdata_org$LogIncome <- round(log(CEdata_org$Income),
                            digits = 1)
CEdata_org$LogExpenditure <- round(log(CEdata_org$Expenditure),
                                 digits = 1)
CEdata_syn$LogIncome <- round(log(CEdata_syn$Income),
                            digits = 1)
CEdata_syn$LogExpenditure <- round(log(CEdata_syn$Expenditure),
                                 digits = 1)

rank_probs= lapply(1:nrow(CEdata_org), importance_sample)%>%
  do.call("rbind", .)


rank_prob_df = data.frame(rank_probs)%>%
  `colnames<-`(c("Rank", "Probability"))%>%
  mutate(i = row_number())
```

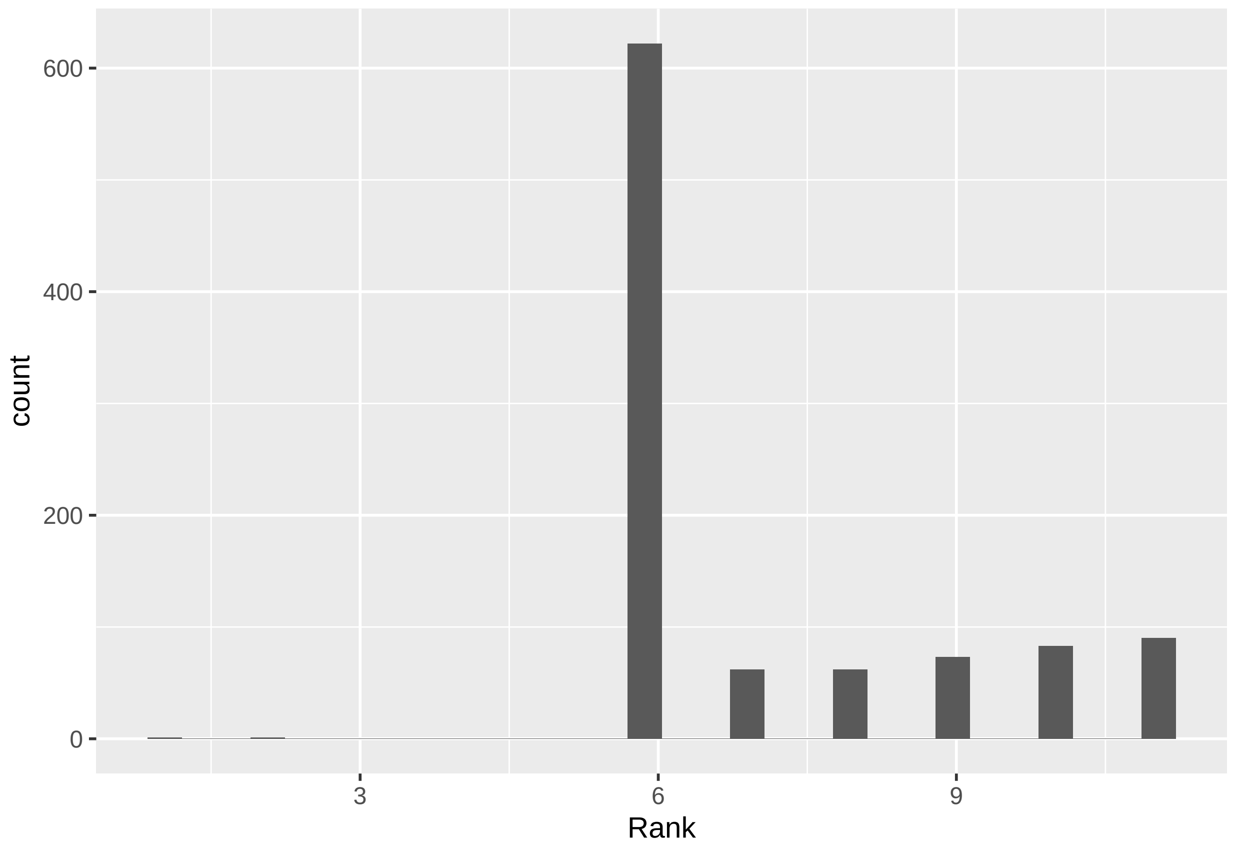
Above, we have importance sampled our ranks and probabilities. Now we briefly analyze them. First, we see that the ranks (remember we considered 12 guesses) were, for the most part, below 6. This is good, as it suggests that the "most obvious" guesses were not the true matches.

```r
ggplot(rank_prob_df, aes(x = i, y = Rank))+geom_point() + geom_path()
```



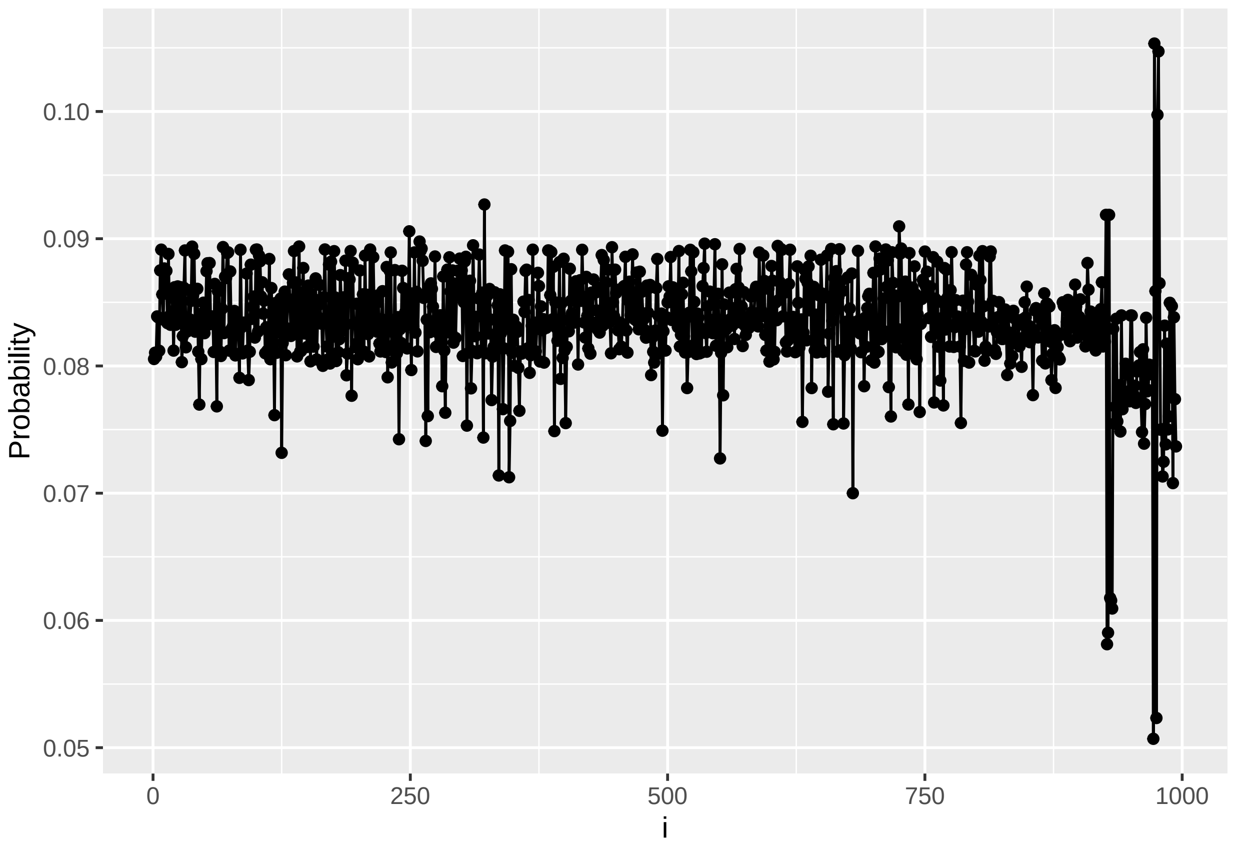
```r
ggplot(rank_prob_df, aes(x = Rank))+geom_histogram()
```

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

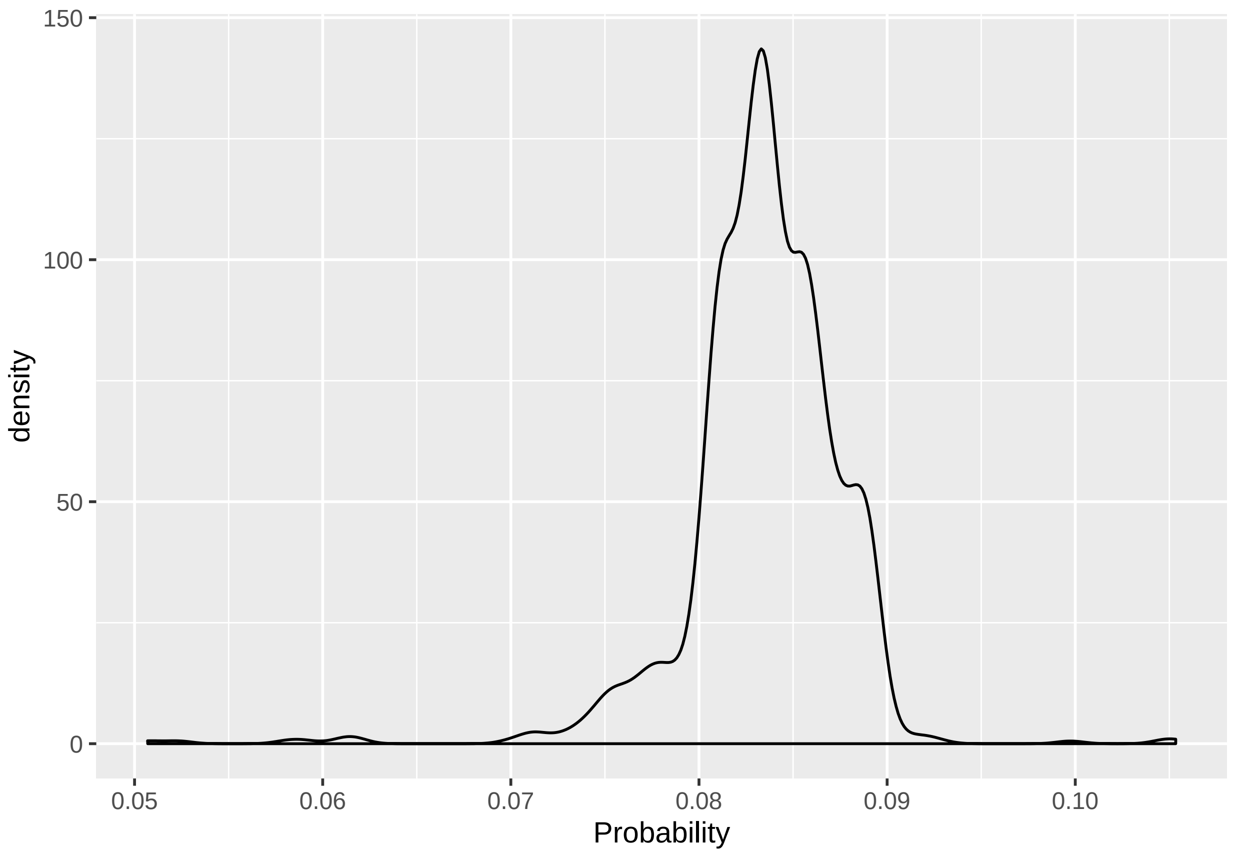


Further, we see that probabilities generally hovered around 8 percent – this seems like reasonably low odds for the intruder.

```r
ggplot(rank_prob_df, aes(x = i, y = Probability))+geom_point() + geom_path()
```



```r
ggplot(rank_prob_df, aes(x = Probability))+geom_density()
```



There are two CU's that stand out here, near the very bottom of `CEdata`. Specifically, these back-to-back CU's both have ranks of 2 (as compared to the rest, which are generally at or above 6); it might be worth examining whether this is in any way anomalous.

Here are the rows:

```
rank_prob_df%>%
  arrange(Rank)%>%
  head(5)
```

```
##   Rank Probability   i
## 1    1  0.10472178 977
## 2    2  0.10533894 973
## 3    6  0.08914084   8
## 4    6  0.08562508   9
## 5    6  0.08766330  11
```

And here is what is contained in them:

```
CEdata_org[rank_prob_df%>%
             filter(Rank <= 3)%>%
             pull(i), ]
```

```
##     UrbanRural Income Race Expenditure logIncome   logExp Race_2 Race_3 Race_4
## 973          0  73200    5     13006.8  11.20095 9.473228      0      0      0
## 977          0  43000    5     11168.4  10.66896 9.320844      0      0      0
##     Race_5 Race_6 LogIncome LogExpenditure
## 973      1      0      11.2            9.5
## 977      1      0      10.7            9.3
```

The scarcity of Race 5's in this dataset seem to be the principal source of the vulnerability here. There are 6 total Race 5's in the dataset; as such, these two synthetic entries are already vulnerable.

```r
CEdata_org%>%
  filter(Race == 5)
```

```
##   UrbanRural Income Race Expenditure logIncome   logExp Race_2 Race_3 Race_4
## 1          0  20712    5    3422.833  9.938469 8.138224      0      0      0
## 2          0  73200    5   13006.800 11.200951 9.473228      0      0      0
## 3          0  43000    5    7301.433 10.668955 8.895826      0      0      0
## 4          0   9600    5    2099.500  9.169518 7.649455      0      0      0
## 5          0  16738    5    8221.500  9.725437 9.014508      0      0      0
## 6          0  43000    5   11168.400 10.668955 9.320844      0      0      0
##   Race_5 Race_6 LogIncome LogExpenditure
## 1      1      0       9.9            8.1
## 2      1      0      11.2            9.5
## 3      1      0      10.7            8.9
## 4      1      0       9.2            7.6
## 5      1      0       9.7            9.0
## 6      1      0      10.7            9.3
```

```r
CEdata_syn%>%
  filter(Race == 5)
```

```
##   UrbanRural     Income Race Expenditure logIncome   logExp Race_2 Race_3
## 1          0   6182.932    5    3422.833  8.729548 8.138224      0      0
## 2          0  37403.526    5   13006.800 10.529520 9.473228      0      0
## 3          0  24685.089    5    7301.433 10.113955 8.895826      0      0
## 4          0 127877.992    5    2099.500 11.758832 7.649455      0      0
## 5          0  14631.277    5    8221.500  9.590917 9.014508      0      0
## 6          0  14972.091    5   11168.400  9.613943 9.320844      0      0
##   Race_4 Race_5 Race_6 LogIncome LogExpenditure
## 1      0      1      0       8.7            8.1
## 2      0      1      0      10.5            9.5
## 3      0      1      0      10.1            8.9
## 4      0      1      0      11.8            7.6
## 5      0      1      0       9.6            9.0
## 6      0      1      0       9.6            9.3
```