Bayesian_latent_class_analysis_HiP_MGR

Elise_van_der_Heijden

10/12/2021

```
# Bayesian latent class model: Hui-Walter & Bayesian Logistic regression
library(rjags)
```

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

```
## Linked to JAGS 4.3.0
```

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

```
library(ggplot2)
```

```
# Load database HiPMGR
btb <- read.delim("~/Re-Analysis Ch3 Bayesian/Database HiPMGR.txt")
View(btb)
str(btb)
```

```
## 'data.frame':    997 obs. of  9 variables:
##  $ Animal_ID     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Herd_ID       : chr  "A" "A" "A" "A" ...
##  $ Study_Year    : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
##  $ Park_ID       : chr  "HiP" "HiP" "HiP" "HiP" ...
##  $ Study_site    : chr  "Masinda" "Masinda" "Masinda" "Masinda" ...
##  $ Sex           : chr  "Female" "Male" "Female" "Male" ...
##  $ Age           : chr  "Adult" "Juvenile" "Adult" "Juvenile" ...
##  $ TST_Result    : chr  "Positive" "Negative" "Negative" "Negative" ...
##  $ TBELISA_Result: chr  "Negative" "Negative" "Positive" "Negative" ...
```

```
# Make a table of the various test result combinations stratified for the two populations
pop <- t(matrix(as.vector(table(btb$TBELISA_Result, btb$TST_Result, btb$Park_ID)),4,2))
View(pop)

# If you want to make this table without stratifying on population
# pop.all <- t(matrix(as.vector(table(btb$TST_Result, btb$TBELISA_Result)),4))

# Give the table column- and rownames and have a look at the data
colnames(pop) <- c("-/-", "-/+", "+/-", "+/+") # test 1 = TST; test 2 = TB ELISA
rownames(pop) <- c("HiP", "MGR")
pop
```

```
##     -/- -/+ +/- +/+
## HiP 597  86  69  14
## MGR 151  39  32   9
```

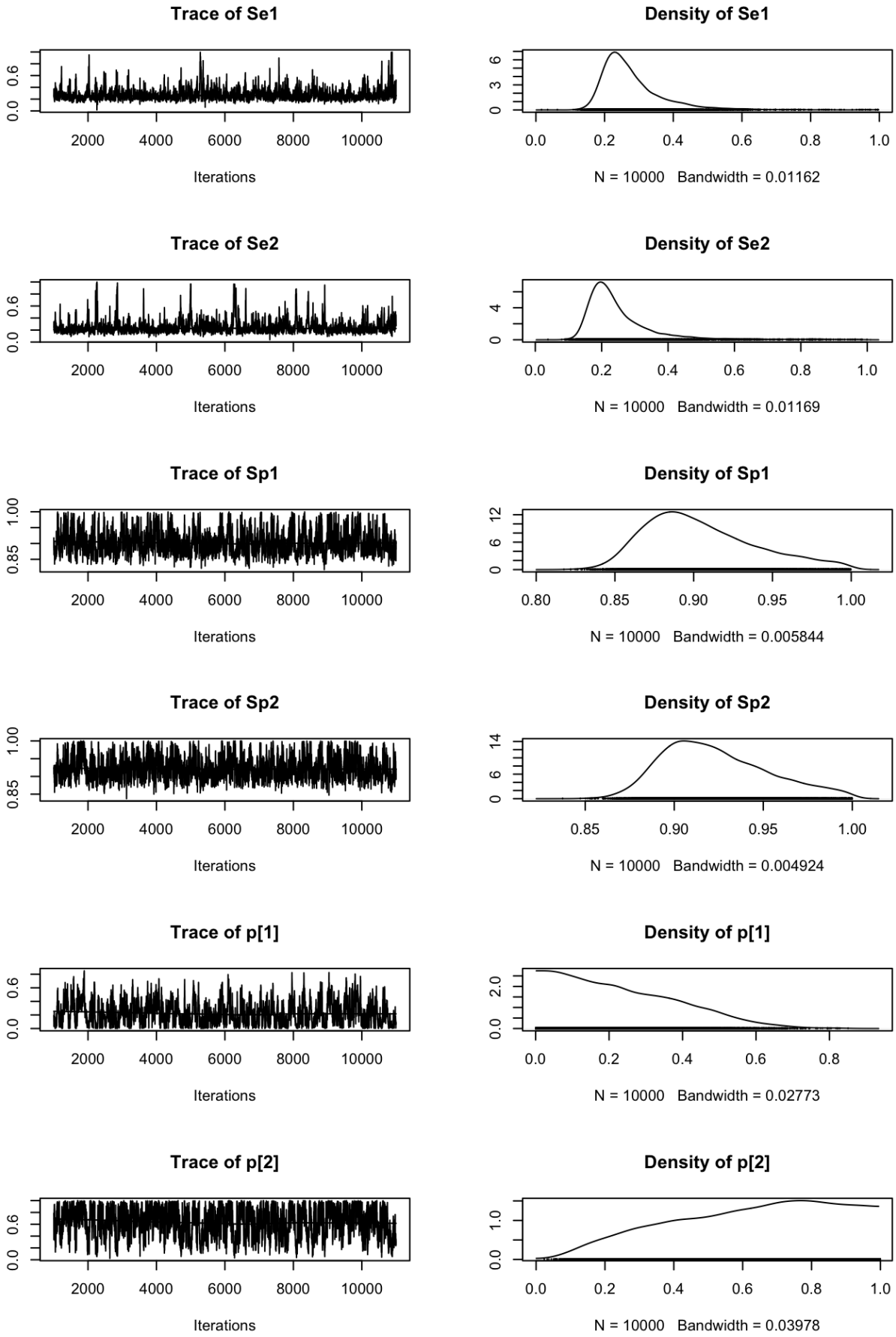
```
# Side note:
# If you want to order the dataframe like in some other publications, change it to:
# pop2 <- pop[,c(4,3,2,1)]
# You will then have to change the order of the definitions in the model too!!!!
# They are now in 4 to 1 order as we have the data from -/- to +/+ like in the Bronsvoort
et al. 2019 publication.

np <- nrow(pop)
# Create vector of population counts
n <- apply(pop,1,sum)
```

```
# Define the model
model1 <- "model{
  for (i in 1:2) {
    p[i] ~ dbeta(1,1)
    pop[i, 1:4] ~ dmulti(par[i,1:4], n[i])
    par[i, 4] <- p[i]*Se1*Se2 + (1-p[i])*(1-Sp1)*(1-Sp2)
    par[i, 2] <- p[i]*Se1*(1-Se2) + (1-p[i])*(1-Sp1)*Sp2
    par[i, 3] <- p[i]*(1-Se1)*Se2 + (1-p[i])*Sp1*(1-Sp2)
    par[i, 1] <- p[i]*(1-Se1)*(1-Se2) + (1-p[i])*Sp1*Sp2
    }
  ## priors
  Se1 ~ dbeta(1, 1)
  Sp1 ~ dbeta(1, 1)
  Se2 ~ dbeta(1, 1)
  Sp2 ~ dbeta(1, 1)

  # these priors are uninformative
  # try this model again with informative priors based on previous research of these test
s in buffaloes

}"

# Compile the model
model1_jags <- jags.model(textConnection(model1),inits=list(.RNG.name = "base::Wichmann-H
ill", .RNG.seed = 2018),
                          data=list(pop=pop, n=n))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2
##    Unobserved stochastic nodes: 6
##    Total graph size: 43
##
## Initializing model
```

```
        # in Bronsvoort et al. own INITS defined (based on??) and also n.chains=3, n.adap
t=50000)
        # in datacamp course usually inits=list(.RNG.name = "base::Wichmann-Hill", .RNG.s
eed = 2018) which is what was used here.
        # should find out how to define our own inits - this seems key for the outcome!


# Simulate the model
model1_sim <- coda.samples(model=model1_jags, variable.names=c("p", "Se1", "Se2", "Sp1",
"Sp2"),
                           n.iter=10000)
        # in Bronsvoort et al. n.iter=250000, n.thin=100
        # the higher the number of iterarions, the more stable and reliable the outcome
        # used 10000 here for now for speed of estimation, should probably be higher!
```

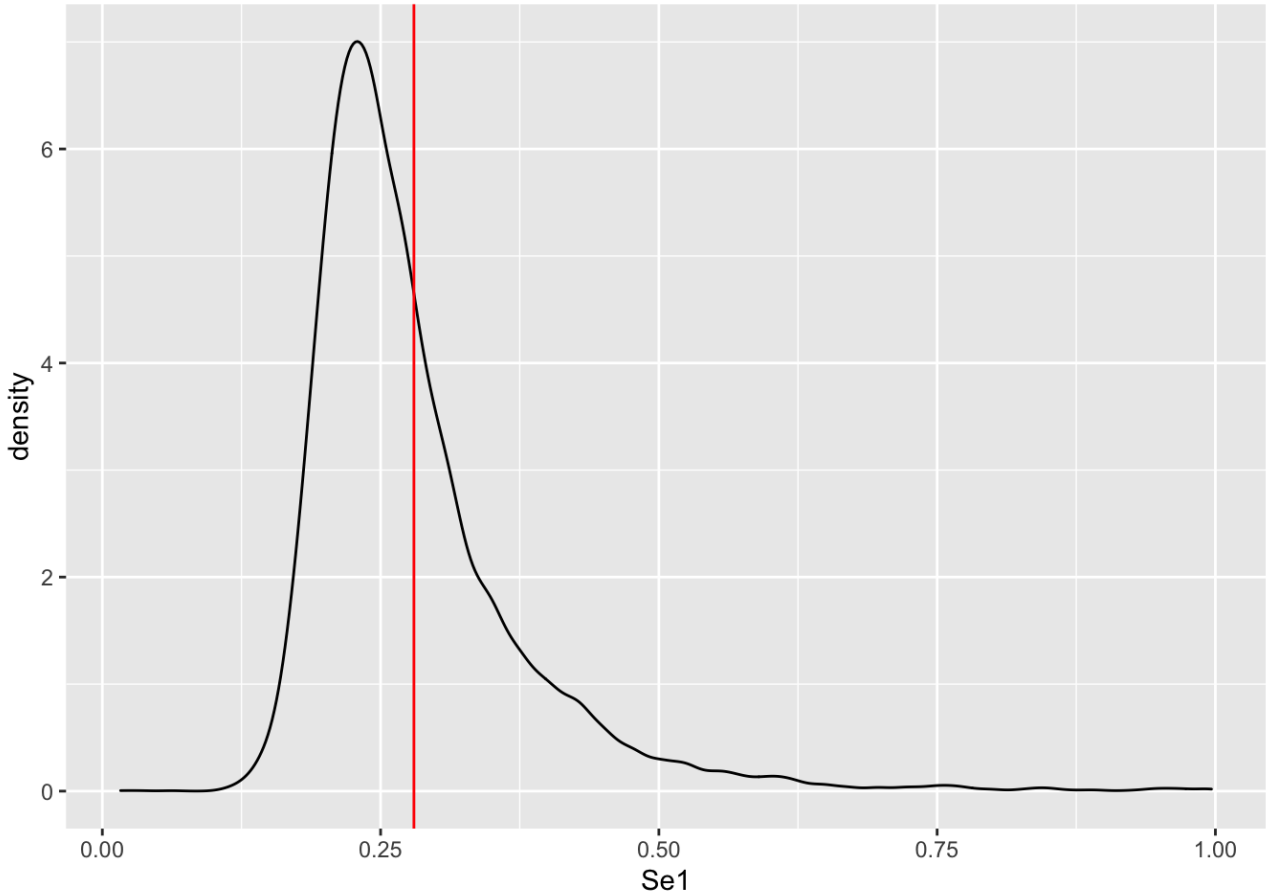
```
# Plot the posterior
plot(model1_sim)
```



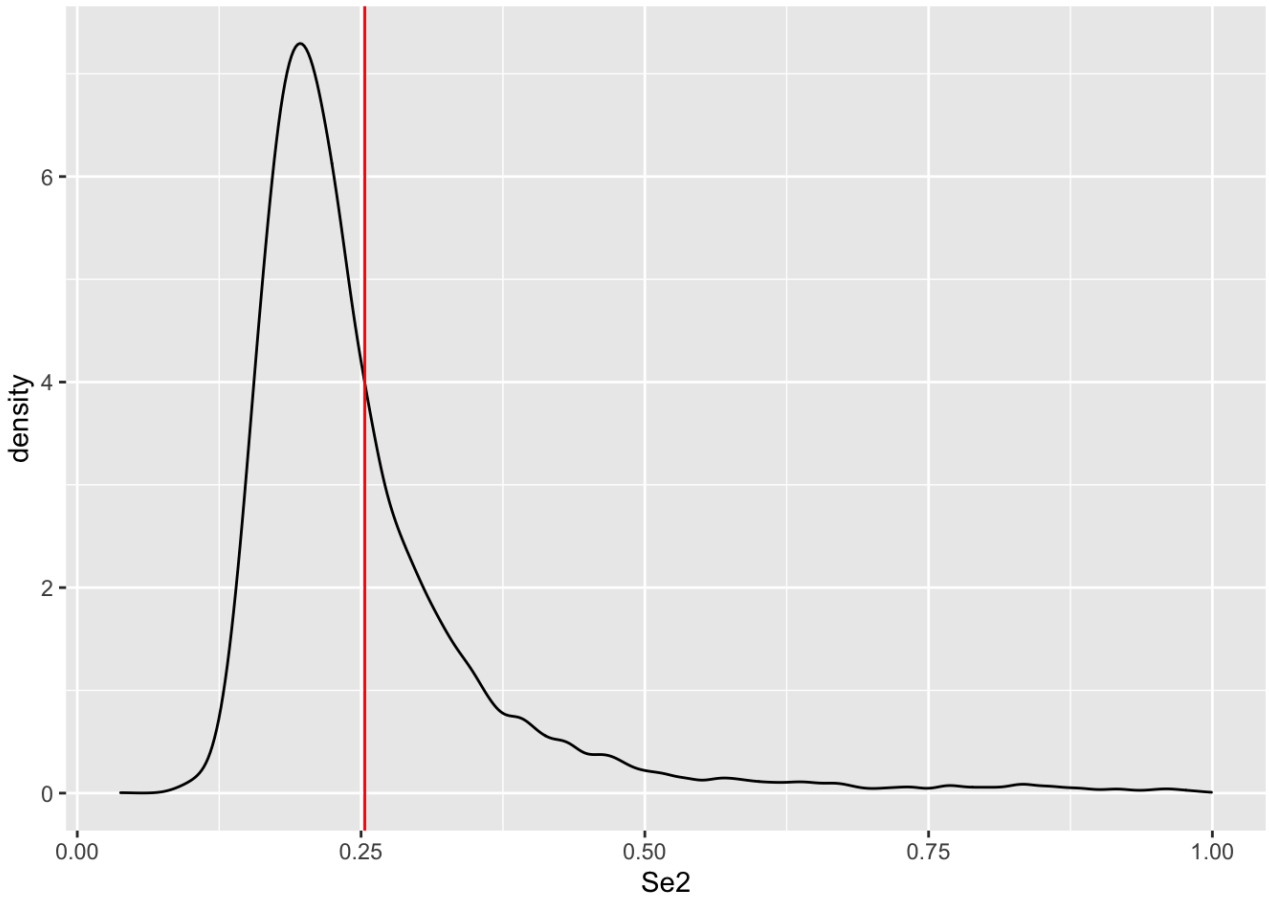

```
# Summarize the data
summary(model1_sim)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##        Mean      SD  Naive SE Time-series SE
## Se1  0.2799 0.09720 0.0009720       0.005127
## Se2  0.2533 0.11511 0.0011511       0.007198
## Sp1  0.9046 0.03479 0.0003479       0.001815
## Sp2  0.9232 0.02931 0.0002931       0.001383
## p[1] 0.2312 0.16505 0.0016505       0.010816
## p[2] 0.6242 0.23679 0.0023679       0.014821
##
## 2. Quantiles for each variable:
##
##         2.5%     25%    50%    75%  97.5%
## Se1  0.16973 0.21875 0.2553 0.3115 0.5335
## Se2  0.14102 0.18610 0.2210 0.2793 0.5940
## Sp1  0.85173 0.87849 0.8986 0.9259 0.9849
## Sp2  0.87649 0.90112 0.9191 0.9421 0.9883
## p[1] 0.01009 0.09268 0.2036 0.3461 0.5938
## p[2] 0.15831 0.43957 0.6523 0.8212 0.9797
```

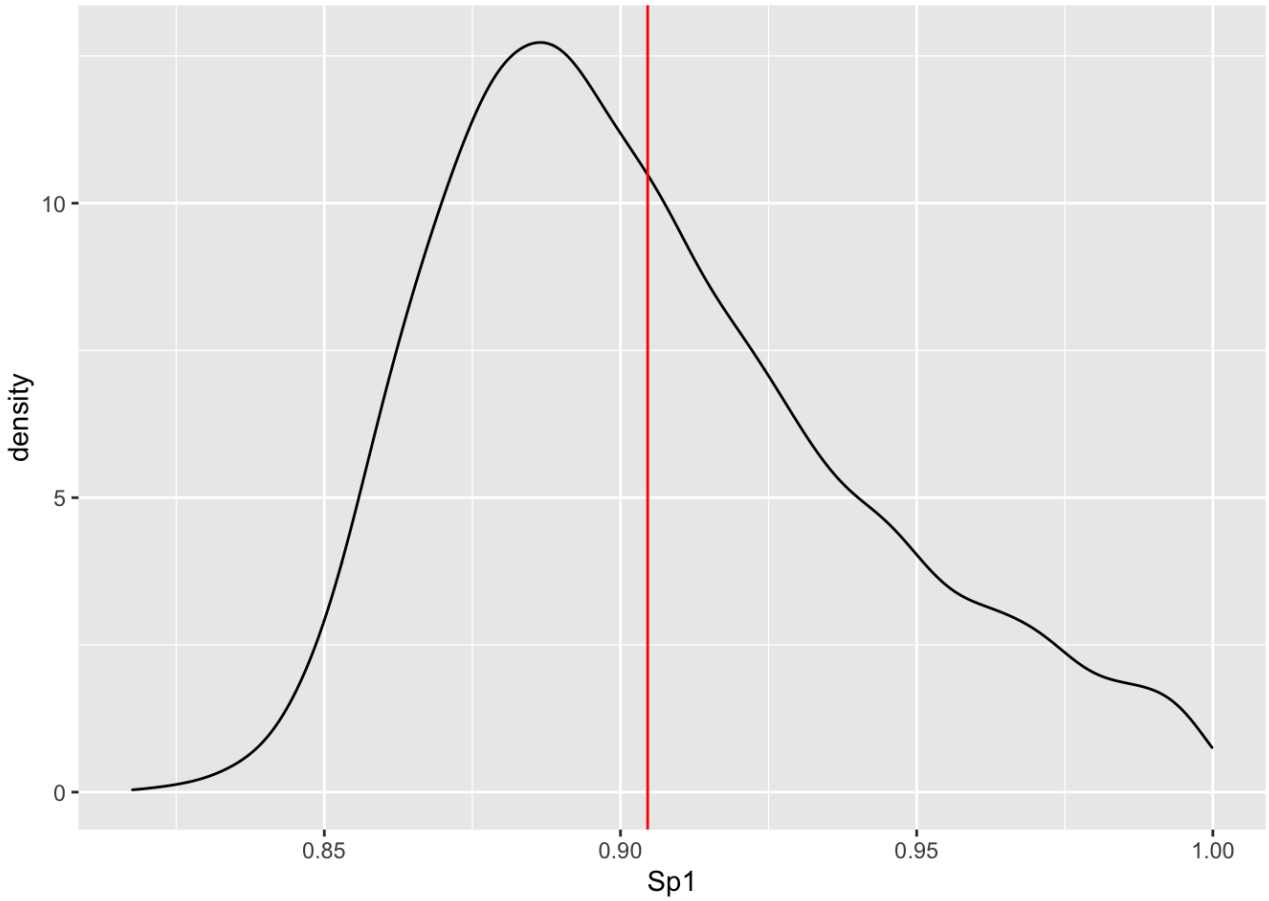
```
# Store the MC chain
model1_chains <- data.frame(model1_sim[[1]], iter = 1:10000)

# Make plots of the different parameters

# Se1 = Se of the TST
ggplot(data=model1_chains, aes(x=Se1)) +
  geom_density() + geom_vline(xintercept=mean(model1_chains$Se1), color="red")
```



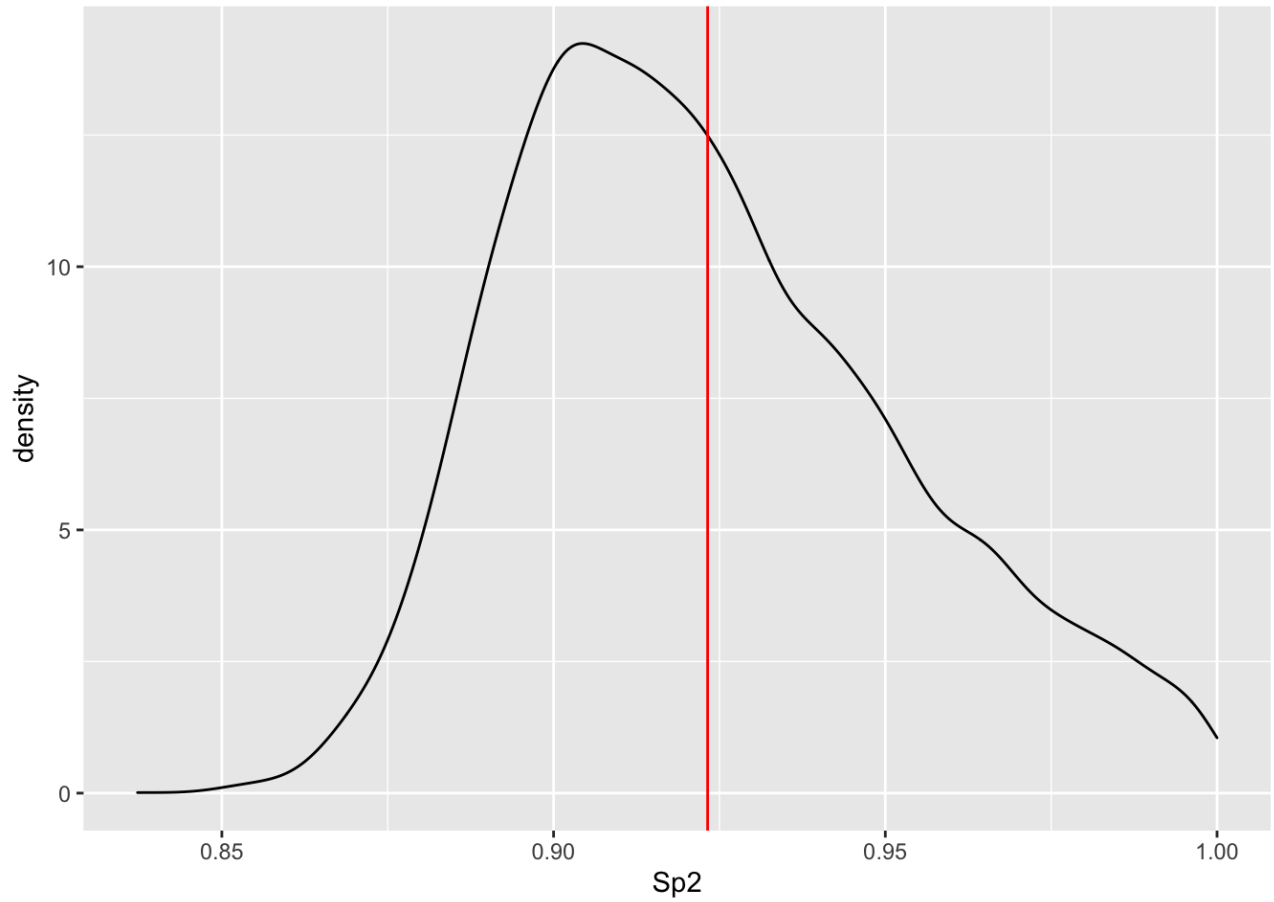

```
# Se2 = Se of the TB ELISA
ggplot(data=model1_chains, aes(x=Se2)) +
  geom_density() + geom_vline(xintercept=mean(model1_chains$Se2), color="red")
```



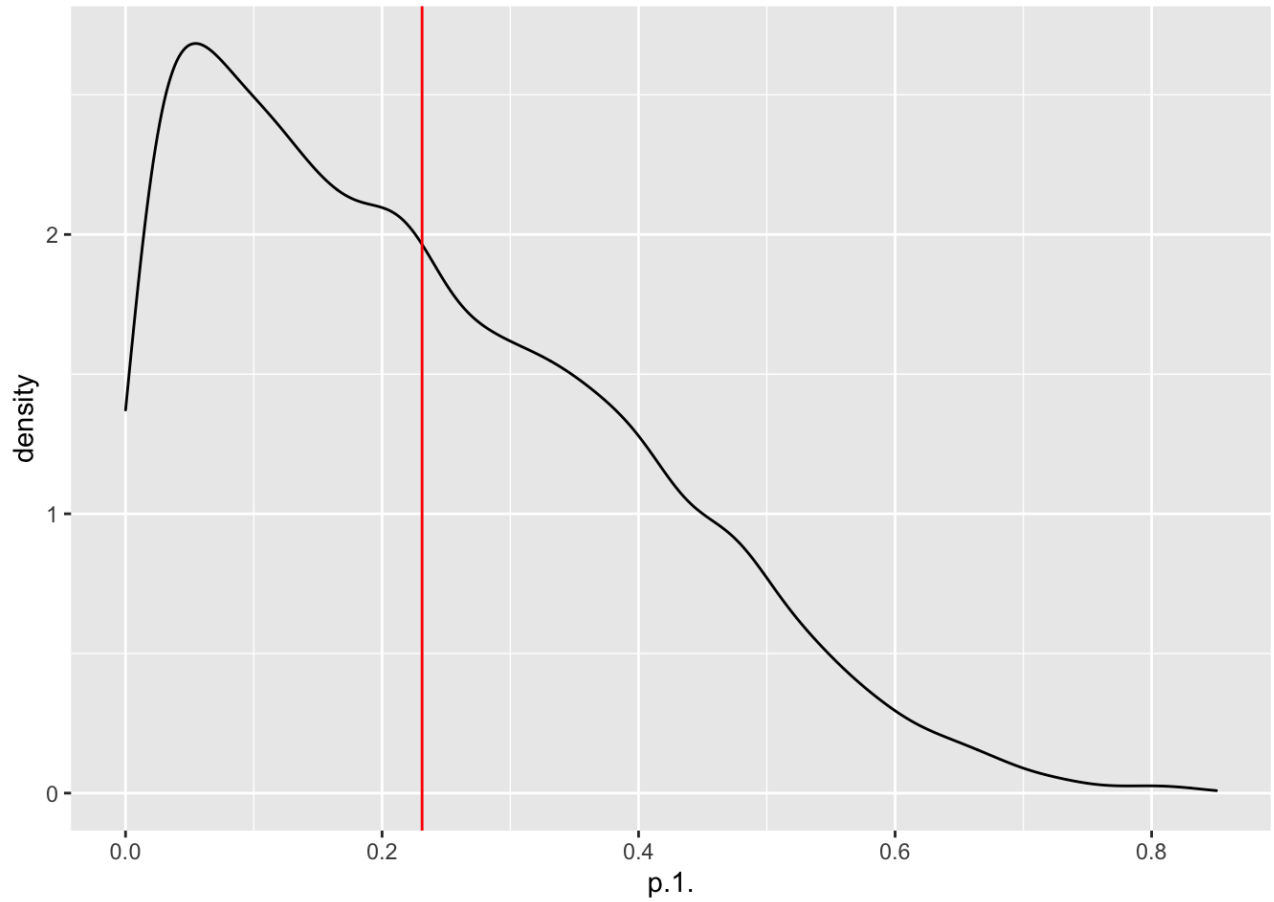
```
# Sp1 = Sp of the TST
ggplot(data=model1_chains, aes(x=Sp1)) +
  geom_density() + geom_vline(xintercept=mean(model1_chains$Sp1), color="red")
```



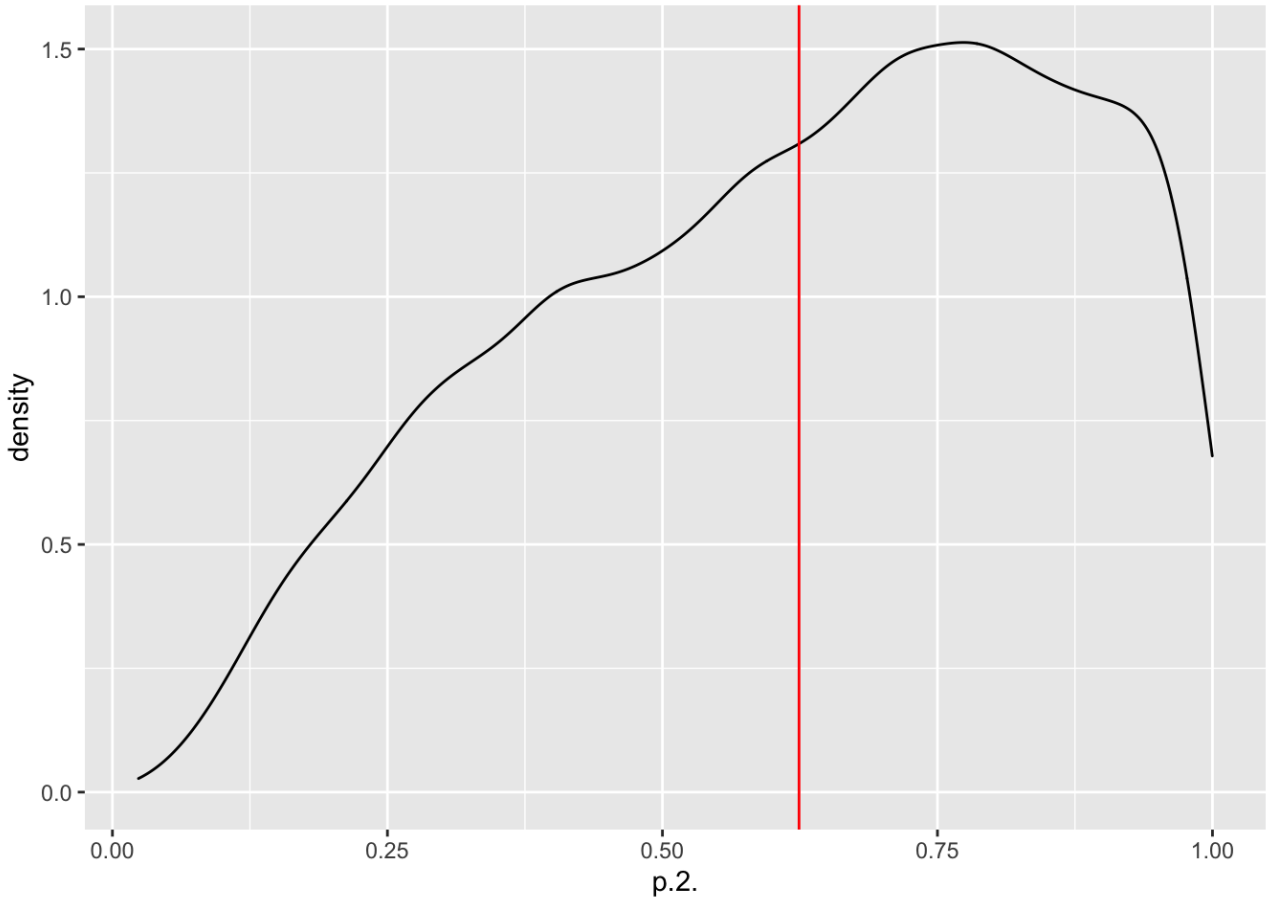

```
# Sp2 = SP of the TB ELISA
ggplot(data=model1_chains, aes(x=Sp2)) +
  geom_density() + geom_vline(xintercept=mean(model1_chains$Sp2), color="red")
```



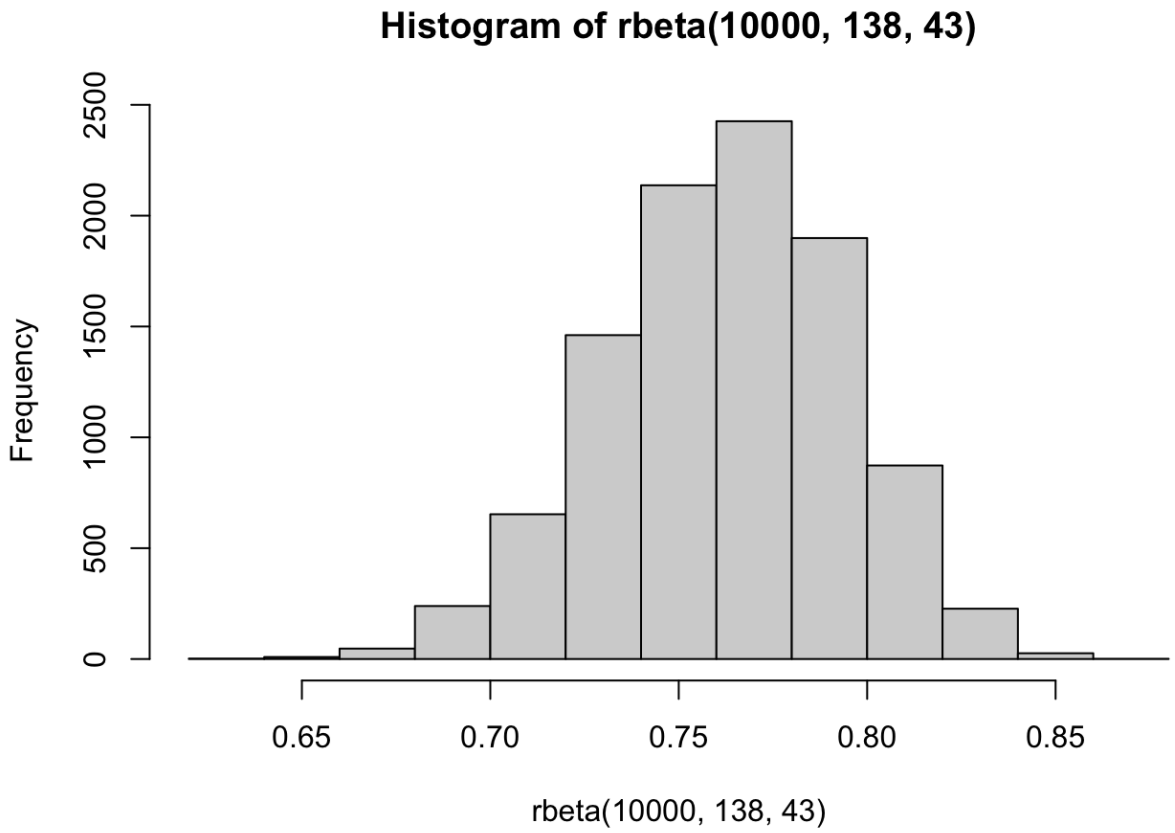

```
# p1 = prevalence in HiP
ggplot(data=model1_chains, aes(x=p.1.)) +
  geom_density() + geom_vline(xintercept=mean(model1_chains$p.1.), color="red")
```



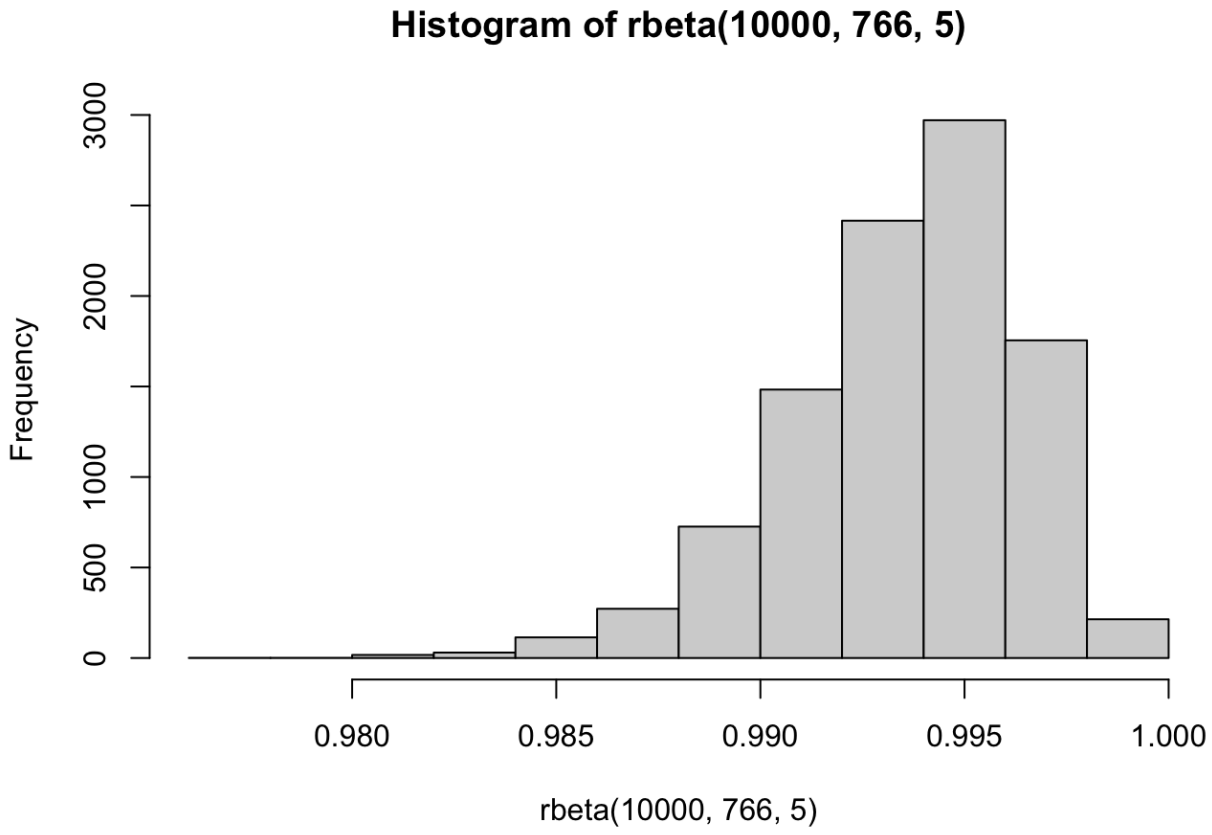
```
# p2 = prevalence in MGR
ggplot(data=model1_chains, aes(x=p.2.)) +
  geom_density() + geom_vline(xintercept=mean(model1_chains$p.2.), color="red")
```



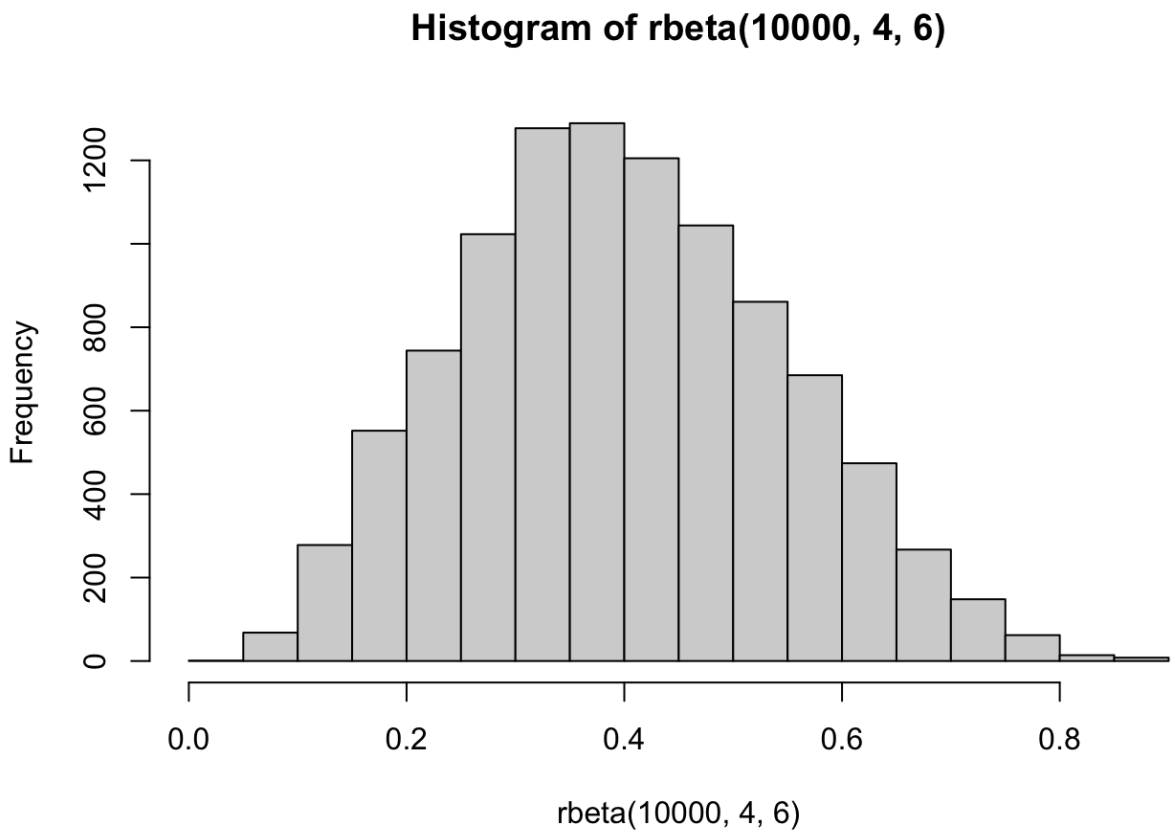

```
# Use informative priors according to known previous estimates for the Se and Sp of TST a
nd TB ELISA
# beta(1,1) * beta(s1,n1-s1) = beta(s1+1,n1-s1+1)
# in other words: from cross-tabs for Se -> beta(a+1,b+1) and for Sp <- beta(d+1,c+1)
# TST Se = 76.5% and Sp = 99.5% (Michel et al. 2011 and unpublished data)
Se1 <- hist(rbeta(10000, 138, 43))
```



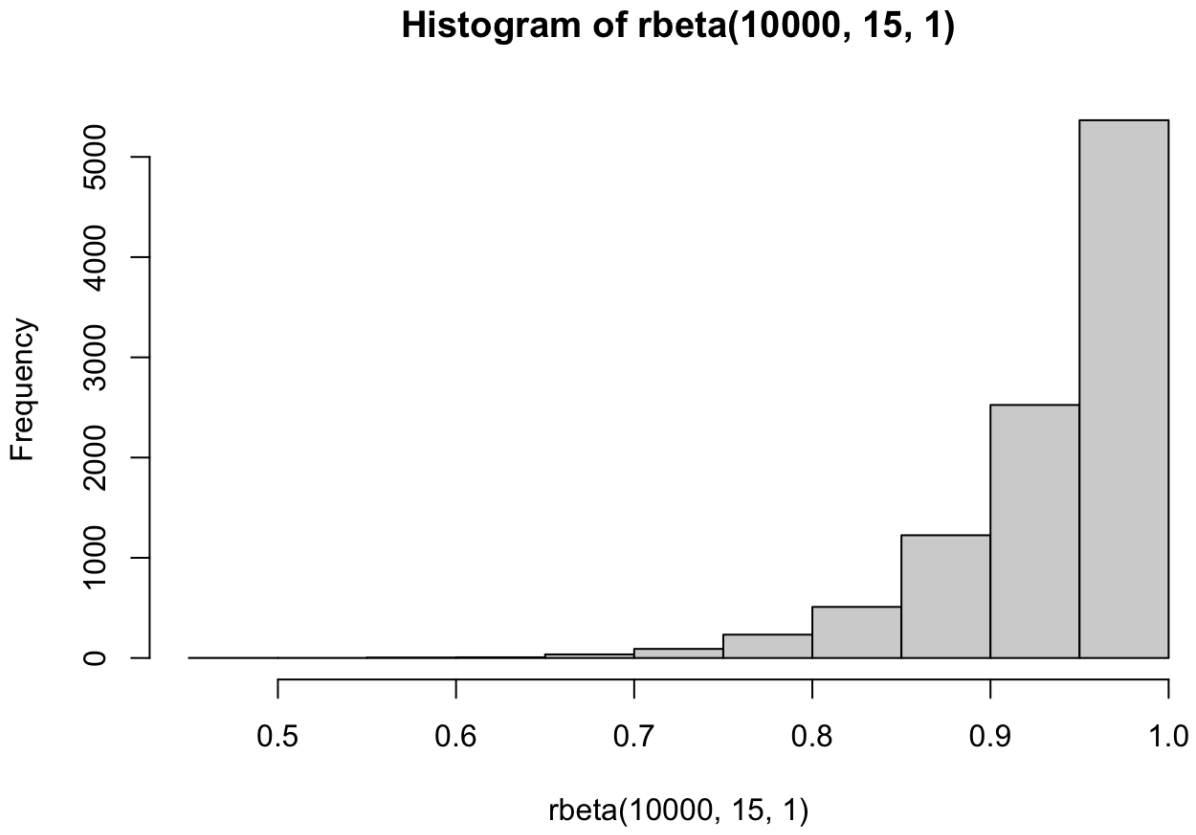

```
Sp1 <- hist(rbeta(10000, 766, 5))
```




```
# TB ELISA Se = 37.5% and Sp = 100% (van der Heijden et al. 2016 and unpublished data)
Se2 <- hist(rbeta(10000, 4, 6))
```




```
Sp2 <- hist(rbeta(10000, 15, 1))
```

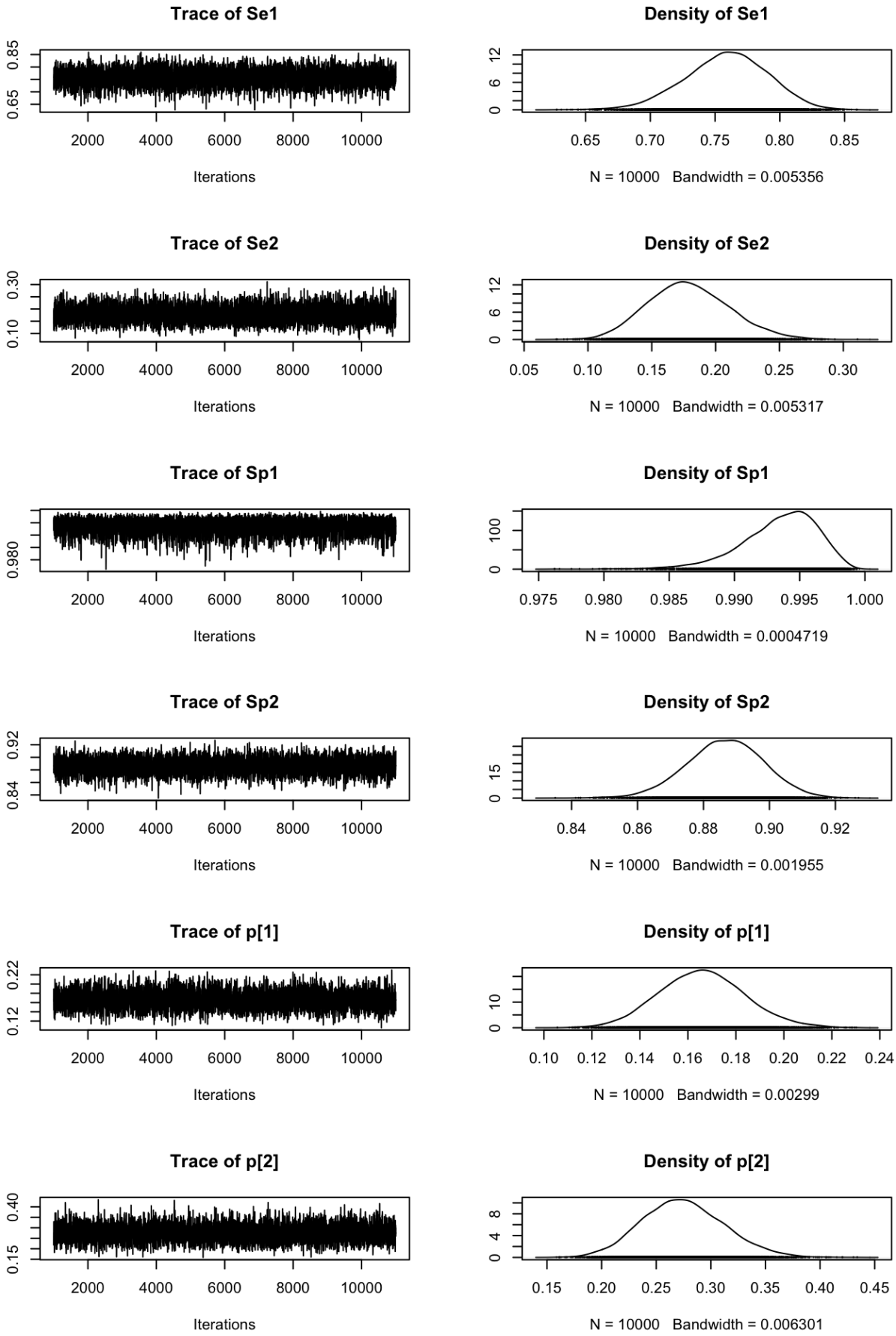


```
# Define the model
model2 <- "model{
  for (i in 1:2) {
p[i] ~ dbeta(1,1)
pop[i, 1:4] ~ dmulti(par[i,1:4], n[i])
par[i, 4] <- p[i]*Se1*Se2 + (1-p[i])*(1-Sp1)*(1-Sp2)
par[i, 2] <- p[i]*Se1*(1-Se2) + (1-p[i])*(1-Sp1)*Sp2
par[i, 3] <- p[i]*(1-Se1)*Se2 + (1-p[i])*Sp1*(1-Sp2)
par[i, 1] <- p[i]*(1-Se1)*(1-Se2) + (1-p[i])*Sp1*Sp2
}
## priors
Se1 ~ dbeta(138, 43)
Sp1 ~ dbeta(766, 5)
Se2 ~ dbeta(4, 6)
Sp2 ~ dbeta(15, 1)

}"

# Compile the model
model2_jags <- jags.model(textConnection(model2),inits=list(.RNG.name = "base::Wichmann-H
ill", .RNG.seed = 2018),
                          data=list(pop=pop, n=n))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2
##    Unobserved stochastic nodes: 6
##    Total graph size: 50
##
## Initializing model
```

```
# in Bronsvoort et al. own INITS defined (based on??) and also n.chains=3, n.adapt=50000)
# in datacamp course usually inits=list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 20
18) which is what was used here.
# should find out how to define our own inits - this seems key for the outcome!


# Simulate the model
model2_sim <- coda.samples(model=model2_jags, variable.names=c("p", "Se1", "Se2", "Sp1",
"Sp2"),
                           n.iter=10000)
# in Bronsvoort et al. n.iter=250000, n.thin=100
# the higher the number of iterarions, the more stable and reliable the outcome
# used 10000 here for now for speed of estimation, should probably be higher!
```

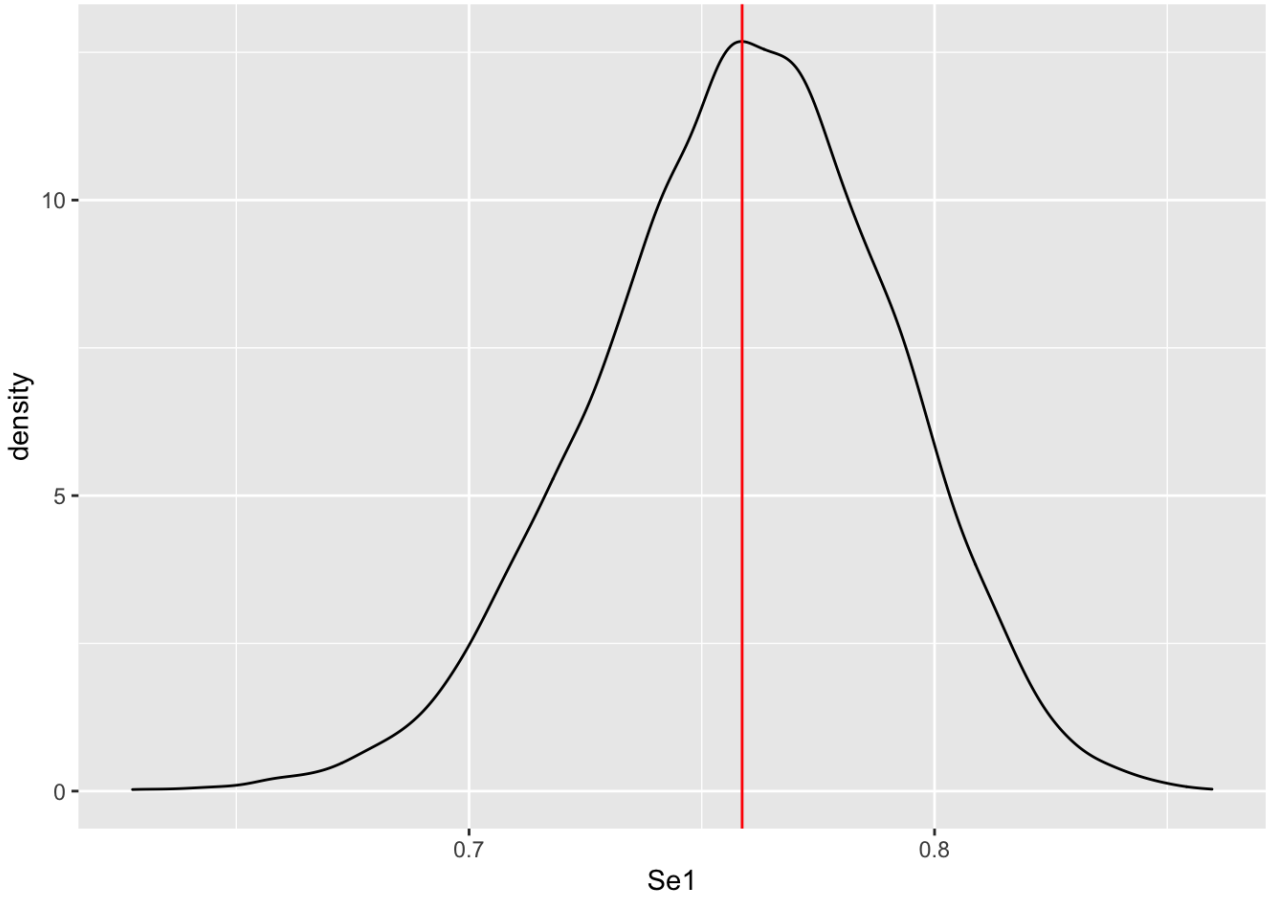
```
# Plot the posterior
plot(model2_sim)
```



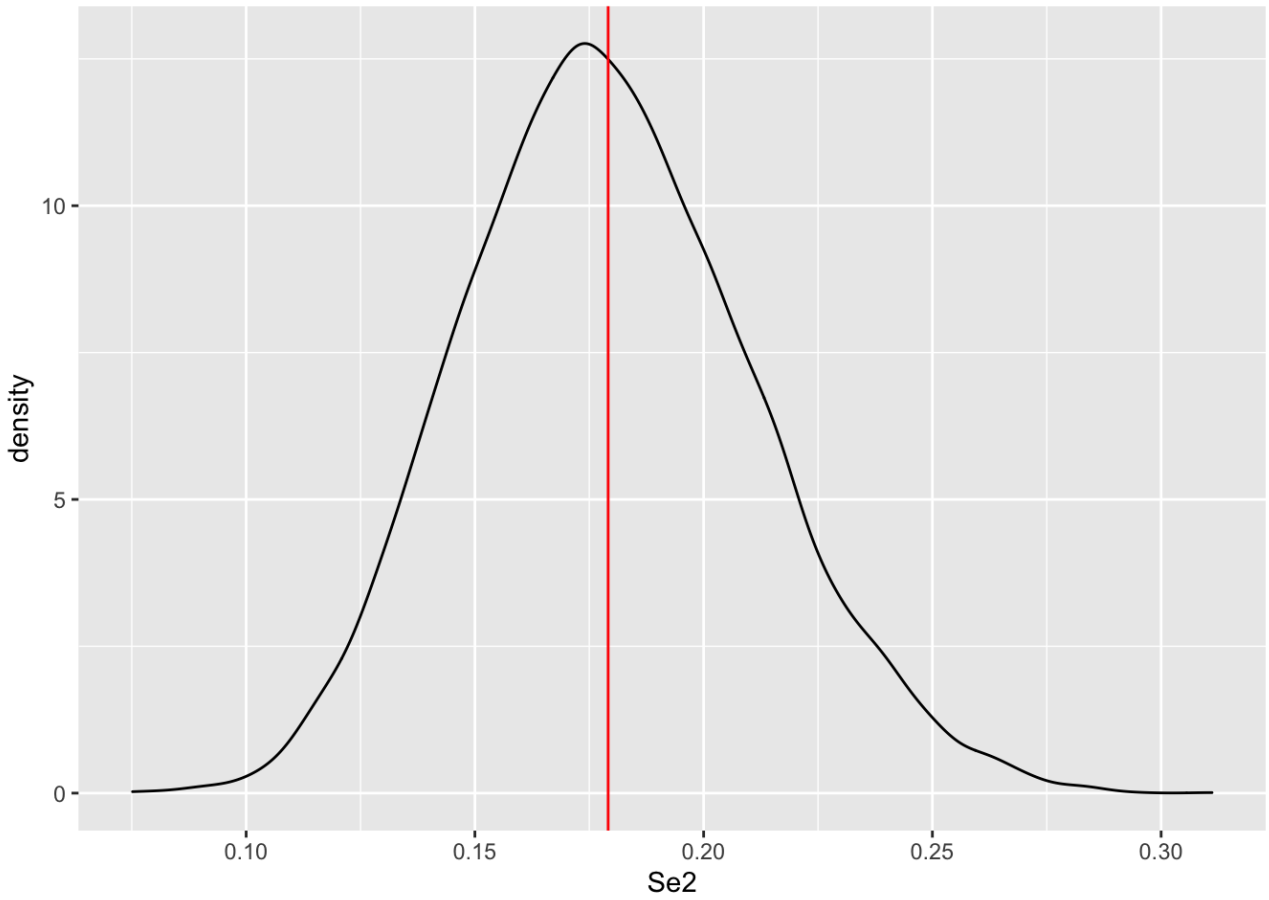

```
# Summarize the data
summary(model2_sim)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##        Mean      SD  Naive SE Time-series SE
## Se1  0.7587 0.03204 0.0003204      5.348e-04
## Se2  0.1791 0.03165 0.0003165      4.233e-04
## Sp1  0.9934 0.00292 0.0000292      4.241e-05
## Sp2  0.8866 0.01167 0.0001167      1.551e-04
## p[1] 0.1660 0.01783 0.0001783      2.799e-04
## p[2] 0.2749 0.03750 0.0003750      5.385e-04
##
## 2. Quantiles for each variable:
##
##        2.5%    25%    50%    75%  97.5%
## Se1  0.6930 0.7381 0.7598 0.7808 0.8176
## Se2  0.1212 0.1571 0.1776 0.1999 0.2449
## Sp1  0.9866 0.9918 0.9939 0.9955 0.9979
## Sp2  0.8630 0.8789 0.8868 0.8945 0.9088
## p[1] 0.1320 0.1538 0.1657 0.1777 0.2022
## p[2] 0.2052 0.2487 0.2736 0.2994 0.3519
```

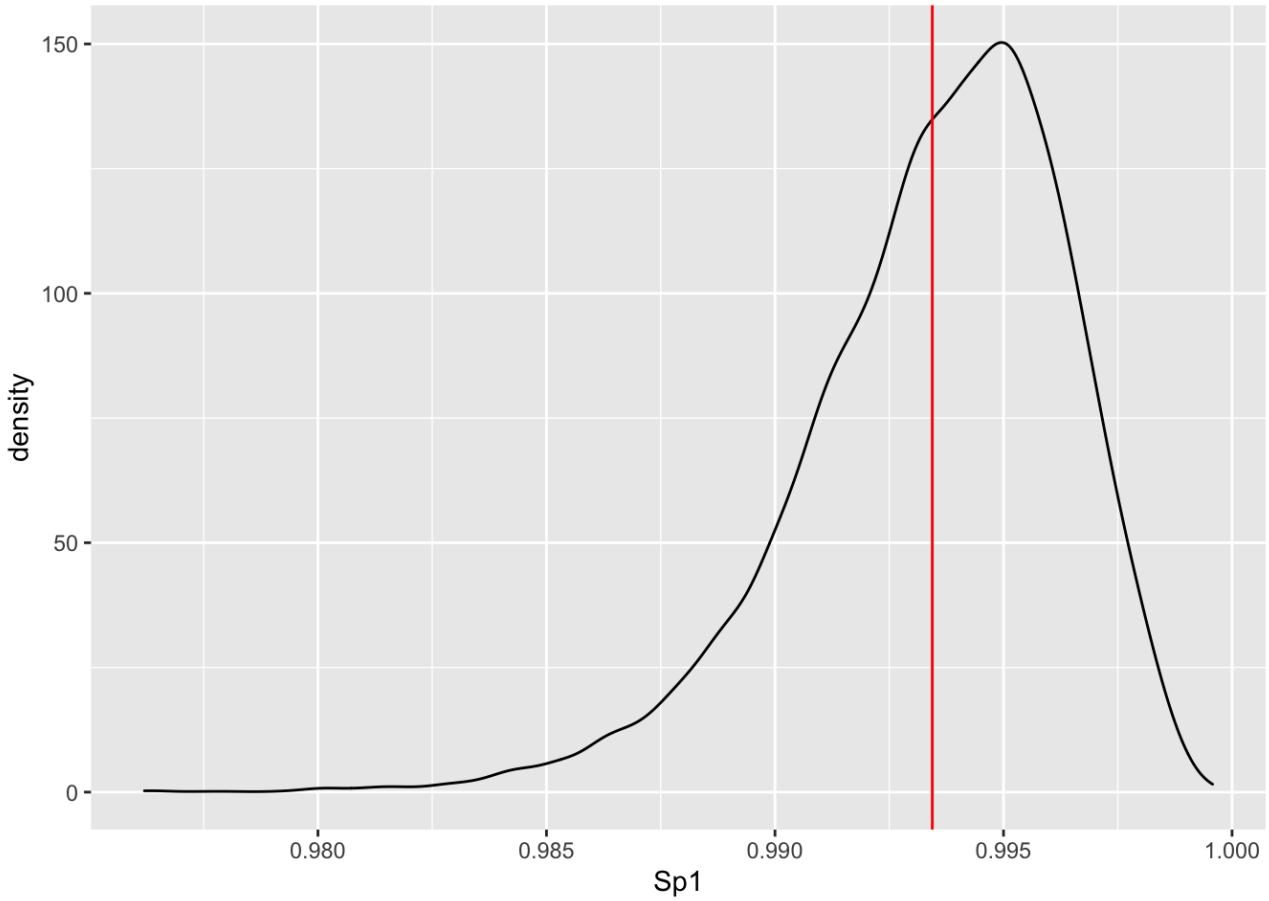
```
# Store the MC chain
model2_chains <- data.frame(model2_sim[[1]], iter = 1:10000)

# Make plots of the different parameters
# Se1 = Se of the TST
ggplot(data=model2_chains, aes(x=Se1)) +
  geom_density() + geom_vline(xintercept=mean(model2_chains$Se1), color="red")
```



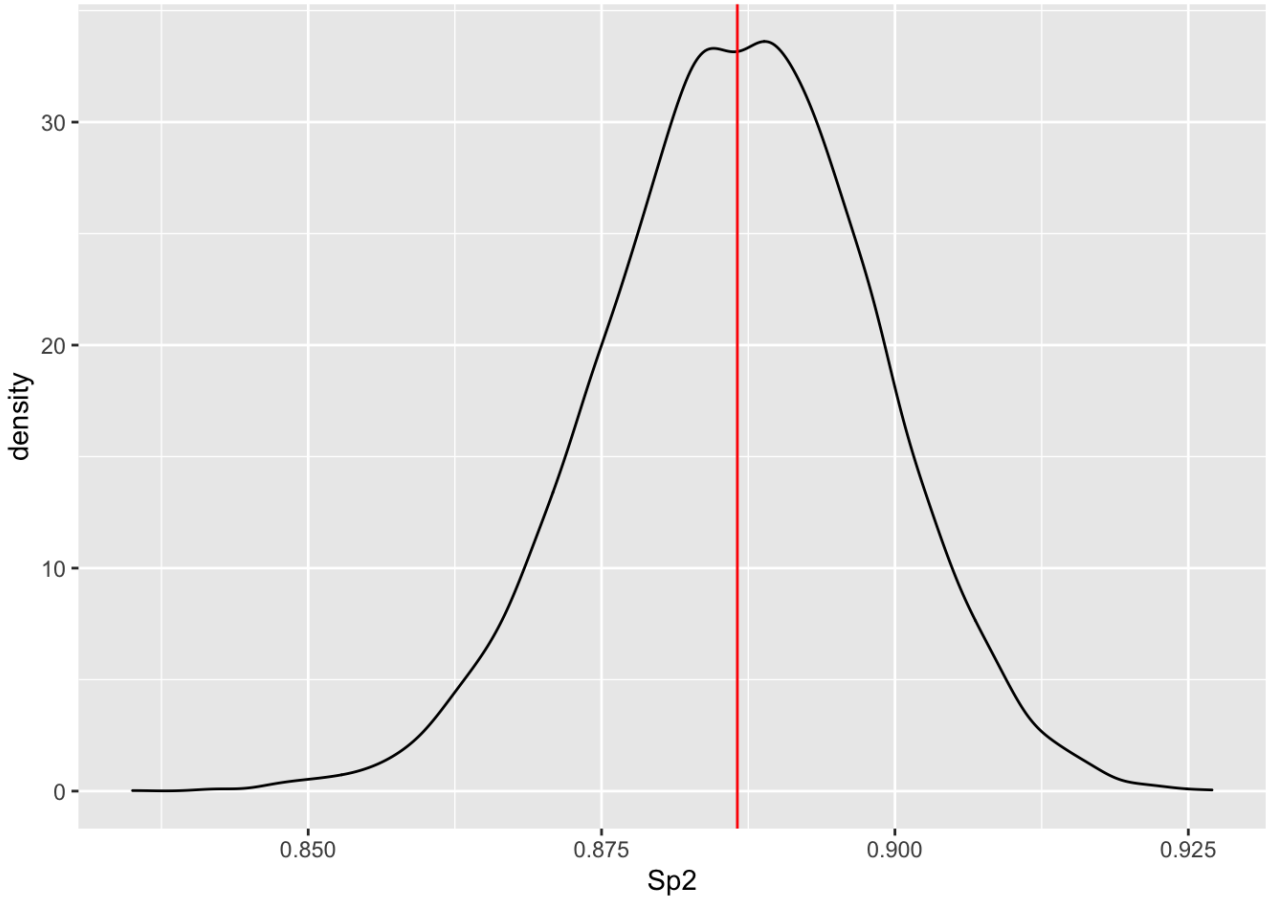

```
# Se2 = Se of the TB ELISA
ggplot(data=model2_chains, aes(x=Se2)) +
  geom_density() + geom_vline(xintercept=mean(model2_chains$Se2), color="red")
```



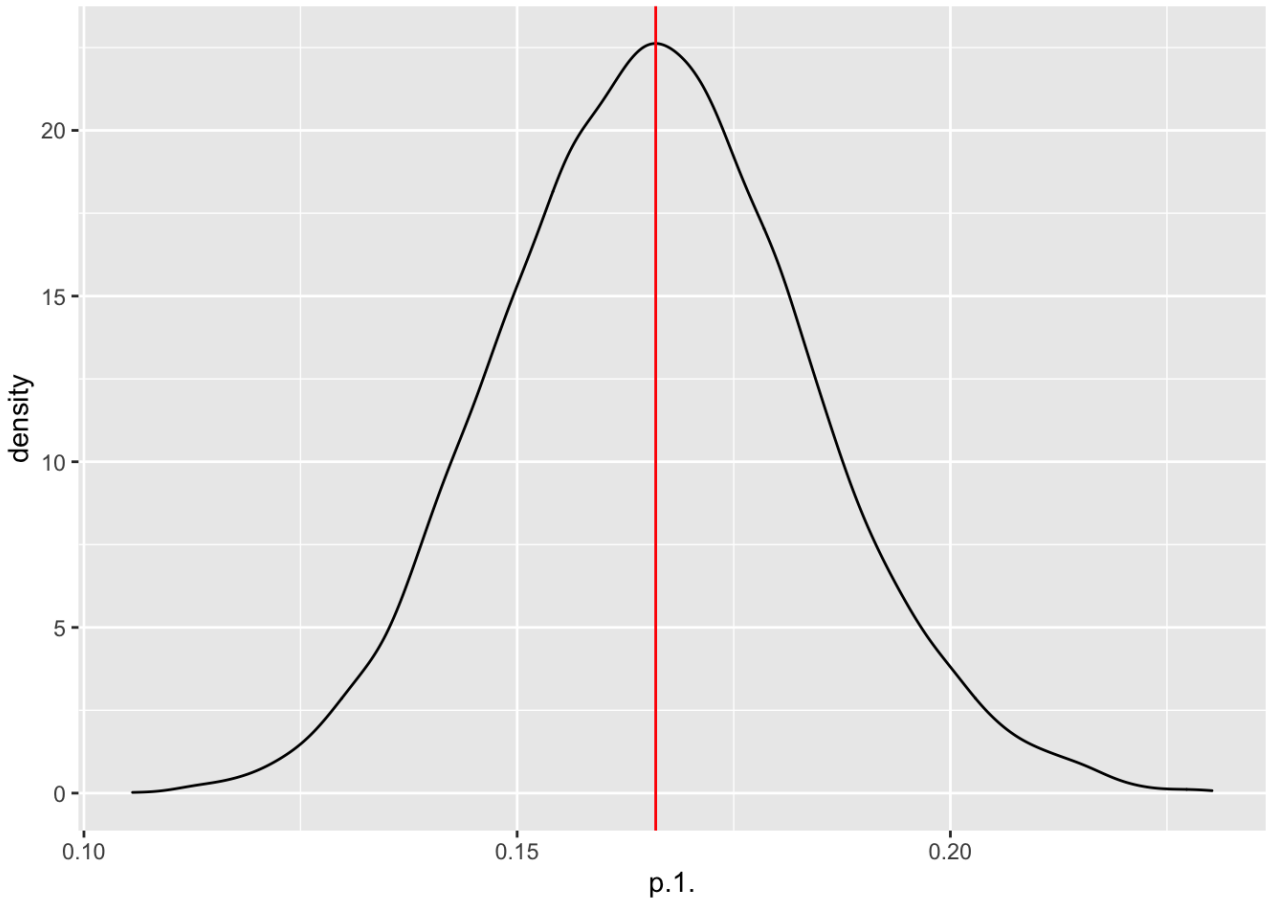
```
# Sp1 = Sp of the TST
ggplot(data=model2_chains, aes(x=Sp1)) +
  geom_density() + geom_vline(xintercept=mean(model2_chains$Sp1), color="red")
```



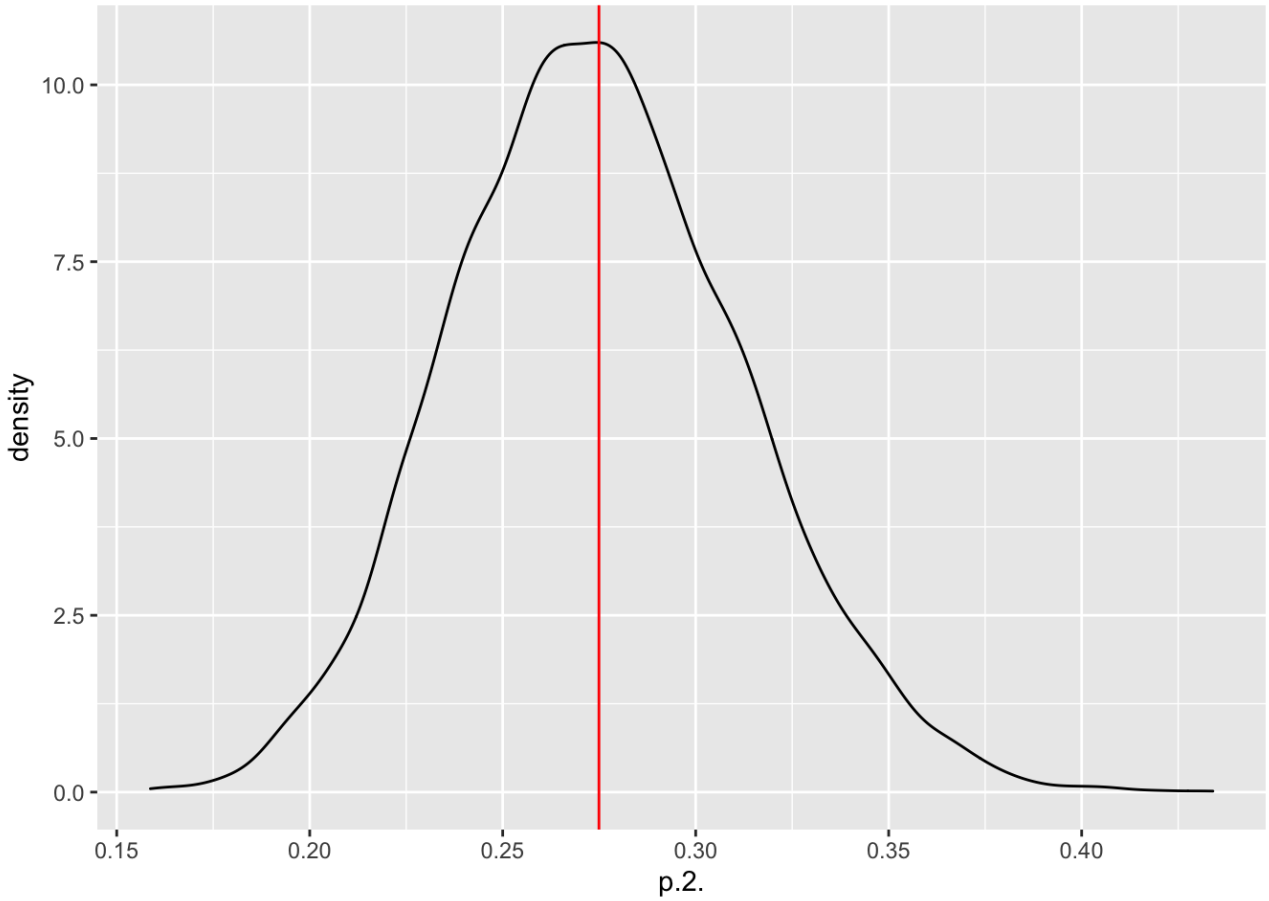

```
# Sp2 = SP of the TB ELISA
ggplot(data=model2_chains, aes(x=Sp2)) +
  geom_density() + geom_vline(xintercept=mean(model2_chains$Sp2), color="red")
```




```
# p1 = prevalence in HiP
ggplot(data=model2_chains, aes(x=p.1.)) +
  geom_density() + geom_vline(xintercept=mean(model2_chains$p.1.), color="red")
```



```
# p2 = prevalence in MGR
ggplot(data=model2_chains, aes(x=p.2.)) +
  geom_density() + geom_vline(xintercept=mean(model2_chains$p.2.), color="red")
```



```
# First change the Herd_ID columns, because currently for HiP there are herds from differ
ent years with the same Herd_ID
btb2 <- btb
btb2$Herd_ID2 <- NA

btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2015 & Herd_ID == "A", "HiP.A", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2015 & Herd_ID == "B", "HiP.B", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2015 & Herd_ID == "C", "HiP.C", H
erd_ID2))

btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "A", "HiP.D", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "B", "HiP.E", H
erd_ID2))

btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2017 & Herd_ID == "A", "HiP.F", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2017 & Herd_ID == "B", "HiP.G", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2017 & Herd_ID == "C", "HiP.H", H
erd_ID2))

btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "1", "MGR.1", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "2", "MGR.2", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "3", "MGR.3", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "4", "MGR.4", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "5", "MGR.5", H
erd_ID2))
btb2 <- transform(btb2, Herd_ID2 = ifelse(Study_Year == 2016 & Herd_ID == "6", "MGR.6", H
erd_ID2))

# Make a table of the various test result combinations stratified for the 14 herds
pop2 <- t(matrix(as.vector(table(btb2$TBELISA_Result, btb2$TST_Result, btb2$Herd_ID2)),4,
14))
View(pop2)

# Add colnames and rownames
# Give the table column- and rownames and have a look at the data
colnames(pop2) <- c("-/-", "-/+", "+/-", "+/+") # test 1 = TST; test 2 = TB ELISA
rownames(pop2) <- unique(btb2$Herd_ID2)
pop2
```

```
##       -/- -/+ +/- +/+
## HiP.A  62  26  15   3
## HiP.B  65   8  16   2
## HiP.C  65  13   8   0
## HiP.D 170   4   8   0
## HiP.E 137   4   9   0
## HiP.F  32   1   3   0
## HiP.G  34   2   4   0
## HiP.H  32  28   6   9
## MGR.1   7   9   4   2
## MGR.2  13  11   4   3
## MGR.3  27  14   0   2
## MGR.4  41   2   6   0
## MGR.5  36   2   3   0
## MGR.6  27   1  15   2
```

```
# Check whether the totals of test results still add up to the same as before
apply(pop2,2,sum) == apply(pop,2,sum) # all true so correct
```

```
##  -/-  -/+  +/-  +/+
## TRUE TRUE TRUE TRUE
```

```
apply(pop2,1,sum) # If you add up all the data per park HiP = 766 and MGR = 231 which is
 correct
```

```
## HiP.A HiP.B HiP.C HiP.D HiP.E HiP.F HiP.G HiP.H MGR.1 MGR.2 MGR.3 MGR.4 MGR.5
##   106    91    86   182   150    36    40    75    22    31    43    49    41
## MGR.6
##    45
```

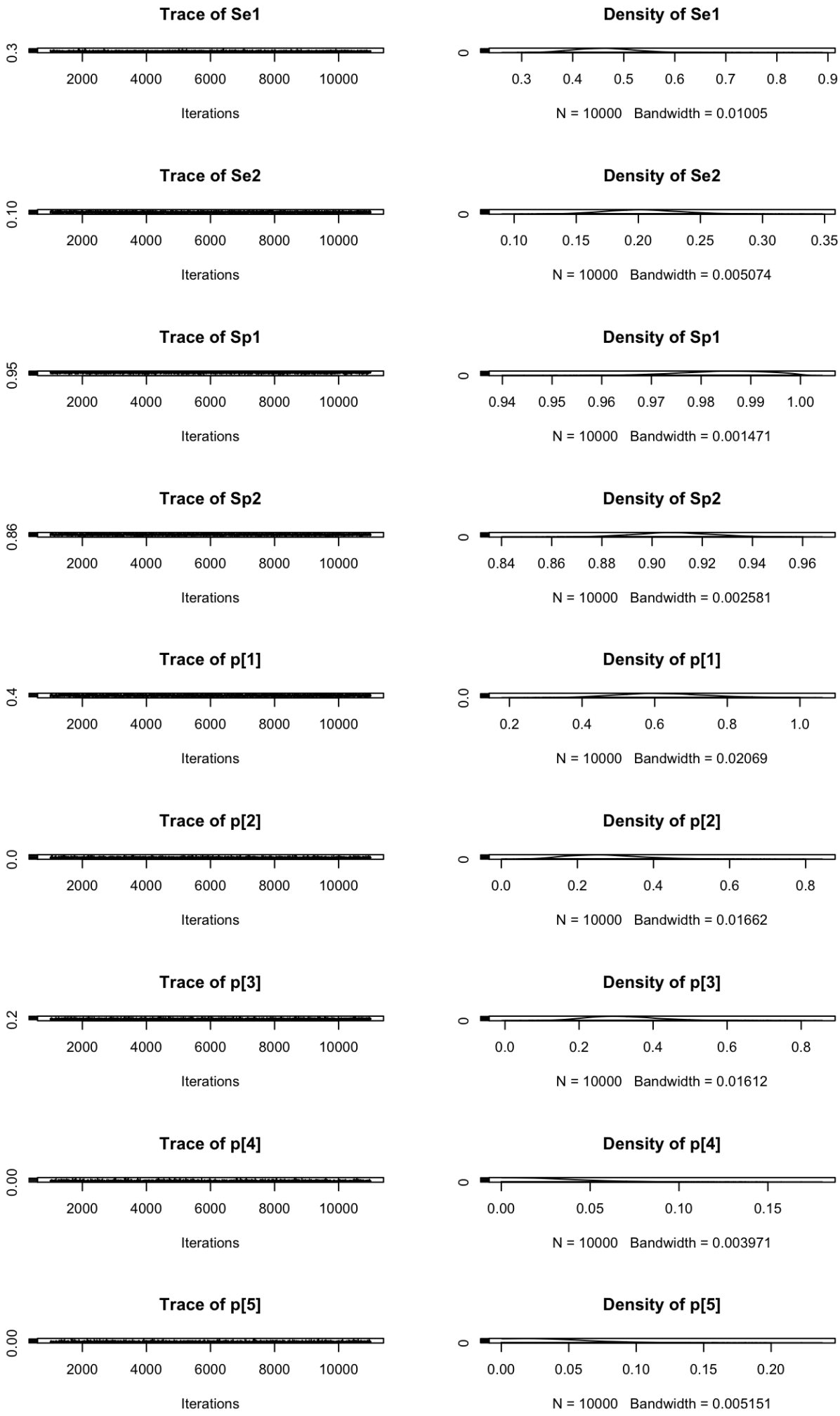
```
# Create vector of population counts
np2 <- nrow(pop2)
n2 <- apply(pop2,1,sum)
```

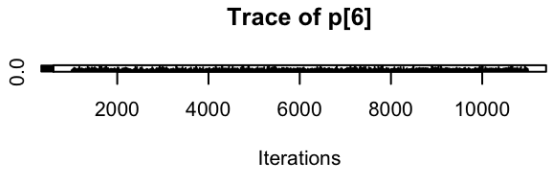
```
# Define the model
model3 <- "model{
  for (i in 1:14) {
    p[i] ~ dbeta(1,1)
    pop[i, 1:4] ~ dmulti(par[i,1:4], n[i])
    par[i, 4] <- p[i]*Se1*Se2 + (1-p[i])*(1-Sp1)*(1-Sp2)
    par[i, 2] <- p[i]*Se1*(1-Se2) + (1-p[i])*(1-Sp1)*Sp2
    par[i, 3] <- p[i]*(1-Se1)*Se2 + (1-p[i])*Sp1*(1-Sp2)
    par[i, 1] <- p[i]*(1-Se1)*(1-Se2) + (1-p[i])*Sp1*Sp2
    }
  ## priors
  Se1 ~ dbeta(1, 1)
  Sp1 ~ dbeta(1, 1)
  Se2 ~ dbeta(1, 1)
  Sp2 ~ dbeta(1, 1)

  # these priors are uninformative
  # try this model again with informative priors based on previous research of these test
s in buffaloes

}"

# Compile the model
model3_jags <- jags.model(textConnection(model3),inits=list(.RNG.name = "base::Wichmann-H
ill", .RNG.seed = 2018),
                          data=list(pop=pop2, n=n2))
```

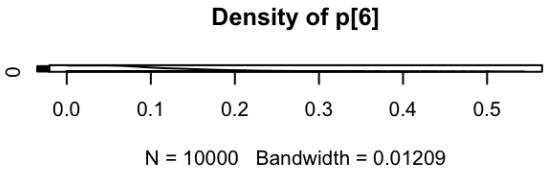
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 14
##    Unobserved stochastic nodes: 18
##    Total graph size: 247
##
## Initializing model
```

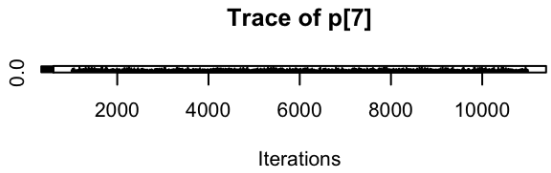
```
        # in Bronsvoort et al. own INITS defined (based on??) and also n.chains=3, n.adap
t=50000)
        # in datacamp course usually inits=list(.RNG.name = "base::Wichmann-Hill", .RNG.s
eed = 2018) which is what was used here.
        # should find out how to define our own inits - this seems key for the outcome!


# Simulate the model
model3_sim <- coda.samples(model=model3_jags, variable.names=c("p", "Se1", "Se2", "Sp1",
"Sp2"),
                           n.iter=10000)
        # in Bronsvoort et al. n.iter=250000, n.thin=100
        # the higher the number of iterarions, the more stable and reliable the outcome,
 but probably depends on volume of data too?
        # used 10000 here for now for speed of estimation, should probably be higher!
```

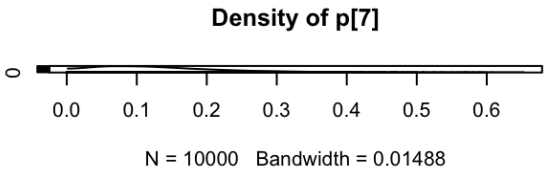
```
# Plot the posterior
plot(model3_sim)
```

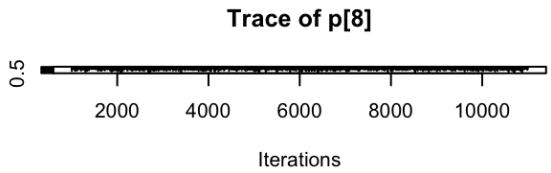

Trace of Se1
Iterations
Density of Se1
N = 10000 Bandwidth = 0.01005
Trace of Se2
Iterations
Density of Se2
N = 10000 Bandwidth = 0.005074
Trace of Sp1
Iterations
Density of Sp1
N = 10000 Bandwidth = 0.001471
Trace of Sp2
Iterations
Density of Sp2
N = 10000 Bandwidth = 0.002581
Trace of p[1]
Iterations
Density of p[1]
N = 10000 Bandwidth = 0.02069
Trace of p[2]
Iterations
Density of p[2]
N = 10000 Bandwidth = 0.01662
Trace of p[3]
Iterations
Density of p[3]
N = 10000 Bandwidth = 0.01612
Trace of p[4]
Iterations
Density of p[4]
N = 10000 Bandwidth = 0.003971
Trace of p[5]
Iterations
Density of p[5]
N = 10000 Bandwidth = 0.005151

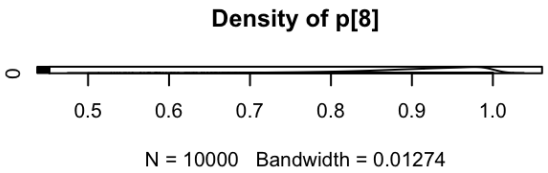

Trace of p[6]
0.0
2000
4000
6000
8000
10000
Iterations

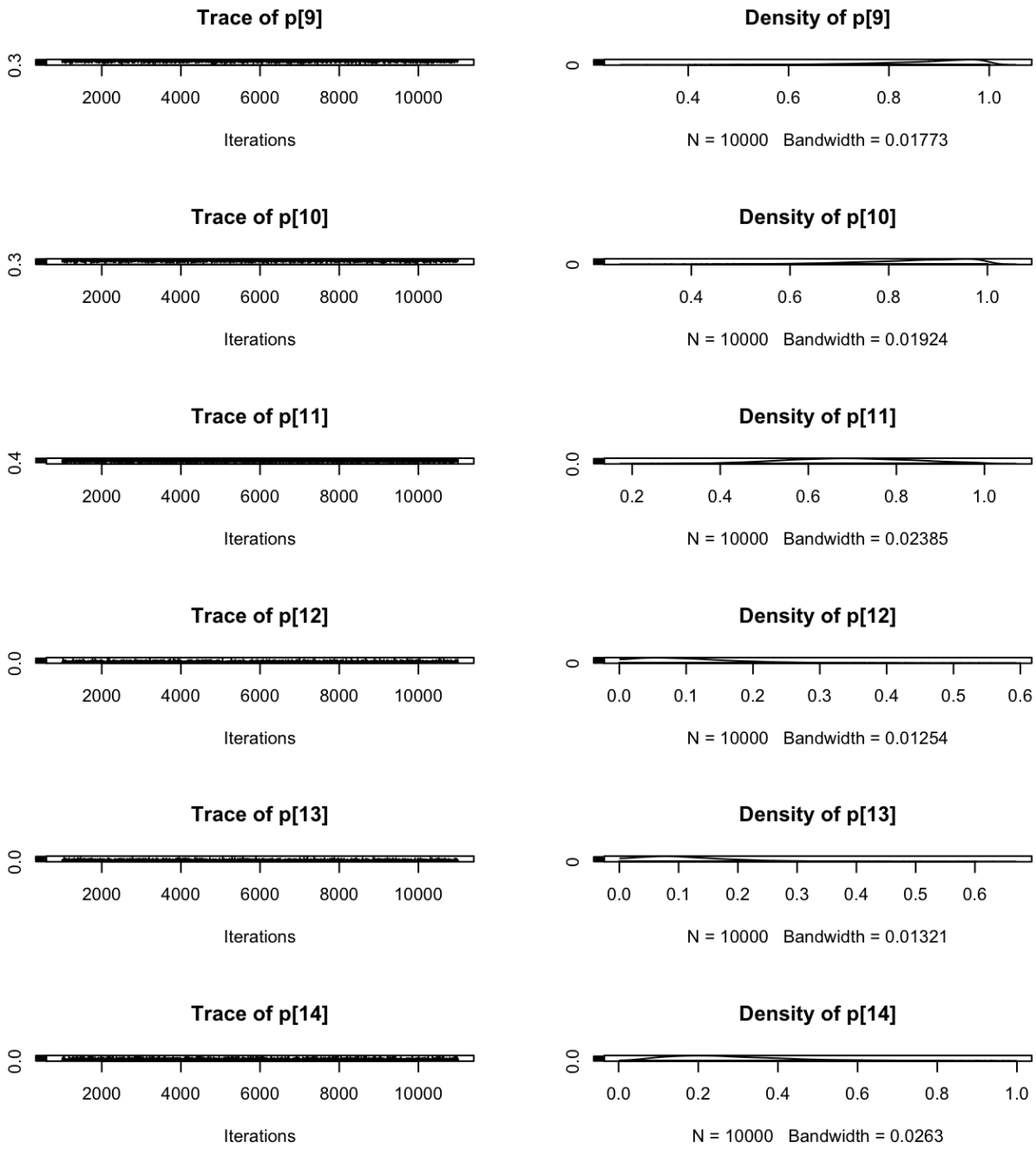

Density of p[6]
0
0.0
0.1
0.2
0.3
0.4
0.5
N = 10000 Bandwidth = 0.01209


Trace of p[7]
0.0
2000
4000
6000
8000
10000
Iterations


Density of p[7]
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
N = 10000 Bandwidth = 0.01488


Trace of p[8]
0.5
2000
4000
6000
8000
10000
Iterations

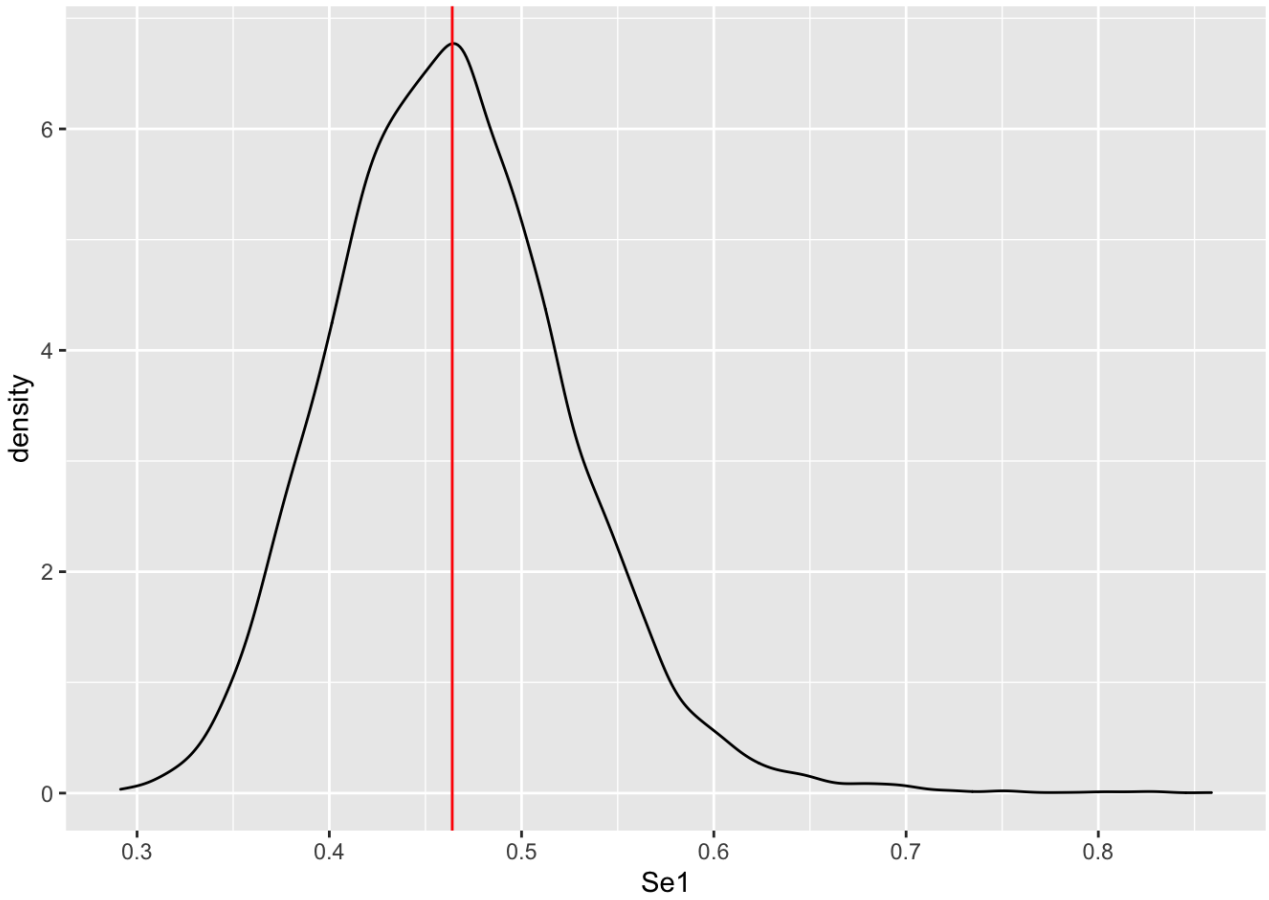

Density of p[8]
0
0.5
0.6
0.7
0.8
0.9
1.0
N = 10000 Bandwidth = 0.01274



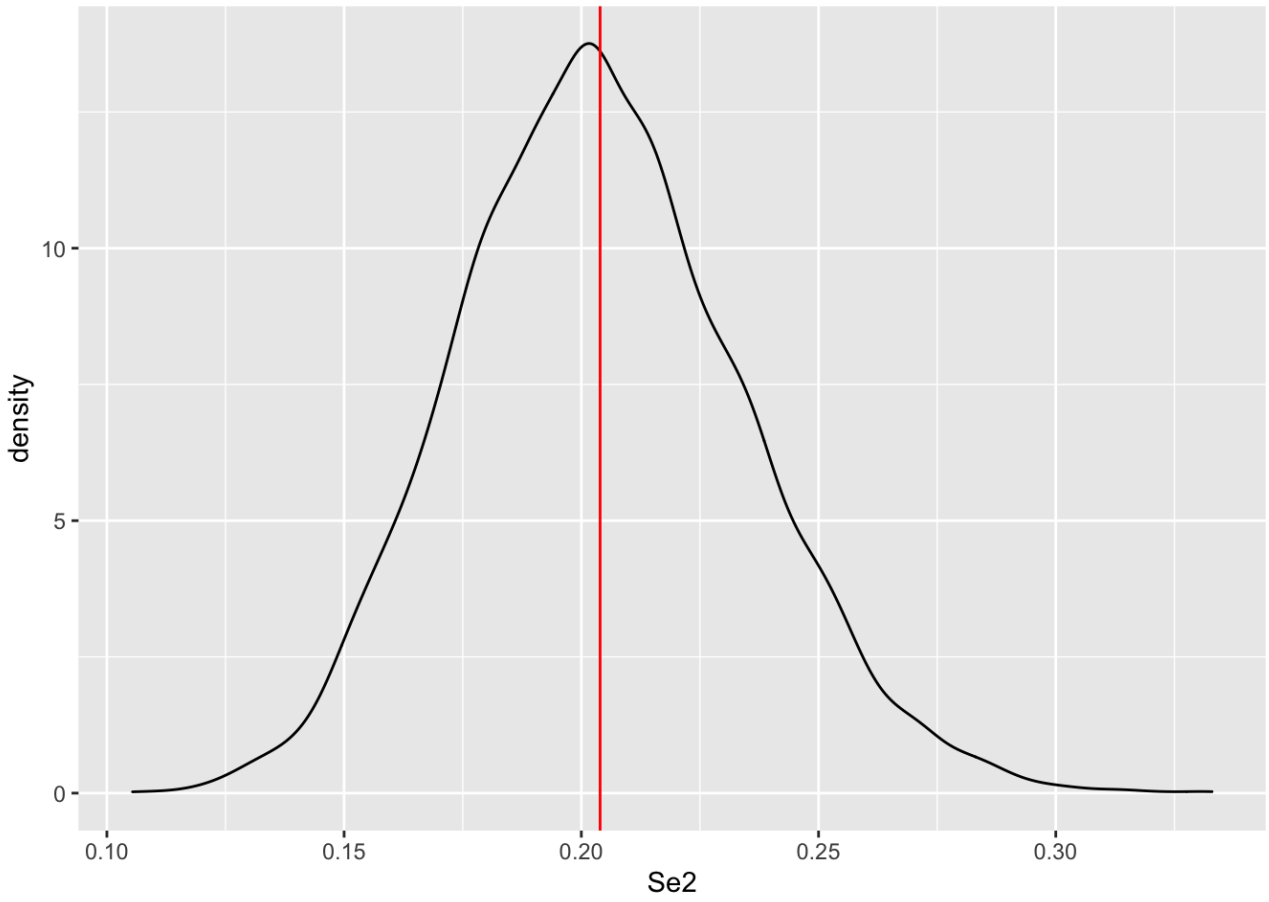

```
# Summarize the data
summary(model3_sim)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD  Naive SE Time-series SE
## Se1   0.46395 0.061684 6.168e-04      0.0020240
## Se2   0.20396 0.030268 3.027e-04      0.0005294
## Sp1   0.98456 0.008758 8.758e-05      0.0001771
## Sp2   0.90871 0.015527 1.553e-04      0.0003547
## p[1]  0.62254 0.124061 1.241e-03      0.0033111
## p[2]  0.27510 0.099346 9.935e-04      0.0022751
## p[3]  0.32065 0.096649 9.665e-04      0.0018243
## p[4]  0.03102 0.024281 2.428e-04      0.0004869
## p[5]  0.04197 0.030933 3.093e-04      0.0006056
## p[6]  0.09519 0.077094 7.709e-04      0.0014185
## p[7]  0.13278 0.091716 9.172e-04      0.0016407
## p[8]  0.90608 0.076633 7.663e-04      0.0019398
## p[9]  0.87153 0.105524 1.055e-03      0.0018838
## p[10] 0.84839 0.114543 1.145e-03      0.0022428
## p[11] 0.69279 0.141936 1.419e-03      0.0024318
## p[12] 0.10744 0.076634 7.663e-04      0.0013917
## p[13] 0.11974 0.083101 8.310e-04      0.0015253
## p[14] 0.28598 0.162851 1.629e-03      0.0039981
##
## 2. Quantiles for each variable:
##
##           2.5%     25%     50%     75%  97.5%
## Se1   0.355505 0.42115 0.46098 0.50133 0.5939
## Se2   0.148578 0.18297 0.20274 0.22344 0.2668
## Sp1   0.965953 0.97881 0.98523 0.99131 0.9987
## Sp2   0.878929 0.89846 0.90843 0.91905 0.9398
## p[1]  0.398012 0.53627 0.61541 0.70128 0.8877
## p[2]  0.111871 0.20307 0.26546 0.33566 0.4950
## p[3]  0.155839 0.25165 0.31200 0.38024 0.5345
## p[4]  0.001250 0.01240 0.02559 0.04407 0.0911
## p[5]  0.001976 0.01805 0.03594 0.05914 0.1161
## p[6]  0.003797 0.03771 0.07538 0.13414 0.2916
## p[7]  0.009461 0.06380 0.11520 0.18245 0.3601
## p[8]  0.717791 0.86500 0.92386 0.96666 0.9971
## p[9]  0.612606 0.81229 0.89980 0.95454 0.9962
## p[10] 0.575921 0.77863 0.87190 0.94086 0.9939
## p[11] 0.418621 0.59135 0.69211 0.79649 0.9635
## p[12] 0.006508 0.04905 0.09253 0.14907 0.2984
## p[13] 0.007940 0.05853 0.10274 0.16389 0.3228
## p[14] 0.057172 0.16557 0.25360 0.37531 0.6836
```

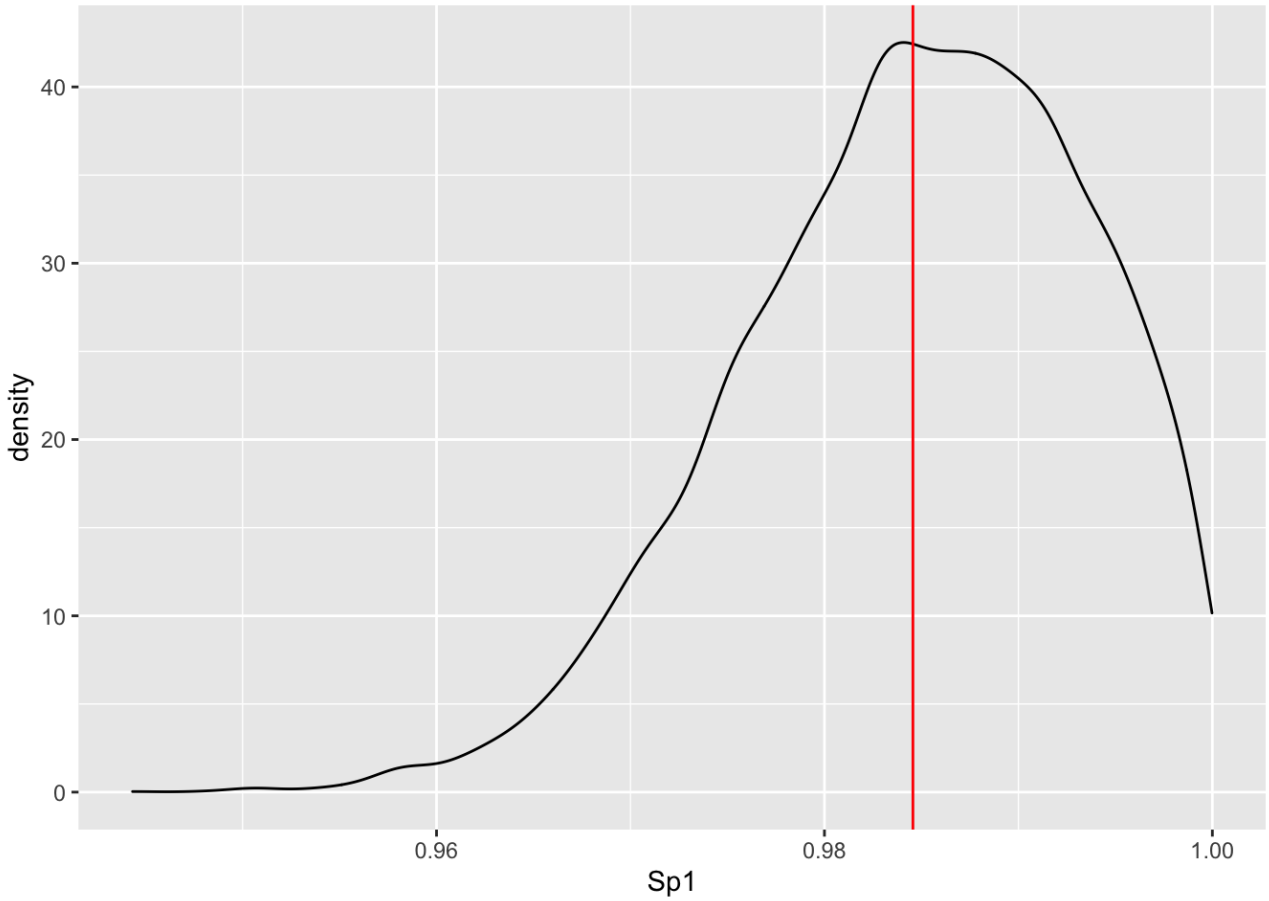
```
# Store the MC chain
model3_chains <- data.frame(model3_sim[[1]], iter = 1:10000)

# Make plots of the different parameters

# Se1 = Se of the TST
ggplot(data=model3_chains, aes(x=Se1)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$Se1), color="red")
```



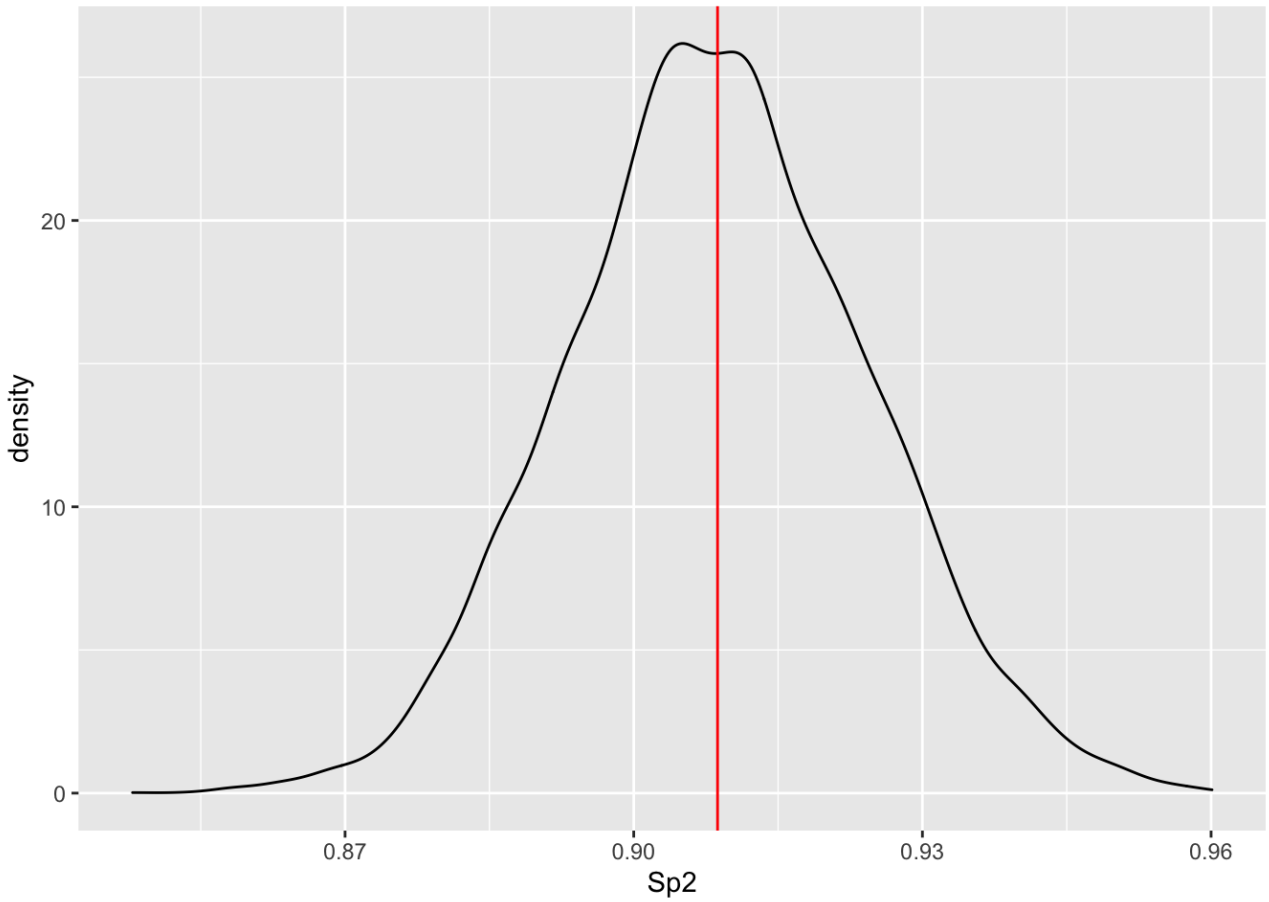

```
# Se2 = Se of the TB ELISA
ggplot(data=model3_chains, aes(x=Se2)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$Se2), color="red")
```



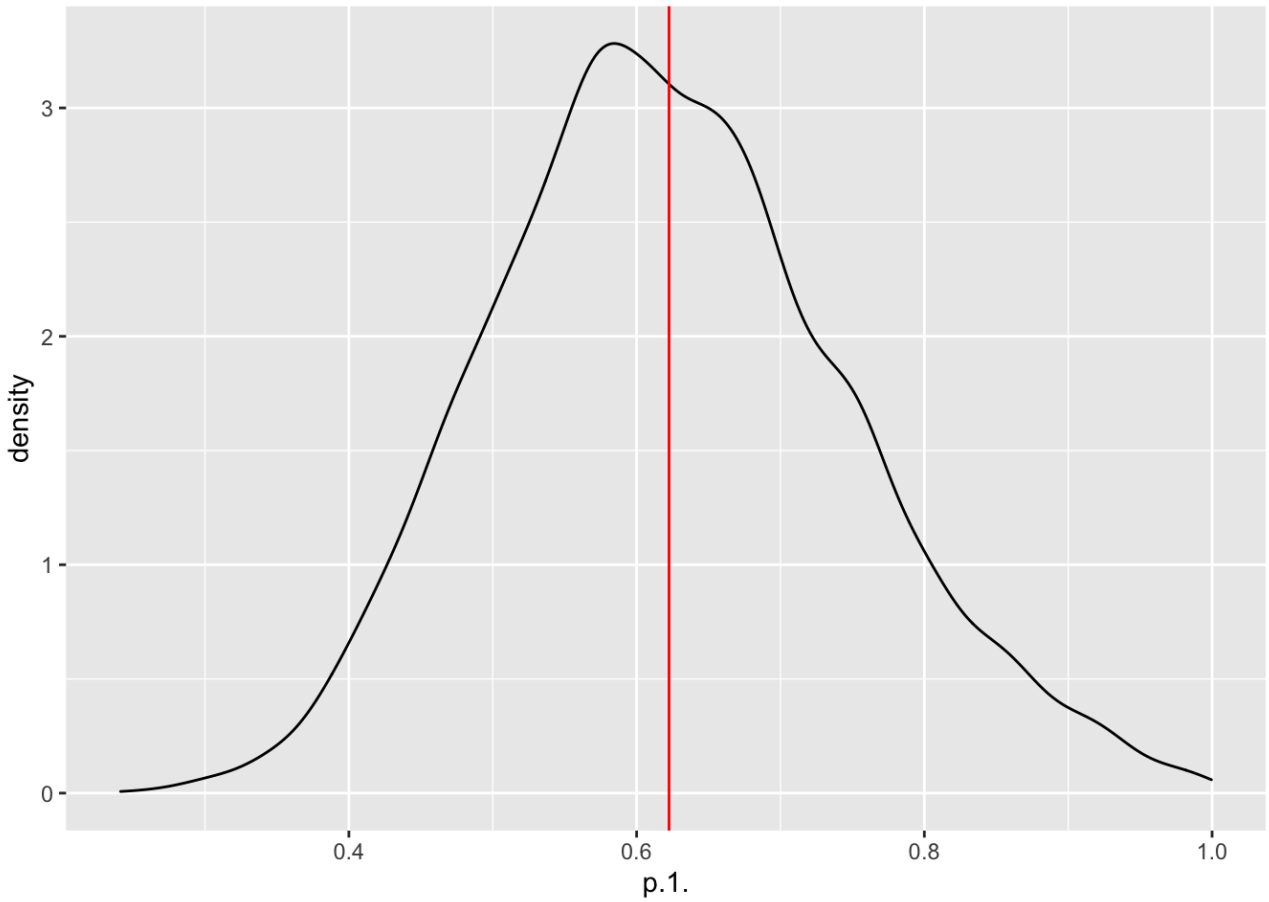
```
# Sp1 = Sp of the TST
ggplot(data=model3_chains, aes(x=Sp1)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$Sp1), color="red")
```



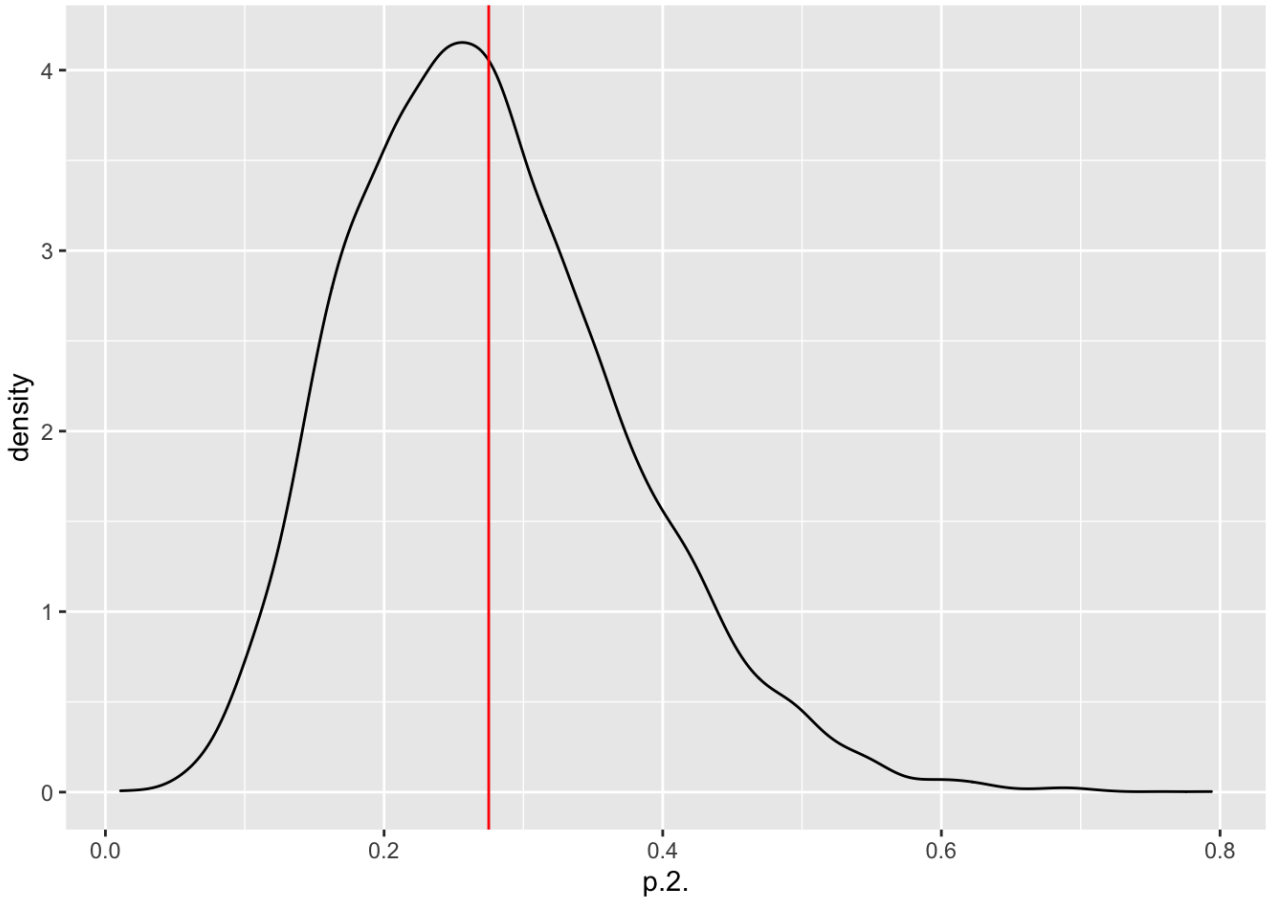

```
# Sp2 = SP of the TB ELISA
ggplot(data=model3_chains, aes(x=Sp2)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$Sp2), color="red")
```



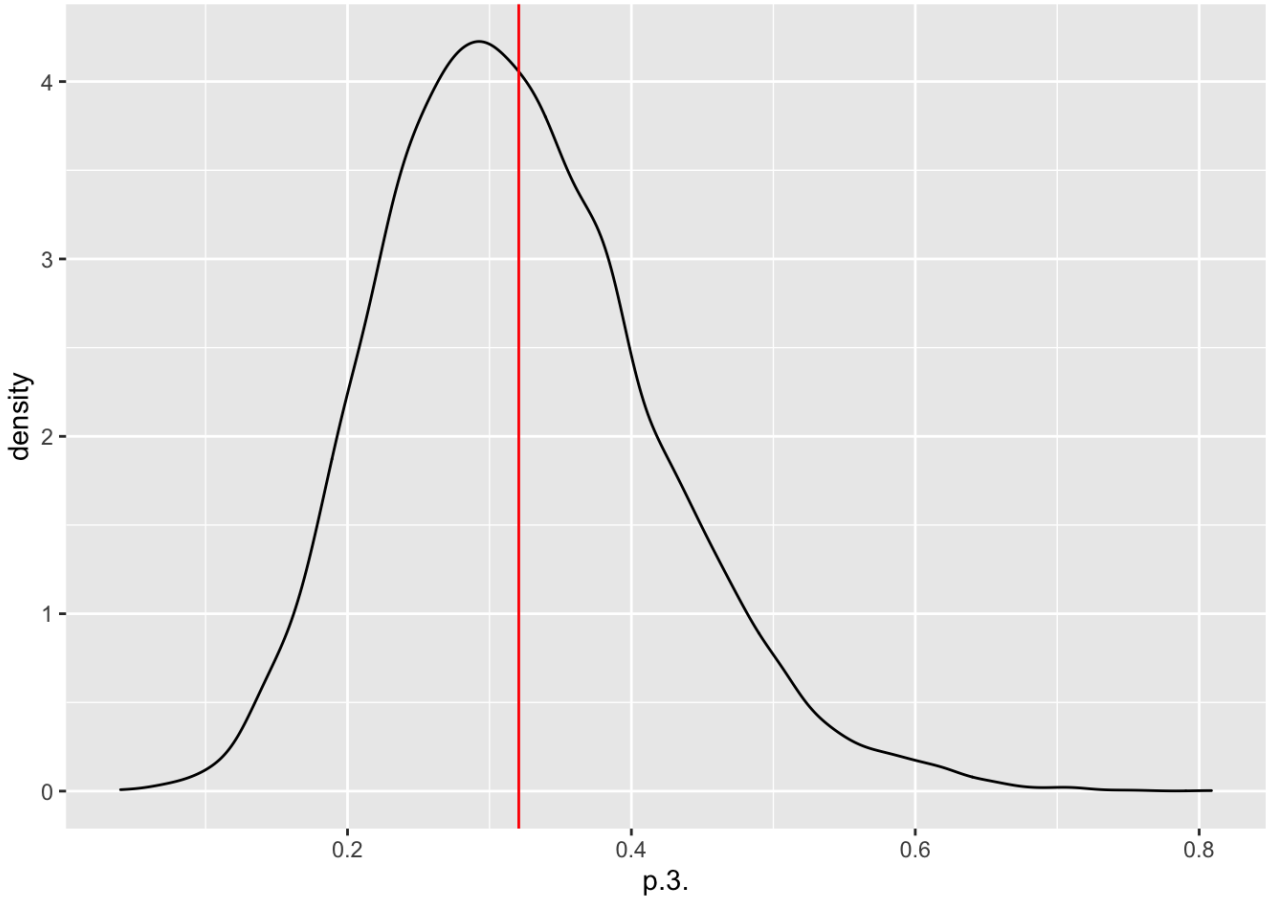

```
# p1 = prevalence in HiP.A
ggplot(data=model3_chains, aes(x=p.1.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.1.), color="red")
```



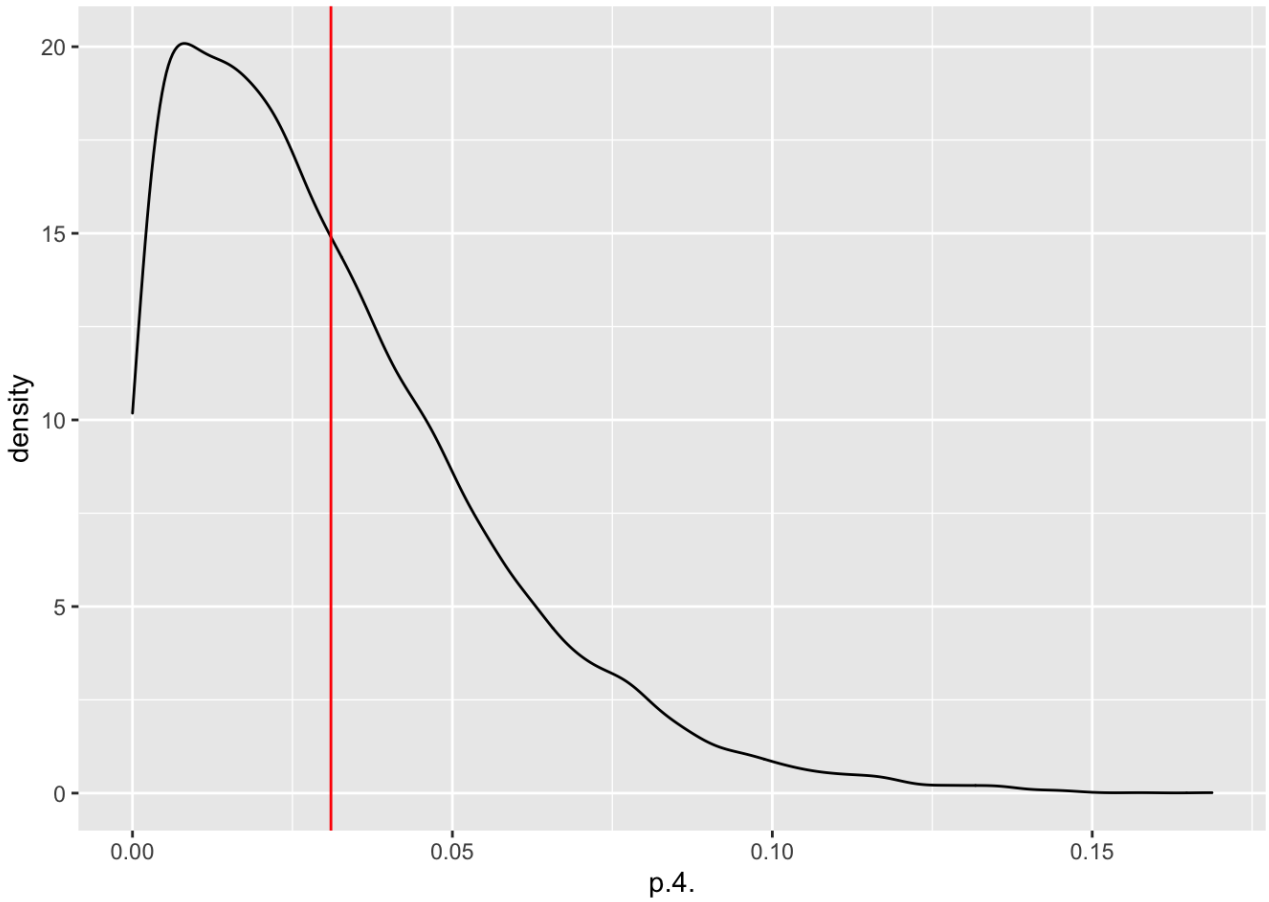
```
# p2 = prevalence in HiP.B
ggplot(data=model3_chains, aes(x=p.2.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.2.), color="red")
```



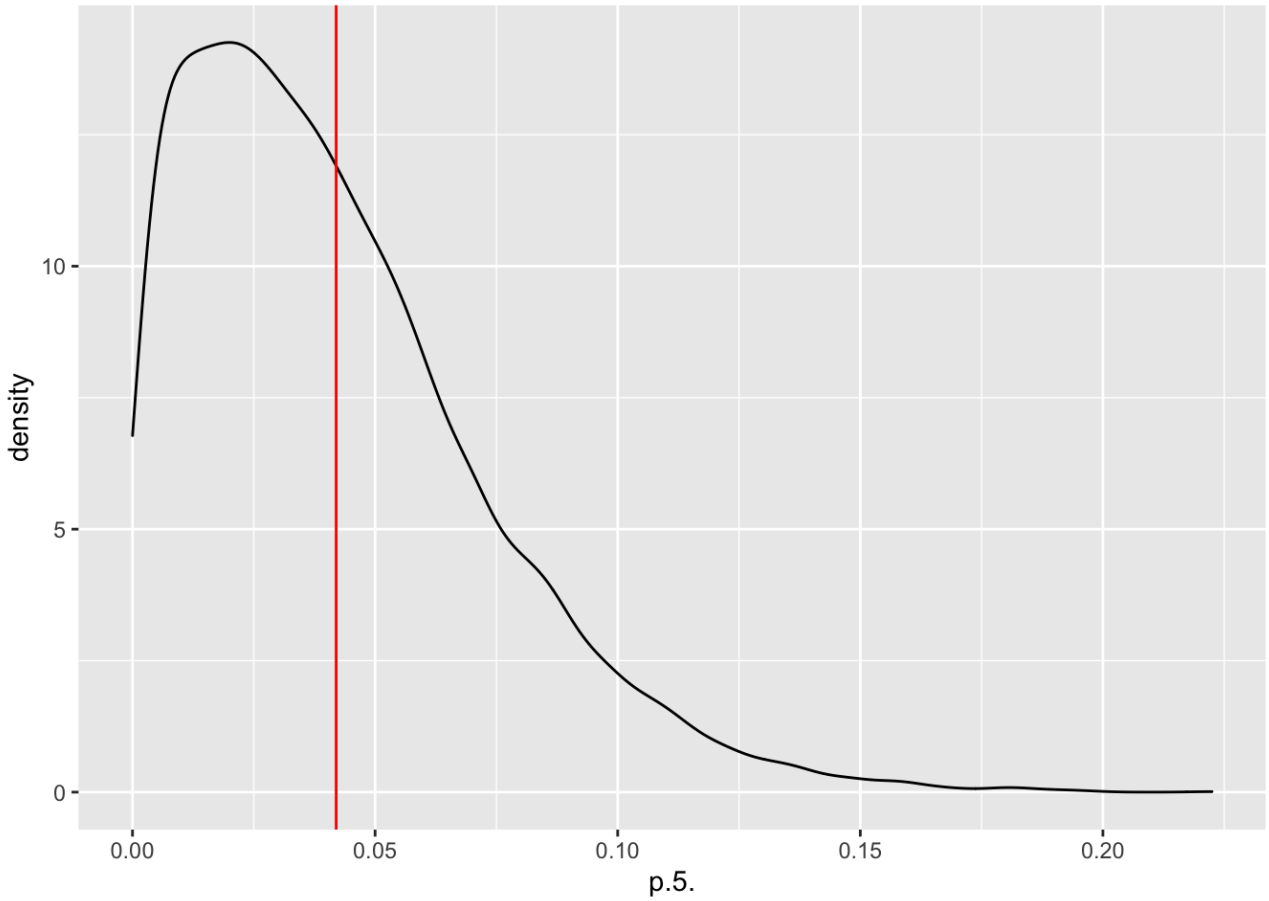

```
# p3 = prevalence in HiP.C
ggplot(data=model3_chains, aes(x=p.3.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.3.), color="red")
```



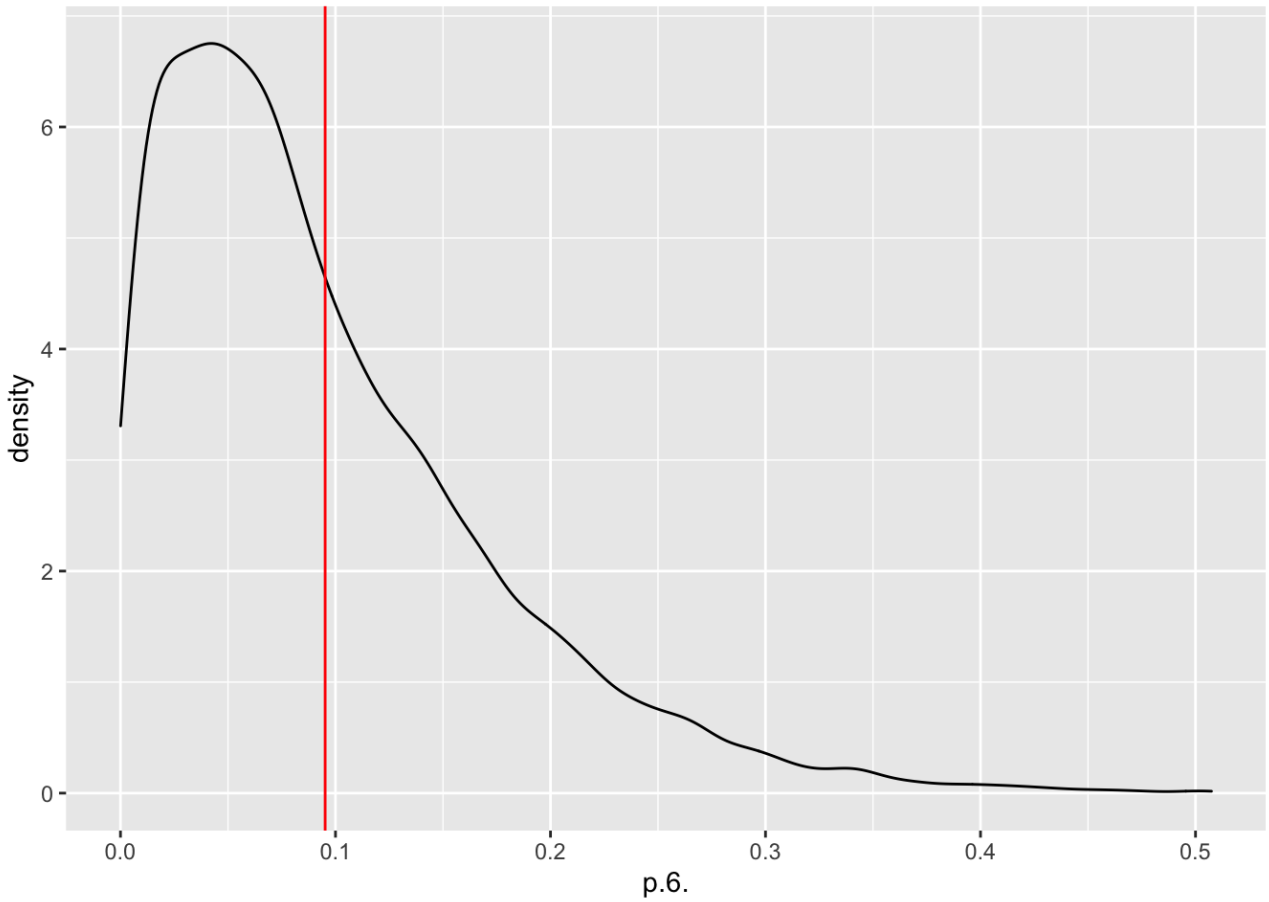

```
# p4 = prevalence in HiP.D
ggplot(data=model3_chains, aes(x=p.4.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.4.), color="red")
```



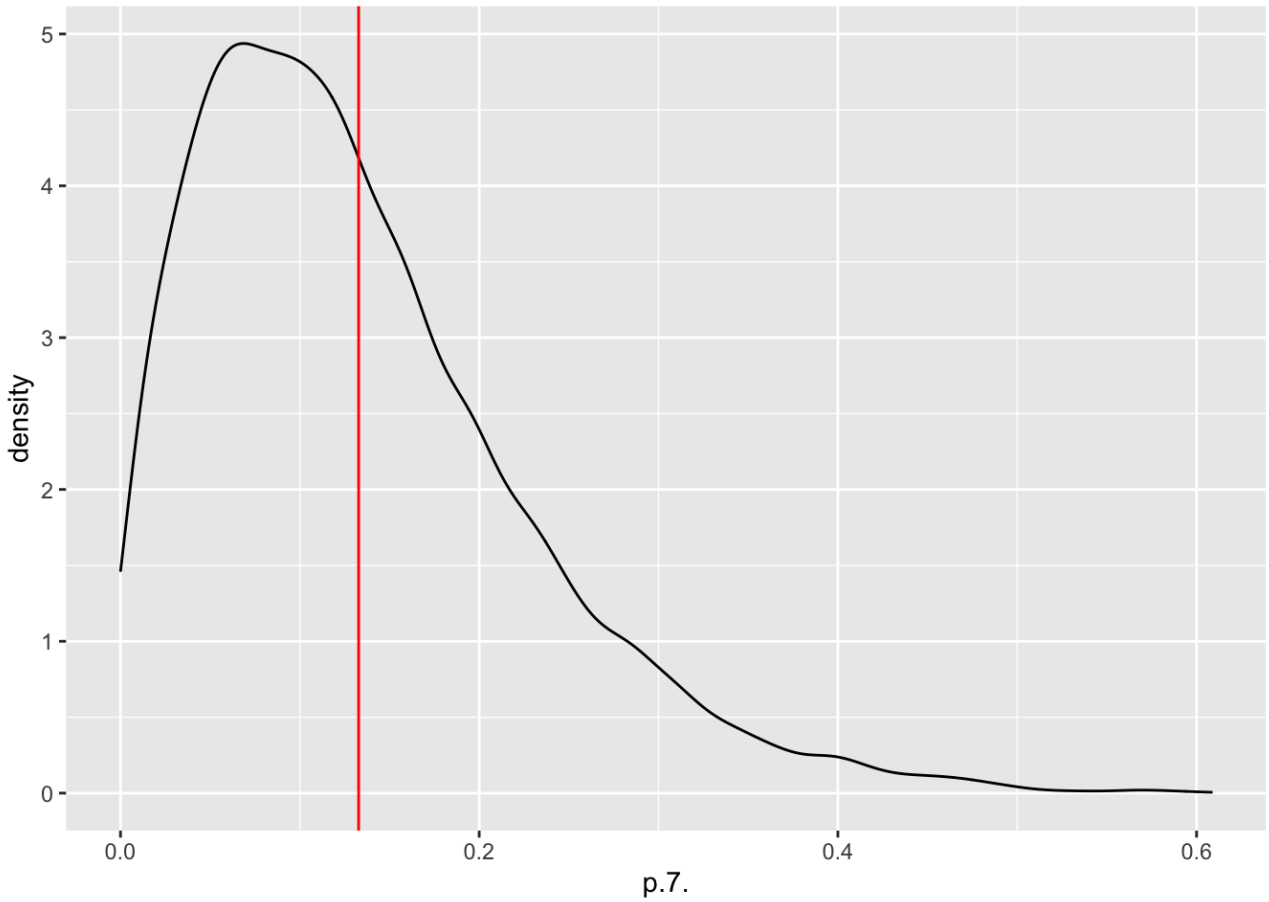
```
# p5 = prevalence in HiP.E
ggplot(data=model3_chains, aes(x=p.5.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.5.), color="red")
```



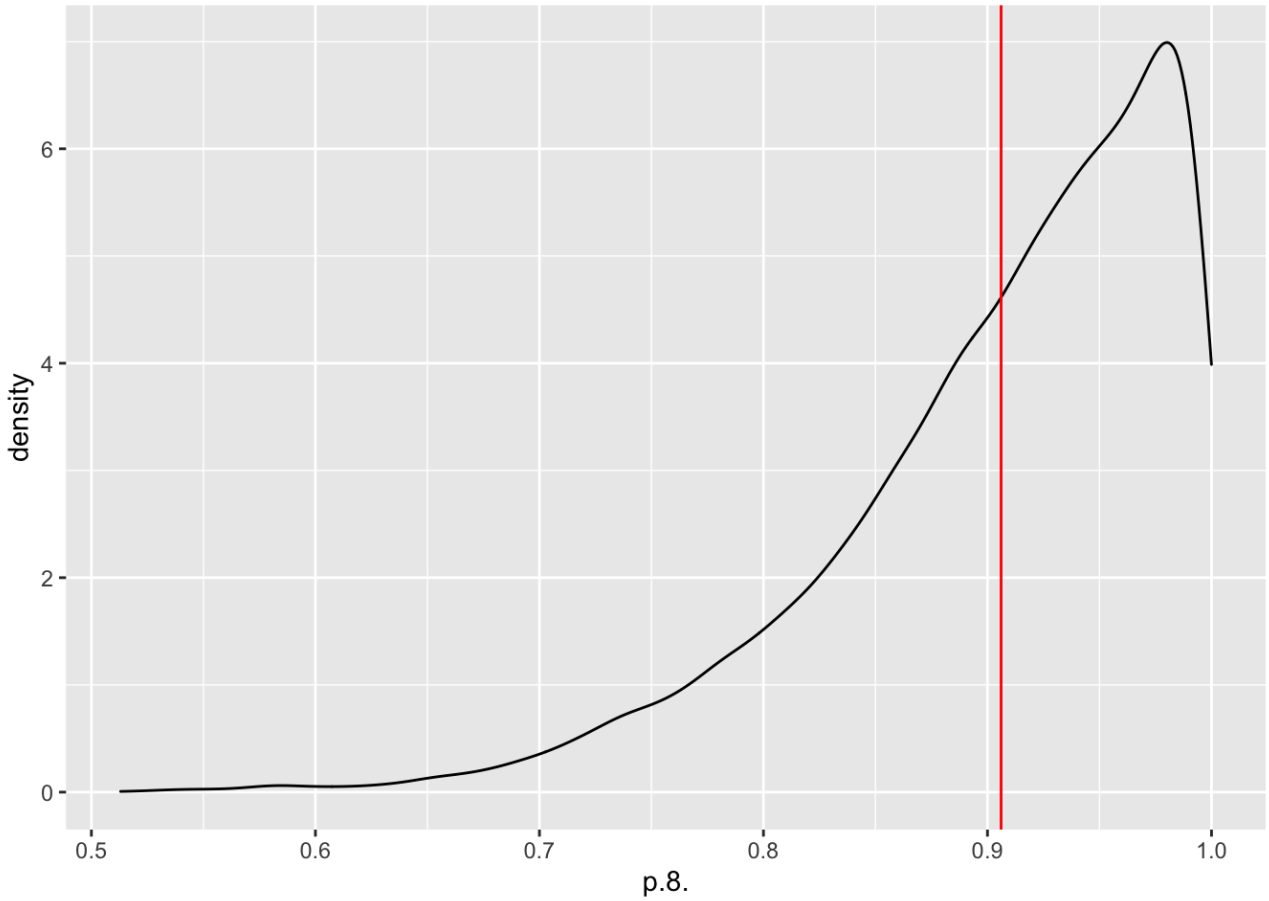

```
# p6 = prevalence in HiP.F
ggplot(data=model3_chains, aes(x=p.6.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.6.), color="red")
```



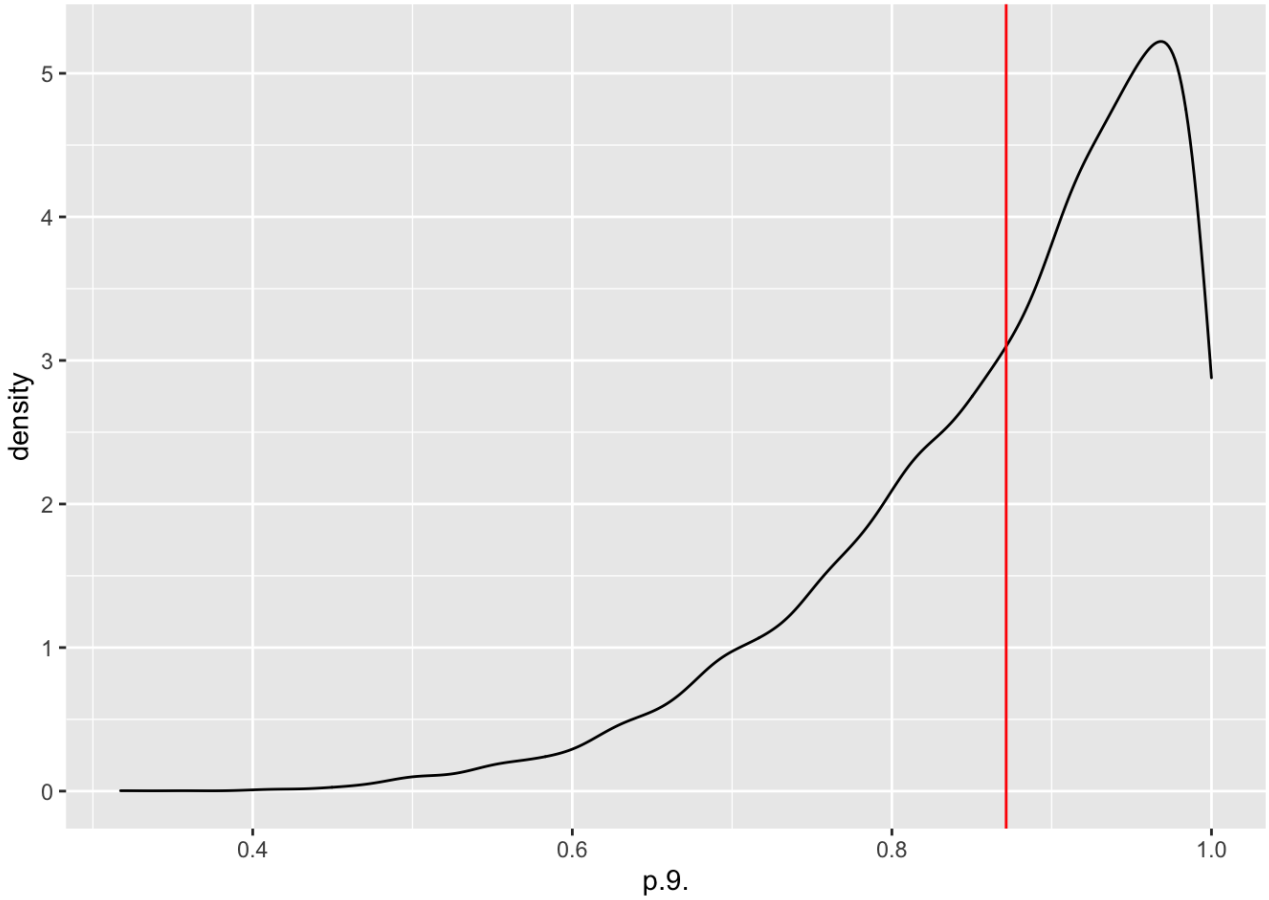

```
# p7 = prevalence in HiP.G
ggplot(data=model3_chains, aes(x=p.7.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.7.), color="red")
```



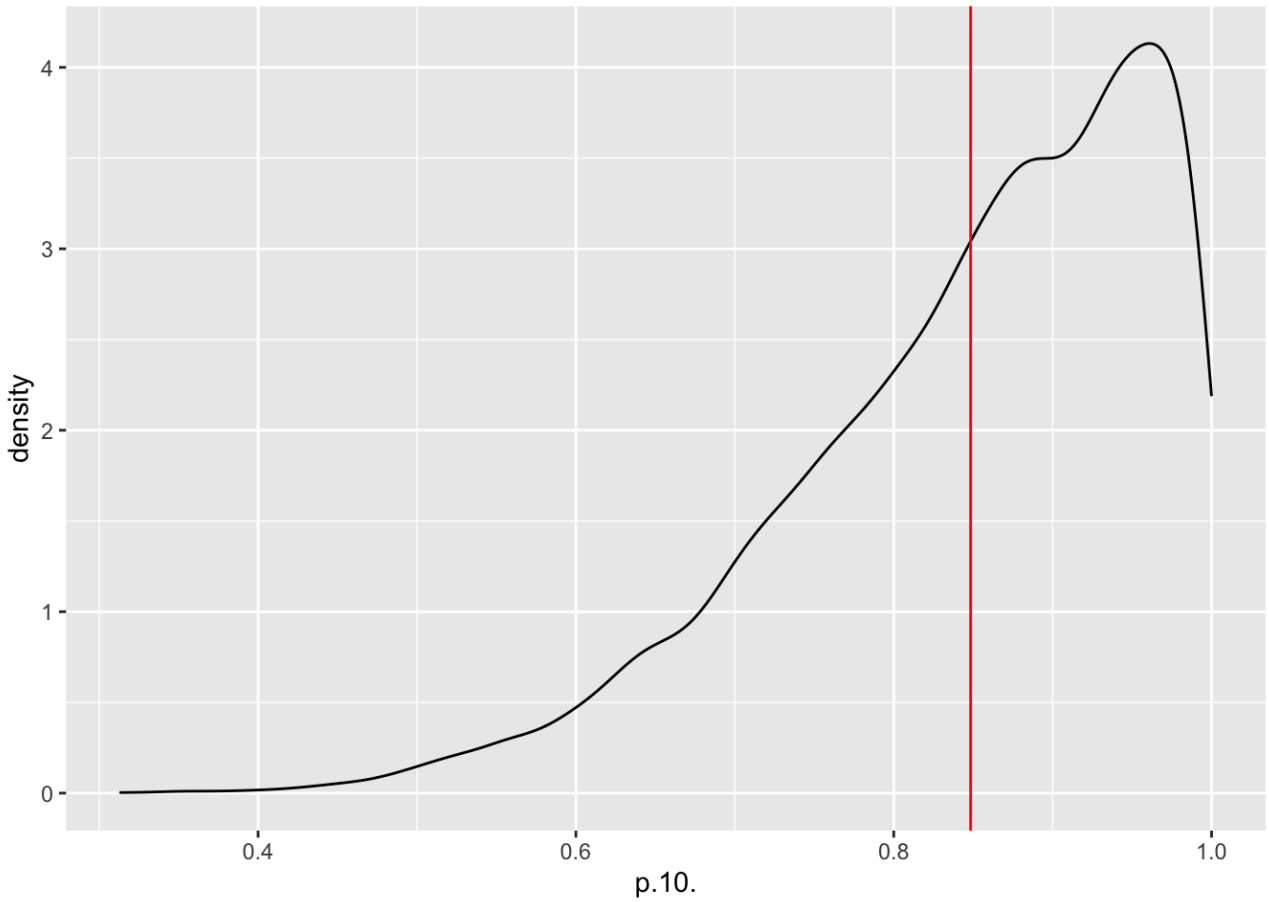
```
# p8 = prevalence in HiP.H
ggplot(data=model3_chains, aes(x=p.8.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.8.), color="red")
```



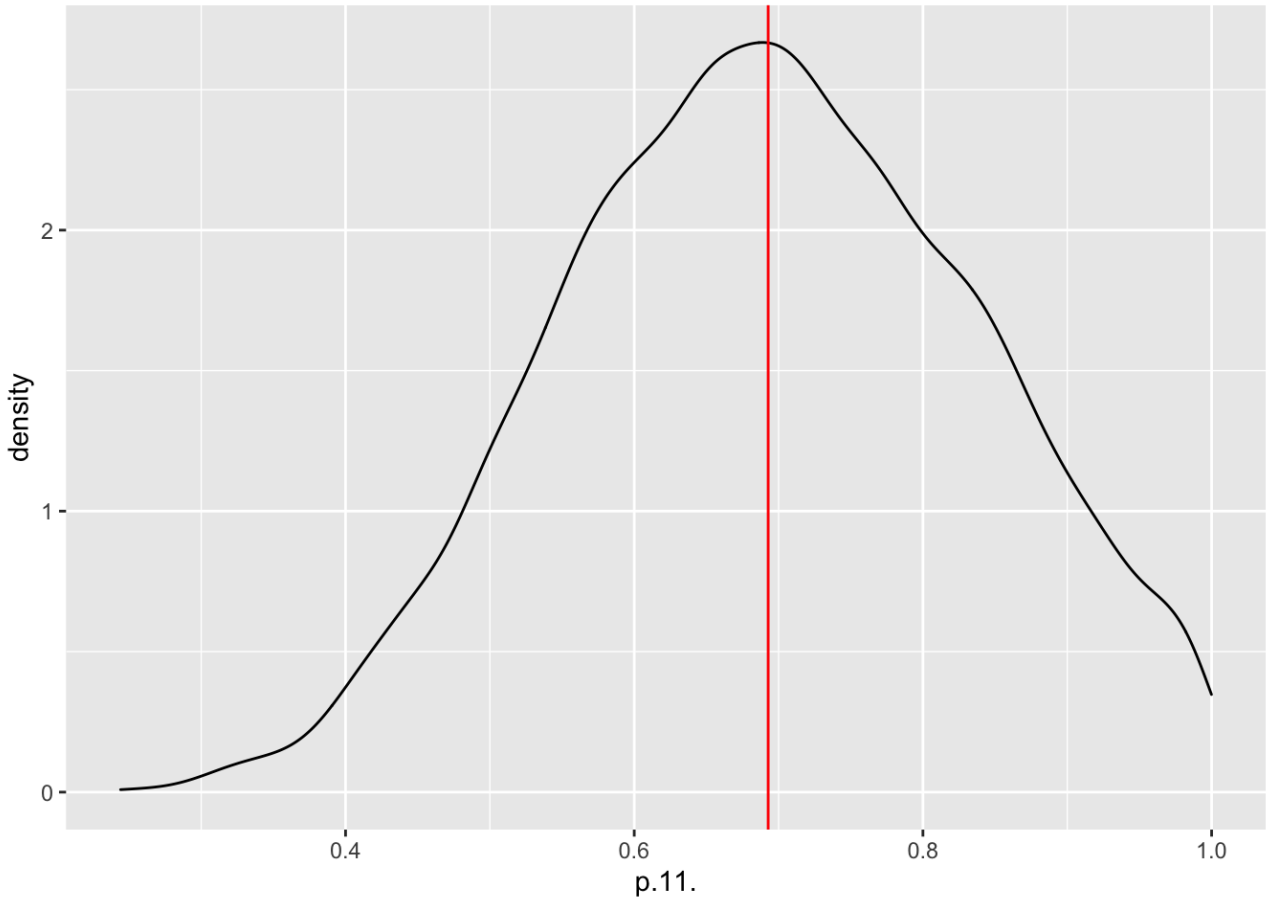

```
# p9 = prevalence in MGR.1
ggplot(data=model3_chains, aes(x=p.9.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.9.), color="red")
```



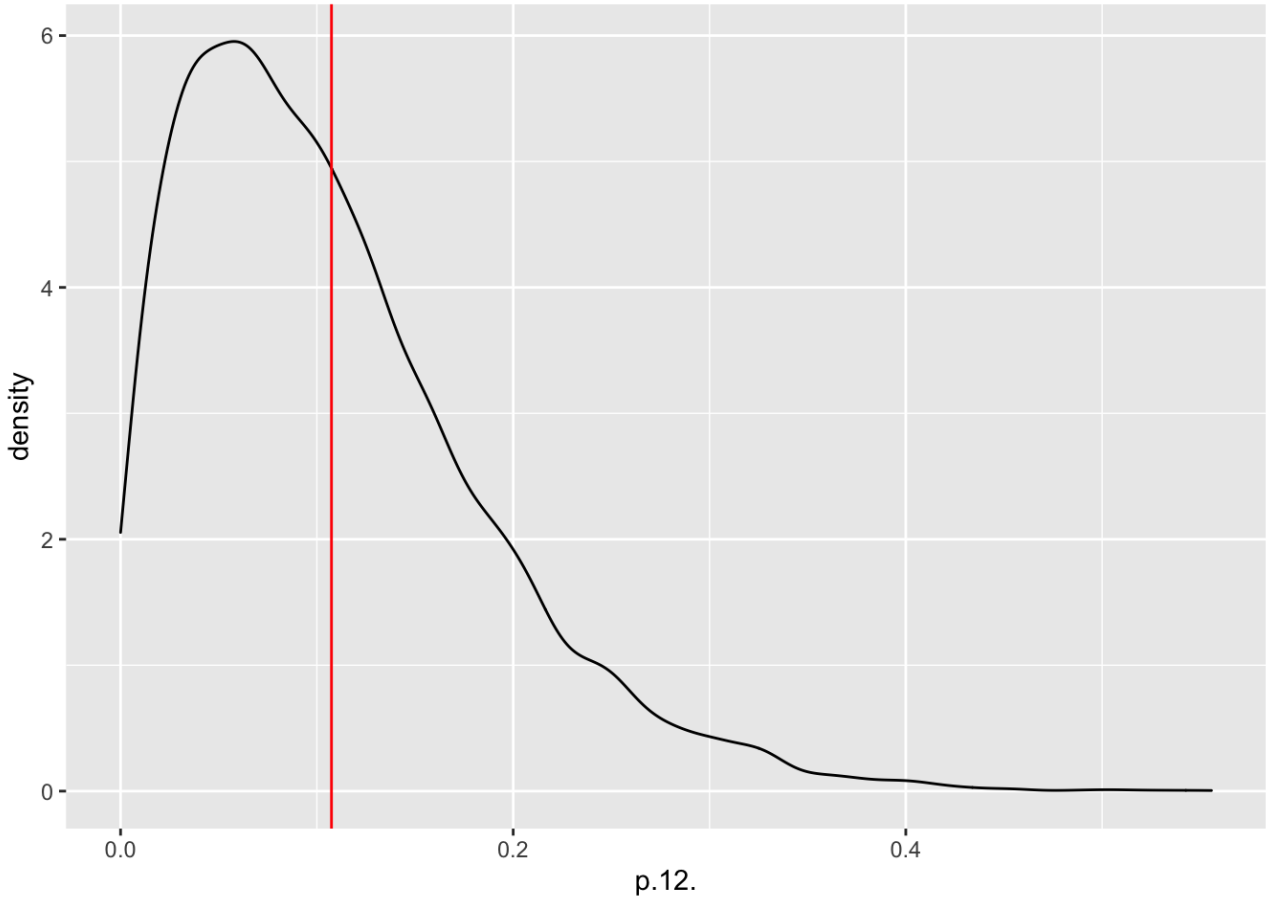

```
# p10 = prevalence in MGR.2
ggplot(data=model3_chains, aes(x=p.10.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.10.), color="red")
```



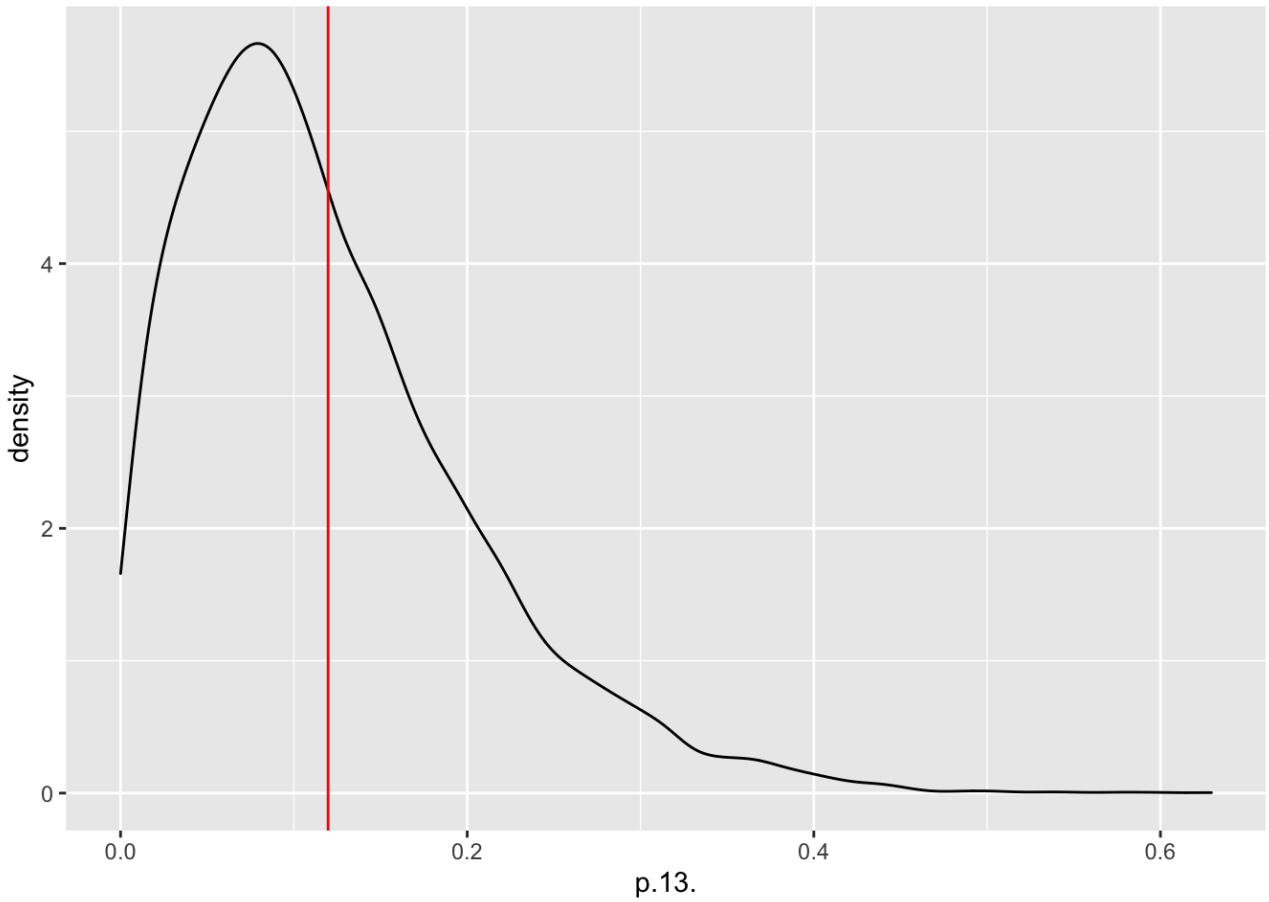
```
# p11 = prevalence in MGR.3
ggplot(data=model3_chains, aes(x=p.11.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.11.), color="red")
```



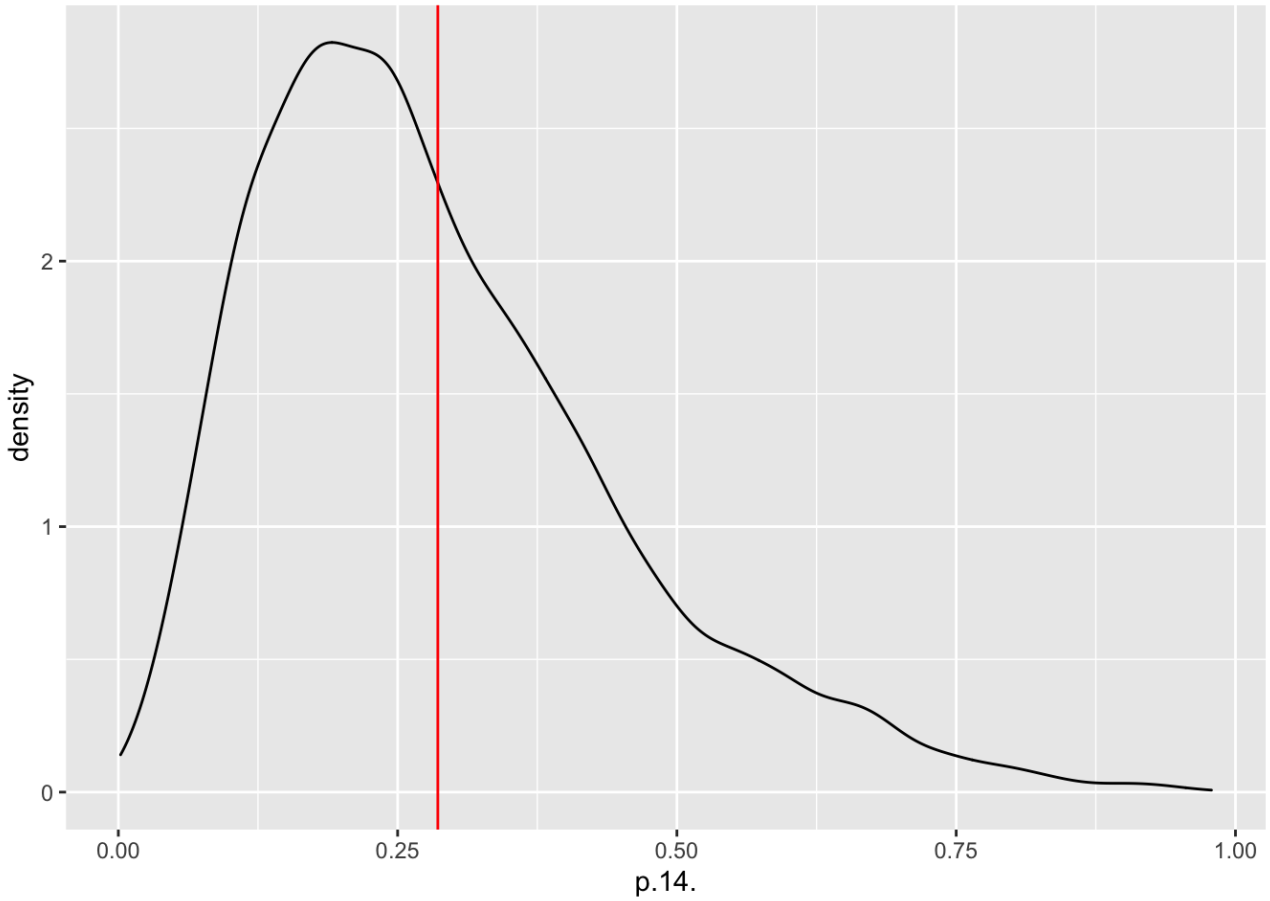

```
# p12 = prevalence in MGR.4
ggplot(data=model3_chains, aes(x=p.12.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.12.), color="red")
```



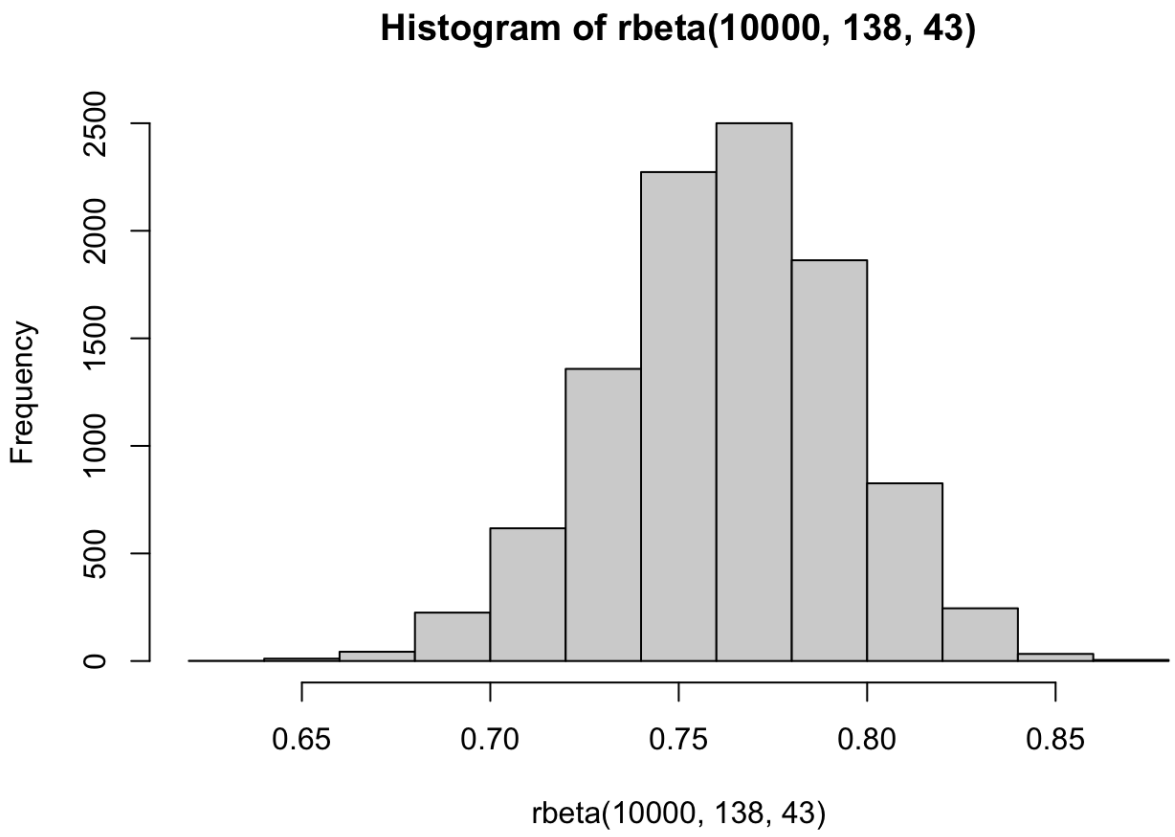

```
# p13 = prevalence in MGR.5
ggplot(data=model3_chains, aes(x=p.13.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.13.), color="red")
```



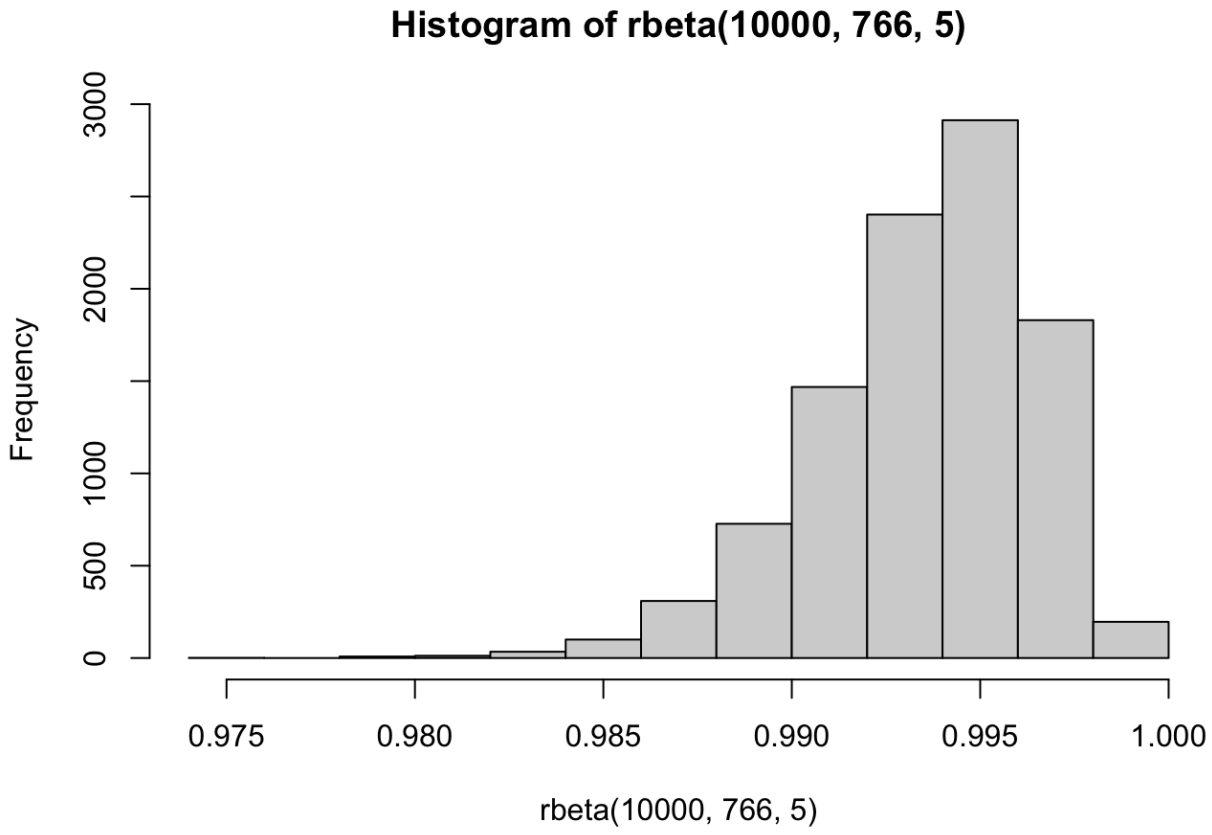
```
# p14 = prevalence in MGR.6
ggplot(data=model3_chains, aes(x=p.14.)) +
  geom_density() + geom_vline(xintercept=mean(model3_chains$p.14.), color="red")
```



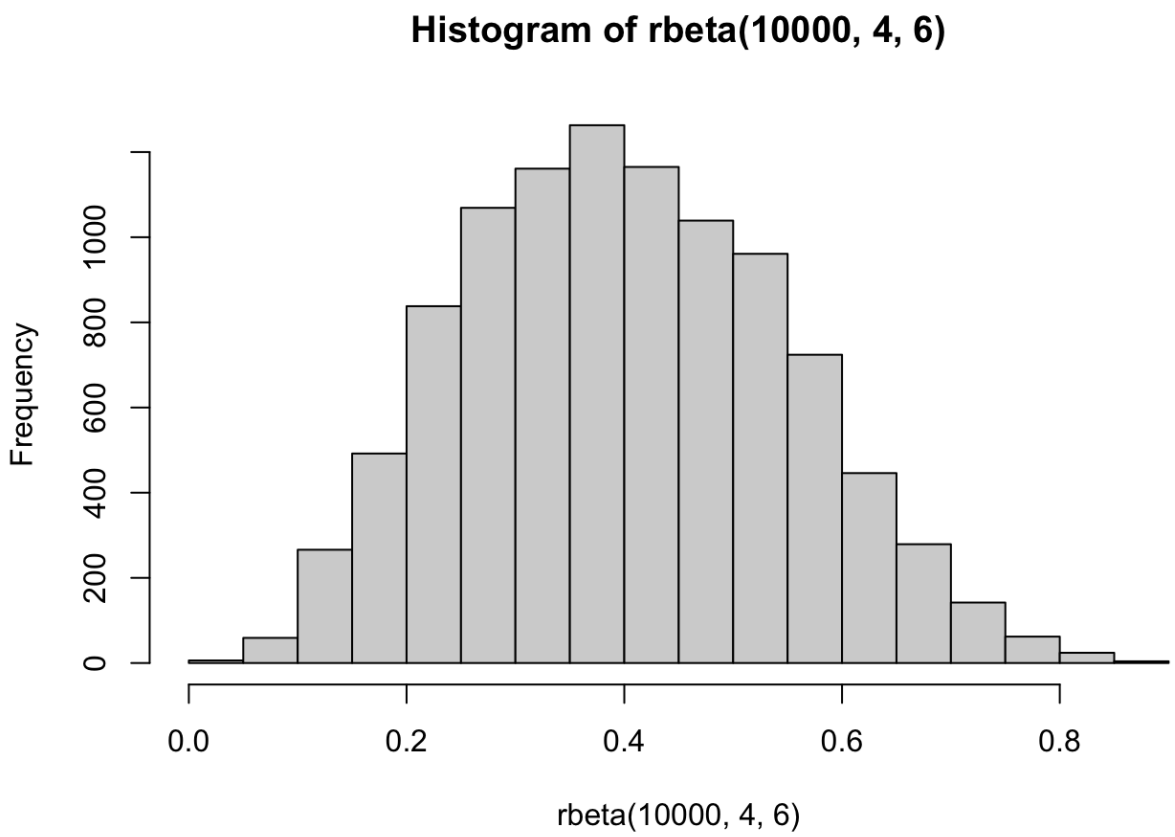

```
# Use informative priors according to known previous estimates for the Se and Sp of TST a
nd TB ELISA
# beta(1,1) * beta(s1,n1-s1) = beta(s1+1,n1-s1+1)
# in other words: from cross-tabs for Se -> beta(a+1,b+1) and for Sp <- beta(d+1,c+1)
# TST Se = 76.5% and Sp = 99.5% (Michel et al. 2011 and unpublished data)
Se1 <- hist(rbeta(10000, 138, 43))
```



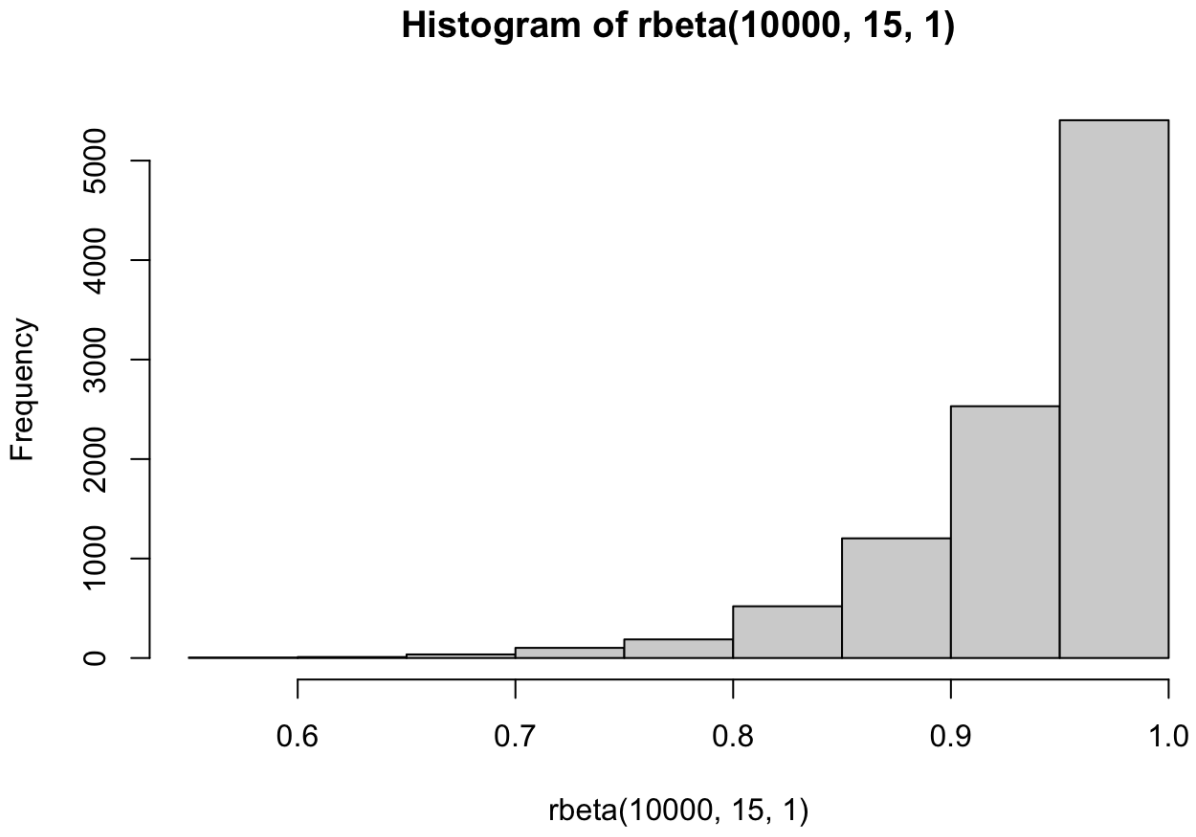

```
Sp1 <- hist(rbeta(10000, 766, 5))
```




```
# TB ELISA Se = 37.5% and Sp = 100% (van der Heijden et al. 2016 and unpublished data)
Se2 <- hist(rbeta(10000, 4, 6))
```




```
Sp2 <- hist(rbeta(10000, 15, 1))
```

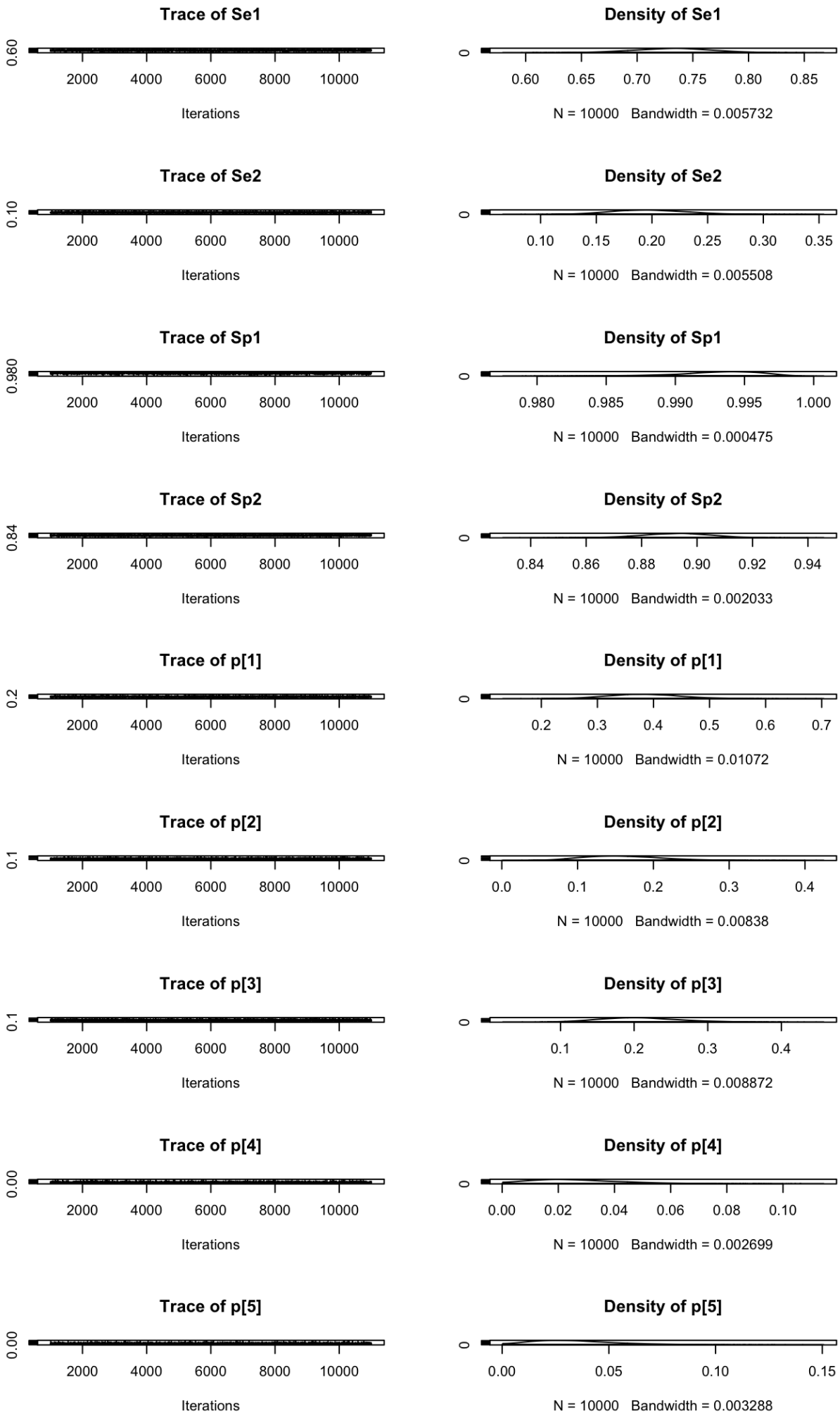


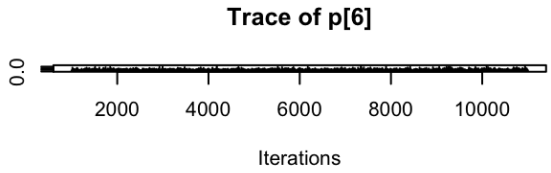
```
# Define the model
model4 <- "model{
  for (i in 1:14) {
    p[i] ~ dbeta(1,1)
    pop[i, 1:4] ~ dmulti(par[i,1:4], n[i])
    par[i, 4] <- p[i]*Se1*Se2 + (1-p[i])*(1-Sp1)*(1-Sp2)
    par[i, 2] <- p[i]*Se1*(1-Se2) + (1-p[i])*(1-Sp1)*Sp2
    par[i, 3] <- p[i]*(1-Se1)*Se2 + (1-p[i])*Sp1*(1-Sp2)
    par[i, 1] <- p[i]*(1-Se1)*(1-Se2) + (1-p[i])*Sp1*Sp2
    }
  ## priors
  Se1 ~ dbeta(138, 43)
  Sp1 ~ dbeta(766, 5)
  Se2 ~ dbeta(4, 6)
  Sp2 ~ dbeta(15, 1)

}"

# Compile the model
model4_jags <- jags.model(textConnection(model4),inits=list(.RNG.name = "base::Wichmann-H
ill", .RNG.seed = 2018),
                          data=list(pop=pop2, n=n2))
```

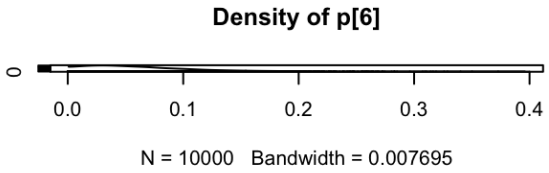
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 14
##    Unobserved stochastic nodes: 18
##    Total graph size: 254
##
## Initializing model
```

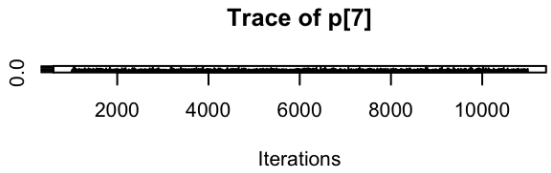
```
        # in Bronsvoort et al. own INITS defined (based on??) and also n.chains=3, n.adap
t=50000)
        # in datacamp course usually inits=list(.RNG.name = "base::Wichmann-Hill", .RNG.s
eed = 2018) which is what was used here.
        # should find out how to define our own inits - this seems key for the outcome!


# Simulate the model
model4_sim <- coda.samples(model=model4_jags, variable.names=c("p", "Se1", "Se2", "Sp1",
"Sp2"),
                           n.iter=10000)
        # in Bronsvoort et al. n.iter=250000, n.thin=100
        # the higher the number of iterarions, the more stable and reliable the outcome,
 but probably depends on volume of data too?
        # used 10000 here for now for speed of estimation, should probably be higher!
```

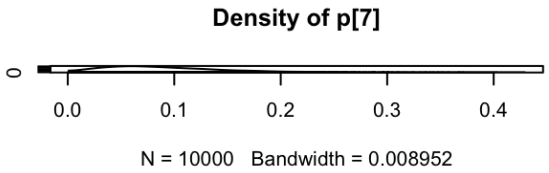
```
# Plot the posterior
plot(model4_sim)
```

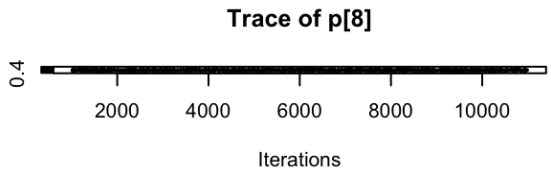

Trace of Se1
Density of Se1
N = 10000 Bandwidth = 0.005732
Trace of Se2
Density of Se2
N = 10000 Bandwidth = 0.005508
Trace of Sp1
Density of Sp1
N = 10000 Bandwidth = 0.000475
Trace of Sp2
Density of Sp2
N = 10000 Bandwidth = 0.002033
Trace of p[1]
Density of p[1]
N = 10000 Bandwidth = 0.01072
Trace of p[2]
Density of p[2]
N = 10000 Bandwidth = 0.00838
Trace of p[3]
Density of p[3]
N = 10000 Bandwidth = 0.008872
Trace of p[4]
Density of p[4]
N = 10000 Bandwidth = 0.002699
Trace of p[5]
Density of p[5]
N = 10000 Bandwidth = 0.003288
Iterations

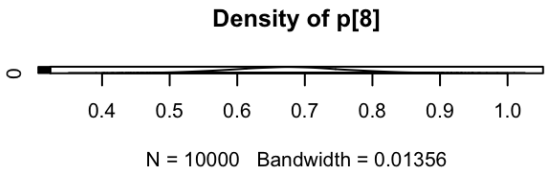

Trace of p[6]
0.0
2000
4000
6000
8000
10000
Iterations

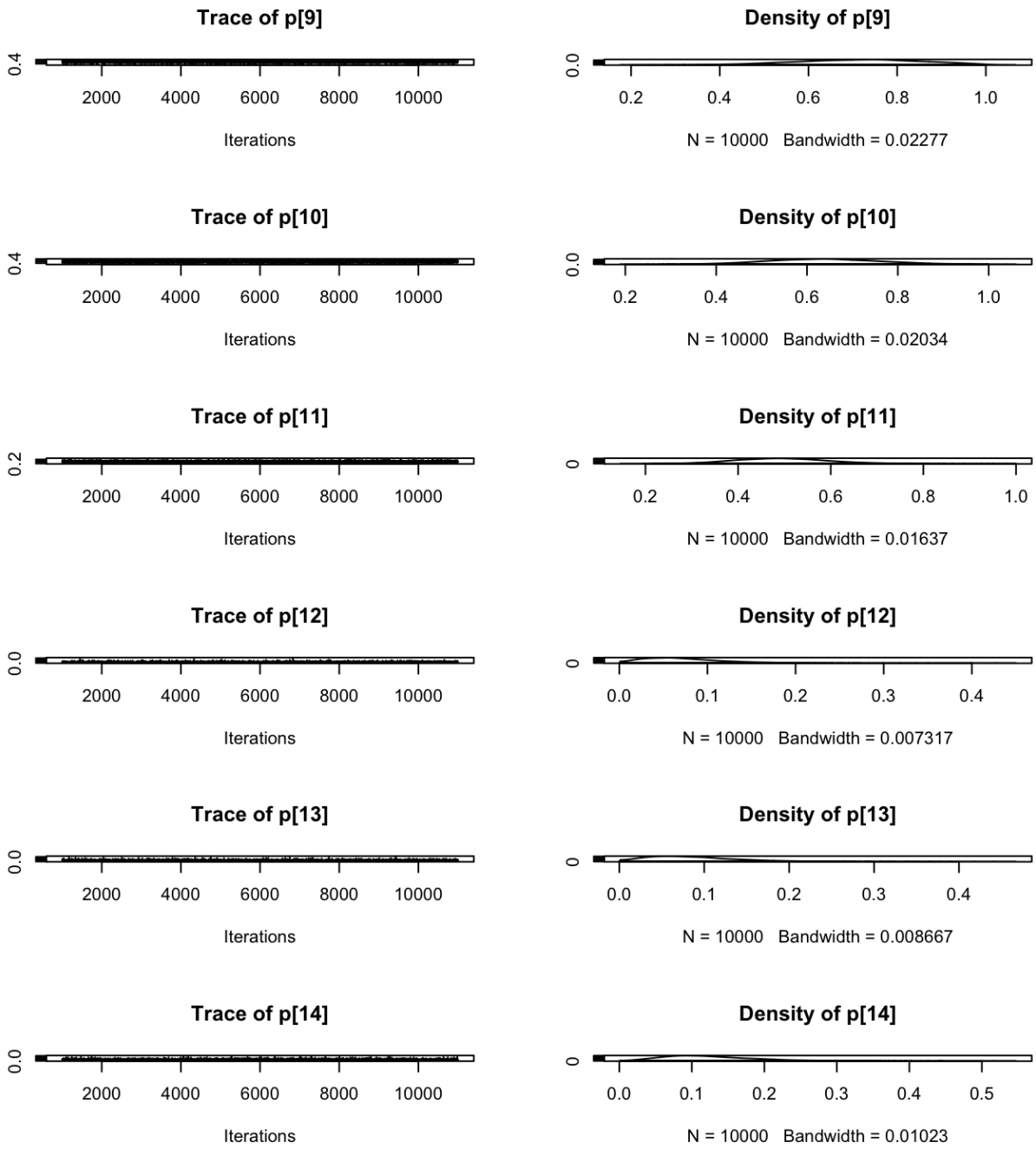

Density of p[6]
0
0.0
0.1
0.2
0.3
0.4
N = 10000 Bandwidth = 0.007695


Trace of p[7]
0.0
2000
4000
6000
8000
10000
Iterations


Density of p[7]
0
0.0
0.1
0.2
0.3
0.4
N = 10000 Bandwidth = 0.008952


Trace of p[8]
0.4
2000
4000
6000
8000
10000
Iterations

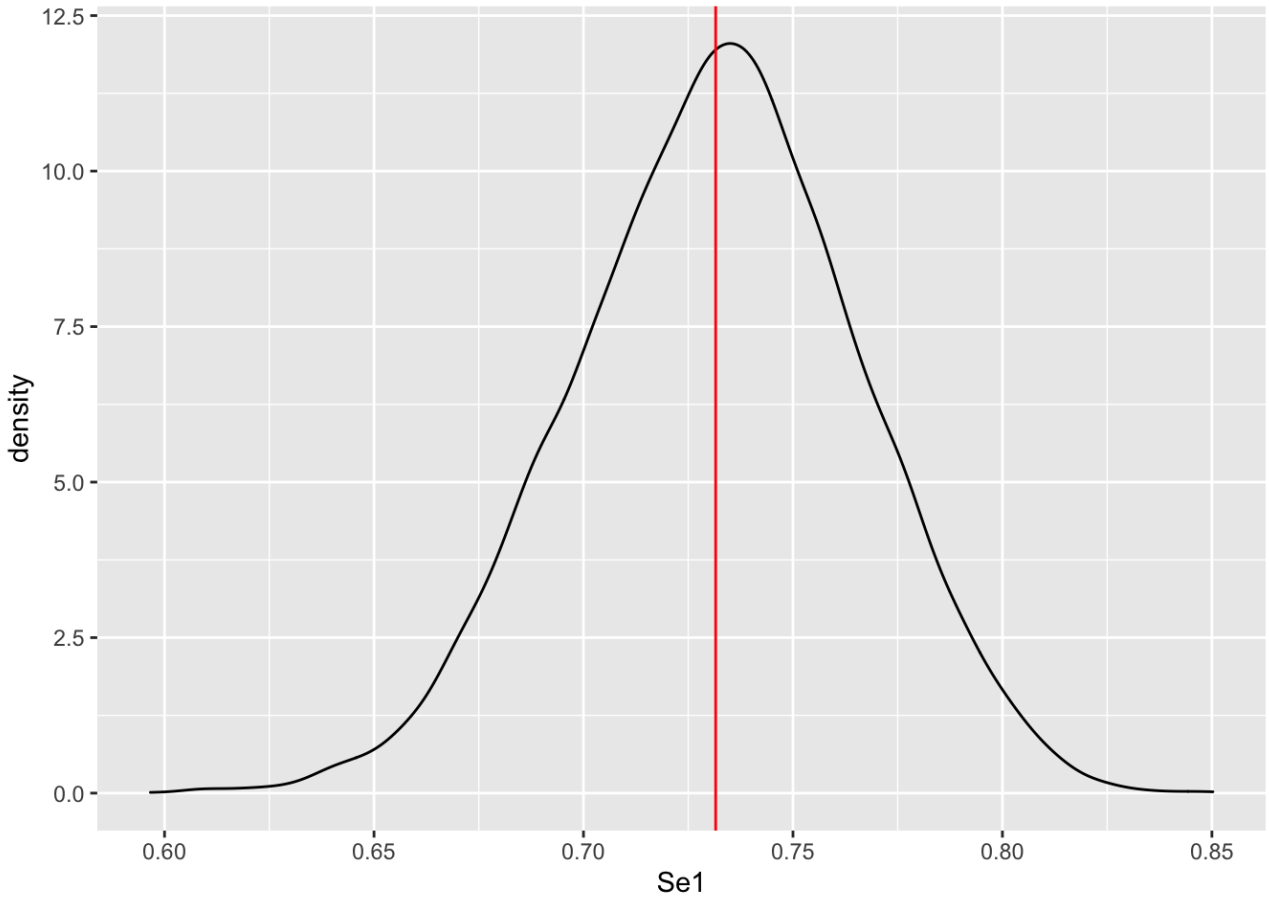

Density of p[8]
0
0.4
0.5
0.6
0.7
0.8
0.9
1.0
N = 10000 Bandwidth = 0.01356



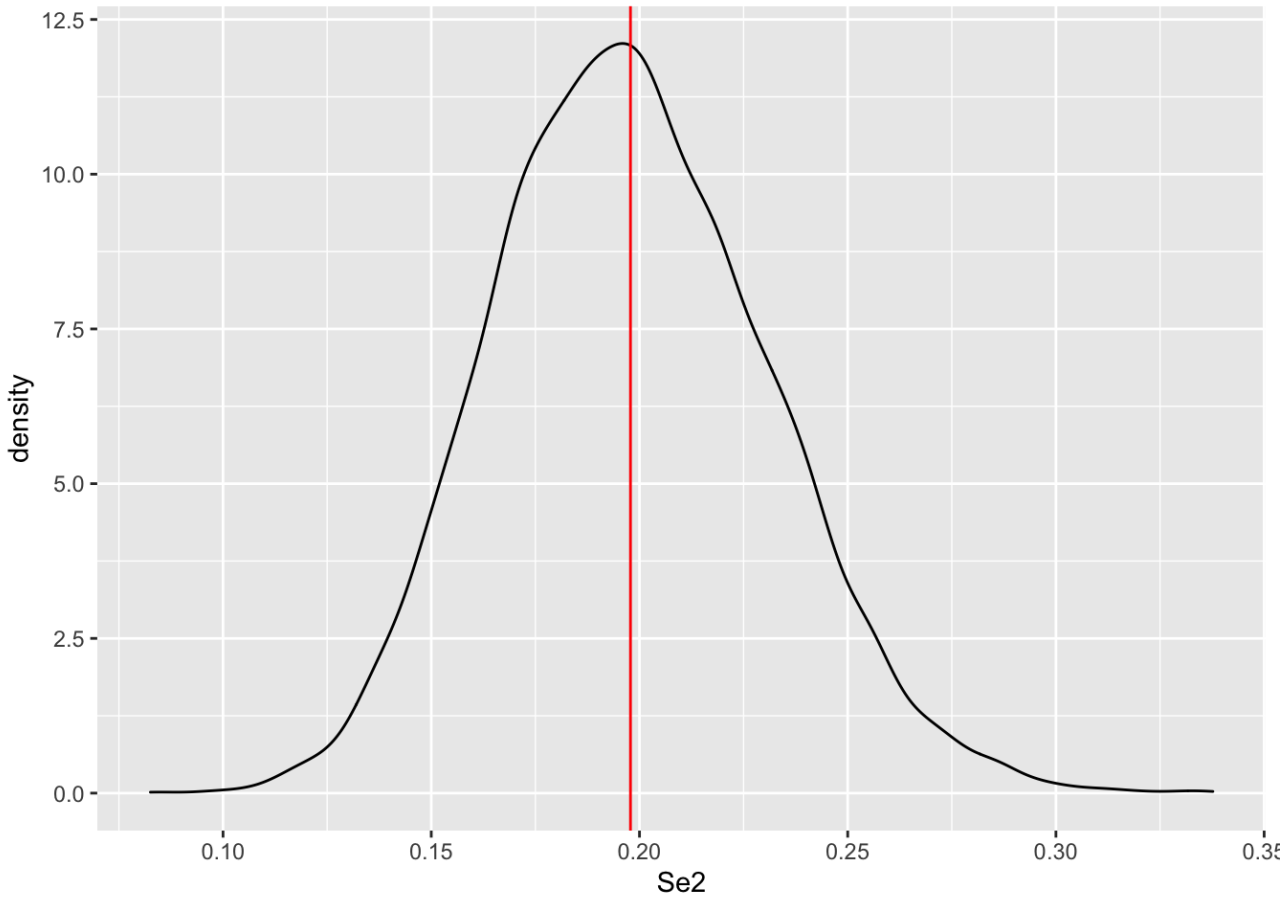

```
# Summarize the data
summary(model4_sim)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD  Naive SE Time-series SE
## Se1   0.73155 0.034119 3.412e-04      0.0006722
## Se2   0.19784 0.032788 3.279e-04      0.0004728
## Sp1   0.99316 0.002933 2.933e-05      0.0000482
## Sp2   0.89257 0.012104 1.210e-04      0.0001680
## p[1]  0.38399 0.063789 6.379e-04      0.0009202
## p[2]  0.15967 0.049880 4.988e-04      0.0007001
## p[3]  0.20813 0.052812 5.281e-04      0.0007250
## p[4]  0.02725 0.016306 1.631e-04      0.0002521
## p[5]  0.03483 0.020194 2.019e-04      0.0003133
## p[6]  0.06397 0.048483 4.848e-04      0.0008500
## p[7]  0.08921 0.054063 5.406e-04      0.0008667
## p[8]  0.67559 0.081901 8.190e-04      0.0012372
## p[9]  0.71585 0.135537 1.355e-03      0.0018745
## p[10] 0.63661 0.121046 1.210e-03      0.0016276
## p[11] 0.49253 0.097448 9.745e-04      0.0013112
## p[12] 0.07237 0.044740 4.474e-04      0.0007745
## p[13] 0.08719 0.053769 5.377e-04      0.0008529
## p[14] 0.12441 0.064039 6.404e-04      0.0009979
##
## 2. Quantiles for each variable:
##
##           2.5%     25%     50%     75%   97.5%
## Se1   0.663852 0.70903 0.73237 0.75476 0.79647
## Se2   0.138056 0.17471 0.19639 0.21954 0.26532
## Sp1   0.986243 0.99151 0.99354 0.99530 0.99767
## Sp2   0.868271 0.88454 0.89298 0.90079 0.91564
## p[1]  0.265504 0.34007 0.38215 0.42575 0.51649
## p[2]  0.075022 0.12335 0.15547 0.19155 0.26711
## p[3]  0.115055 0.17069 0.20465 0.24194 0.32082
## p[4]  0.003024 0.01528 0.02475 0.03681 0.06536
## p[5]  0.004653 0.01997 0.03180 0.04620 0.08265
## p[6]  0.003477 0.02767 0.05331 0.08905 0.18536
## p[7]  0.012588 0.04900 0.07957 0.12041 0.21943
## p[8]  0.517256 0.62100 0.67498 0.72917 0.83782
## p[9]  0.444850 0.62130 0.72131 0.81467 0.95851
## p[10] 0.406236 0.55160 0.63489 0.71891 0.87779
## p[11] 0.310097 0.42445 0.49004 0.55659 0.69032
## p[12] 0.009378 0.03921 0.06460 0.09757 0.17656
## p[13] 0.010904 0.04735 0.07828 0.11649 0.21722
## p[14] 0.028455 0.07840 0.11415 0.16003 0.27596
```

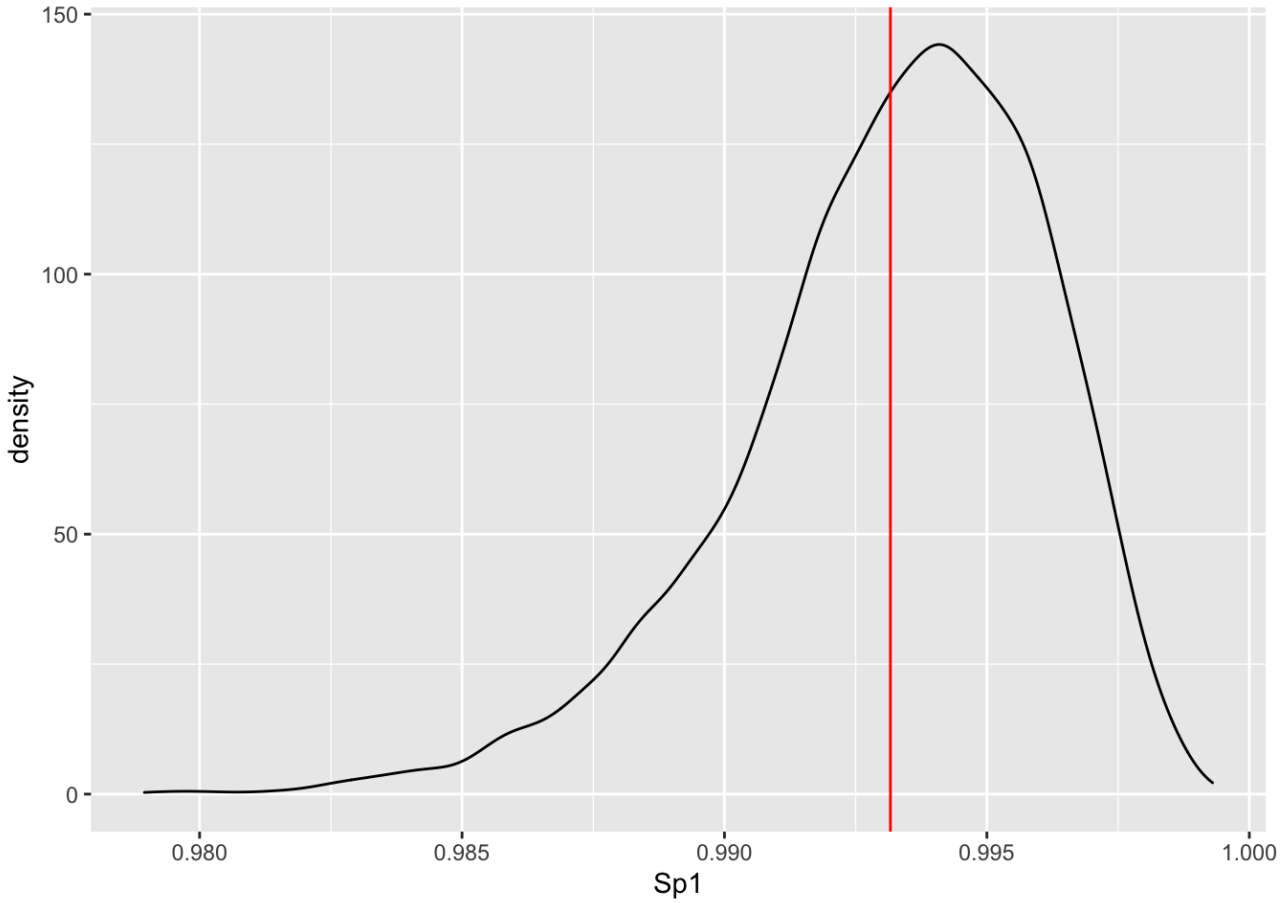
```
# Store the MC chain
model4_chains <- data.frame(model4_sim[[1]], iter = 1:10000)

# Make plots of the different parameters

# Se1 = Se of the TST
ggplot(data=model4_chains, aes(x=Se1)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$Se1), color="red")
```



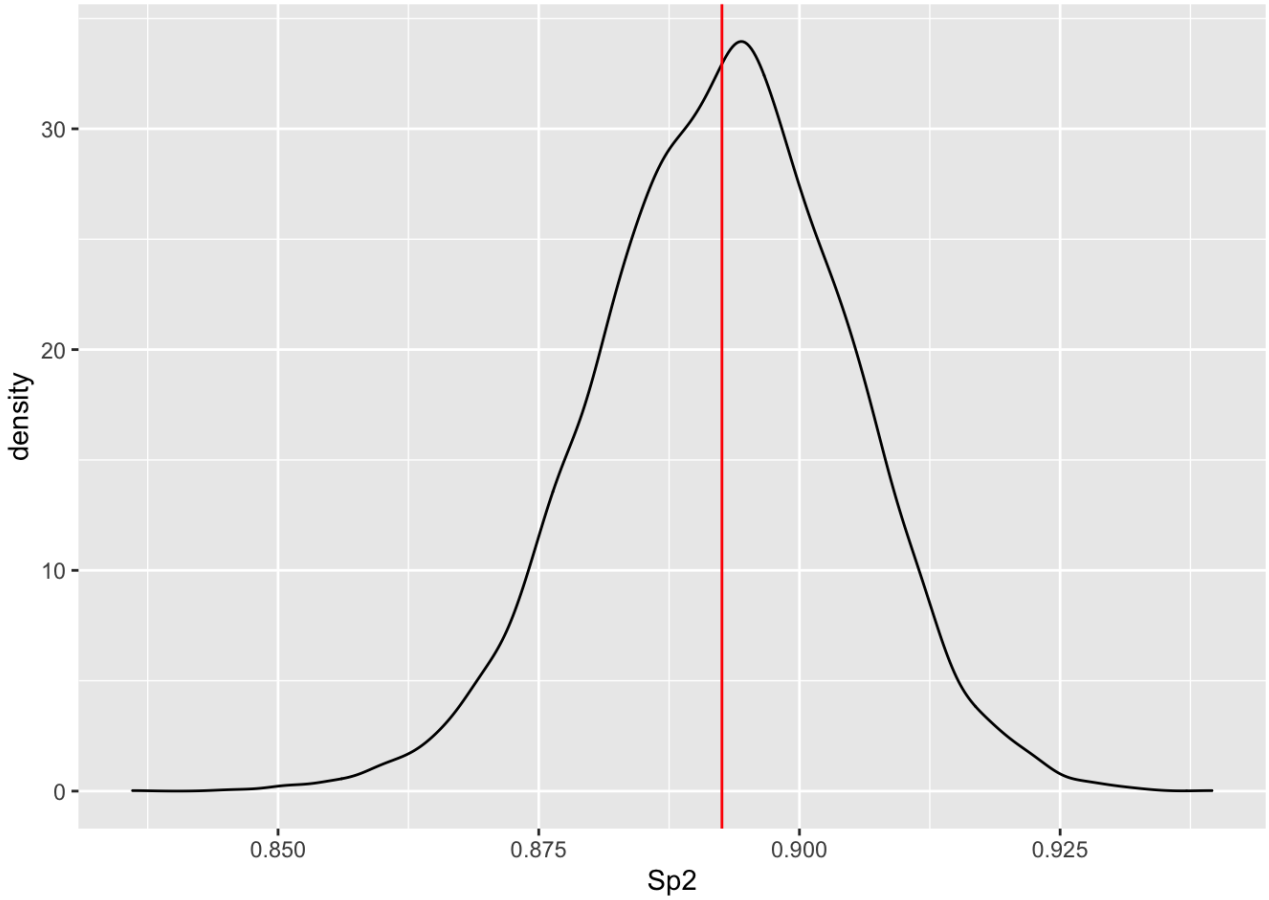

```
# Se2 = Se of the TB ELISA
ggplot(data=model4_chains, aes(x=Se2)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$Se2), color="red")
```



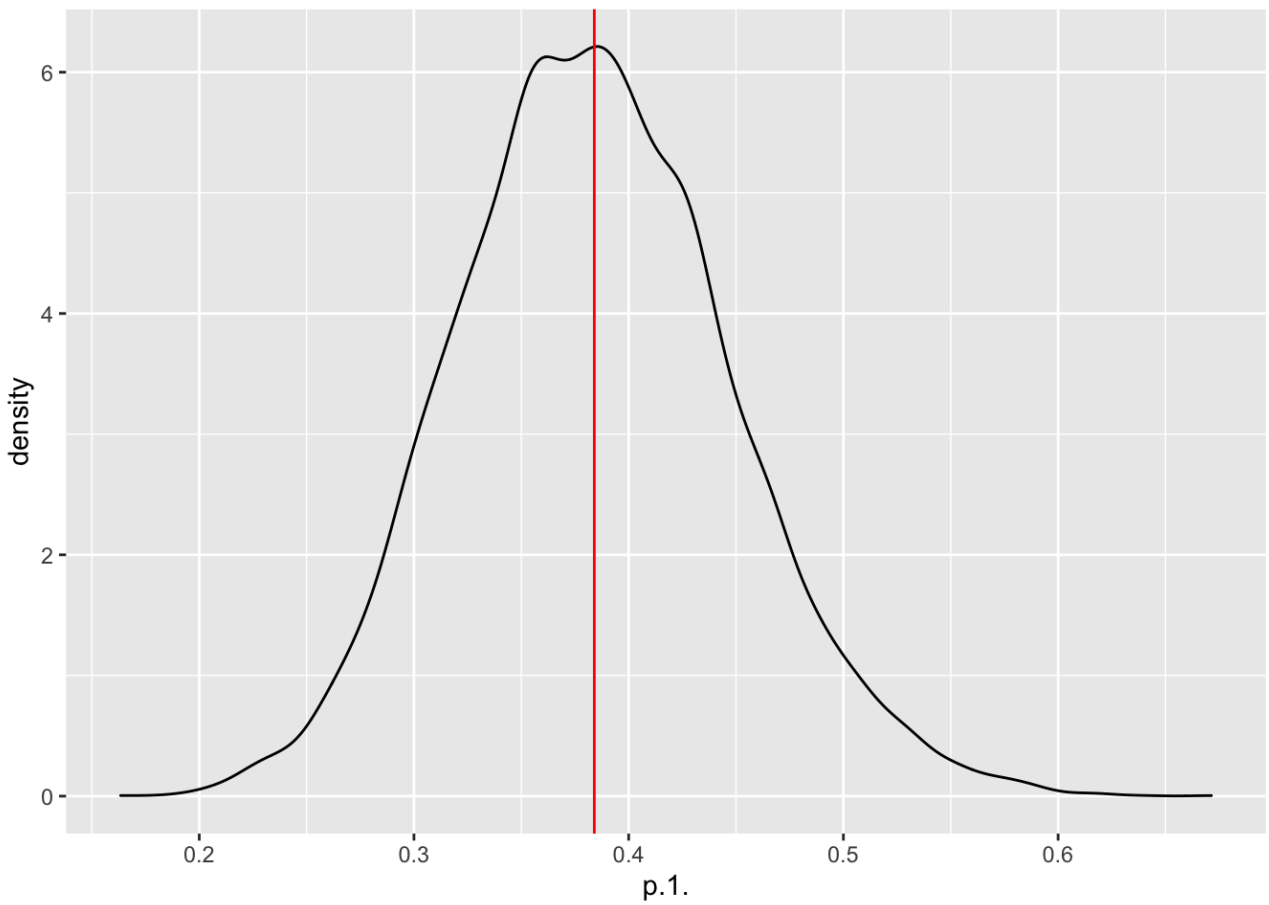
```
# Sp1 = Sp of the TST
ggplot(data=model4_chains, aes(x=Sp1)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$Sp1), color="red")
```



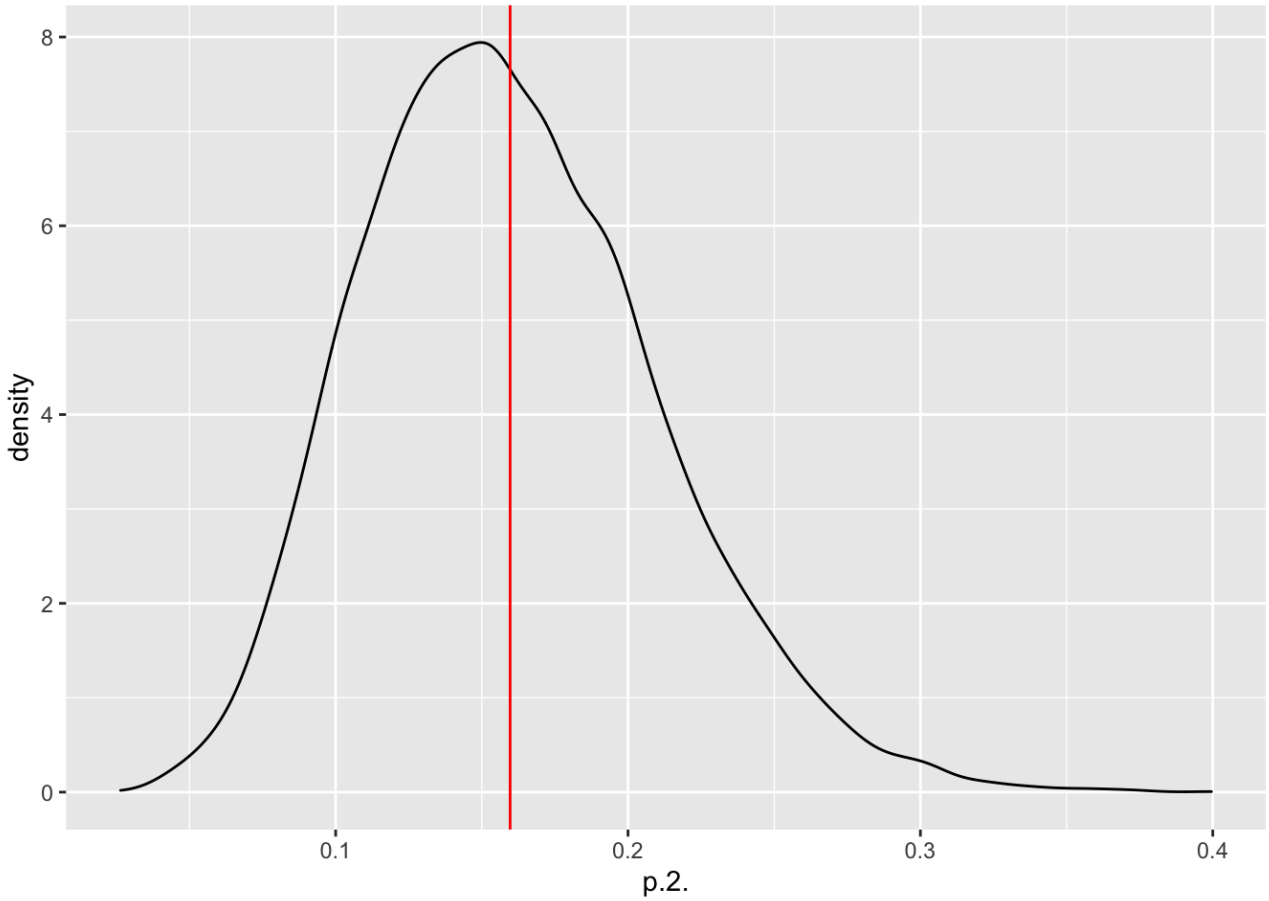

```
# Sp2 = SP of the TB ELISA
ggplot(data=model4_chains, aes(x=Sp2)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$Sp2), color="red")
```



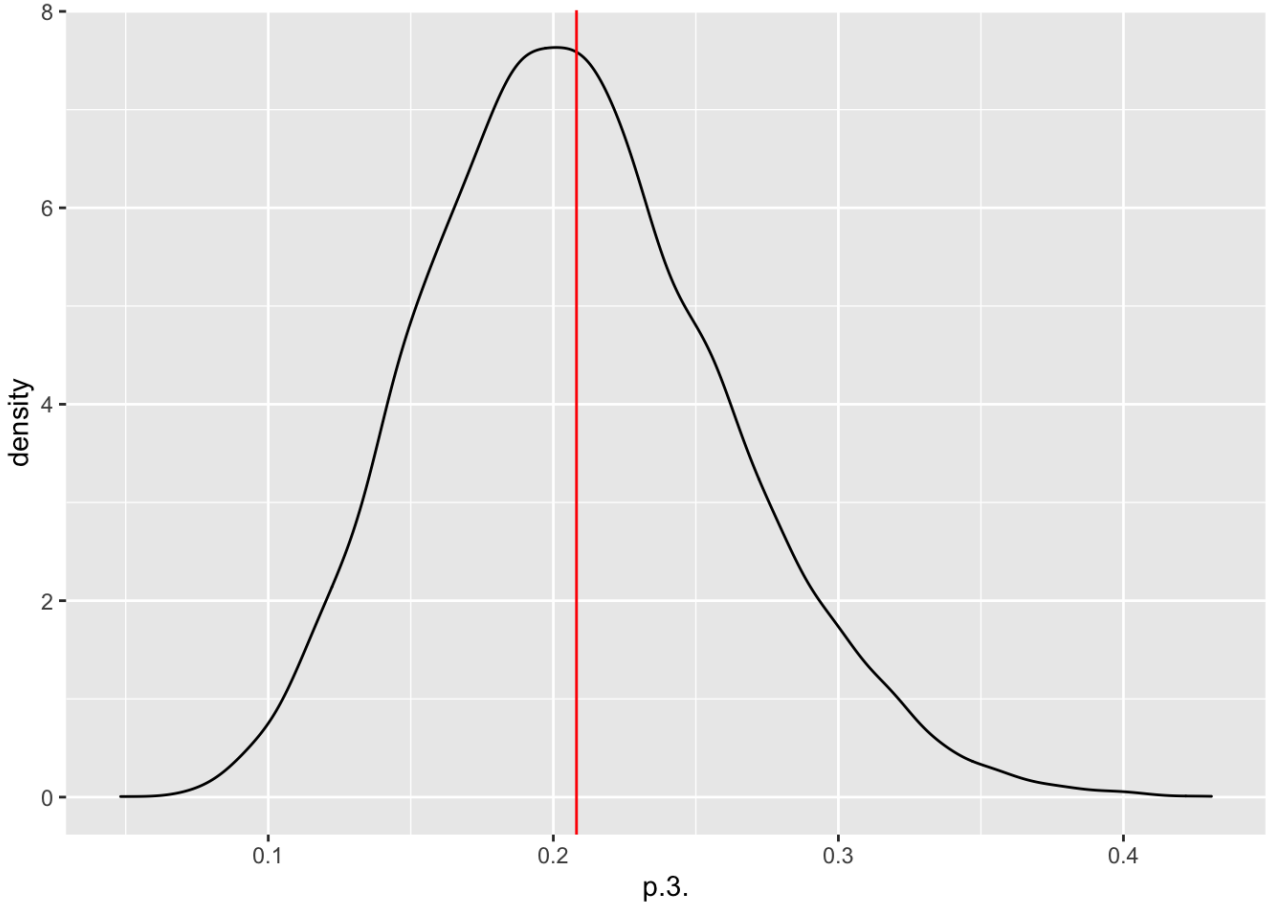

```
# p1 = prevalence in HiP.A
ggplot(data=model4_chains, aes(x=p.1.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.1.), color="red")
```



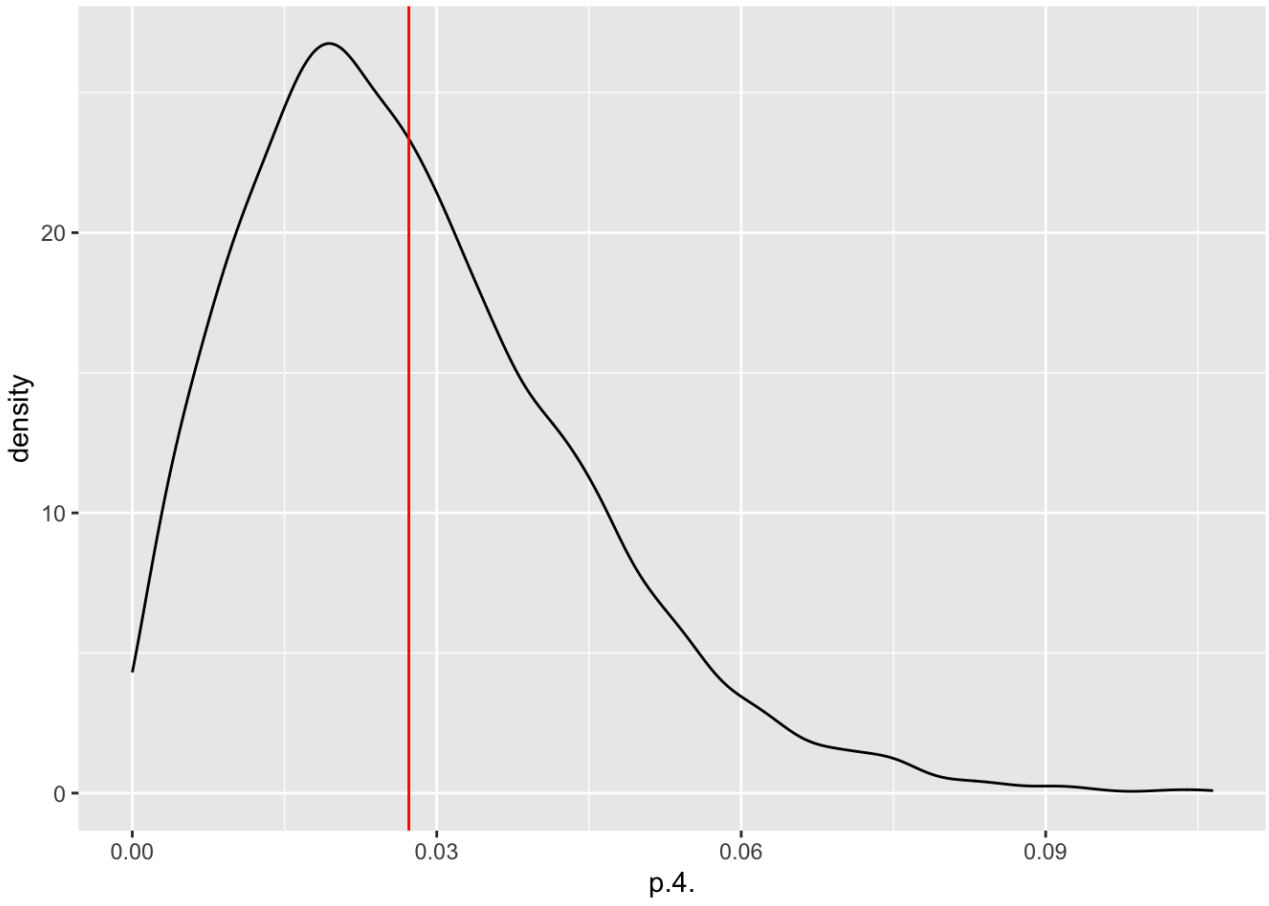
```
# p2 = prevalence in HiP.B
ggplot(data=model4_chains, aes(x=p.2.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.2.), color="red")
```



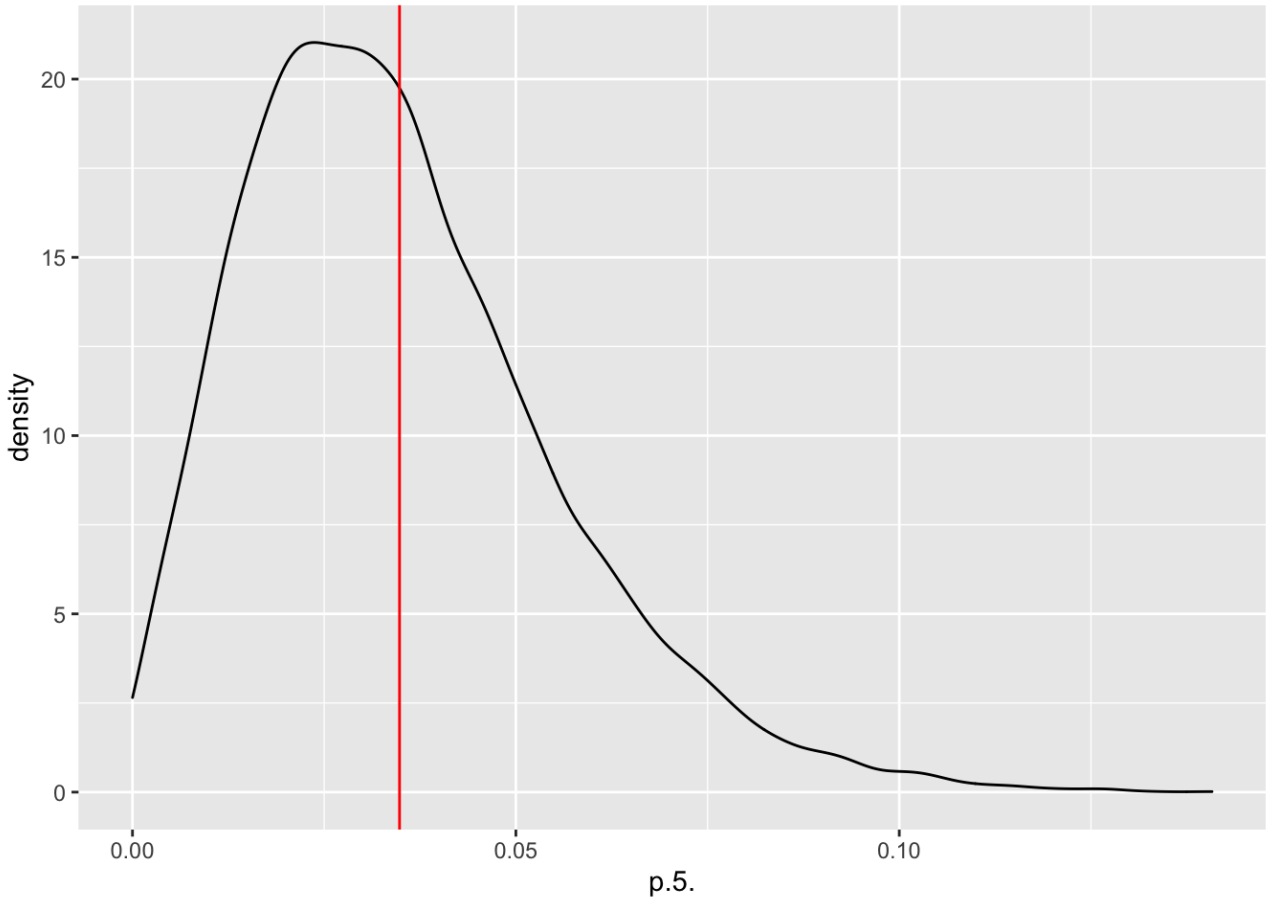

```
# p3 = prevalence in HiP.C
ggplot(data=model4_chains, aes(x=p.3.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.3.), color="red")
```



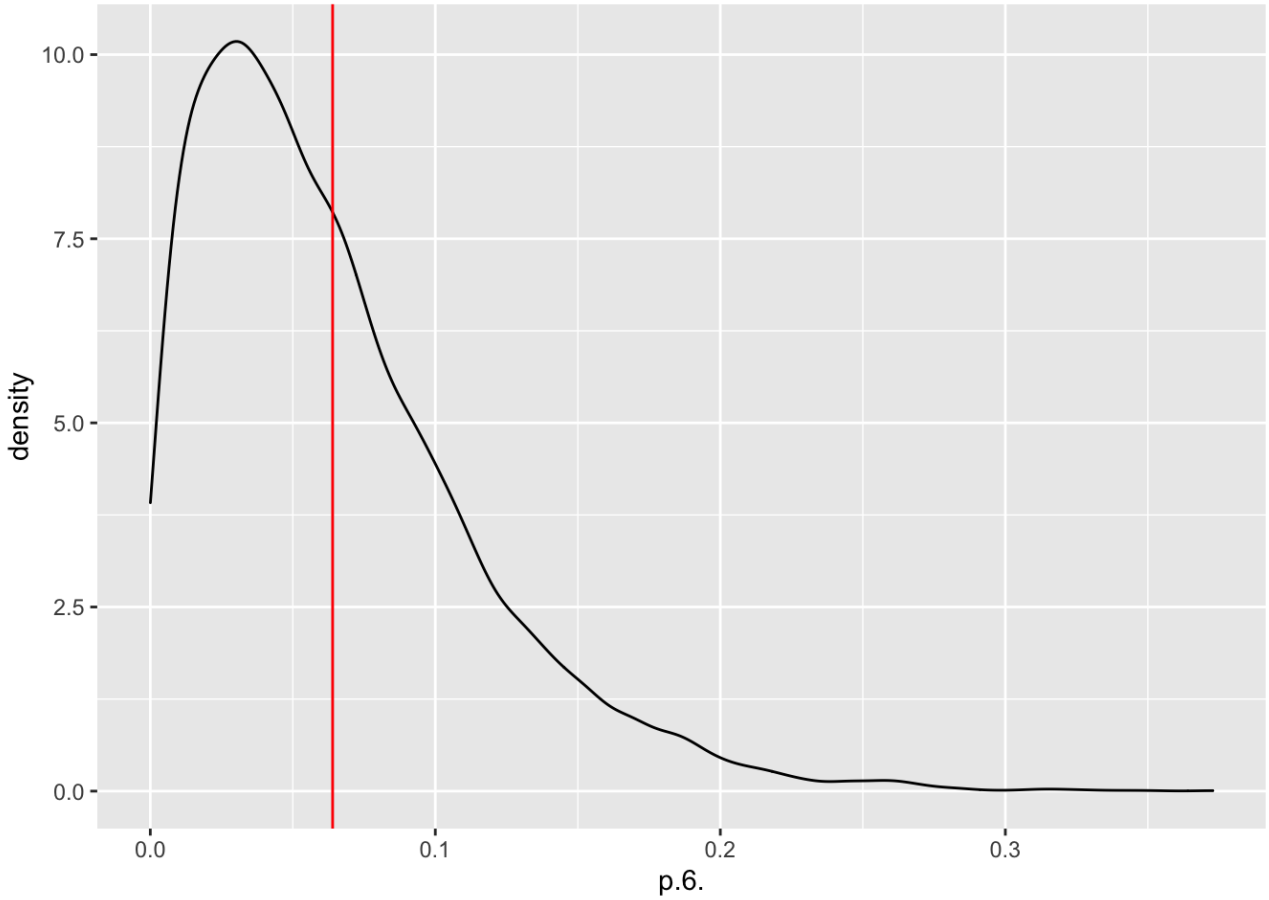

```
# p4 = prevalence in HiP.D
ggplot(data=model4_chains, aes(x=p.4.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.4.), color="red")
```



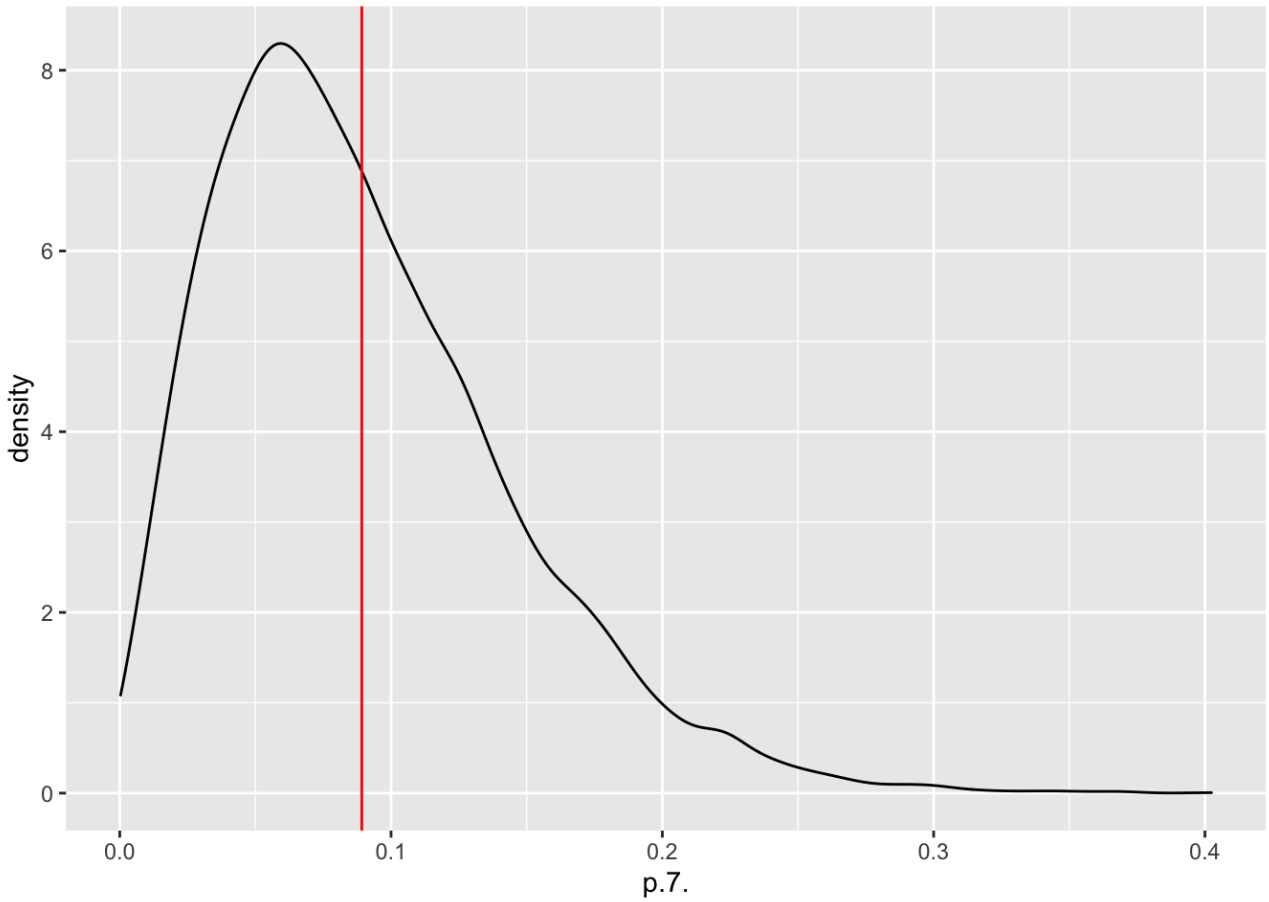
```
# p5 = prevalence in HiP.E
ggplot(data=model4_chains, aes(x=p.5.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.5.), color="red")
```



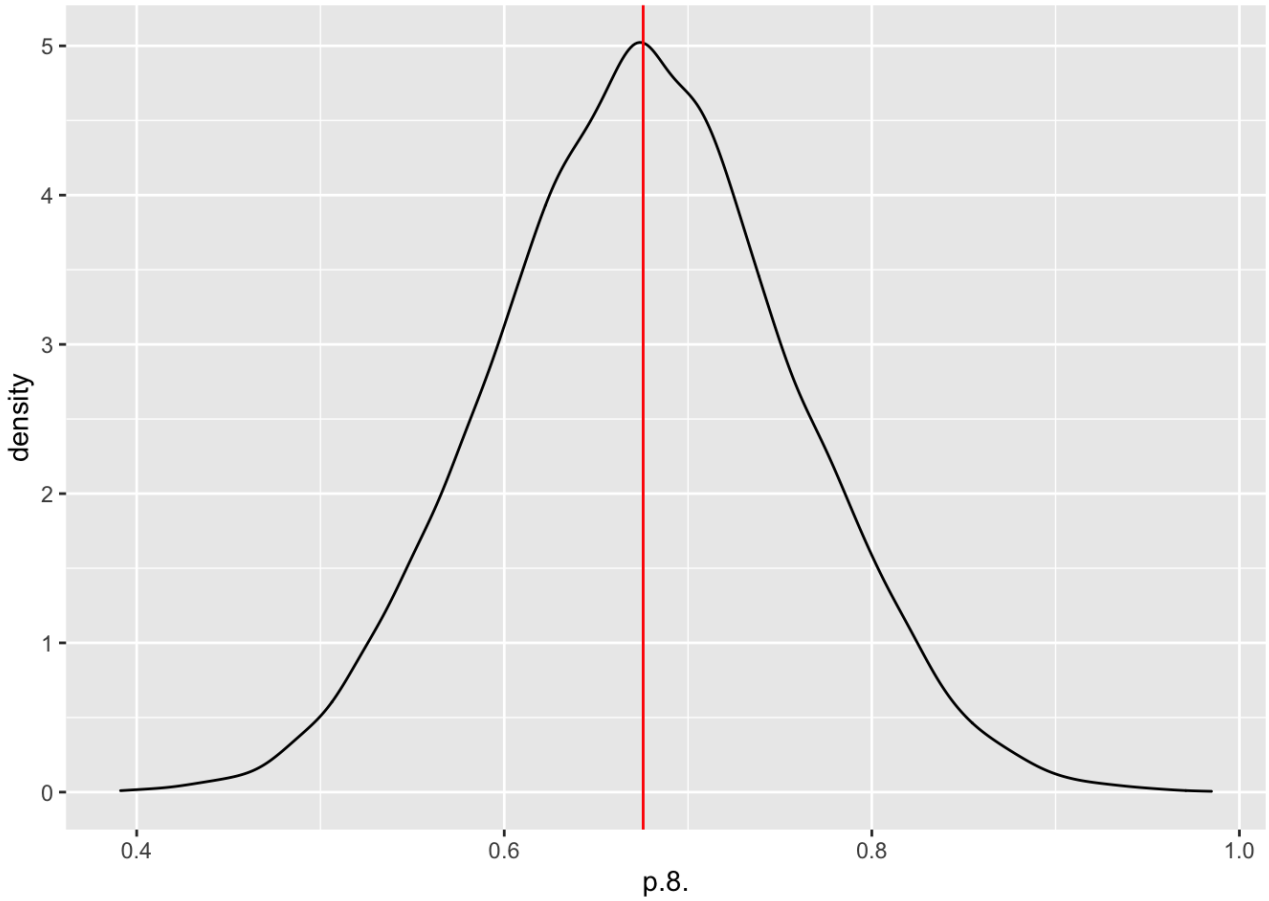

```
# p6 = prevalence in HiP.F
ggplot(data=model4_chains, aes(x=p.6.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.6.), color="red")
```



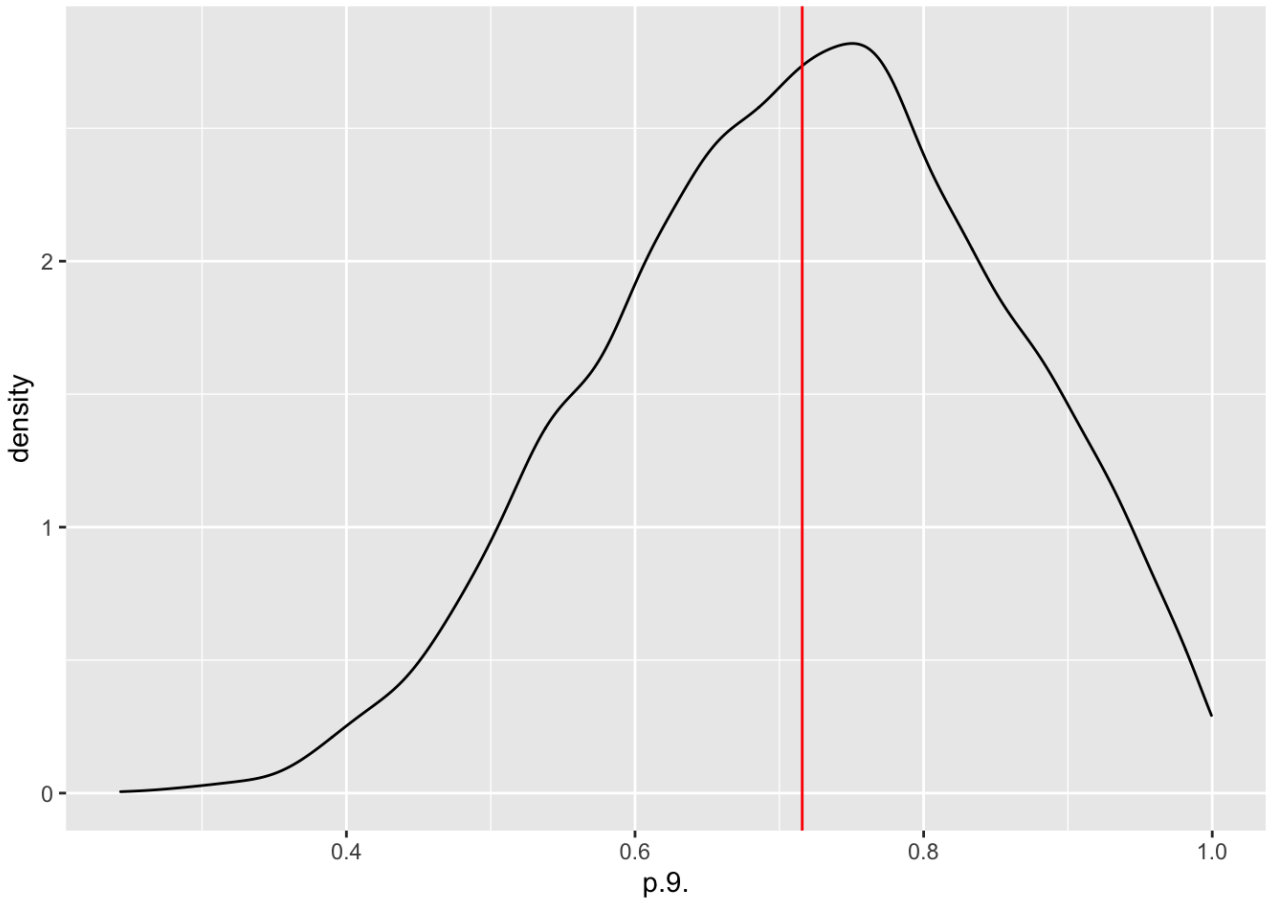

```
# p7 = prevalence in HiP.G
ggplot(data=model4_chains, aes(x=p.7.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.7.), color="red")
```



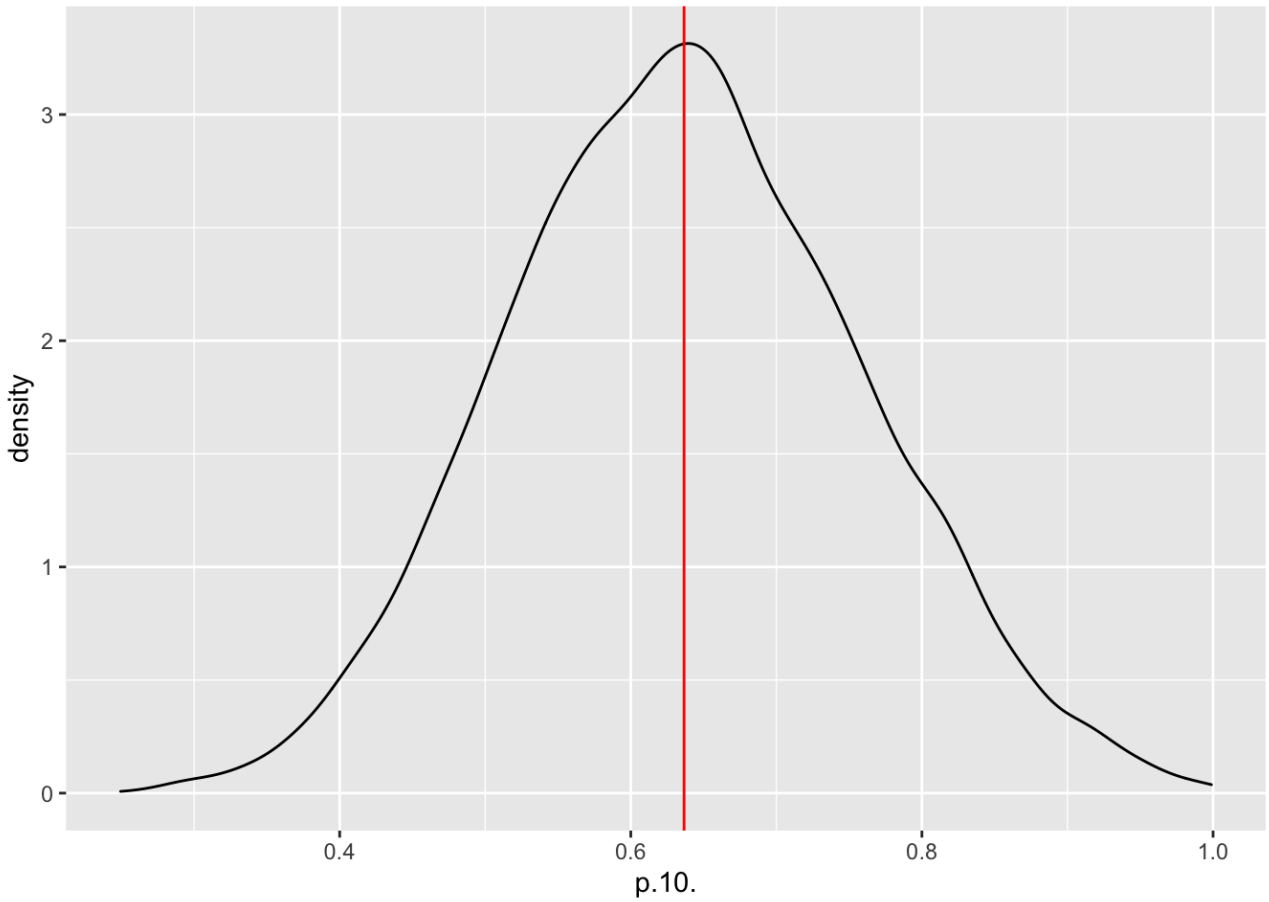
```
# p8 = prevalence in HiP.H
ggplot(data=model4_chains, aes(x=p.8.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.8.), color="red")
```



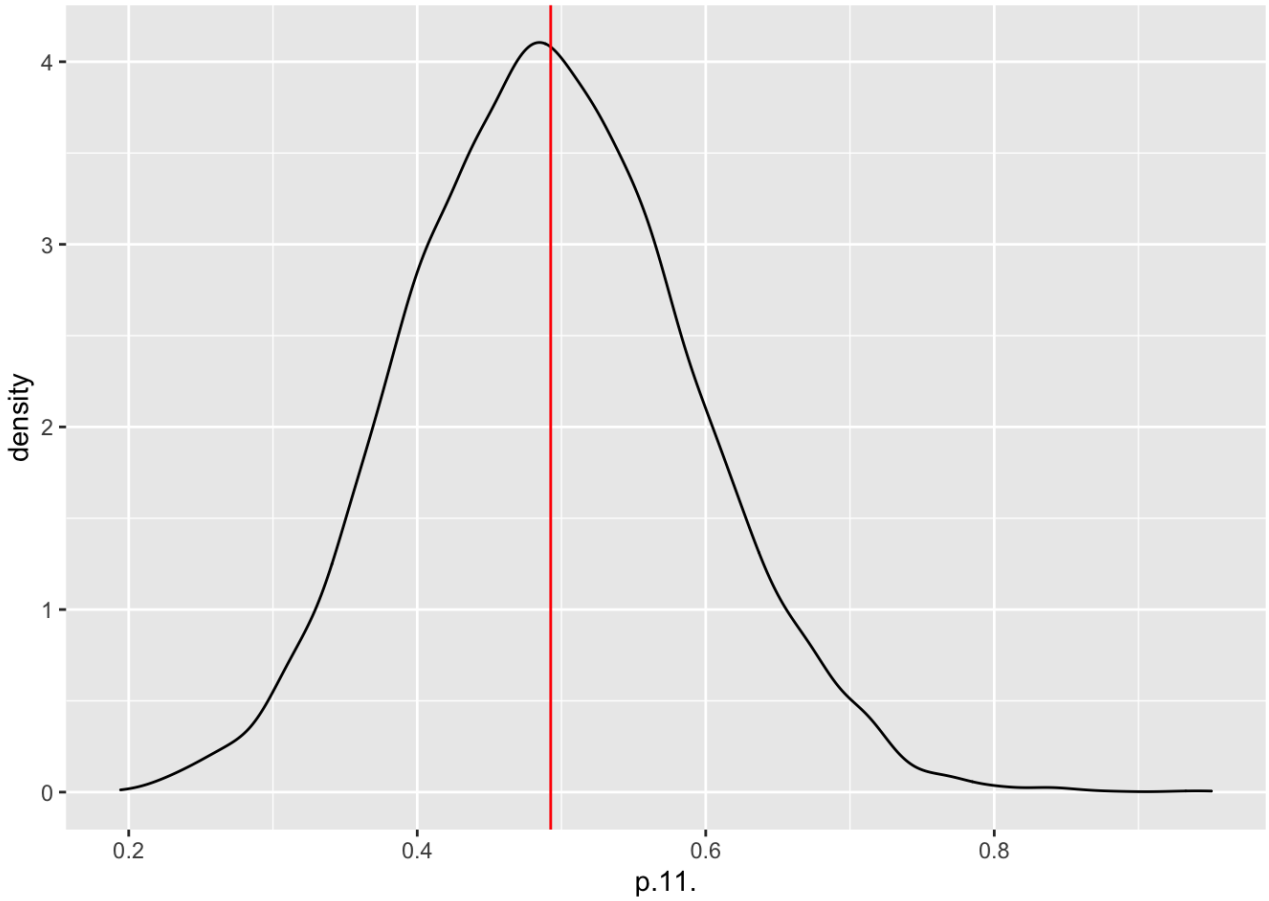

```
# p9 = prevalence in MGR.1
ggplot(data=model4_chains, aes(x=p.9.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.9.), color="red")
```



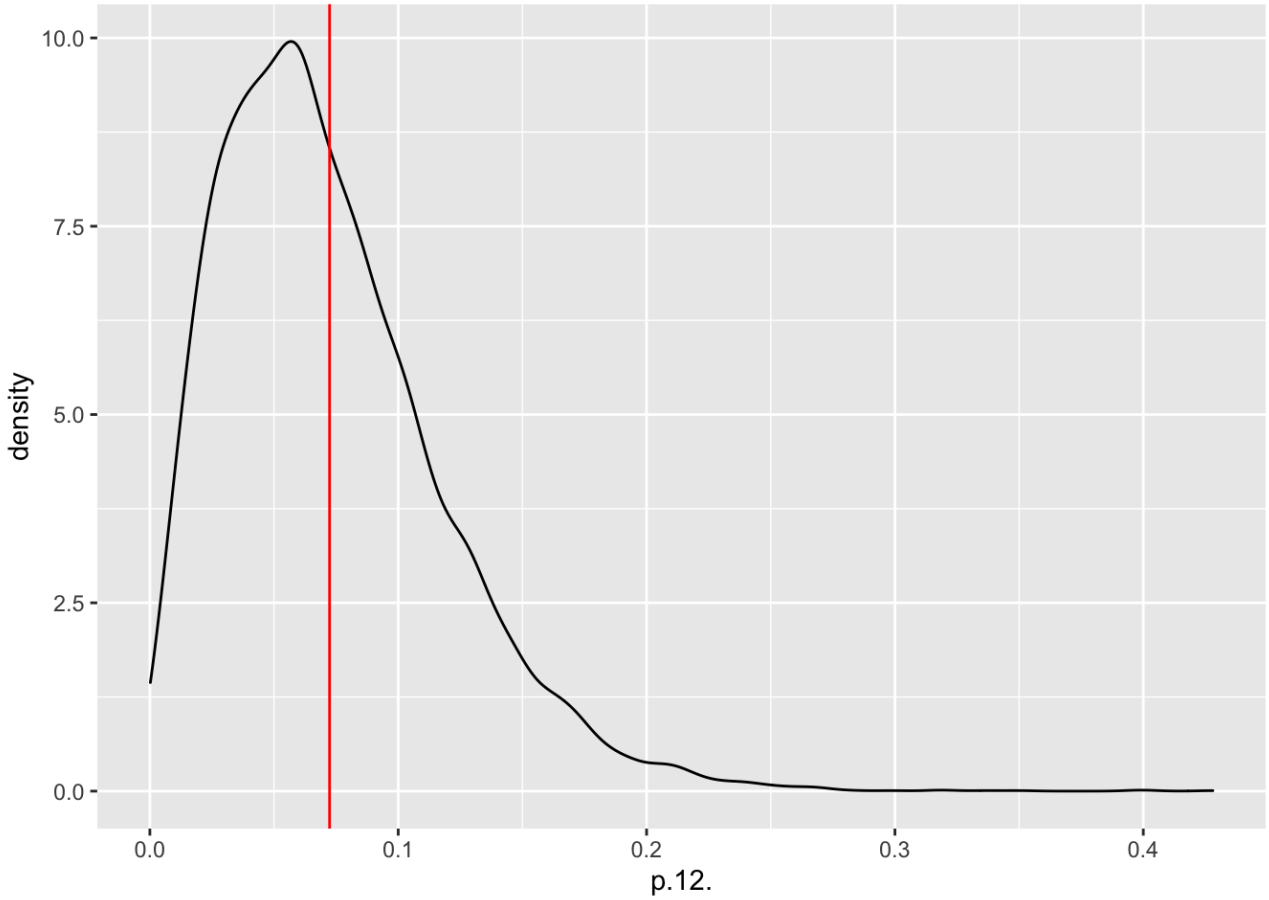

```
# p10 = prevalence in MGR.2
ggplot(data=model4_chains, aes(x=p.10.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.10.), color="red")
```



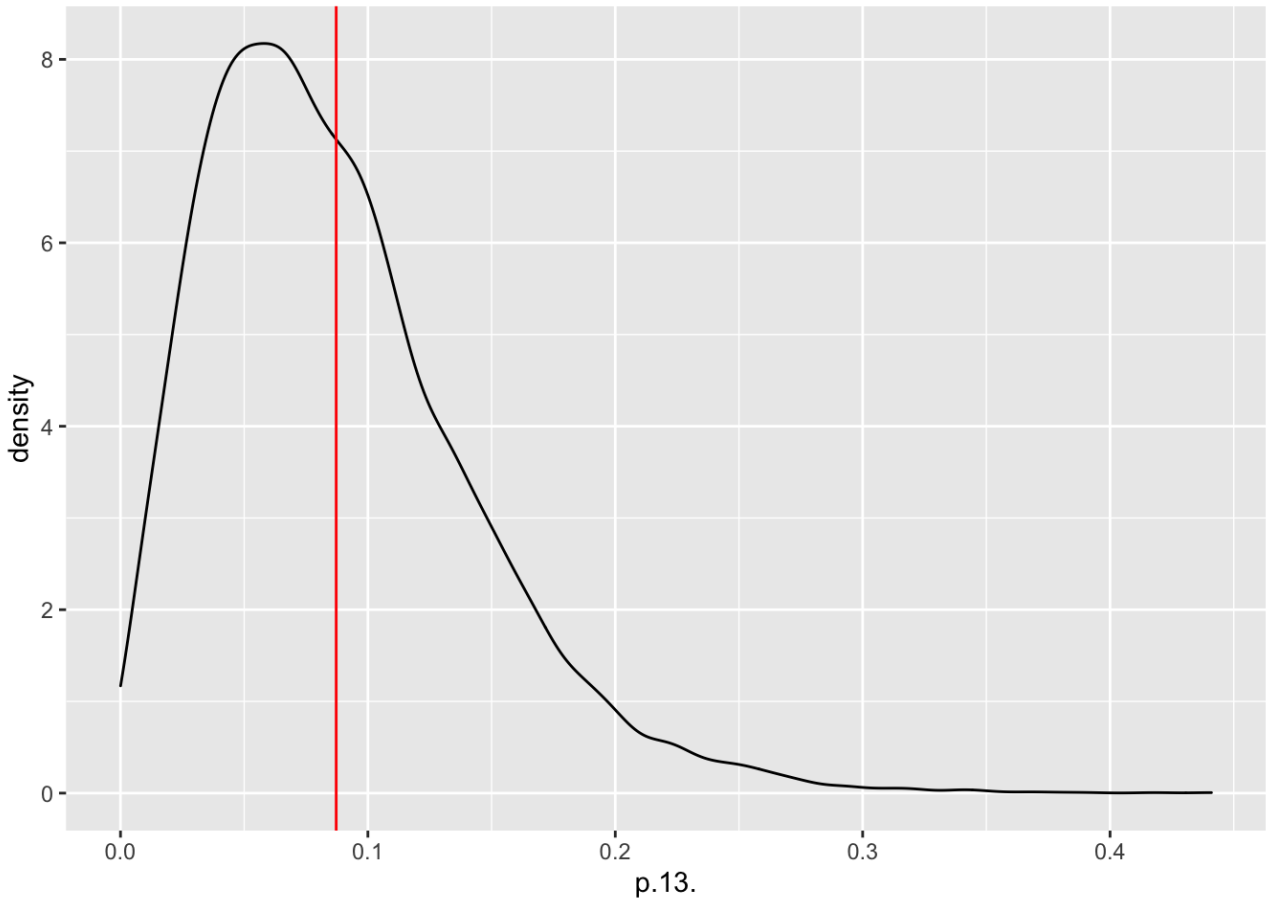
```
# p11 = prevalence in MGR.3
ggplot(data=model4_chains, aes(x=p.11.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.11.), color="red")
```



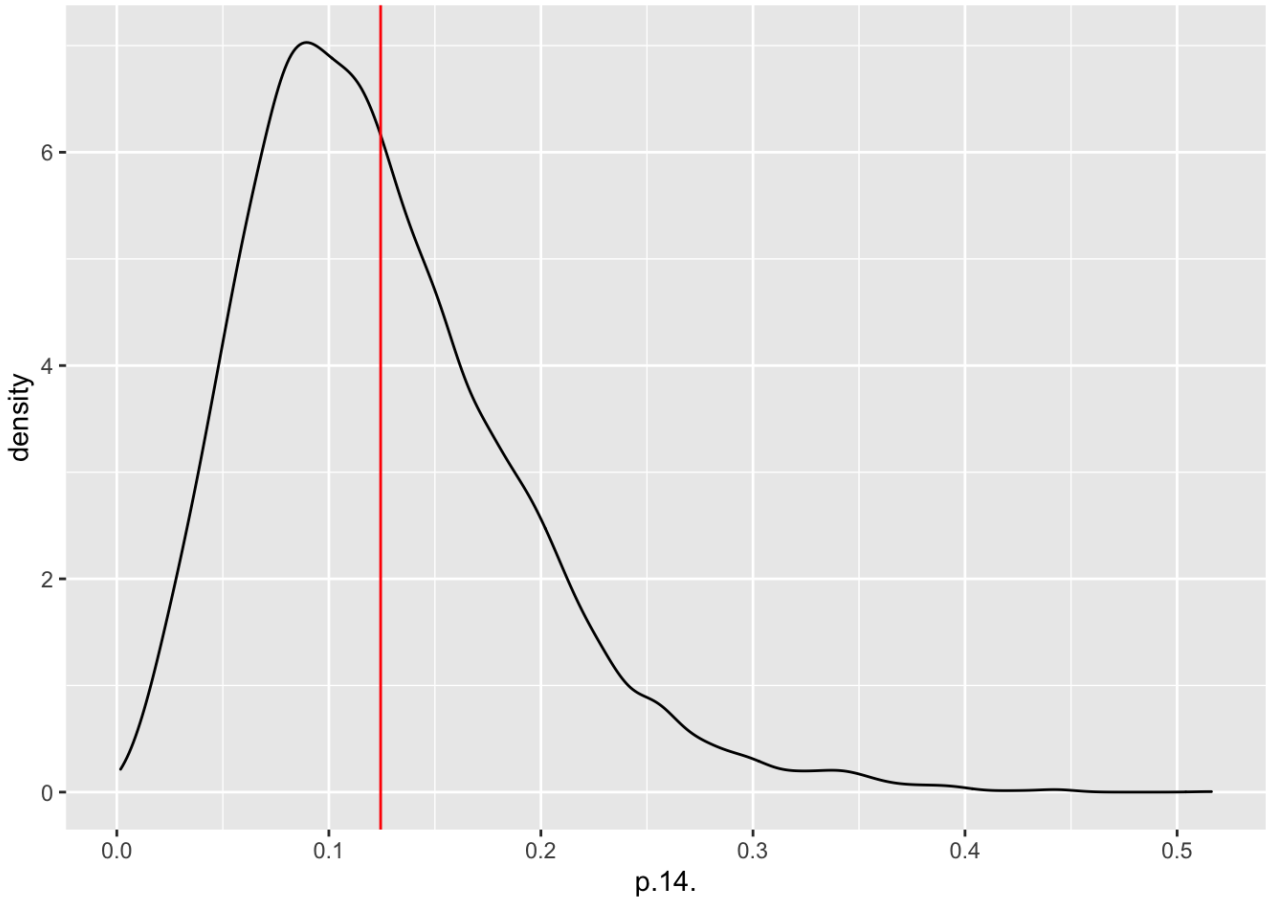

```
# p12 = prevalence in MGR.4
ggplot(data=model4_chains, aes(x=p.12.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.12.), color="red")
```




```
# p13 = prevalence in MGR.5
ggplot(data=model4_chains, aes(x=p.13.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.13.), color="red")
```



```
# p14 = prevalence in MGR.6
ggplot(data=model4_chains, aes(x=p.14.)) +
  geom_density() + geom_vline(xintercept=mean(model4_chains$p.14.), color="red")
```




```
# dd<-list(n = n, pop = pop)
# modelData <- dump.format(dd)
# modelInit1 <- dump.format(list(Se1=0.6, Se2=0.2, Sp1=0.99, Sp2=0.92, pi=runif(length(n
s), 0.1, 0.3)))
# modelInit2 <- dump.format(list(Se1=0.5, Se2=0.1, Sp1=0.97, Sp2=0.90, pi=runif(length(n
s), 0.1, 0.3)))
# modelInit3 <- dump.format(list(Se1=0.7, Se2=0.3, Sp1=0.94, Sp2=0.94, pi=runif(length(n
s), 0.1, 0.3)))
# modelInits <- c(modelInit1, modelInit2, modelInit3)
```

```
# We also want to know whether there is a difference in test performance between the park
s

# Might be good also to include a more severe interpretation (compare the different TST i
nterpretations described in the VPN)
# We can only include this for the MGR data and HiP2015, as we don't have all the skin te
st readings for HiP unfortunately.
# We could also include time point in this regression as we have longitudinal data for MG
R.

# We also want to know whether there is a difference in test performance between the park
s
# So we should do a multivariate model including location in the model as well as animal-
level factors (can we?)

# IDEA: Consider including the LIOdetect in this paper? This one is not conditionally ind
ependent from the IDEXX, so then we have to take into account the covDp and covDn like in
the Bronsvoort et al. 2019 paper.
```