AdvancedStatsAssignment

Table of contents

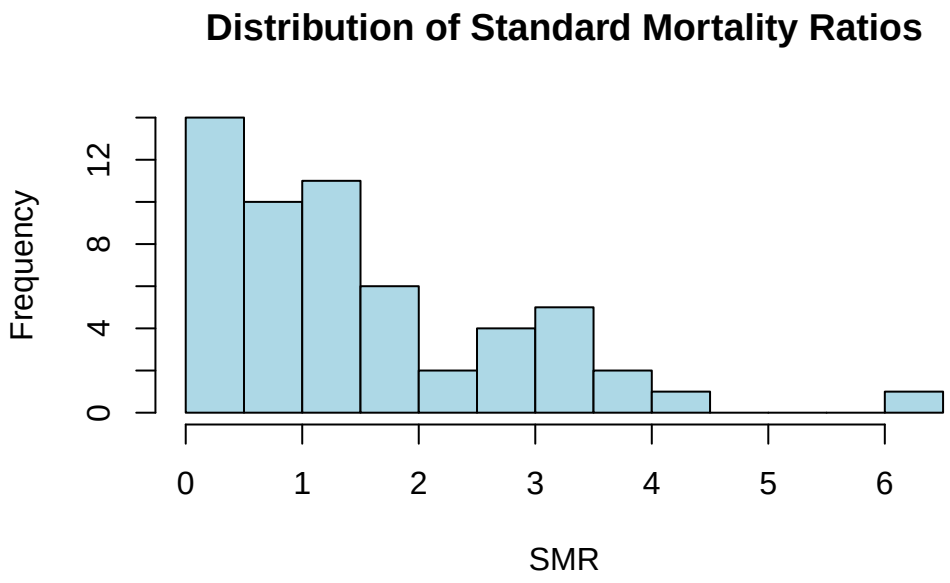
Question 1

```
source("ScotlandMap.R")  # need to read in the Scotland Map function
testdat = runif(56)  # generate random numbers to use as observations
ScotlandMap(testdat, figtitle = "Scotland random numbers")
```

Question 1 a)

```
ScotlandDF$SMR = ScotlandDF$Observed/ScotlandDF$Expected
```

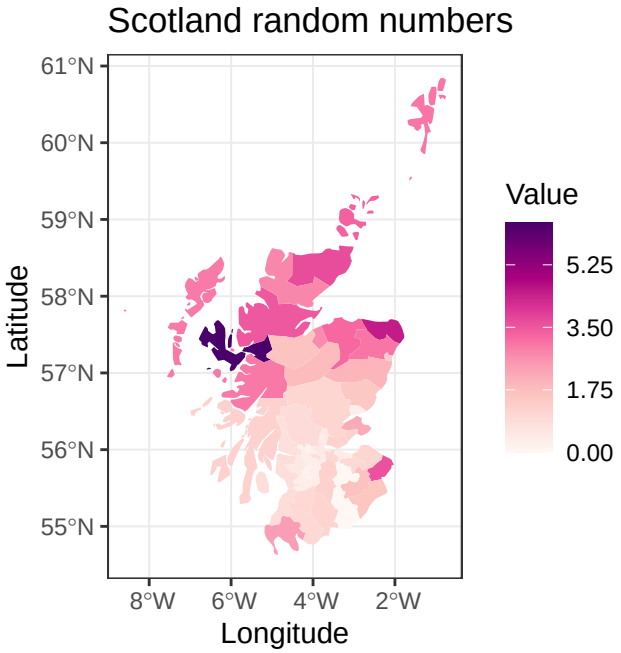
```
hist(ScotlandDF$SMR, breaks = 20, col = "lightblue", main = "Distribution of Standard Mort
    xlab = "SMR")
```




```
source("ScotlandMap.R")  # need to read in the ScotlandMap function
testdat = ScotlandDF$SMR  # generate random numbers to use as observations
ScotlandMap(testdat, figtitle = "Scotland random numbers")
```

```
Warning: OGR support is provided by the sf and terra packages among others

Warning: OGR support is provided by the sf and terra packages among others

Warning: OGR support is provided by the sf and terra packages among others

Warning: OGR support is provided by the sf and terra packages among others
```

```
Warning: OGR support is provided by the sf and terra packages among others

Warning: OGR support is provided by the sf and terra packages among others

OGR data source with driver: ESRI Shapefile
Source: "C:\AppliedDataScienceAndStatistics\Applied-Data-Science-and-Statistics\Term2\Advanc
with 56 features
It has 1 fields
```



Question 1 b)

##fix this latex TODO

$\beta 0$ is the intercept that represents the baseline for when all the other variables are 0 which can also be interpreted as the log number expected of cases per administrative area.

θi is the effect that captures the variation in the observed number of cases that cannot be explained by the expected numbers alone. θi also represents the deviation of the log observed number of cases from the log expected number of cases for area i.

In the Poisson model, θi represents the deviation of the observed number of cases in area i from the expected number of cases in area i, given the reference rates. In other words, it captures

how much higher or lower the observed number of cases is compared to what we would expect based on the reference rates

The role of θ_i in the estimation of the relative risk is to allow for variation in the risk of disease across different areas, over and above what is explained by the reference rates. By including a random effect for each area in the model, we are able to estimate the relative risk for each area, while accounting for the fact that some areas may have higher or lower risk due to factors that are not captured by the reference rates alone.

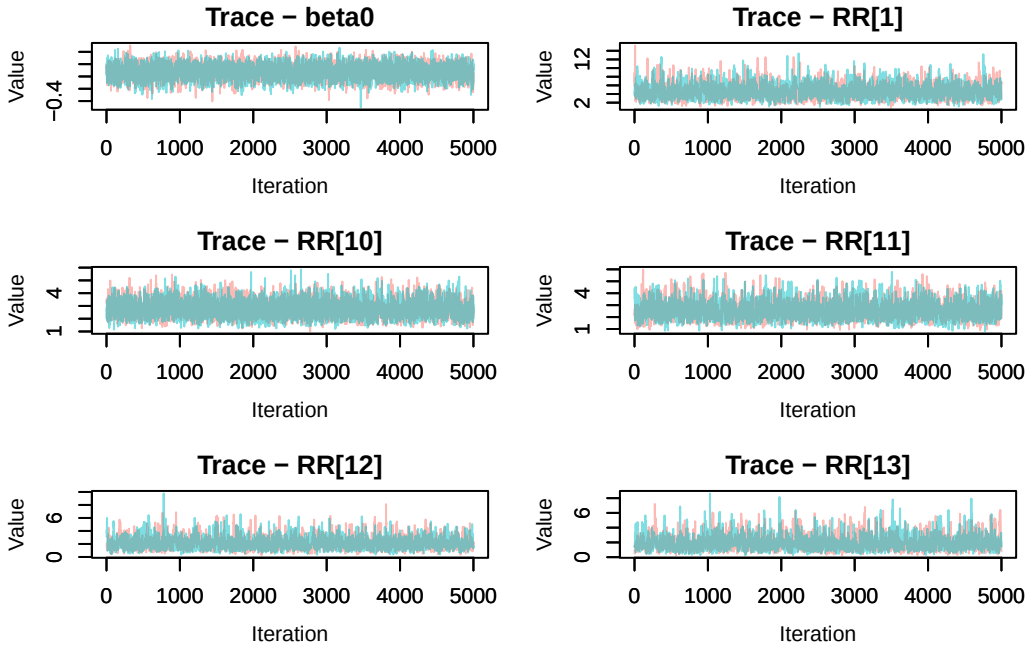
Question 1 c)

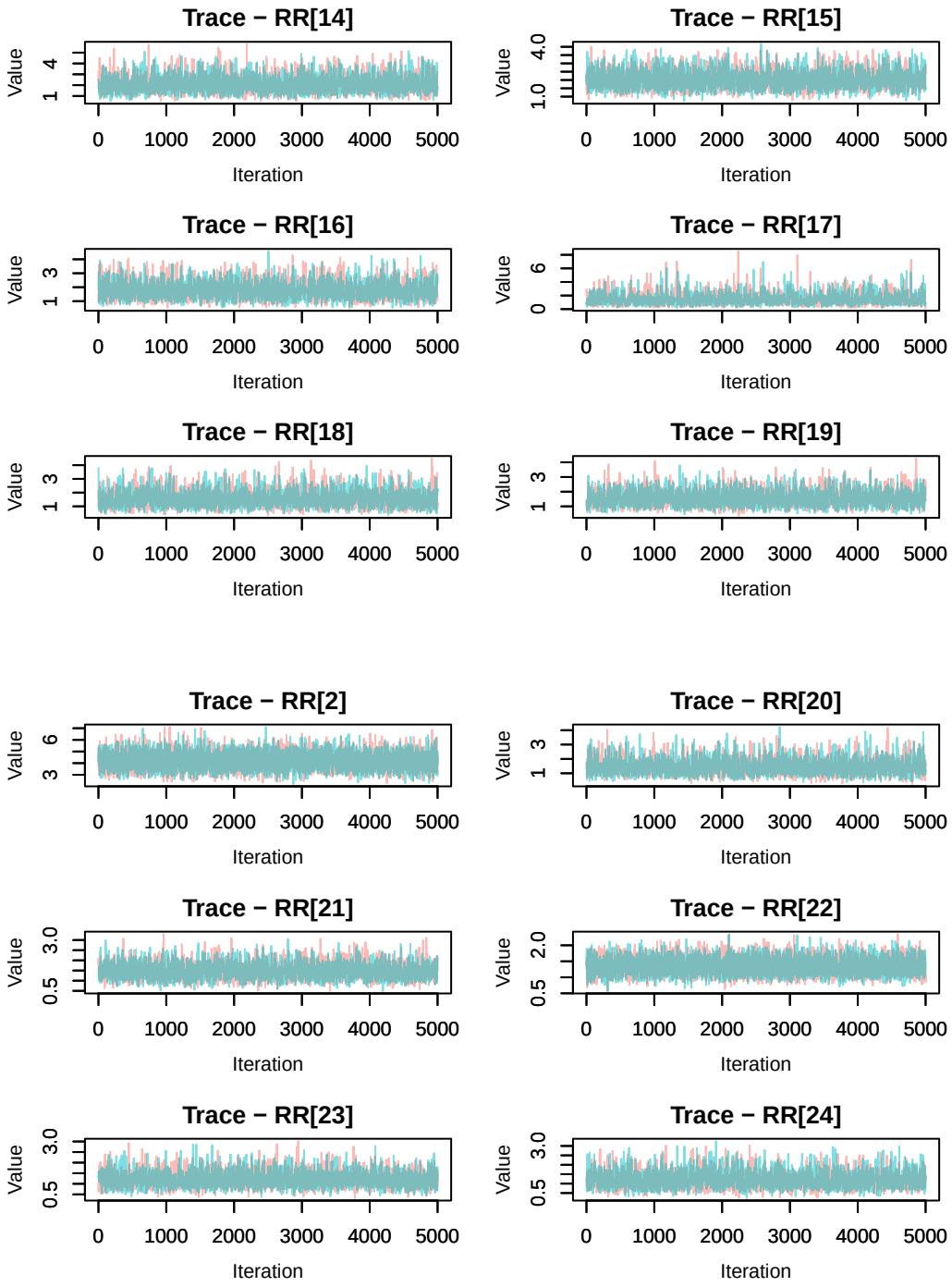
```
jags.mod.lipCancer = function(){
  # Priors
  beta0 ~ dnorm(0, 1/1000)
  tau ~ dgamma(1, 0.05)

  # Likelihood
  for (i in 1:N) {
    obs[i] ~ dpois(mu[i])
    log(mu[i]) = log(expo[i]) + beta0 + theta[i]
    theta[i] ~ dnorm(0, tau)
    RR[i] = exp(beta0) * exp(theta[i])
  }
}

# Parameters to monitor
jags.param.lipCancer = c("beta0", "theta", "tau", "RR")

# Data
jags.data.lipCancer <- list(N = nrow(ScotlandDF),
     obs = ScotlandDF$Observed,
     expo = ScotlandDF$Expected)

## initial values
inits1 <- list('tau' = 100, 'sigma' = 20, 'theta'=c(10,-5,-25))
inits2 <- list('tau'=100000,'sigma'=-100,'theta'=c(-100,100,500))

jags.inits.lipCancer = list(inits1,inits2)

## fit the jags model
```

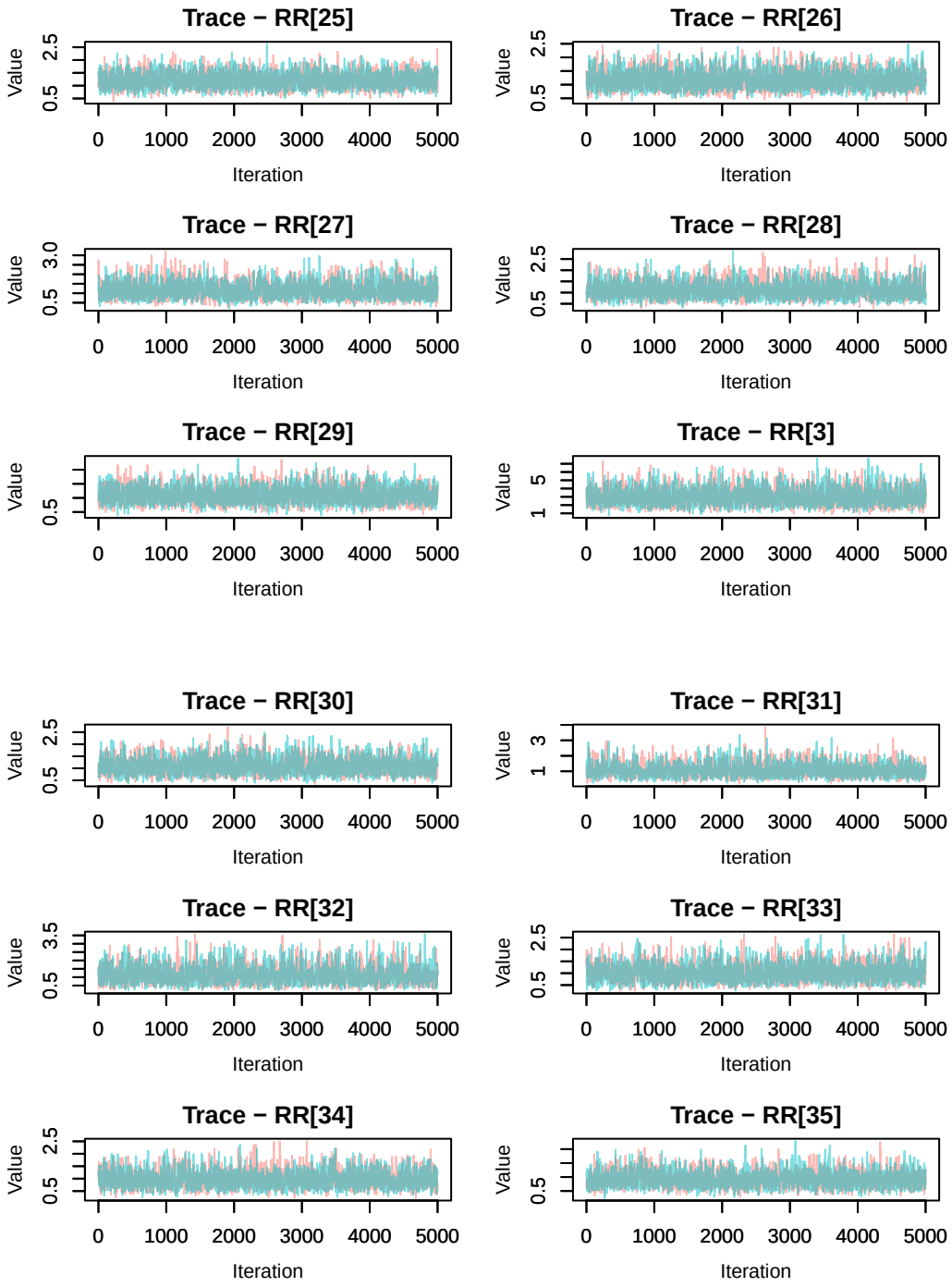
```
jags.mod.fit.lipCancer <- jags(data = jags.data.lipCancer, #inits = jags.inits.lipCancer,
parameters.to.save = jags.param.lipCancer, n.chains = 2, n.iter = 10000,
n.burnin = 5000,n.thin=1,model.file = jags.mod.lipCancer, DIC=FALSE)
```

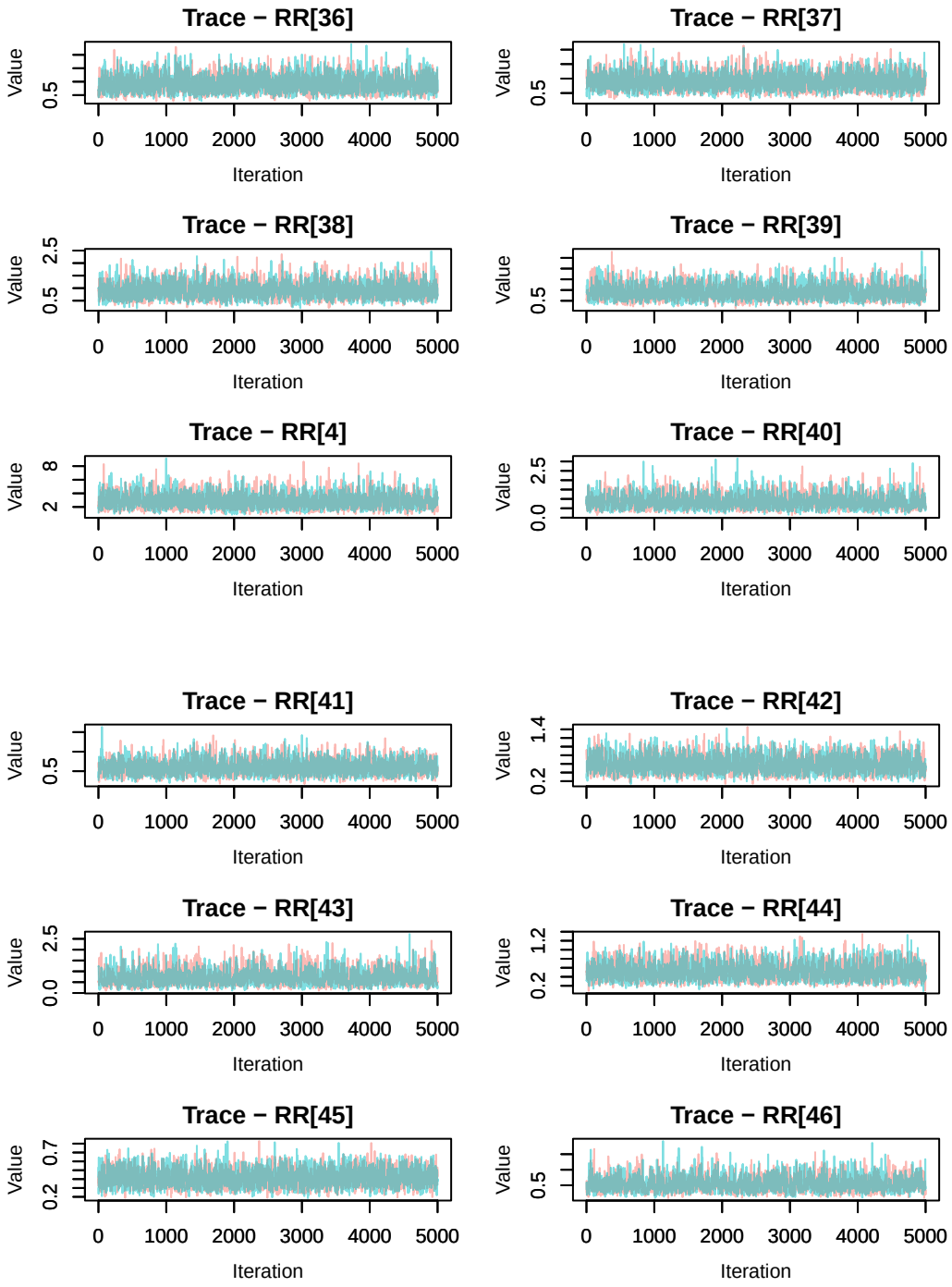
```
module glm loaded

module dic loaded

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 56
   Unobserved stochastic nodes: 58
   Total graph size: 450

Initializing model
```

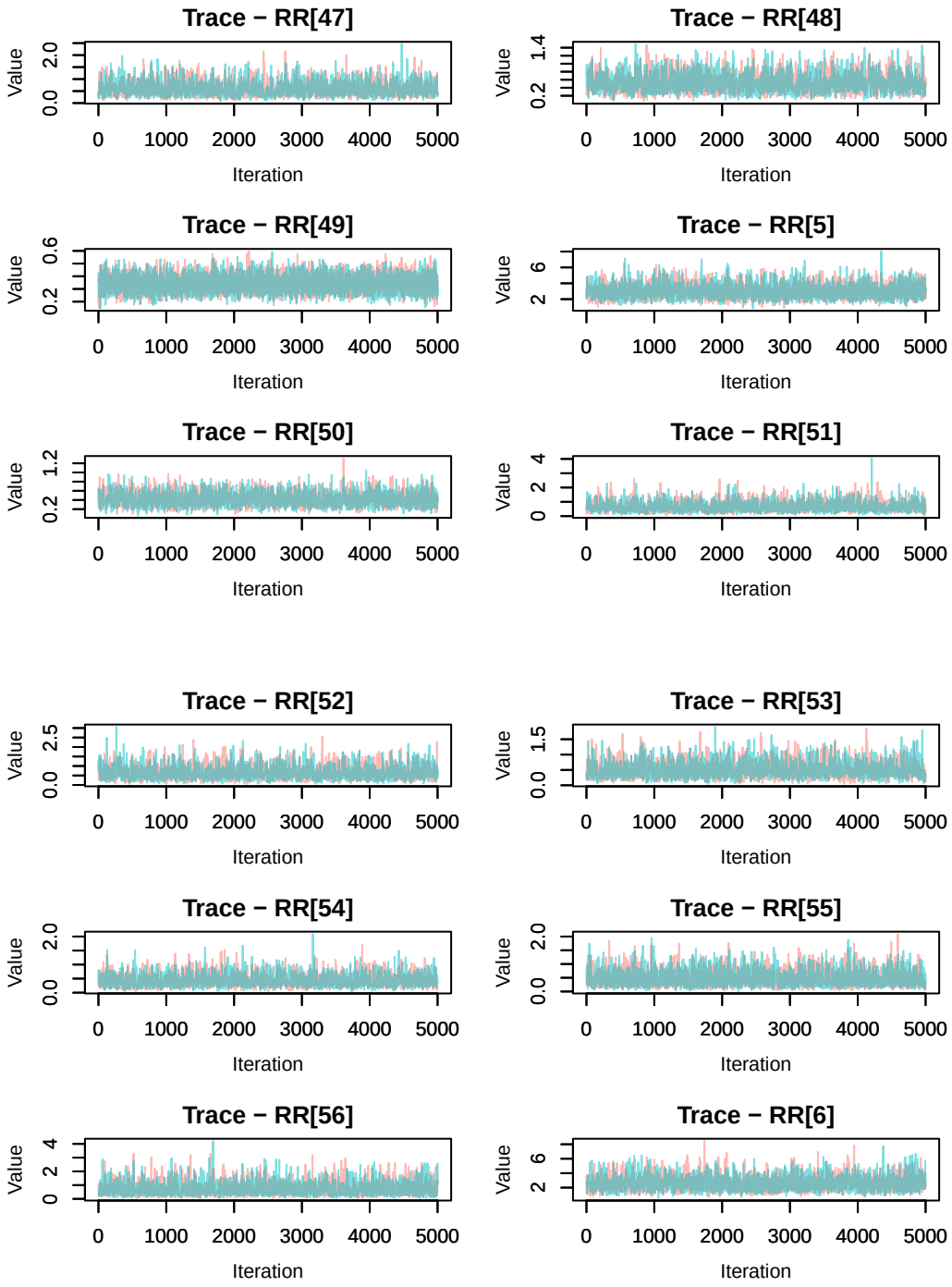
```
jagsfitlipCancer.mcmc <- as.mcmc(jags.mod.fit.lipCancer)
MCMCtrace(jagsfitlipCancer.mcmc, type = "trace", ind = TRUE, pdf = FALSE)
```

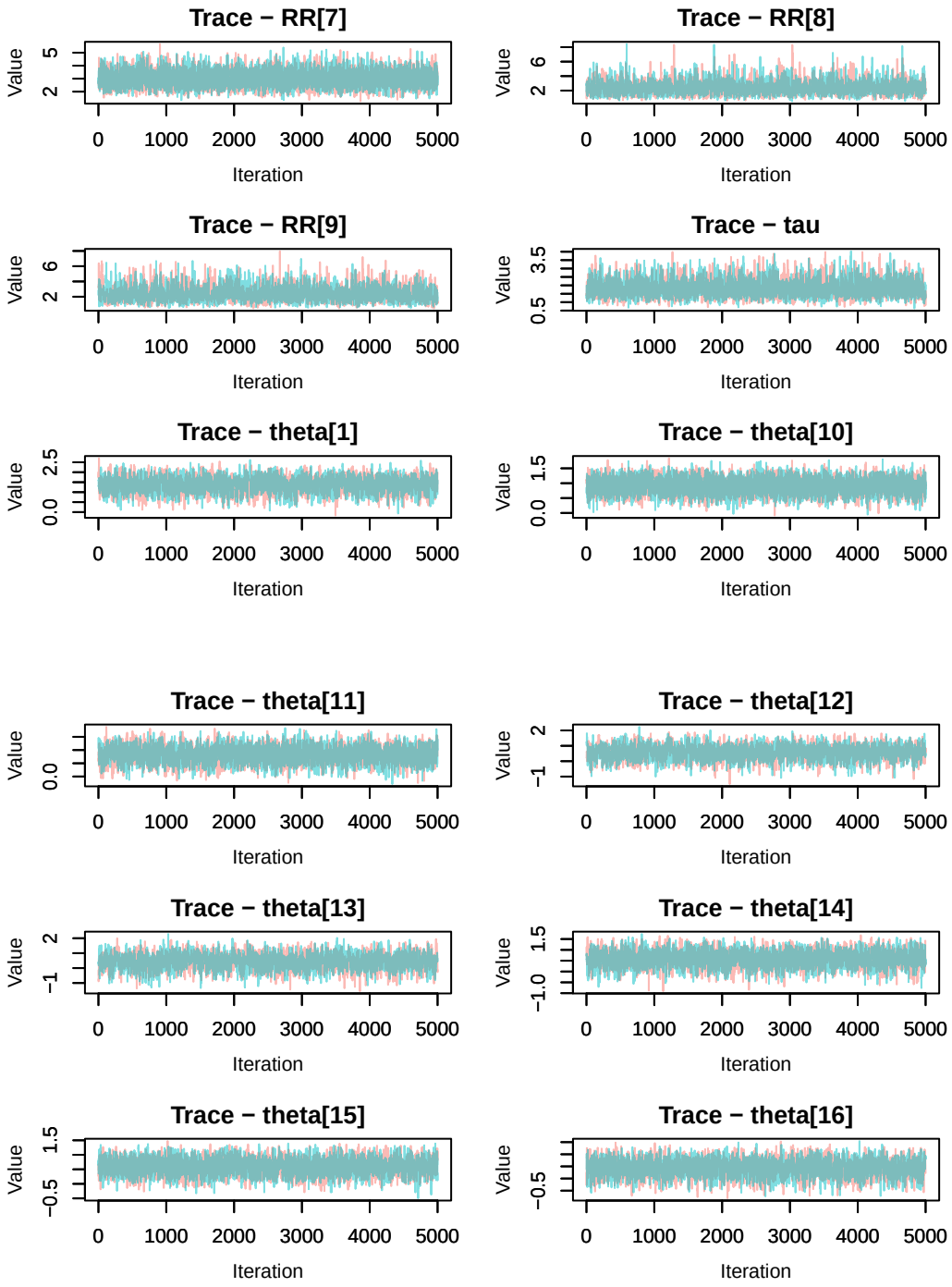


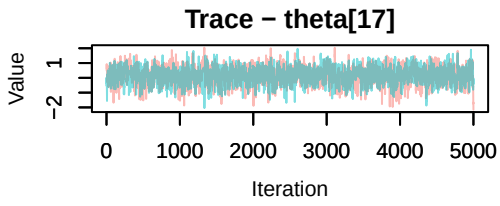

Trace − RR[14]
Value
Iteration
Trace − RR[15]
Value
Iteration
Trace − RR[16]
Value
Iteration
Trace − RR[17]
Value
Iteration
Trace − RR[18]
Value
Iteration
Trace − RR[19]
Value
Iteration
Trace − RR[2]
Value
Iteration
Trace − RR[20]
Value
Iteration
Trace − RR[21]
Value
Iteration
Trace − RR[22]
Value
Iteration
Trace − RR[23]
Value
Iteration
Trace − RR[24]
Value
Iteration

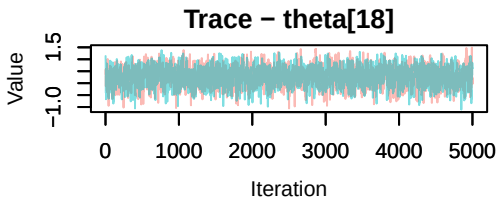

Trace − RR[25]
Trace − RR[26]
Trace − RR[27]
Trace − RR[28]
Trace − RR[29]
Trace − RR[3]
Trace − RR[30]
Trace − RR[31]
Trace − RR[32]
Trace − RR[33]
Trace − RR[34]
Trace − RR[35]
Value
Iteration
0
1000
2000
3000
4000
5000

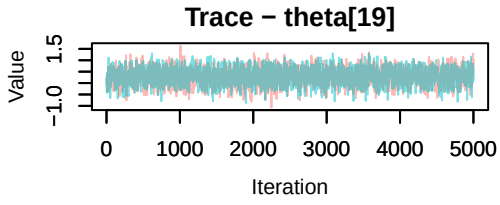

Trace − RR[36]
Trace − RR[37]
Trace − RR[38]
Trace − RR[39]
Trace − RR[4]
Trace − RR[40]
Trace − RR[41]
Trace − RR[42]
Trace − RR[43]
Trace − RR[44]
Trace − RR[45]
Trace − RR[46]
Value
Iteration
0
1000
2000
3000
4000
5000

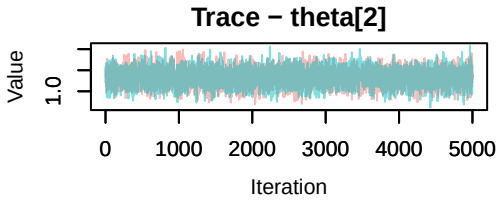

Trace − RR[47]
Value
Iteration
Trace − RR[48]
Trace − RR[49]
Trace − RR[5]
Trace − RR[50]
Trace − RR[51]
Trace − RR[52]
Trace − RR[53]
Trace − RR[54]
Trace − RR[55]
Trace − RR[56]
Trace − RR[6]
0
1000
2000
3000
4000
5000

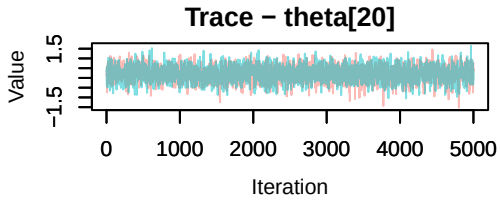

Trace − RR[7]
Trace − RR[8]
Trace − RR[9]
Trace − tau
Trace − theta[1]
Trace − theta[10]
Trace − theta[11]
Trace − theta[12]
Trace − theta[13]
Trace − theta[14]
Trace − theta[15]
Trace − theta[16]
Value
Iteration
0
1000
2000
3000
4000
5000

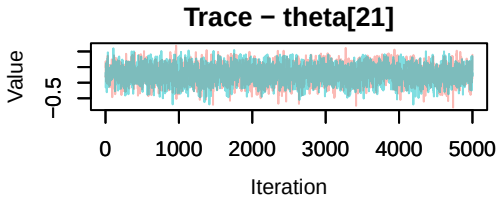

Trace − theta[17]
Value
Iteration

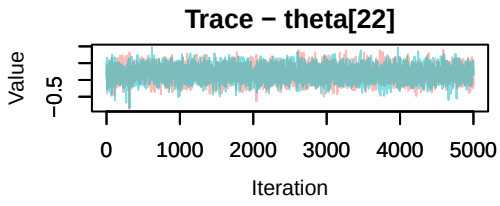

Trace − theta[18]
Value
Iteration

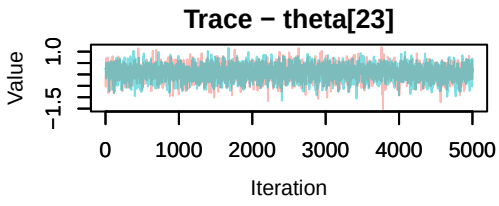

Trace − theta[19]
Value
Iteration

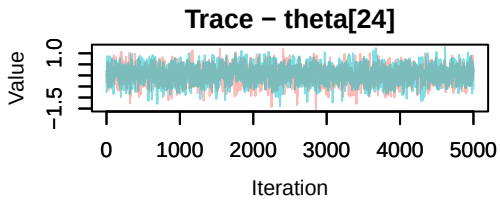

Trace − theta[2]
Value
Iteration

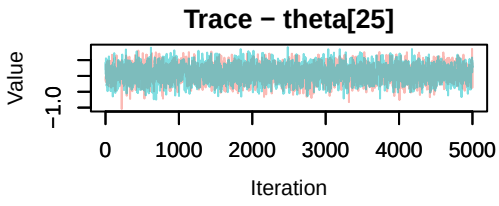

Trace − theta[20]
Value
Iteration

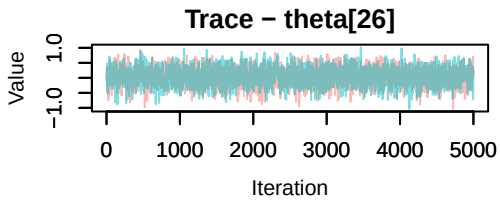

Trace − theta[21]
Value
Iteration

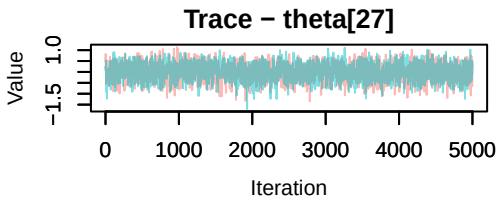

Trace − theta[22]
Value
Iteration

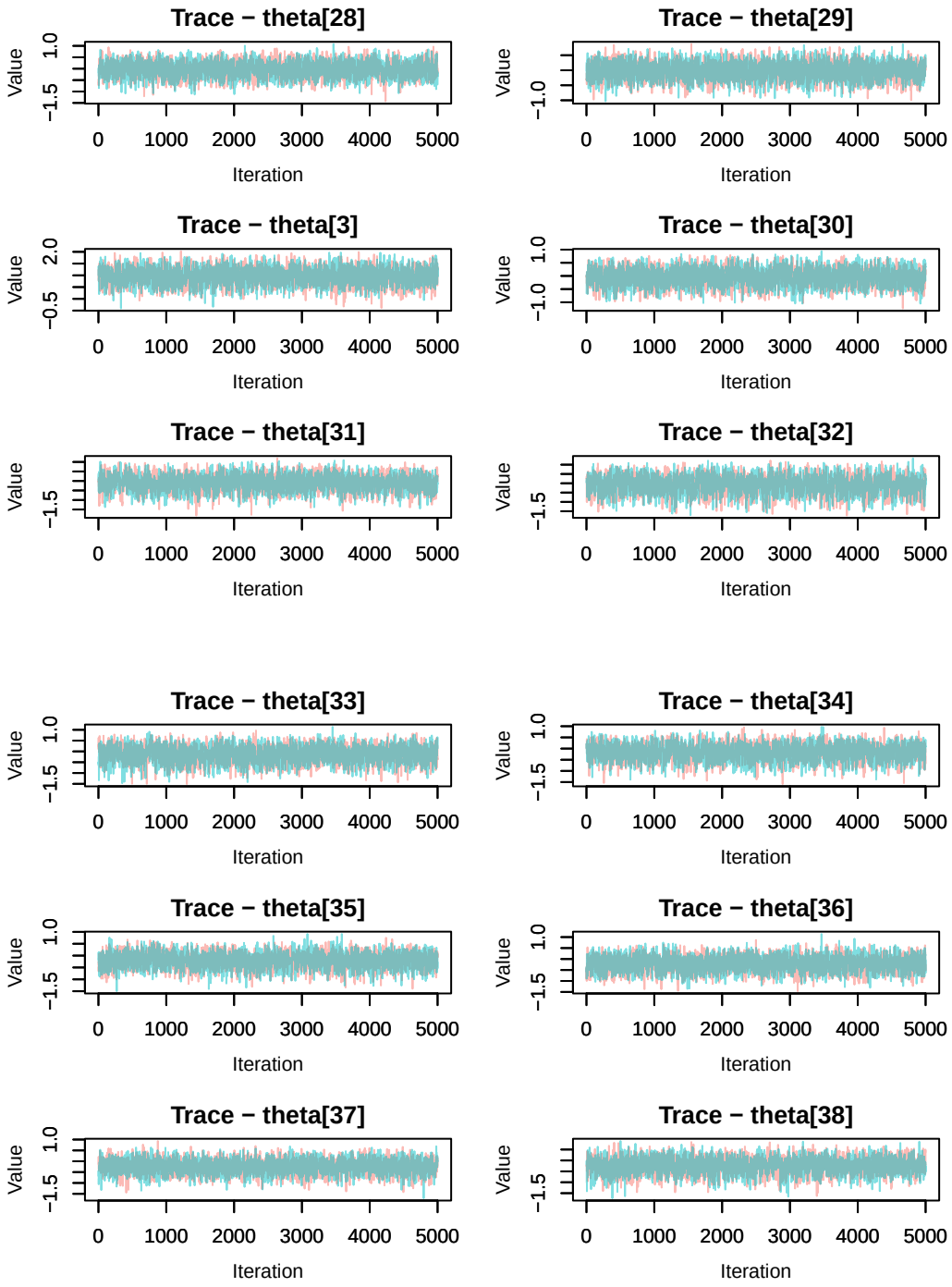

Trace − theta[23]
Value
Iteration

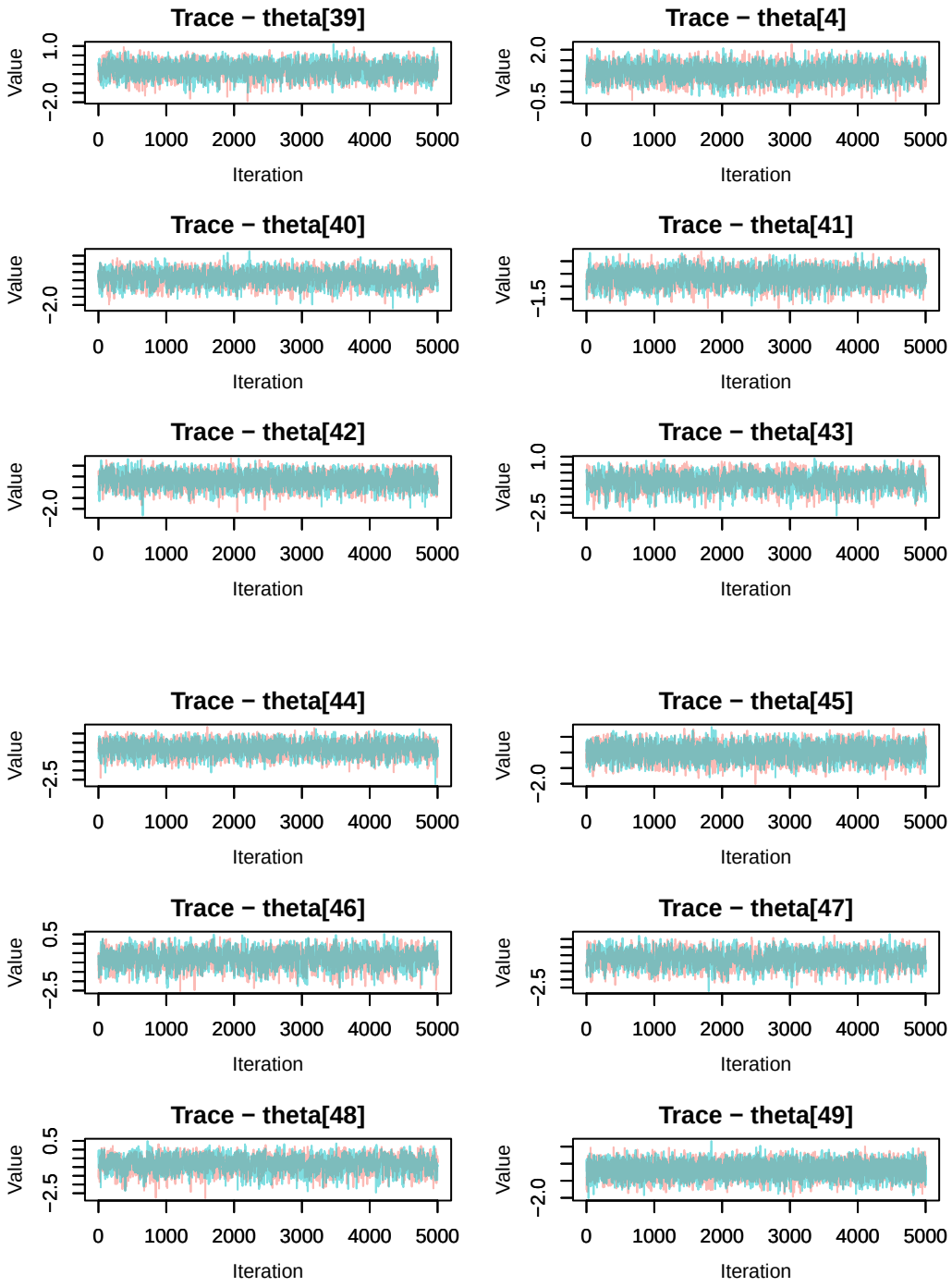

Trace − theta[24]
Value
Iteration

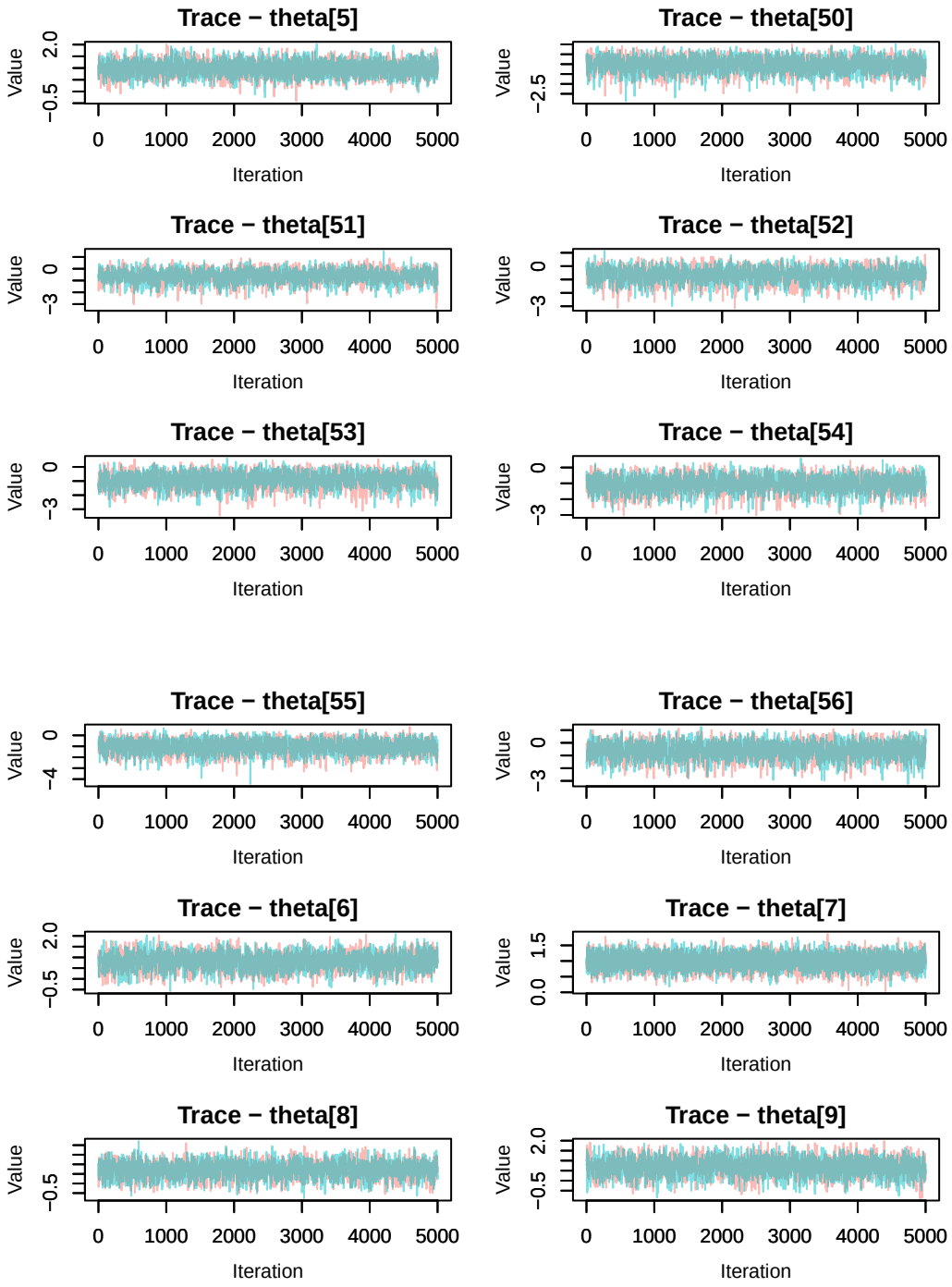

Trace − theta[25]
Value
Iteration


Trace − theta[26]
Value
Iteration


Trace − theta[27]
Value
Iteration


Trace − theta[28]
Value
Iteration
Trace − theta[29]
Value
Iteration
Trace − theta[3]
Value
Iteration
Trace − theta[30]
Value
Iteration
Trace − theta[31]
Value
Iteration
Trace − theta[32]
Value
Iteration
Trace − theta[33]
Value
Iteration
Trace − theta[34]
Value
Iteration
Trace − theta[35]
Value
Iteration
Trace − theta[36]
Value
Iteration
Trace − theta[37]
Value
Iteration
Trace − theta[38]
Value
Iteration


Trace − theta[39]
Trace − theta[4]
Trace − theta[40]
Trace − theta[41]
Trace − theta[42]
Trace − theta[43]
Trace − theta[44]
Trace − theta[45]
Trace − theta[46]
Trace − theta[47]
Trace − theta[48]
Trace − theta[49]
Value
Iteration
0
1000
2000
3000
4000
5000




```
gelman.diag(jagsfitlipCancer.mcmc)
```

Potential scale reduction factors:

	Point est.	Upper C.I.
beta0	1.00	1.00
RR[1]	1.00	1.00
RR[10]	1.00	1.00
RR[11]	1.00	1.00
RR[12]	1.00	1.00
RR[13]	1.00	1.00
RR[14]	1.00	1.00
RR[15]	1.00	1.00
RR[16]	1.00	1.00
RR[17]	1.00	1.00
RR[18]	1.00	1.00
RR[19]	1.00	1.00
RR[2]	1.00	1.00
RR[20]	1.00	1.00
RR[21]	1.00	1.00
RR[22]	1.00	1.01
RR[23]	1.00	1.01
RR[24]	1.01	1.03
RR[25]	1.00	1.00
RR[26]	1.00	1.00
RR[27]	1.00	1.00
RR[28]	1.00	1.01
RR[29]	1.00	1.00
RR[3]	1.00	1.00
RR[30]	1.00	1.00
RR[31]	1.00	1.00
RR[32]	1.00	1.00
RR[33]	1.00	1.00
RR[34]	1.00	1.01
RR[35]	1.00	1.00
RR[36]	1.00	1.01
RR[37]	1.00	1.00
RR[38]	1.00	1.00
RR[39]	1.00	1.00
RR[4]	1.00	1.01
RR[40]	1.00	1.00
RR[41]	1.00	1.00
RR[42]	1.00	1.00
RR[43]	1.01	1.03
RR[44]	1.00	1.00

RR[45]	1.00	1.00
RR[46]	1.00	1.00
RR[47]	1.00	1.01
RR[48]	1.00	1.00
RR[49]	1.00	1.00
RR[5]	1.00	1.00
RR[50]	1.00	1.00
RR[51]	1.00	1.02
RR[52]	1.00	1.01
RR[53]	1.00	1.00
RR[54]	1.00	1.00
RR[55]	1.00	1.00
RR[56]	1.00	1.00
RR[6]	1.00	1.00
RR[7]	1.00	1.01
RR[8]	1.00	1.00
RR[9]	1.00	1.02
tau	1.00	1.00
theta[1]	1.00	1.00
theta[10]	1.00	1.00
theta[11]	1.00	1.00
theta[12]	1.00	1.00
theta[13]	1.00	1.00
theta[14]	1.00	1.00
theta[15]	1.00	1.00
theta[16]	1.00	1.00
theta[17]	1.00	1.00
theta[18]	1.00	1.00
theta[19]	1.00	1.00
theta[2]	1.00	1.00
theta[20]	1.00	1.00
theta[21]	1.00	1.00
theta[22]	1.00	1.00
theta[23]	1.00	1.00
theta[24]	1.01	1.03
theta[25]	1.00	1.00
theta[26]	1.00	1.01
theta[27]	1.00	1.00
theta[28]	1.00	1.01
theta[29]	1.00	1.00
theta[3]	1.00	1.01
theta[30]	1.00	1.00
theta[31]	1.00	1.00

```
theta[32]       1.00       1.00
theta[33]       1.00       1.00
theta[34]       1.00       1.00
theta[35]       1.00       1.00
theta[36]       1.00       1.01
theta[37]       1.00       1.00
theta[38]       1.00       1.00
theta[39]       1.00       1.01
theta[4]        1.00       1.01
theta[40]       1.00       1.00
theta[41]       1.00       1.01
theta[42]       1.00       1.01
theta[43]       1.01       1.03
theta[44]       1.00       1.00
theta[45]       1.00       1.01
theta[46]       1.00       1.00
theta[47]       1.00       1.01
theta[48]       1.00       1.00
theta[49]       1.00       1.00
theta[5]        1.00       1.00
theta[50]       1.00       1.00
theta[51]       1.00       1.01
theta[52]       1.00       1.00
theta[53]       1.00       1.01
theta[54]       1.00       1.00
theta[55]       1.00       1.00
theta[56]       1.00       1.00
theta[6]        1.00       1.00
theta[7]        1.00       1.01
theta[8]        1.00       1.00
theta[9]        1.00       1.01

Multivariate psrf

1.04
```

question 1.c)

we know have to monitor RR as well

```
## extract posterior RR
posRR <- substr(rownames(jags.mod.fit.lipCancer$BUGSoutput$summary), 1, 2) ==
    "RR"

RRRegions = jags.mod.fit.lipCancer$BUGSoutput$summary[posRR, 1]

ScotlandDF$RR = RRRegions
```

Checking the difference between the average RR and the SMR for each region

should I do RR/SMR and see how close it is to 1?

```
ScotlandDF$SMR/ScotlandDF$RR
```

```
    RR[1]     RR[2]     RR[3]     RR[4]     RR[5]     RR[6]     RR[7]     RR[8]
1.3791503 1.0733573 1.1949174 1.2305619 1.1415387 1.2515693 1.0745066 1.2520283
    RR[9]    RR[10]    RR[11]    RR[12]    RR[13]    RR[14]    RR[15]    RR[16]
1.2553943 1.0937287 1.1300741 1.2905314 1.4079491 1.1418594 1.0626658 1.0926245
   RR[17]    RR[18]    RR[19]    RR[20]    RR[21]    RR[22]    RR[23]    RR[24]
1.2280622 1.0737967 1.0556862 1.0822555 1.0257926 1.0171952 1.0141712 1.0195314
   RR[25]    RR[26]    RR[27]    RR[28]    RR[29]    RR[30]    RR[31]    RR[32]
1.0080821 1.0131707 1.0074349 0.9953973 1.0057442 0.9957580 0.9738316 0.9353699
   RR[33]    RR[34]    RR[35]    RR[36]    RR[37]    RR[38]    RR[39]    RR[40]
0.9855707 0.9655924 0.9684404 0.9623293 0.9688204 0.9423093 0.9268801 0.8944802
   RR[41]    RR[42]    RR[43]    RR[44]    RR[45]    RR[46]    RR[47]    RR[48]
0.8925594 0.8699347 0.6645624 0.8058865 0.9082096 0.6798684 0.6090051 0.6403321
   RR[49]    RR[50]    RR[51]    RR[52]    RR[53]    RR[54]    RR[55]    RR[56]
0.9297654 0.7565279 0.4619581 0.4457409 0.3694338 0.3470364 0.0000000 0.0000000
```

Question 1.e)

Now I will have to check when RR is > 1 using similarly the summary Confidicence intervals

```
# ##extract posterior RR posProbRR <-
# substr(rownames(jags.mod.fit.lipCancer$BUGSoutput$summary),1,6)=='ProbRR'
# jags.mod.fit.lipCancer$BUGSoutput$summary[posProbRR,1] ##mean
# jags.mod.fit.lipCancer$BUGSoutput$summary[posProbRR,3] # 2.5
# percentile jags.mod.fit.lipCancer$BUGSoutput$summary[posProbRR,7] #
# 97.5 percentile

## to see which regions RR are in the 95% C.I we will have to see if
```

```
## the 2.5% is small or equal to 1 and if 97.5% is equal or greater
## than 1

## extract posterior RR
posProbRR <- substr(rownames(jags.mod.fit.lipCancer$BUGSoutput$summary),
    1, 2) == "RR"

# jags.mod.fit.lipCancer$BUGSoutput$summary[posProbRR,1] ##mean
lowerCI = jags.mod.fit.lipCancer$BUGSoutput$summary[posProbRR, 3]  # 2.5 percentile
upperCI = jags.mod.fit.lipCancer$BUGSoutput$summary[posProbRR, 7]  # 97.5 percentile
```

Since we know that the 97,5 percentile is always bigger than the the 2,5 percentile to check if the 95% CI of RR is lower than 1, meaning there is no excessive risk, all we have to do is check if a region 97,5% percentile is smaller than 1

```
for (i in 1:nrow(ScotlandDF)) {
    if (upperCI[i] < 1) {
        print(ScotlandDF$Name[i])
    }
}
```

```
[1] "Renfrew"
[1] "Falkirk"
[1] "Motherwell"
[1] "Edinburgh"
[1] "Hamilton"
[1] "Glasgow"
[1] "Dundee"
[1] "Strathkelvin"
```

And as so here are the regions without any excessive lip cancer.

Question 2 a)

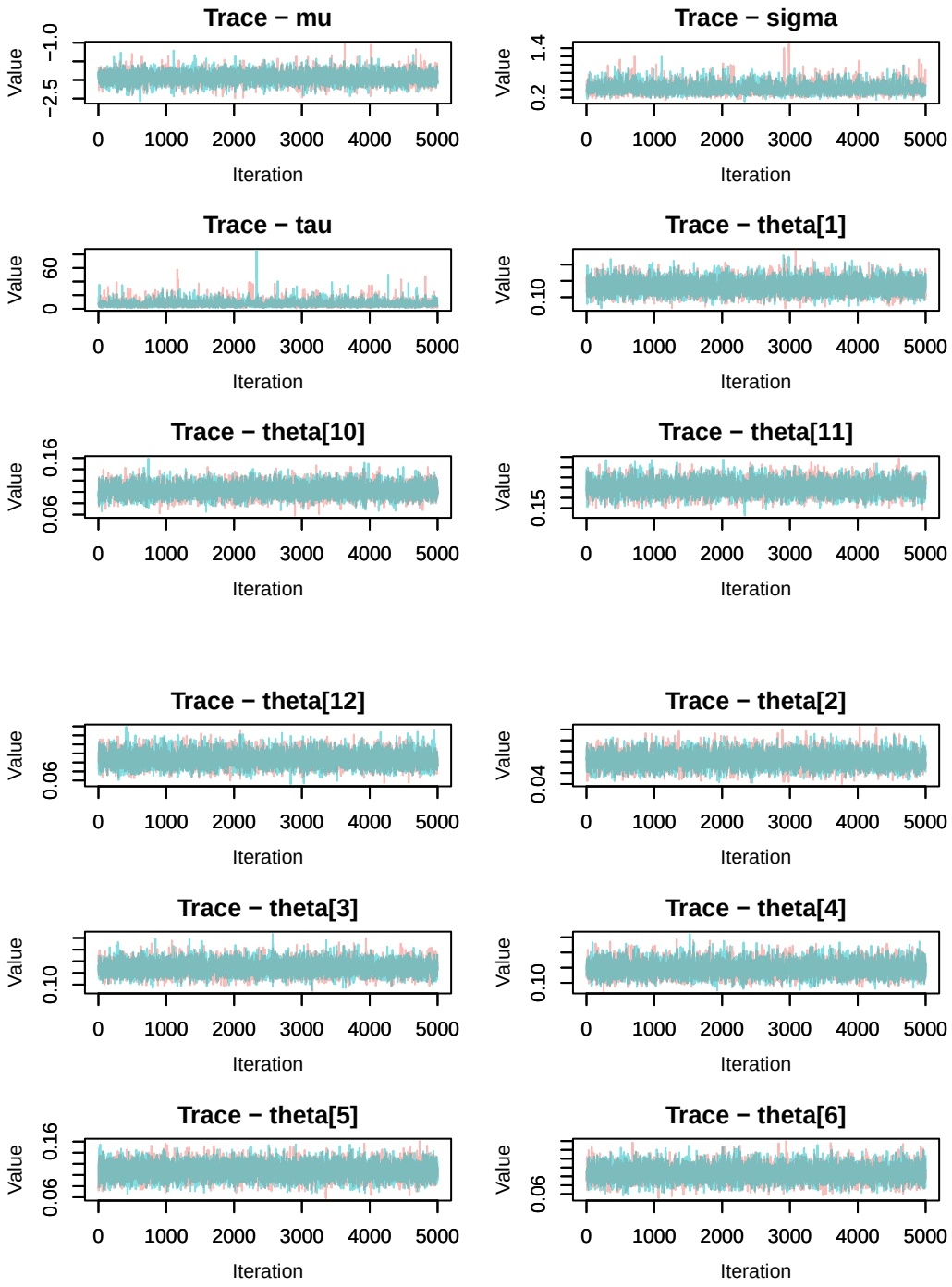
```
jags.mod.infantDeath <- function(){
for(i in 1:12){
r[i] ~ dbin(theta[i],n[i])
logit(theta[i]) <- logit.theta[i]
logit.theta[i] ~ dnorm(mu, tau)
}
```

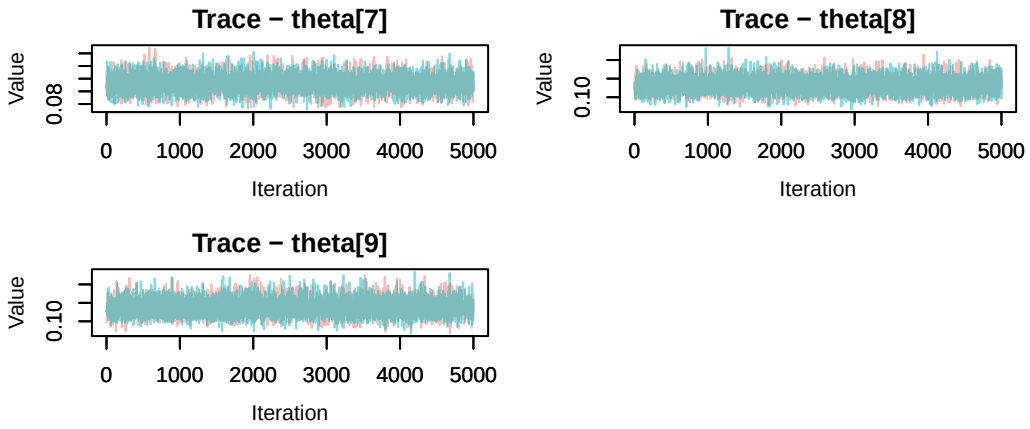
```
mu ~ dnorm(0,0.001)
tau <- 1/sigma^2
sigma ~ dunif(0,100)
}

# Parameters to monitor
jags.param.infantDeath <- c("mu", "sigma", "theta", "tau")

## initial values
inits1 <- list('tau' = 100, 'sigma' = 20, 'theta'=c(10,-5,-25))
inits2 <- list('tau'=100000,'sigma'=-100,'theta'=c(-100,100,500))

jags.inits = list(inits1,inits2)

## data

jags.data.infantDeath <- list(r = surgicalDF$r, n = surgicalDF$n)

## fit the jags model

jags.mod.fit.infantDeath <- jags(data = jags.data.infantDeath, #inits = jags.inits,
parameters.to.save = jags.param.infantDeath, n.chains = 2, n.iter = 10000,
n.burnin = 5000,n.thin=1,model.file = jags.mod.infantDeath, DIC=FALSE)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 12
   Unobserved stochastic nodes: 14
   Total graph size: 57

Initializing model
```

Now that we have the fitted the model we will observe the plots to check for possible convergence during the recorded 5000 iterations.

```
jagsfitinfantDeath.mcmc <- as.mcmc(jags.mod.fit.infantDeath)
MCMCtrace(jagsfitinfantDeath.mcmc, type = "trace", ind = TRUE, pdf = FALSE)
```


Trace − mu
Value
−2.5
−1.0
0
1000
2000
3000
4000
5000
Iteration
Trace − sigma
Value
0.2
1.4
Iteration
Trace − tau
Value
0
60
Iteration
Trace − theta[1]
Value
0.10
Iteration
Trace − theta[10]
Value
0.06
0.16
Iteration
Trace − theta[11]
Value
0.15
Iteration
Trace − theta[12]
Value
0.06
Iteration
Trace − theta[2]
Value
0.04
Iteration
Trace − theta[3]
Value
0.10
Iteration
Trace − theta[4]
Value
0.10
Iteration
Trace − theta[5]
Value
0.06
0.16
Iteration
Trace − theta[6]
Value
0.06
Iteration



As we can see from the plots, there seems to be no visible patterns on the chains and all parameters have stable oscillation around a common value. We can also observe that the chains seem to be well mixed with each other as it is difficult to distinguish between chains.

However visualization alone is not enough to fully access the convergence so we will use the Gelman-Rubin diagnostics to check for convergence.

```
gelman.diag(jagsfitinfantDeath.mcmc)
```

```
Potential scale reduction factors:

          Point est. Upper C.I.
mu              1.00       1.00
sigma           1.01       1.01
tau             1.00       1.00
theta[1]        1.00       1.00
theta[10]       1.00       1.00
theta[11]       1.00       1.00
theta[12]       1.00       1.00
theta[2]        1.00       1.00
theta[3]        1.00       1.00
theta[4]        1.00       1.00
```

```
theta[5]        1.00       1.00
theta[6]        1.00       1.00
theta[7]        1.00       1.00
theta[8]        1.00       1.00
theta[9]        1.00       1.00

Multivariate psrf

1
```

As we can see from the Upper Confidence Interval, the values of all parameters are approximately 1, indicating that the parameters have indeed converged.

Question 2 b)

```
jags.mod.infantDeath <- function(){
for(i in 1:12){
r[i] ~ dbin(theta[i],n[i])
logit(theta[i]) <- logit.theta[i]
logit.theta[i] ~ dnorm(mu, tau)
}
mu ~ dnorm(0,0.001)
tau <- 1/sigma^2
sigma ~ dunif(0,100)

## pi = P (rpred i > ri) + 1/2 * P (rpred i = ri)

#P[i] = (r > r[i]) + 0.5*

}

# Parameters to monitor
jags.param.infantDeath <- c("mu", "sigma", "theta", "tau")

## initial values
inits1 <- list('tau' = 100, 'sigma' = 20, 'theta'=c(10,-5,-25))
inits2 <- list('tau'=100000,'sigma'=-100,'theta'=c(-100,100,500))

jags.inits = list(inits1,inits2)

## data
```

```r
jags.data.infantDeath <- list(r = surgicalDF$r, n = surgicalDF$n)

## fit the jags model

jags.mod.fit.infantDeath <- jags(data = jags.data.infantDeath, #inits = jags.inits,
parameters.to.save = jags.param.infantDeath, n.chains = 2, n.iter = 10000,
n.burnin = 5000,n.thin=1,model.file = jags.mod.infantDeath, DIC=FALSE)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 12
   Unobserved stochastic nodes: 14
   Total graph size: 57

Initializing model
```

Question 2 c)

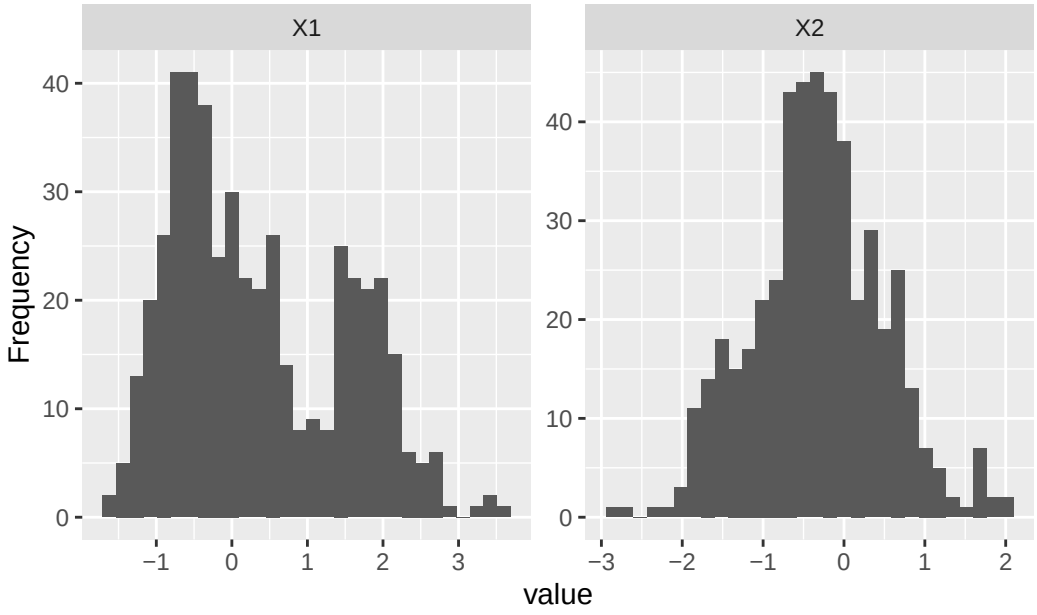
Question B - classification

Question B.1

Group 0 is characterised for always having x1 > -1,65 and x2 < 2,05 with the majority of the occurences of group 0 having

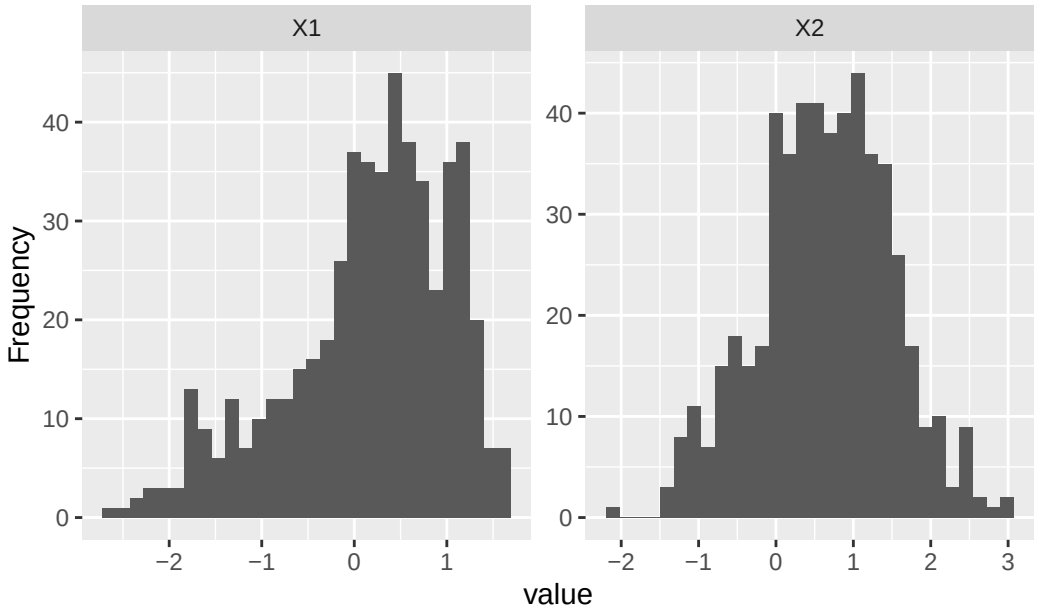
```r
## I am just going to split the dataframe in two so its easier to show
## an histogram of each variable for both groups

group0 = subset(classificationDF, Group == 0)
group1 = subset(classificationDF, Group == 1)

## removing the id variable from both of these new data frames so they
## are not included in the histograms
group0 = group0 %>%
    select(-(1))
group1 = group1 %>%
    select(-(1))
```

```
plot_histogram(group0)
```



As we can see from the histograms from group 0 it seems that X2 seems to follow a normal distribution that is very slightly skewed to the left and has mean -0,5. X1 is clearly skewed to the right possibly following a normal distribution and again mean equal to -0,5.

```
plot_histogram(group1)
```



Group 1 histograms reveal that X2 is normally distributed with mean equal to 0,5 and sightly skewed to the left. The variable X1 also seems to follow a normal distribution that has a very prolonged skewness to the left and mean 0,5.0

Question B.2

Both LDA and QDA assume normality which is true from what we can see from the histograms on 2.1

Starting with Linear discriminant analysis (LDA) we can easily see from the initial graph that we can see that the any sensible decision boundary is highly non-linear. As analysed from histograms this data violates the initial assumption of homogeneity of variance (homoscedasticity). The non-linearity of the data would also lead to LDA having terrible performance. As such we can conclude that LDA is not suitable for this data.

Secondly, quadratic discriminant analysis (QDA) follow the assumption of normality of each of the variables which does seem to hold even with how highly non-linear the data is. However judging from the plot with the observations, it does seem that QDA might not have enough flexibility to account for all the variability in what is the theoretical Bayes decision boundary. As such QDA could be appropriate for this data but not the best performing classification method for such data.

Thirdly, K-nearest neighbour classification (KNN) has no assumptions made about the shape of the decision, as such as we choose a adequate level of smoothness to prevent overfitting KNN classification is an adequate method for the data.

Support Vector Machines (SVM) are most commonly used for binary classification, which holds true for our data. The high non-linear decision boundary however requires a careful selection of a non linear kernel function to ensure that the boundary becomes non-linear when converted back to the regular space. As such I deem SVM an appropriate method to classify the data.

Finally, Random forest works by constructing multiple decision trees randomly selecting a portion of the variables and de-correlate them using multiple bootstrapped sets. Since our data has only 2 variables there isn't enough variables to overcome overfitting. To conclude, Random Forest classification is not very appropriate to this type of data.

Question B.3

```
# make this example reproducible
set.seed(26041999)
# use 80% of dataset as training set and 20% as test set
sample = sample.split(classificationDF$Group, SplitRatio = 0.8)
## note that the column selected above can be any column.
train = classificationDF[sample, ]
test = classificationDF[!sample, ]

# split the data using the indices returned by the createDataPartition
# function
X_train = train[, c("X1", "X2")]
y_train = train$Group
X_test = test[, c("X1", "X2")]
y_test = test$Group

# check the dimensions
dim(train)
```

```
[1] 800   4
```

```
dim(test)
```

```
[1] 200   4
```

Question B.4

The 3 method I will be using are QDA, KNN and SVM as deemed to be the most appropriate methods for this data.

QDA - quadratic discriminant analysis

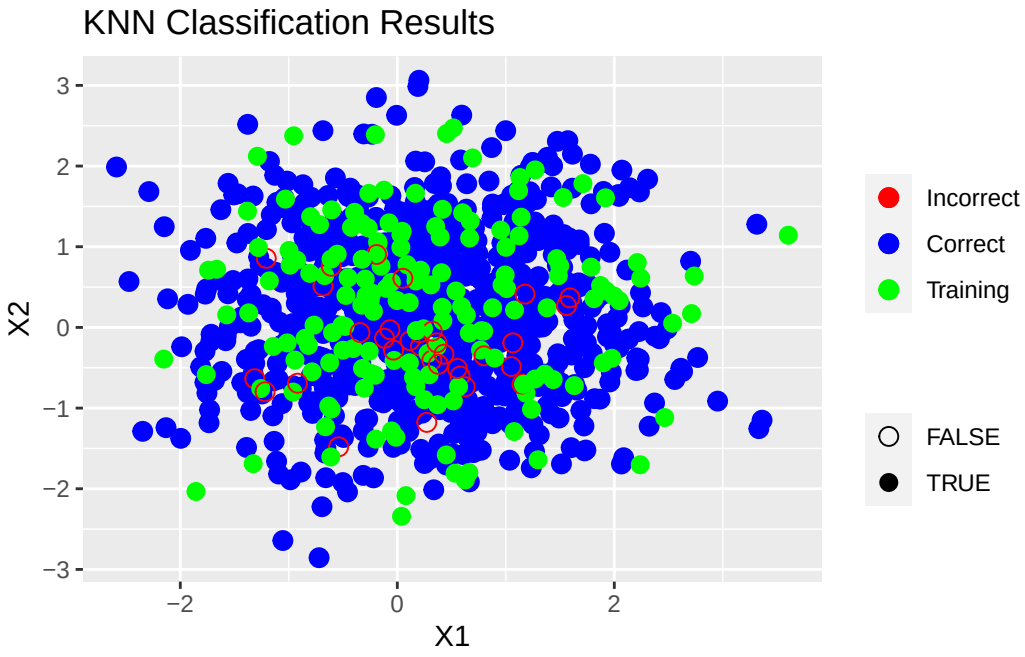
KNN - K-nearest neighbour

```
# fit the model
fit = knn(train = X_train, test = X_test, cl = y_train, k = 3)
# fit = knn(xTrain,xTest,yTrain,k=3)

# produce the confusion matrix when numbers are used as class labels we
# might need
confusion = confusionMatrix(as.factor(fit), as.factor(y_test))

confusion
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 80 17
         1 15 88

               Accuracy : 0.84
                 95% CI : (0.7817, 0.8879)
    No Information Rate : 0.525
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6795

 Mcnemar's Test P-Value : 0.8597

            Sensitivity : 0.8421
            Specificity : 0.8381
         Pos Pred Value : 0.8247
         Neg Pred Value : 0.8544
             Prevalence : 0.4750
         Detection Rate : 0.4000
```

```
Detection Prevalence : 0.4850
   Balanced Accuracy : 0.8401

     'Positive' Class : 0
```

```
test$predicted_group = fit
test$correct_prediction = test$predicted_group == test$Group

# plot the data points
p = ggplot() + geom_point(data = train, aes(x = X1, y = X2, color = "Training"),
    size = 3) + geom_point(data = test, aes(x = X1, y = X2, color = correct_prediction,
    shape = factor(correct_prediction)), size = 3) + scale_color_manual(values = c("red",
    "blue", "green"), labels = c("Incorrect", "Correct", "Training")) + scale_shape_manual
    16)) + labs(x = "X1", y = "X2", color = "", shape = "") + ggtitle("KNN Classification

# display the plot
print(p)
```



SVM - Support Vector Machines

```r
## checking the model performance with multiple difference C constants
mdl = train(x = X_train, y = y_train, method = "svmLinear", trControl = trainControl("cv",
    number = 5), tuneGrid = expand.grid(C = seq(0, 2, length = 20)))
```

```
Warning in train.default(x = X_train, y = y_train, method = "svmLinear", : You
are trying to do regression and your outcome only has two possible values Are
you trying to do classification? If so, use a 2 level factor as your outcome
column.
```

```
Warning: Setting row names on a tibble is deprecated.
```

```
Warning: model fit failed for Fold1: C=0.0000 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
```

```
Warning: Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
```

```
Warning: model fit failed for Fold2: C=0.0000 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters
```

```
Warning: Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.

Warning: model fit failed for Fold3: C=0.0000 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters

Warning: Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
```

```
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.

Warning: model fit failed for Fold4: C=0.0000 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters

Warning: Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.

Warning: model fit failed for Fold5: C=0.0000 Error in .local(x, ...) :
  No Support Vectors found. You may want to change your parameters

Warning: Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
```

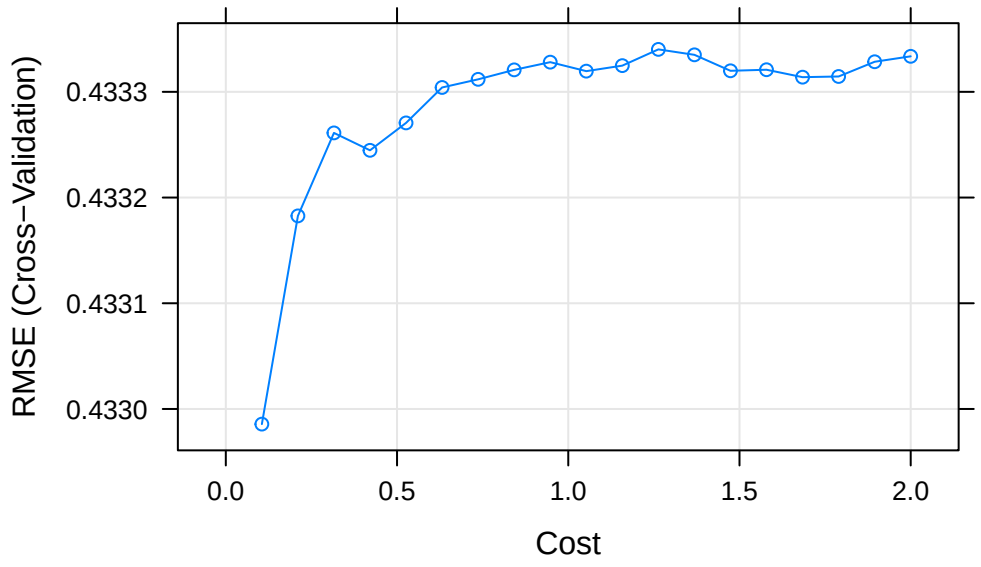
```
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
Setting row names on a tibble is deprecated.
```

```
Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
There were missing values in resampled performance measures.
```

```
Warning in train.default(x = X_train, y = y_train, method = "svmLinear", :
missing values found in aggregated results
```

```
Warning: Setting row names on a tibble is deprecated.
```

```r
# Plot model accuracy vs different values of Cost
plot(mdl)
```



```
# Print the best tuning parameter C that maximises model accuracy
mdl$bestTune
```

```
         C
2 0.1052632
```

#TODO talk about how there is a balance on how the number of miss classified

```
## the SVM not working part

# # Test model on testing data yTestPred = predict(mdl, newdata=X_test)
# confusionMatrix(yTestPred, y_test) # predicted/true
```

quick debug que funciona CARALHOOOOOOOOOOOOOO

TODO this is using radial but I should have one with polynimial as well to show that polinomial is better

```
## https://github.com/MatheusSchaly/Online-Courses/blob/master/Machine_Learning_A-Z_Hands-
## to add visualization

mdl = train(x=X_train,y=y_train, method='svmRadial')
print(mdl)
```

```
Support Vector Machines with Radial Basis Function Kernel

800 samples
  2 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 800, 800, 800, 800, 800, 800, ...
Resampling results across tuning parameters:

  C     RMSE       Rsquared   MAE
  0.25  0.3165339  0.6025039  0.2003828
  0.50  0.3139614  0.6107481  0.1952854
  1.00  0.3113789  0.6183418  0.1920298
```

```
Tuning parameter 'sigma' was held constant at a value of 1.432896
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 1.432896 and C = 1.
```

```r
mdl = svm(Group ~ .,
                  data = train,
                  type = 'C-classification',
                  kernel = 'radial')

# Test model on testing data
yTestPred = predict(mdl, newdata=test)
# yTestPred = mdl %>% predict(xTest)
confusionMatrix(as.factor(yTestPred), as.factor(y_test)) # predicted/true
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 80 20
         1 15 85

               Accuracy : 0.825
                 95% CI : (0.7651, 0.875)
    No Information Rate : 0.525
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.65

 Mcnemar's Test P-Value : 0.499

            Sensitivity : 0.8421
            Specificity : 0.8095
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.8500
             Prevalence : 0.4750
         Detection Rate : 0.4000
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.8258

       'Positive' Class : 0
```

Question B.5