JAGS Model

Tom Park

2020-02-08

Basic Model: Ambulance + Overdose

Ambulance Call-outs Model

n_A: sample size
x_A: the total number who confirmed they did call an ambulance
p_A: probability of a person call an ambulance

$$x_A \sim Bin(n_A, p_A)$$

We assume $n_A = 1000, p_A = 0.8$.
Suppose the prior of p_A is noninformative.

$$p(p_A) \sim Beta(1, 1)$$

Overdose Model

Now we plug in this values into the overdose model and obtain possible O_t values **assuming we have U_t values.**

Also, we have priors.

$$z_t \sim N(\mu, \sigma^2)$$
$$\lambda_t^{OD} = \exp(z_t)$$
$$O_t \sim Poi(\lambda_t^{OD} N)$$
$$U_t \sim Bin(O_t, p_A)$$

For simplicity we set N =10000 for now. We need to generate reasonable U_t values first. Note that U_t comes from μ, σ following all the way through the overdose model.
$\mu = \log 0.05, \sigma = 1, N = 10000$.
We suppose survey data exists: (n_A, x_A) known.

We set for our prior parameters:

$$\mu \sim U(-10, 0)$$
$$\sigma \sim U(0, 5)$$

```
# install packages
if (!require(rjags)) install.packages("rjags", dependencies = TRUE)
if (!require(coda)) install.packages("coda", dependencies = TRUE)
if (!require(tidyverse)) install.packages("tidyverse", dependencies = TRUE)
if (!require(tinytex)) install.packages("tinytex", dependencies = TRUE)

library('rjags')
library('coda')
library('tidyverse')
library('tinytex')
```

The data is the same data from pymc3 with Python.
Todo: build a pipeline to connect the python (pymc3) and R (JAGS)

```
df <- read.csv('./basic_data.csv')
df$X <- NULL
head(df)
```

```
##    o_t  u_t x_a
## 1 2475 1969 799
## 2  262  217 798
## 3  318  253 795
## 4  149  119 816
## 5 1151  934 805
## 6   39   34 794
```

Now we set the model which defines the relations of overdose model and ambulance call model.

The model defined as follows.

```
cat("model{
## define the priors

p_a ~ dbeta(alpha, beta)
mu_z ~ dunif(mu_a, mu_b)
sigma_z ~ dunif(sigma_a, sigma_b)


 ## ambulance model
for (i in 1:n) {
  #Likelihood
  x_a[i] ~ dbin(p_a, n_a) # each survey result for month

}

# overdose
for (i in 1:n) {

  ## the latent variables
  z_t[i]~ dnorm(mu_z, 1/(sigma_z^2))
  lmb_t[i] <- exp(z_t[i])

  ## overdose model
  o_t[i] ~ dpois(lmb_t[i]*N) # total overdoses per month
  # Note that from pymc3 gamma was used instead of Pois dist
  u_t[i] ~ dbin(p_a, o_t[i]) # ambulanced overdoses per month
}

}", file='basic_model.txt')
```

Pre-set variables.

```
n_T <- length(df$o_t)
n_a <- 1000
N <- 10000
u_t <- df$u_t
x_a <- df$x_a
```

Define the list providing the values of the variables and the parameters for the priors of the model.

```
dat <- list(
            # priors for ambulance model
            'alpha' = 1,
            'beta' =  1,

            # priors for overdose model
            'mu_a'=(-10),
            'mu_b'=0,
            'sigma_a'=0,
            'sigma_b'=5,

            # likelihood
            'u_t'=u_t, # giving data
            'x_a'=x_a,   # giving data
            'N'=N, # the population 10000
            'n'= n_T, # total months 12
            'n_a'=n_a # survey size 1000

            )
```

Note: for the list object usually named 'data' or 'dat' in JAGS context, do not use arrow but use equal sign to define elements of the list.

```
chains=2
# inits = list()
simple.model <- jags.model(file='basic_model.txt',
                           data=dat,
                           n.chains = chains)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 27
##    Total graph size: 88
##
## Initializing model
```

```
simple.model
```

```
## JAGS model:
##
## model{
## ## define the priors
##
## p_a ~ dbeta(alpha, beta)
## mu_z ~ dunif(mu_a, mu_b)
## sigma_z ~ dunif(sigma_a, sigma_b)
##
##
##  ## ambulance model
## for (i in 1:n) {
##   #Likelihood
##   x_a[i] ~ dbin(p_a, n_a) # each survey result for month
```

```
##
## }
##
## # overdose
## for (i in 1:n) {
##
##   ## the latent variables
##   z_t[i]~ dnorm(mu_z, 1/(sigma_z^2))
##   lmb_t[i] <- exp(z_t[i])
##
##   ## overdose model
##   o_t[i] ~ dpois(lmb_t[i]*N) # total overdoses per month
##   # Note that from pymc3 gamma was used instead of Pois dist
##   u_t[i] ~ dbin(p_a, o_t[i]) # ambulanced overdoses per month
## }
##
## }
## Fully observed variables:
##  N alpha beta mu_a mu_b n n_a sigma_a sigma_b u_t x_a
```

O__t

```
params= c('o_t','p_a')
samples <- coda.samples(simple.model, params, n.iter = 1000)
# guess it's getting posterior samples ?
```

```
interations = 1000
burnin= floor(interations/2)
summary(window(samples), start=burnin)
```

```
##
## Iterations = 1001:2000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean        SD  Naive SE Time-series SE
## o_t[1]  2479.3015 28.200731 6.306e-01      0.9967438
## o_t[2]   273.1110  8.486206 1.898e-01      0.2429151
## o_t[3]   318.5485  9.157925 2.048e-01      0.2762984
## o_t[4]   149.8915  6.268682 1.402e-01      0.1807071
## o_t[5]  1175.2455 18.803298 4.205e-01      0.6518428
## o_t[6]    43.0195  3.406893 7.618e-02      0.0981701
## o_t[7]  3012.3680 31.213342 6.980e-01      1.1634191
## o_t[8]   246.8405  8.132904 1.819e-01      0.2317924
## o_t[9]   716.9095 14.509499 3.244e-01      0.4483718
## o_t[10]  387.6210 10.071002 2.252e-01      0.2844543
## o_t[11] 2082.3580 26.071569 5.830e-01      0.9749012
## o_t[12]   69.6605  4.370331 9.772e-02      0.1312886
## p_a        0.7942  0.003736 8.355e-05      0.0001564
##
```

```
## 2. Quantiles for each variable:
##
##              2.5%       25%       50%       75%     97.5%
## o_t[1]  2423.0000 2460.0000 2479.0000 2498.0000 2534.0000
## o_t[2]   258.0000  267.0000  273.0000  279.0000  291.0000
## o_t[3]   301.0000  312.0000  318.0000  325.0000  337.0000
## o_t[4]   138.0000  145.0000  150.0000  154.0000  162.0000
## o_t[5]  1140.0000 1162.0000 1175.0000 1188.0000 1213.0250
## o_t[6]    37.0000   40.0000   43.0000   45.0000   50.0000
## o_t[7]  2954.0000 2992.0000 3011.0000 3034.0000 3075.0250
## o_t[8]   232.0000  241.0000  246.0000  252.0000  263.0000
## o_t[9]   689.0000  707.0000  717.0000  727.0000  746.0000
## o_t[10]  369.0000  381.0000  387.0000  394.0000  408.0250
## o_t[11] 2032.0000 2065.0000 2082.0000 2100.0000 2136.0000
## o_t[12]   62.0000   67.0000   69.0000   73.0000   79.0000
## p_a        0.7867    0.7918    0.7942    0.7968    0.8018
```

q1: what is the equivalent plot that we can see we have enough iteration?

Boxplots of O_t

q2: I see two elements from the samples list. Which one I should use it or should I use both?

```
pst_mtx = as.matrix(samples)
temp = pst_mtx[,1:12]
p_a <- pst_mtx[,13]
head(temp)
```

```
##      o_t[1] o_t[2] o_t[3] o_t[4] o_t[5] o_t[6] o_t[7] o_t[8] o_t[9]
## [1,]   2466    275    325    147   1157     45   3044    246    726
## [2,]   2494    270    304    145   1167     42   3024    236    713
## [3,]   2511    278    334    153   1170     44   3068    241    722
## [4,]   2502    278    325    159   1187     51   3018    238    705
## [5,]   2454    266    318    166   1183     44   3053    261    720
## [6,]   2552    263    313    149   1194     49   2995    238    720
##      o_t[10] o_t[11] o_t[12]
## [1,]     379    2075      67
## [2,]     378    2114      72
## [3,]     385    2069      71
## [4,]     378    2086      73
## [5,]     384    2049      71
## [6,]     389    2087      69
```

```
length(p_a)
```

```
## [1] 2000
```

```
colnames(temp) <- seq(1,12)
df_o_t <- as.data.frame(temp)
head(df_o_t)
```

```
##      1   2   3   4    5  6    7   8   9  10   11 12
```

```
## 1 2466 275 325 147 1157 45 3044 246 726 379 2075 67
## 2 2494 270 304 145 1167 42 3024 236 713 378 2114 72
## 3 2511 278 334 153 1170 44 3068 241 722 385 2069 71
## 4 2502 278 325 159 1187 51 3018 238 705 378 2086 73
## 5 2454 266 318 166 1183 44 3053 261 720 384 2049 71
## 6 2552 263 313 149 1194 49 2995 238 720 389 2087 69
```

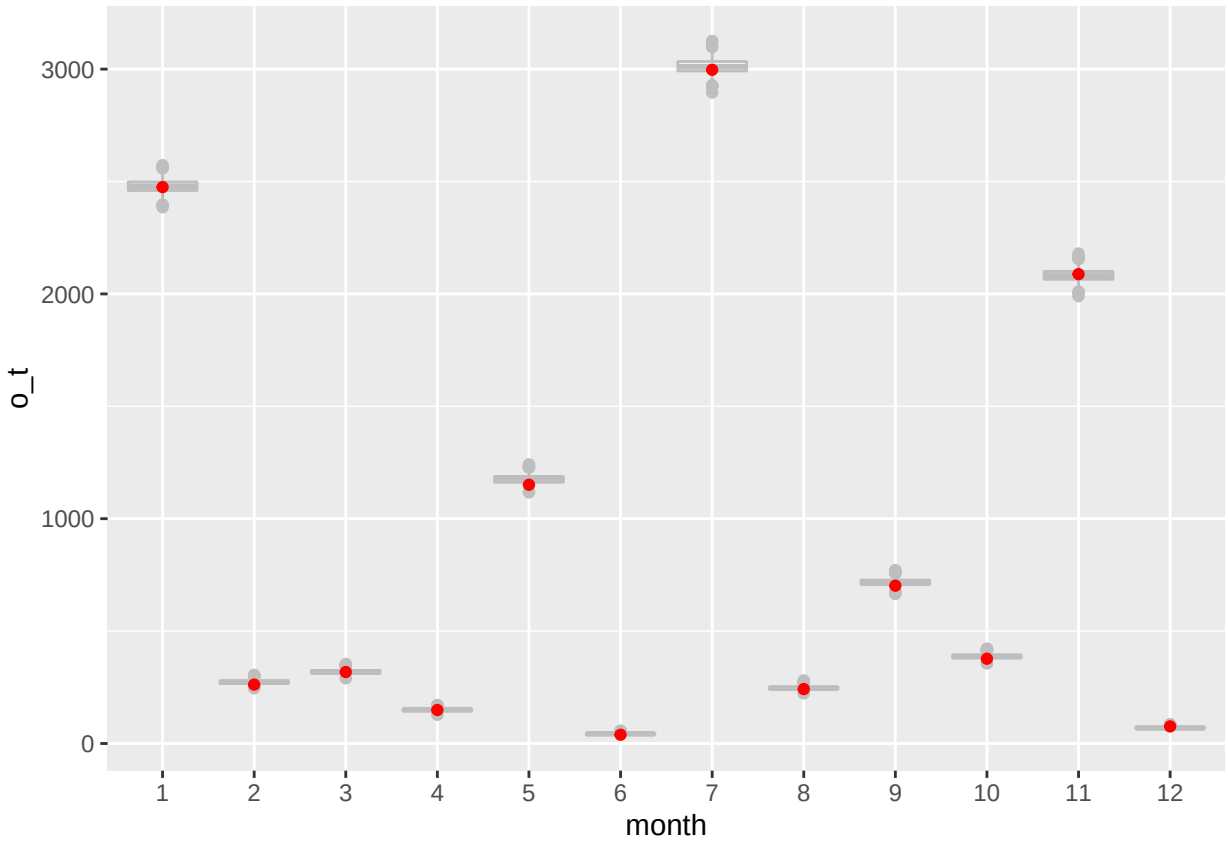
```
trace_o_t <- gather(df_o_t, key = 'month',value = 'o_t')
trace_o_t$month <- factor(trace_o_t$month,levels = seq(1,12))
str(trace_o_t)
```

```
## 'data.frame':    24000 obs. of  2 variables:
##  $ month: Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ o_t  : num  2466 2494 2511 2502 2454 ...
```

```
head(trace_o_t,n = 24)
```

```
##    month  o_t
## 1      1 2466
## 2      1 2494
## 3      1 2511
## 4      1 2502
## 5      1 2454
## 6      1 2552
## 7      1 2479
## 8      1 2486
## 9      1 2488
## 10     1 2457
## 11     1 2509
## 12     1 2492
## 13     1 2501
## 14     1 2462
## 15     1 2477
## 16     1 2436
## 17     1 2501
## 18     1 2479
## 19     1 2516
## 20     1 2497
## 21     1 2513
## 22     1 2523
## 23     1 2532
## 24     1 2492
```

```
# real values of the data set from pymc3 samples
o_t_values=data.frame('month'=seq(1,12),'o_t'=df$o_t)
# factorizing the months for box plot visualization
o_t_values$month <- factor(o_t_values$month,levels = seq(1,12))

ggplot()+
  # boxplot from the trace
  geom_boxplot(aes(x=month,y=o_t), color='grey',data = trace_o_t)+
  # real values as red dots
  geom_point(aes(x=o_t_values$month, y=o_t_values$o_t),color='red')
```



Predictive Posterior Checks

```
df_u_t <- data.frame()
m=length(p_a)
for (i in 1:m) {
  obs <- rbinom(n=12,size = as.numeric(df_o_t[i,]),prob =p_a[i])
  df_u_t=rbind(df_u_t,obs)
}
colnames(df_u_t) <- factor(seq(1,12),levels = seq(1,12))
str(df_u_t)
```

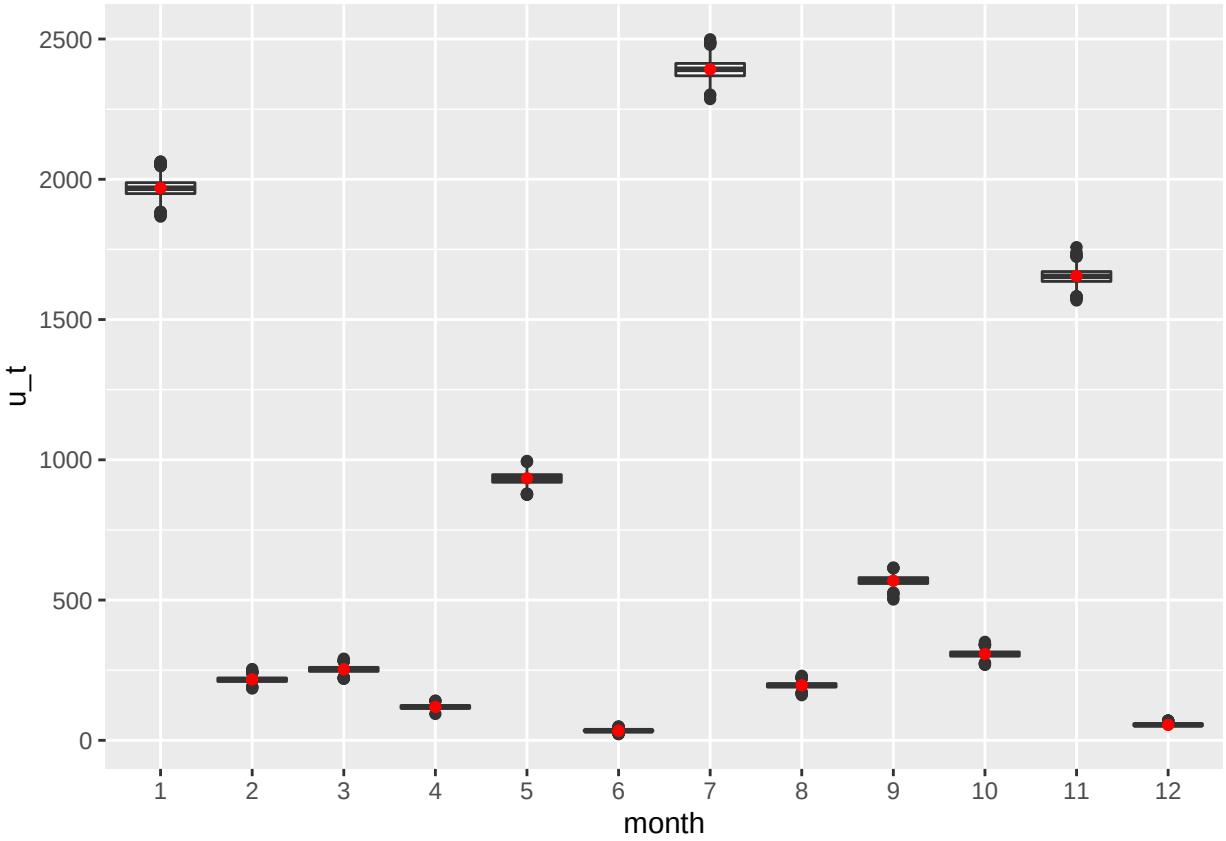
```
## 'data.frame':    2000 obs. of  12 variables:
##  $ 1 : int  1971 1983 2001 1988 1947 2048 1963 1992 1967 1953 ...
##  $ 2 : int  219 211 217 222 204 216 226 238 215 223 ...
##  $ 3 : int  259 248 269 255 247 247 263 243 243 263 ...
##  $ 4 : int  122 114 123 128 133 116 110 128 117 114 ...
##  $ 5 : int  898 910 916 941 923 963 960 902 894 936 ...
##  $ 6 : int  38 36 34 40 32 38 41 30 34 37 ...
##  $ 7 : int  2434 2383 2499 2388 2366 2360 2433 2391 2431 2407 ...
##  $ 8 : int  196 182 181 191 208 195 187 197 195 184 ...
##  $ 9 : int  573 576 578 564 569 589 586 593 585 561 ...
##  $ 10: int  301 283 294 302 309 308 294 310 315 317 ...
##  $ 11: int  1656 1698 1617 1624 1649 1641 1659 1677 1692 1630 ...
##  $ 12: int  54 55 61 57 59 56 56 49 60 55 ...
```

U_t: Predictive Posterior Checks

```
ppc_u_t <- gather(df_u_t, key = 'month',value = 'u_t')
summary(ppc_u_t)
```

```
##     month                u_t
##  Length:24000       Min.   :  22.0
##  Class :character   1st Qu.: 156.0
##  Mode  :character   Median : 279.5
##                     Mean   : 725.0
##                     3rd Qu.:1139.0
##                     Max.   :2499.0
```

```
u_t_values=data.frame('month'=seq(1,12),'u_t'=df$u_t)
# u_t_values$month <- factor(u_t_values$month,levels = seq(1,12))
ppc_u_t$month <- factor(ppc_u_t$month,levels = seq(1,12))

ggplot()+geom_boxplot(aes(x=month,y=u_t),data = ppc_u_t)+geom_point(aes(x=u_t_values$month, y=u_t_values
```



todo: finish this part.

x_A: Predictive Posterior Checks

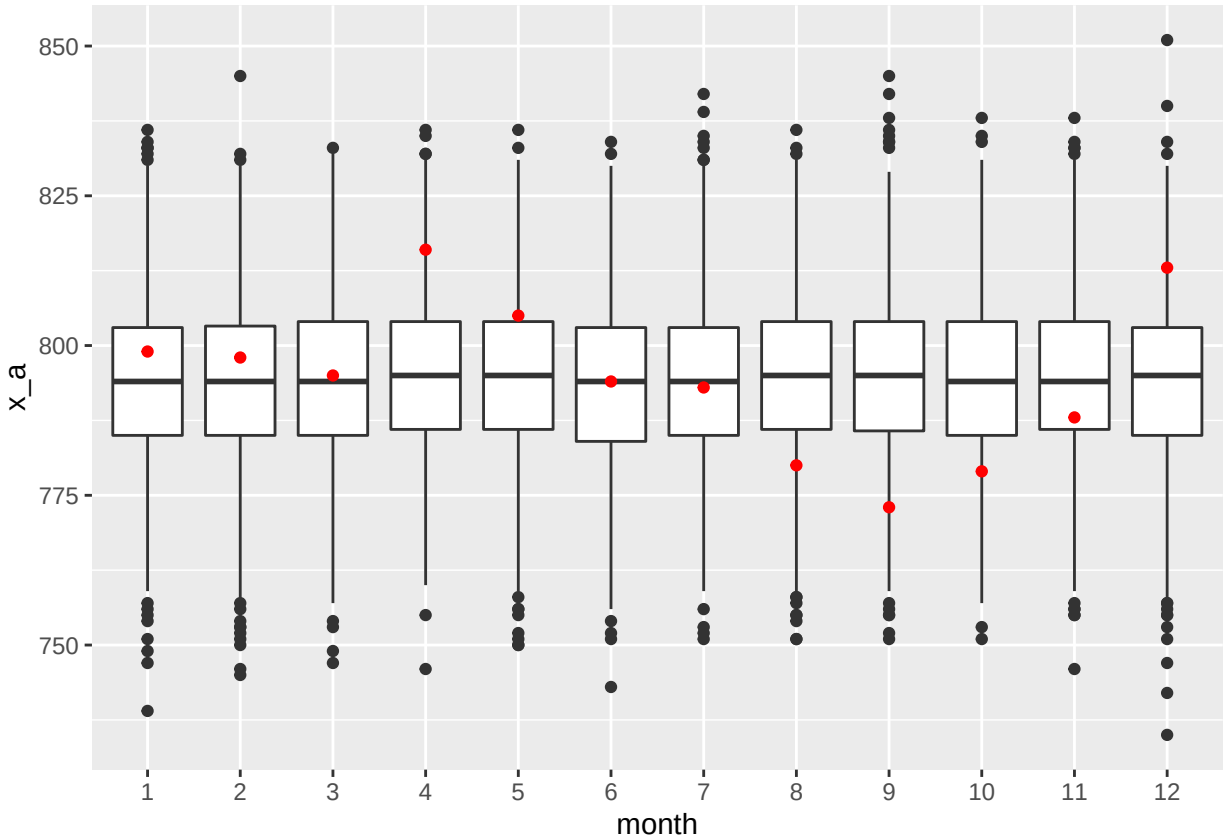
```
df_x_a <- vector()
for (i in 1:n_T) {
  obs <- rbinom(m,n_a,p_a)
  df_x_a <- cbind(df_x_a,obs)
```

```
}

head(df_x_a)
```

```
##      obs obs obs obs obs obs obs obs obs obs obs obs
## [1,] 775 807 810 809 793 773 788 812 802 788 794 809
## [2,] 776 797 769 799 771 791 766 800 796 808 800 784
## [3,] 788 805 796 771 806 818 803 792 802 793 792 811
## [4,] 795 796 782 804 789 809 779 799 803 767 798 783
## [5,] 819 795 800 775 814 790 783 771 796 816 782 793
## [6,] 818 807 792 813 796 767 805 796 795 786 798 791
```

```
colnames(df_x_a) <- seq(1:12)
```

```
df_x_a <- as.data.frame(df_x_a)
ppc_x_a <- gather(df_x_a, key = 'month',value = 'x_a')


ppc_x_a$month <- factor(ppc_x_a$month,levels = seq(1,12))


x_a_values=data.frame('month'=seq(1,12),'x_a'=df$x_a)
x_a_values$month <- factor(x_a_values$month,levels = seq(1,12))

ggplot()+geom_boxplot(aes(x=month,y=x_a),data = ppc_x_a)+geom_point(aes(x=x_a_values$month, y=x_a_values
```



Contamination of p_A

Now, suppose the survey data gives us a wrong (biased) p_A value.

Bias $= \theta - \hat{\theta} = p_A - \hat{p}_A$
$\hat{p}_A = p_A + bias(p_A)$

Three more data sets are given: unbiased, overestimated, underestimated p_A.

```
## write a function that led to compare o_t, u_t and x_a.
#first we need a fuction that gives us data, model, trace, and ppc.
test_robust <- function(file=file, random=1, N=10000, p_a=0.8, bias = -0.2, n_a=1000, n_T=12) {
  df <- read.csv(file = file)
  df$X <- NULL
  df$month <- seq(1:12)
# obtain the (biased) data
  dat <- list(
            # priors for ambulance model
            'alpha' = 1,
            'beta' =  1,

            # priors for overdose model
            'mu_a'=(-10),
            'mu_b'=0,
            'sigma_a'=0,
            'sigma_b'=5,

            # likelihood
            'u_t'=df$u_t, # giving data
            'x_a'=df$x_a,   # giving data
            'N'=N, # the population 10000
            'n'= n_T, # total months 12
            'n_a'=n_a # survey size 1000

            )
# run the model
  chains=2
  # target 1 to save
  simple.model <- jags.model(file='basic_model.txt',
                          data=dat,
                          n.chains = chains)

# get the samples of O_t, p_a
  params= c('o_t','p_a')
  # target 2 to save
  samples <- coda.samples(simple.model, params, n.iter = 2000)

  # tidy p_a: target 3
  p_a <- pst_mtx[,13]

  ## tidy o_t
  pst_mtx = as.matrix(samples)
  temp = pst_mtx[,1:12]
  colnames(temp) <- seq(1,12)
  df_o_t <- as.data.frame(temp)
```

```
  trace_o_t <- gather(df_o_t, key = 'month',value = 'o_t')
  trace_o_t$month <- factor(trace_o_t$month,levels = seq(1,12))


  # tidy u_t pp samples
  df_u_t <- data.frame()
  for (i in 1:m) {
   obs <- rbinom(n=12,size = as.numeric(df_o_t[i,]),prob =p_a[i])
   df_u_t=rbind(df_u_t,obs)
  }
  colnames(df_u_t) <- factor(seq(1,12),levels = seq(1,12))
  ppc_u_t <- gather(df_u_t, key = 'month',value = 'u_t')
  ppc_u_t$month <- factor(ppc_u_t$month,levels = seq(1,12))

  ## tidy x_a
  df_x_a <- vector()
  for (i in 1:n_T) {
    obs <- rbinom(m,n_a,p_a)
    df_x_a <- cbind(df_x_a,obs)
  }
  colnames(df_x_a) <- seq(1:12)
  df_x_a <- as.data.frame(df_x_a)
  ppc_x_a <- gather(df_x_a, key = 'month',value = 'x_a')
  ppc_x_a$month <- factor(ppc_x_a$month,levels = seq(1,12))
  ppc = list('u_t'=ppc_u_t, 'x_a'=ppc_x_a)
  mylist=list('data'=df,'model'=simple.model,'trace'=trace_o_t,'ppc'=ppc)
  return (mylist)
}

# then visualization function which gives us three different types of boxplots.
```

```
my_list_unbiased = test_robust(file = './basic_data.csv', bias =0)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 27
##    Total graph size: 88
##
## Initializing model
```

```
my_list_under = test_robust(file = './under_p_a_data.csv',bias=-0.2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 27
##    Total graph size: 88
##
## Initializing model
```

```
my_list_over = test_robust(file='./over_p_a_data.csv',bias = +0.1)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 27
##    Total graph size: 88
##
## Initializing model
```

```
print(my_list_unbiased$data$u_t)
```

```
##  [1] 1969  217  253  119  934   34 2392  196  569  308 1655   55
```

```
print(my_list_under$data$u_t)
```

```
##  [1] 1969  217  253  119  934   34 2392  196  569  308 1655   55
```

```
print(my_list_over$data$u_t)
```

```
##  [1] 1969  217  253  119  934   34 2392  196  569  308 1655   55
```

```
print(my_list_unbiased$data$x_a)
```

```
##  [1] 799 798 795 816 805 794 793 780 773 779 788 813
```

```
print(my_list_under$data$x_a)
```

```
##  [1] 602 598 622 608 595 593 575 638 586 618 574 582
```

```
print(my_list_over$data$x_a)
```

```
##  [1] 898 891 910 902 894 921 881 909 879 913 908 914
```

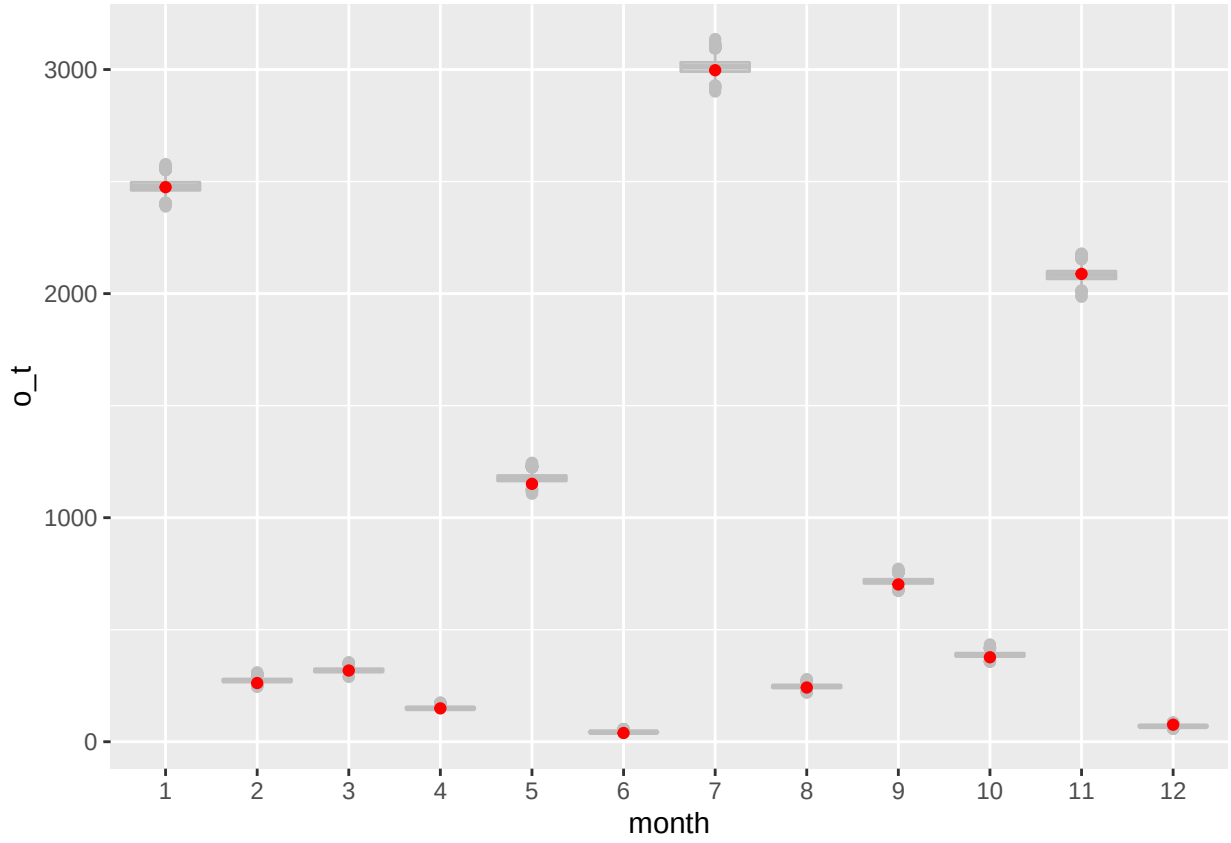
```
 head(my_list_unbiased$ppc$u_t)
```

```
##   month  u_t
## 1     1 1944
## 2     1 1911
## 3     1 1948
## 4     1 1938
## 5     1 1935
## 6     1 1948
```

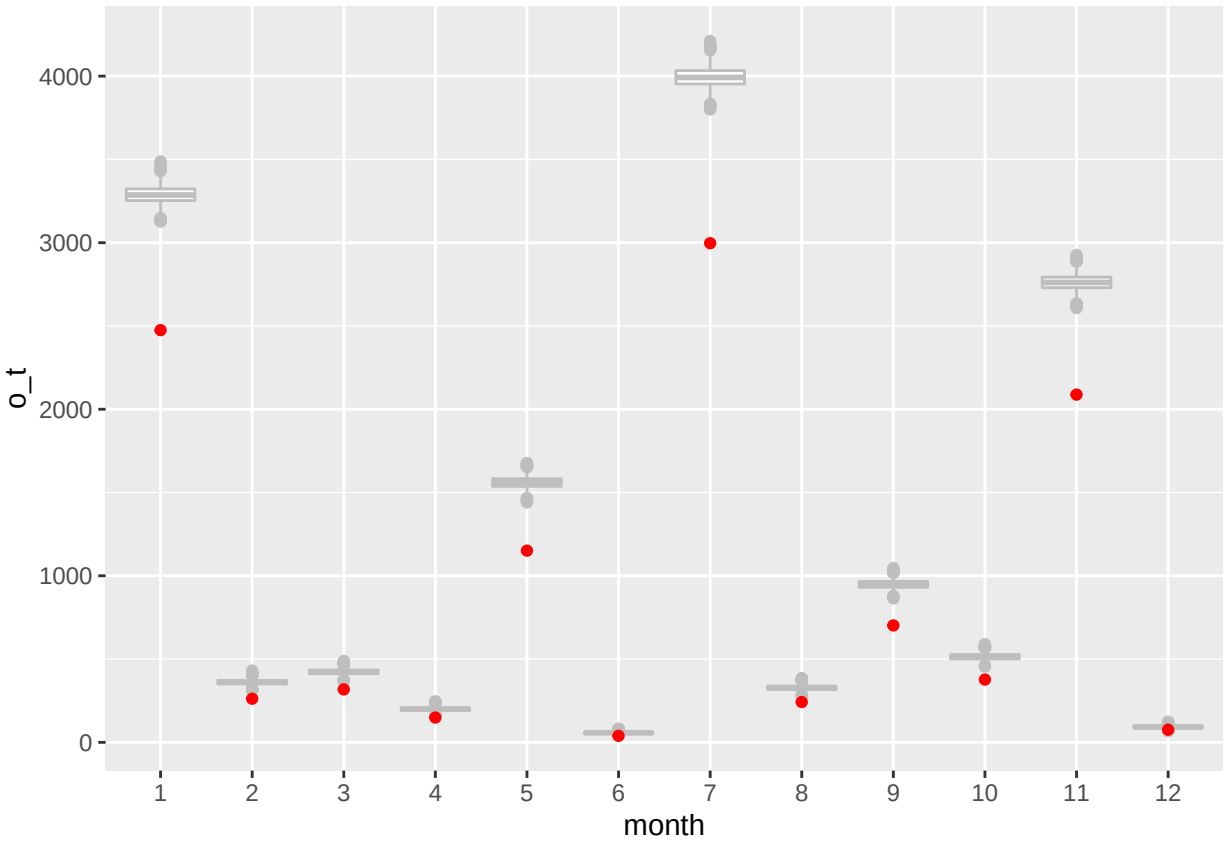
```
# finish this function
visualization <- function(mylist= None, post= F, u_t = F, x_a = F, string='string') {
  data= mylist$data
  if (post == T) {
    # boxplots of o_t
    trace_o_t = mylist$trace
  p <- ggplot()+
  # boxplot from the trace
  geom_boxplot(aes(x=month,y=o_t), color='grey',data = trace_o_t)+
  # real values as red dots
  geom_point(aes(x=o_t_values$month, y=o_t_values$o_t),color='red')

  }
```

```
  if (u_t == T) {
    ppc_u_t = mylist$ppc$u_t
    # boxplots of u_t
  p <- ggplot()+geom_boxplot(aes(x=month,y=u_t),data = ppc_u_t)+geom_point(aes(x=data$month, y=data$u_t)

  }

  if (x_a == T) {
    ppc_x_a = mylist$ppc$x_a
    p <- ggplot()+geom_boxplot(aes(x=month,y=x_a),data = ppc_x_a)+geom_point(aes(x=data$month, y=data$x_
  }
  return(p)
}
```

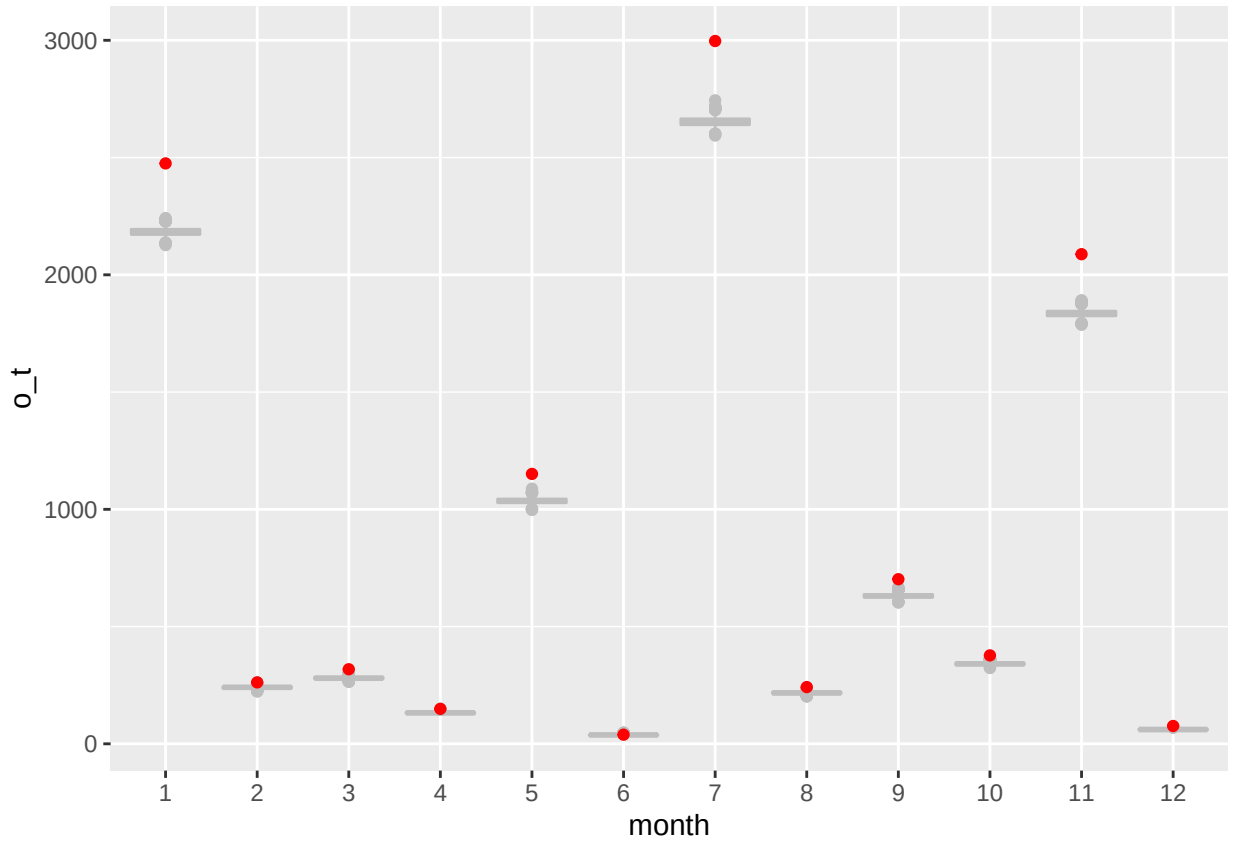
```
visualization(my_list_unbiased, post=T)
```



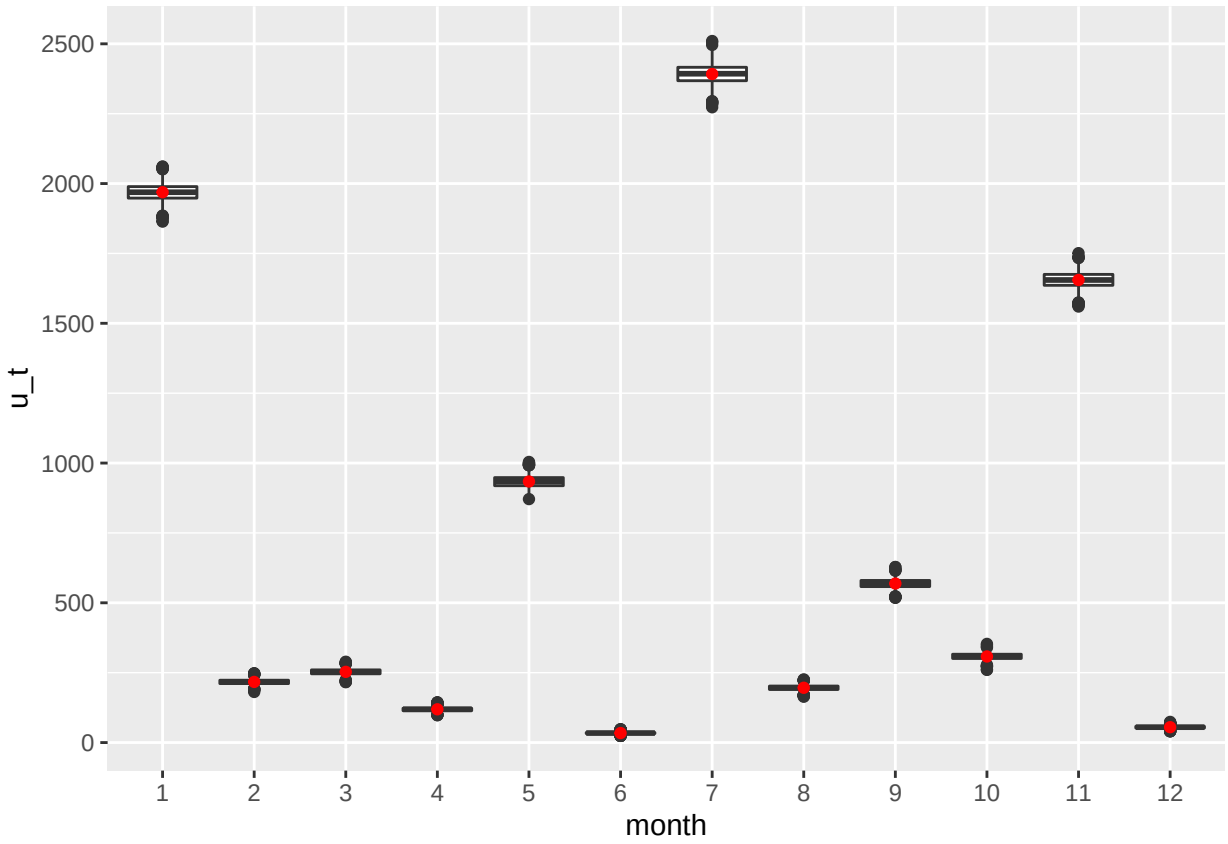

```
visualization(my_list_under,post= T)
```



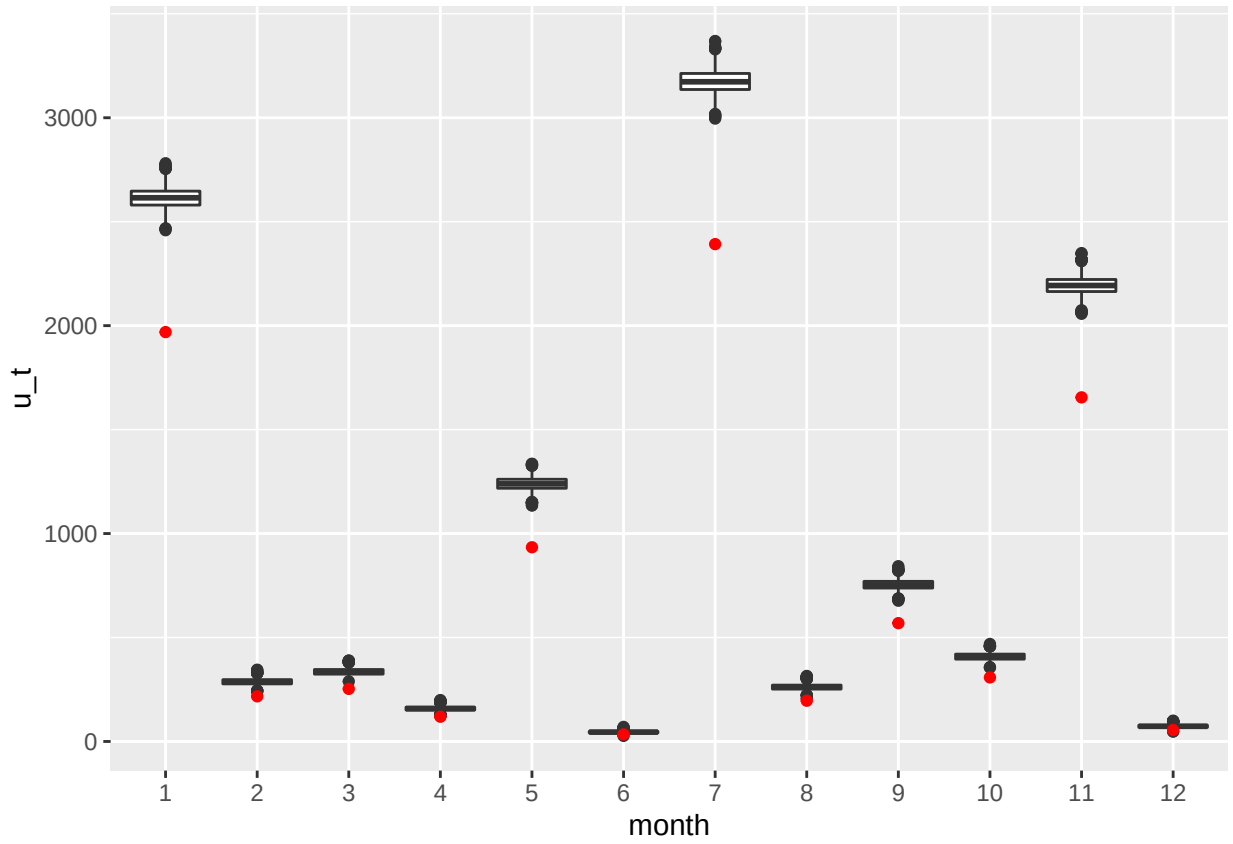

```
visualization(my_list_over,post=T)
```



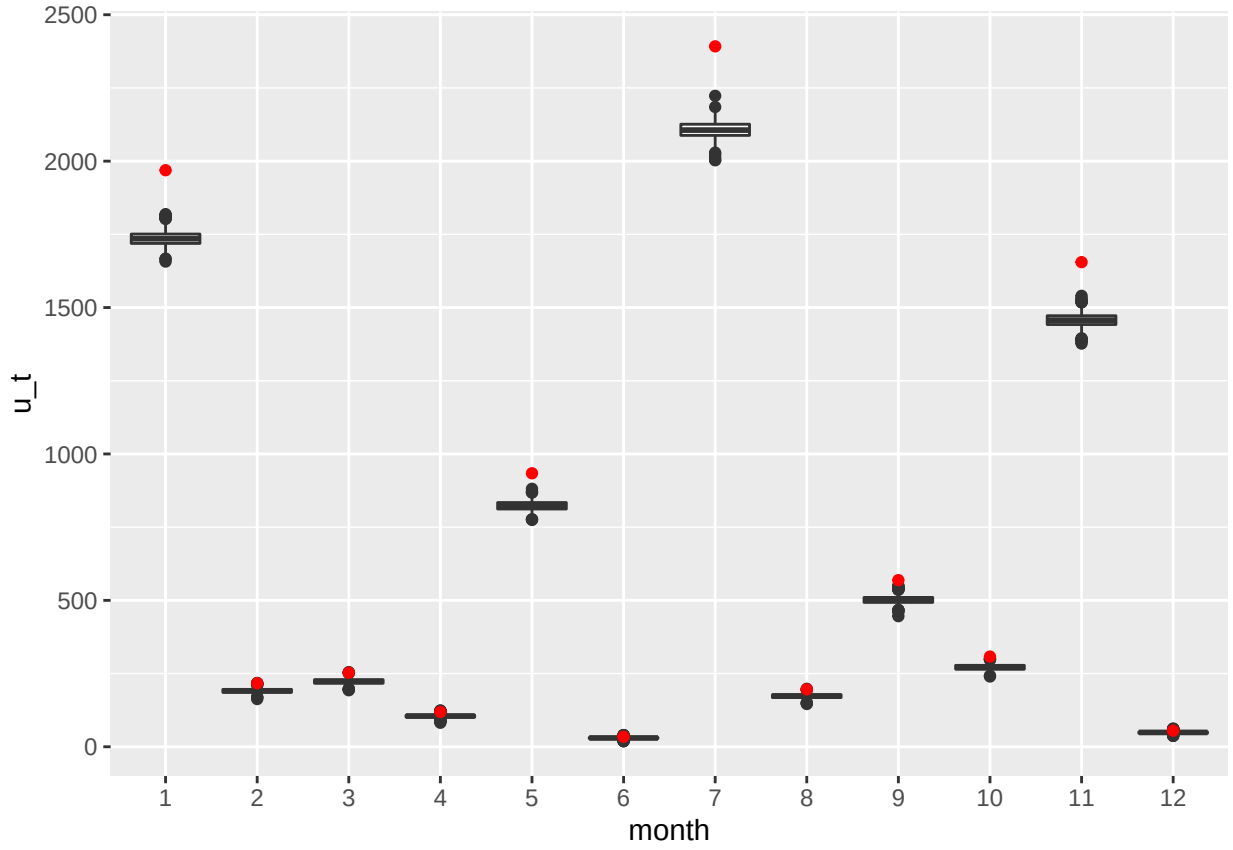

```
visualization(my_list_unbiased, u_t=T)
```



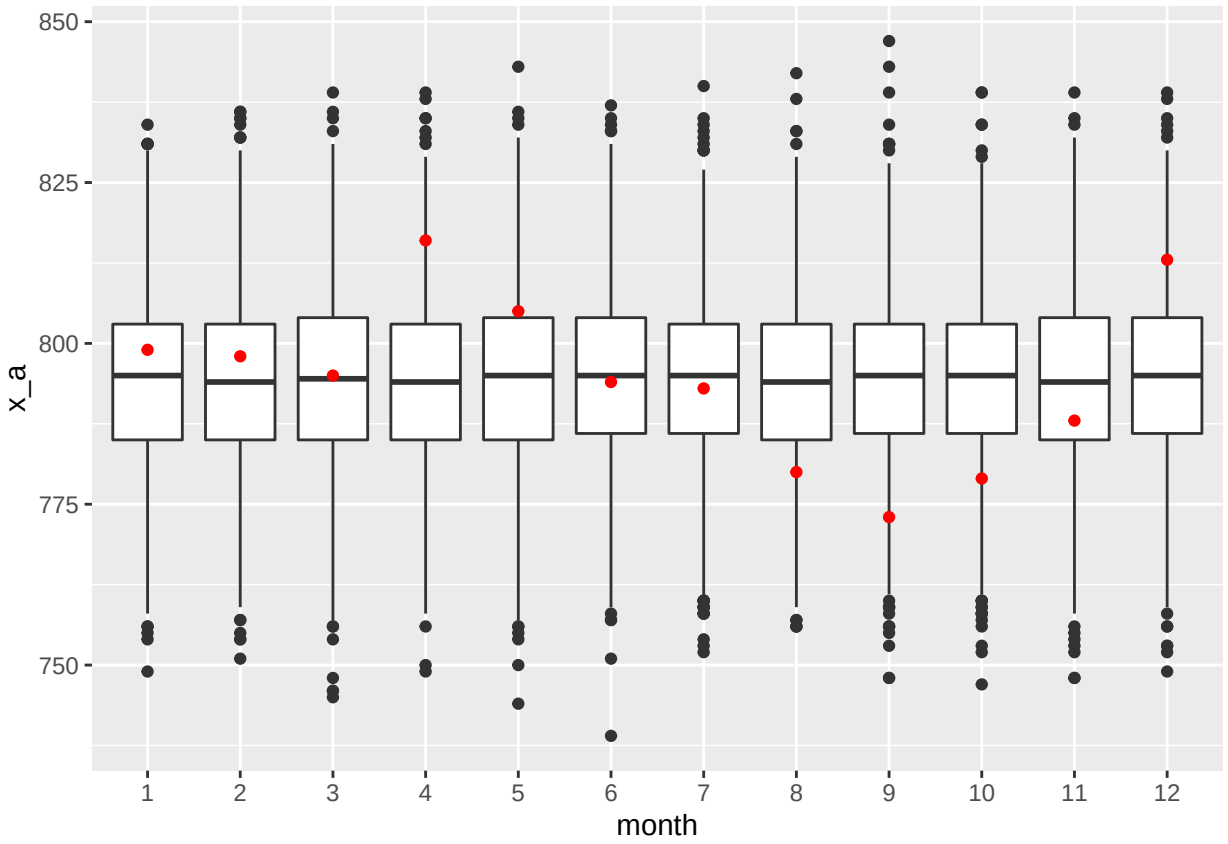

```
visualization(my_list_under,u_t= T)
```



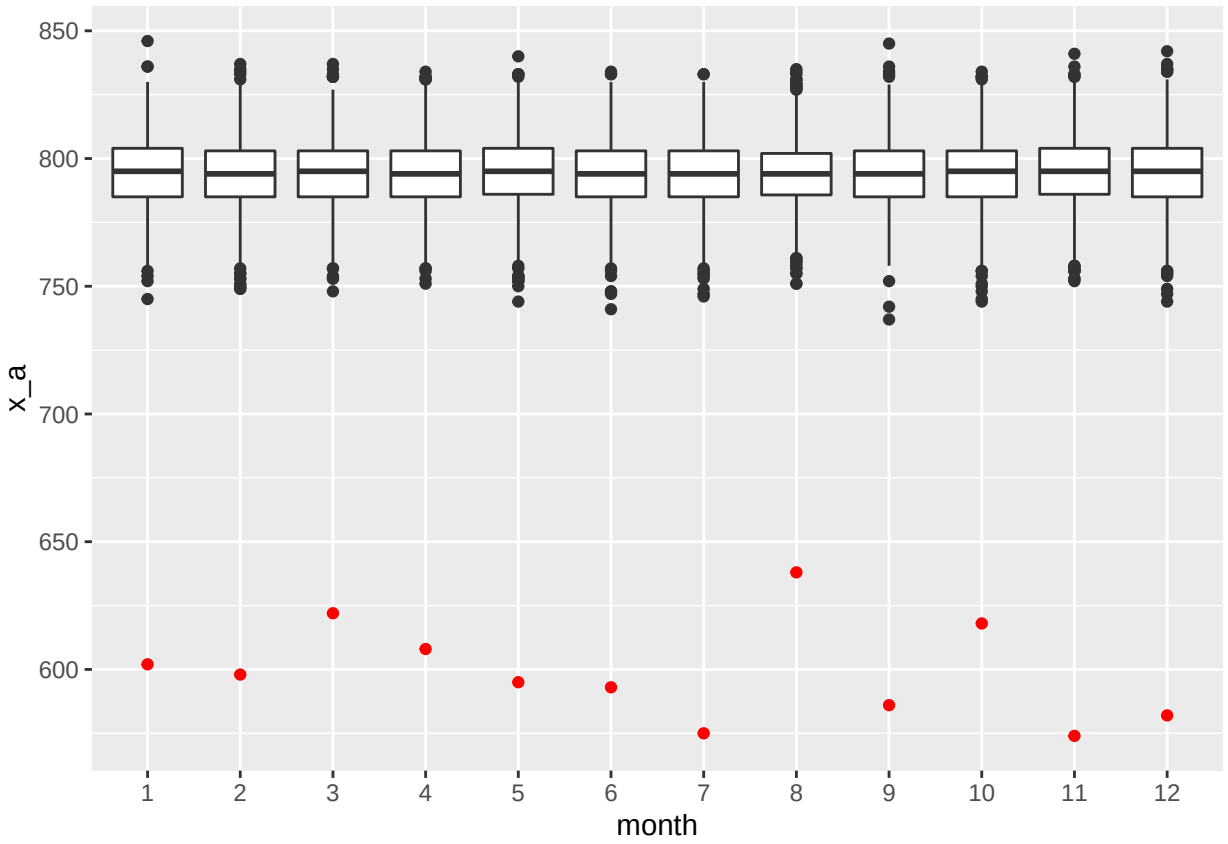

```
visualization(my_list_over,u_t=T)
```



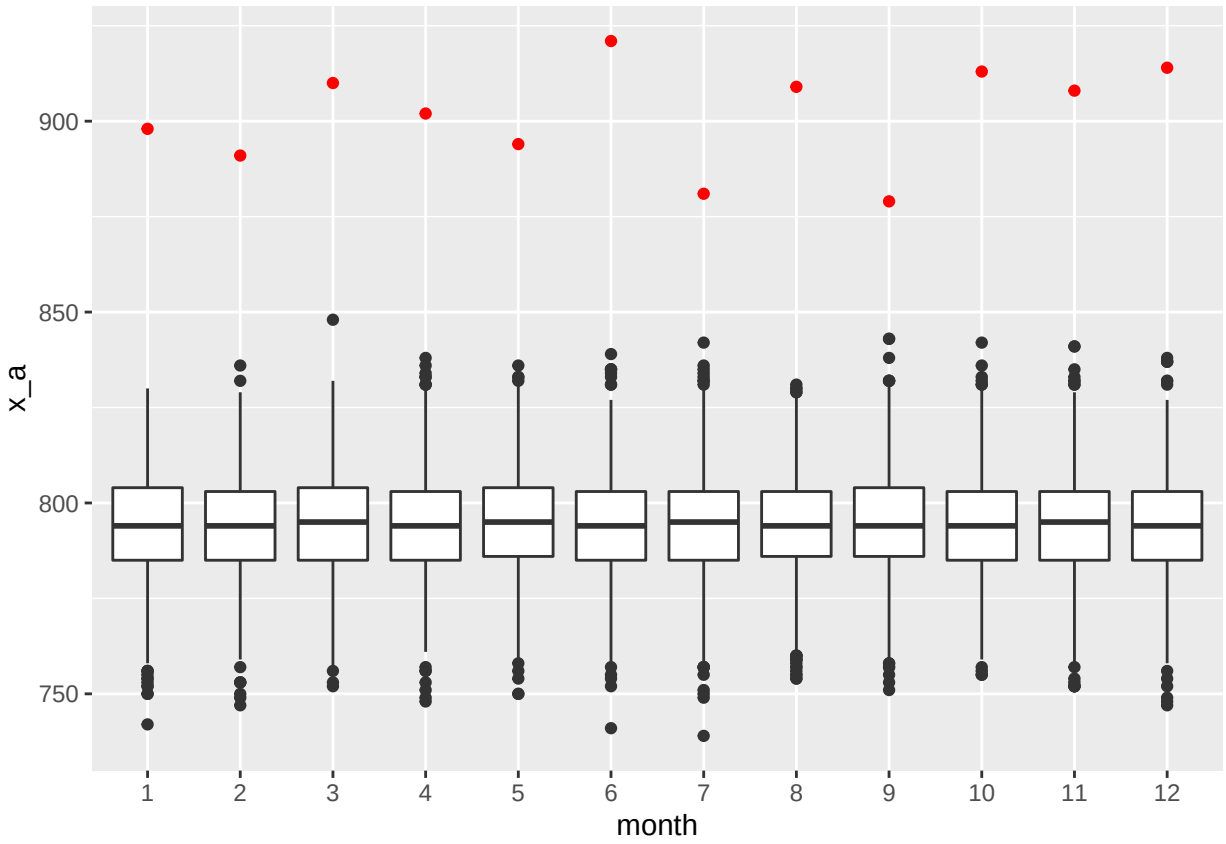

```
visualization(my_list_unbiased, x_a=T)
```




```
visualization(my_list_under,x_a= T)
```




```
visualization(my_list_over,x_a=T)
```



Trim here and make a function and give us the plots.

Reference

first tutorial
second tutorial
JAGS manual
error handling guide