Diffusion results

Report Goal: Provide a minimalistic report prototype for future reports.
Report Description: This is a prototype of a simple report. It should represent the one side of the spectrum of MIECHV automated reports..

Cohort: 1980

Andrey -write something here.

```
cohortYear <- 1980
```

```
require(rjags)
```

```
## Loading required package: rjags
```

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

```
## Loading required package: lattice
```

```
## linking to JAGS 3.3.0
```

```
## module basemod loaded
```

```
## module bugs loaded
```

```
if (Sys.info()["nodename"] == "MICKEY") pathDirectory <- "F:/Users/wibeasley/Documents/Consulting/EmosaM
# pathDirectory <-
# 'F:/Users/wibeasley/Documents/Consulting/EmosaMcmc/Dev/EMOSA/OneShot_Only1984Diffusion'
if (Sys.info()["nodename"] == "MERKANEZ-PC") pathDirectory <- "F:/Users/wibeasley/Documents/SSuccess/Int

# pathModel <- file.path(pathDirectory,
# 'DiffusionOnly/DiffusionGauss.bugs') pathModel <-
# file.path(pathDirectory, 'DiffusionOnly/DiffusionLogit.bugs')
pathModel <- file.path(pathDirectory, "DiffusionOnly/DiffusionBeta.bugs")
pathData <- file.path(pathDirectory, "Data/SummaryBirthYearByTime.csv")
# curve(dbeta(x, 1,1)) curve(dbeta(x, 10,10)) curve(dlogis(x, location =
# .25, scale = 1), xlim=c(-5, 5))


ds <- read.csv(pathData, stringsAsFactors = FALSE)
ds <- ds[ds$byear == cohortYear, ]  #Select only the desired cohort
ds <- ds[order(ds$time), ]  #Sort, just, to make sure values will be passed to JAGS in the correct order

pg <- ds$ProportionGoers
pi <- ds$ProportionIrregulars
pa <- ds$ProportionAbsentees

# Proportion of Goers, of Irregulars, or Nongoers (or absentees) {Check
# these with data; I may have messed up the order} For the 1984 cohort pg
# <- c(0.401088929, 0.340290381, 0.249546279, 0.218693285, 0.180580762,
# 0.167876588, 0.157894737, 0.158802178, 0.161524501) pi <- c(0.233212341,
# 0.256805808, 0.288566243, 0.305807623, 0.27676951, 0.270417423,
# 0.229582577, 0.250453721, 0.237749546) pa <- c(0.36569873, 0.402903811,
# 0.461887477, 0.475499093, 0.542649728, 0.561705989, 0.612522686,
# 0.590744102, 0.600725953)
timeCount <- length(pg)
if (length(pi) != timeCount) stop("The proportions have a different number of time points.")
if (length(pa) != timeCount) stop("The proportions have a different number of time points.")
mean(c(pg, pi, pa))
```

```
## [1] 0.3333
```

```r
jagsData <- list(pg = pg, pi = pi, pa = pa, timeCount = timeCount)

# parameters <- c('mu')
parametersToTrack <- c("Kgi", "Kga", "Kig", "Kia", "Kag", "Kai", "sumG", "sumI")
# parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia', 'Kag', 'Kai', 'sumG',
# 'sumI', 'sumA') parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia',
# 'Kag', 'Kai', 'sigmaG', 'sigmaI') inits <- function(){
# list(Kgi=rnorm(1), Kga=rnorm(1), Kig=rnorm(1), Kia=rnorm(1),
# Kag=rnorm(1), Kai=rnorm(1)) }

countChains <- 6  #3 #6
countIterations <- 1e+05

startTime <- Sys.time()

jagsModel <- jags.model(file = pathModel, data = jagsData, n.chains = countChains)  #, inits=inits)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 183
##
## Initializing model
```

```r
# print(jagsModel) update(jagsModel, 1000) #modifies the original object
# and returns NULL
dic <- dic.samples(jagsModel, n.iter = countIterations)
dic
```

```
## Mean deviance:  -97.7
## penalty 7.46
## Penalized deviance: -90.3
```

```r
# mcarray <- jags.samples(model=jagsModel, c('mu'),
# n.iter=countIterations) #If I understand correctly, the following line
# is similar, but better
chains <- coda.samples(jagsModel, variable.names = parametersToTrack, n.iter = countIterations)  # updat
elapsed <- Sys.time() - startTime
(condensed <- summary(chains))
```

```
##
## Iterations = 101001:201000
## Thinning interval = 1
## Number of chains = 6
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD Naive SE Time-series SE
## Kag     0.178 6.38e-02 8.24e-05        0.00189
## Kai     0.091 6.49e-02 8.38e-05        0.00149
## Kga     0.432 2.44e-01 3.14e-04        0.00802
## Kgi     0.530 2.17e-01 2.80e-04        0.00757
## Kia     0.304 1.73e-01 2.23e-04        0.00545
## Kig     0.218 1.22e-01 1.58e-04        0.00403
## sumG 2066.257 1.03e+03 1.33e+00        7.31173
## sumI 1646.732 8.24e+02 1.06e+00        6.38972
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%      75%    97.5%
## Kag  5.20e-02 1.35e-01 1.79e-01    0.221    0.303
## Kai  4.56e-03 4.07e-02 7.94e-02    0.128    0.246
## Kga  3.23e-02 2.40e-01 4.15e-01    0.608    0.925
## Kgi  9.14e-02 3.76e-01 5.39e-01    0.693    0.921
## Kia  2.68e-02 1.73e-01 2.87e-01    0.415    0.684
## Kig  1.82e-02 1.25e-01 2.09e-01    0.298    0.481
## sumG 5.78e+02 1.31e+03 1.89e+03 2633.575 4531.681
## sumI 4.57e+02 1.04e+03 1.51e+03 2099.256 3619.648
```

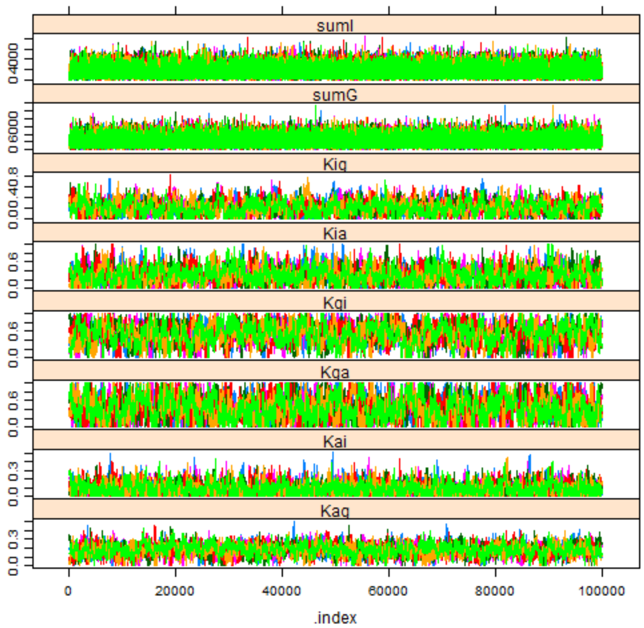
```r
# windows() # dev.off()
gelman.diag(chains, autoburnin = FALSE)  #This is R-hat; the burnin period is manually specified above,
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## Kag           1       1.00
## Kai           1       1.01
## Kga           1       1.00
## Kgi           1       1.00
## Kia           1       1.00
## Kig           1       1.01
## sumG          1       1.00
## sumI          1       1.00
##
## Multivariate psrf
##
## 1
```

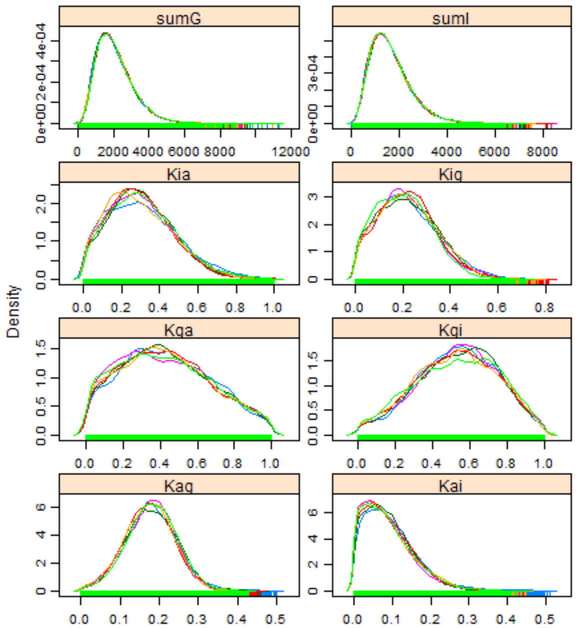
```
effectiveSize(chains)  #Sample size adjusted for autocorrelation
```

```
##      Kag      Kai      Kga      Kgi      Kia      Kig     sumG     sumI
##   1480.4   2471.4   1204.8    997.3   1449.5   1185.5  43260.6  37083.4
```

```
xyplot(chains)  #Needs at least two parameters; else throws an error.
```




```
densityplot(chains)
```




```
# gelman.plot(chains) print(rbind(paste('estimated mu: ',
# condensed$statistics['mu0', 'Mean']), paste('observed mean:', mean(y,
# na.rm=T))))
elapsed
```

```
## Time difference of 10.26 mins
```

Cohort: 1981

```
cohortYear <- 1981
```

```
require(rjags)



if (Sys.info()["nodename"] == "MICKEY") pathDirectory <- "F:/Users/wibeasley/Documents/Consulting/EmosaM
# pathDirectory <-
# 'F:/Users/wibeasley/Documents/Consulting/EmosaMcmc/Dev/EMOSA/OneShot_Only1984Diffusion'
if (Sys.info()["nodename"] == "MERKANEZ-PC") pathDirectory <- "F:/Users/wibeasley/Documents/SSuccess/Int

# pathModel <- file.path(pathDirectory,
# 'DiffusionOnly/DiffusionGauss.bugs') pathModel <-
# file.path(pathDirectory, 'DiffusionOnly/DiffusionLogit.bugs')
pathModel <- file.path(pathDirectory, "DiffusionOnly/DiffusionBeta.bugs")
pathData <- file.path(pathDirectory, "Data/SummaryBirthYearByTime.csv")
# curve(dbeta(x, 1,1)) curve(dbeta(x, 10,10)) curve(dlogis(x, location =
# .25, scale = 1), xlim=c(-5, 5))


ds <- read.csv(pathData, stringsAsFactors = FALSE)
ds <- ds[ds$byear == cohortYear, ]  #Select only the desired cohort
ds <- ds[order(ds$time), ]  #Sort, just, to make sure values will be passed to JAGS in the correct order

pg <- ds$ProportionGoers
pi <- ds$ProportionIrregulars
pa <- ds$ProportionAbsentees

# Proportion of Goers, of Irregulars, or Nongoers (or absentees) {Check
# these with data; I may have messed up the order} For the 1984 cohort pg
# <- c(0.401088929, 0.340290381, 0.249546279, 0.218693285, 0.180580762,
# 0.167876588, 0.157894737, 0.158802178, 0.161524501) pi <- c(0.233212341,
# 0.256805808, 0.288566243, 0.305807623, 0.27676951, 0.270417423,
# 0.229582577, 0.250453721, 0.237749546) pa <- c(0.36569873, 0.402903811,
# 0.461887477, 0.475499093, 0.542649728, 0.561705989, 0.612522686,
# 0.590744102, 0.600725953)
timeCount <- length(pg)
if (length(pi) != timeCount) stop("The proportions have a different number of time points.")
if (length(pa) != timeCount) stop("The proportions have a different number of time points.")
mean(c(pg, pi, pa))
```

```
## [1] 0.3333
```

```
jagsData <- list(pg = pg, pi = pi, pa = pa, timeCount = timeCount)

# parameters <- c('mu')
parametersToTrack <- c("Kgi", "Kga", "Kig", "Kia", "Kag", "Kai", "sumG", "sumI")
# parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia', 'Kag', 'Kai', 'sumG',
# 'sumI', 'sumA') parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia',
# 'Kag', 'Kai', 'sigmaG', 'sigmaI') inits <- function(){
# list(Kgi=rnorm(1), Kga=rnorm(1), Kig=rnorm(1), Kia=rnorm(1),
# Kag=rnorm(1), Kai=rnorm(1)) }

countChains <- 6  #3 #6
countIterations <- 1e+05

startTime <- Sys.time()

jagsModel <- jags.model(file = pathModel, data = jagsData, n.chains = countChains)  #, inits=inits)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 185
##
## Initializing model
```

```
# print(jagsModel) update(jagsModel, 1000) #modifies the original object
# and returns NULL
dic <- dic.samples(jagsModel, n.iter = countIterations)
dic
```

```
## Mean deviance:  -101
## penalty 7.99
## Penalized deviance: -92.6
```

```
# mcarray <- jags.samples(model=jagsModel, c('mu'),
# n.iter=countIterations) #If I understand correctly, the following line
# is similar, but better
chains <- coda.samples(jagsModel, variable.names = parametersToTrack, n.iter = countIterations)  # updat
elapsed <- Sys.time() - startTime
(condensed <- summary(chains))
```

```
##
## Iterations = 101001:201000
## Thinning interval = 1
## Number of chains = 6
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD Naive SE Time-series SE
## Kag  1.28e-01 3.78e-02 4.87e-05       5.65e-04
## Kai  5.87e-02 4.33e-02 5.58e-05       6.97e-04
## Kga  2.61e-01 1.60e-01 2.07e-04       5.28e-03
## Kgi  4.05e-01 1.80e-01 2.32e-04       7.18e-03
## Kia  1.99e-01 1.20e-01 1.55e-04       3.08e-03
## Kig  2.41e-01 1.15e-01 1.48e-04       3.89e-03
## sumG 3.23e+03 1.64e+03 2.12e+00       1.03e+01
## sumI 1.80e+03 9.06e+02 1.17e+00       5.55e+00
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%      75%    97.5%
## Kag  5.38e-02 1.04e-01 1.27e-01 1.51e-01    0.205
## Kai  2.89e-03 2.61e-02 5.06e-02 8.17e-02    0.164
## Kga  1.53e-02 1.33e-01 2.47e-01 3.69e-01    0.609
## Kgi  6.97e-02 2.76e-01 4.01e-01 5.30e-01    0.762
## Kia  1.27e-02 1.04e-01 1.89e-01 2.78e-01    0.456
## Kig  3.59e-02 1.60e-01 2.34e-01 3.14e-01    0.485
## sumG 8.79e+02 2.03e+03 2.96e+03 4.13e+03 7185.096
## sumI 4.83e+02 1.13e+03 1.64e+03 2.29e+03 3952.185
```

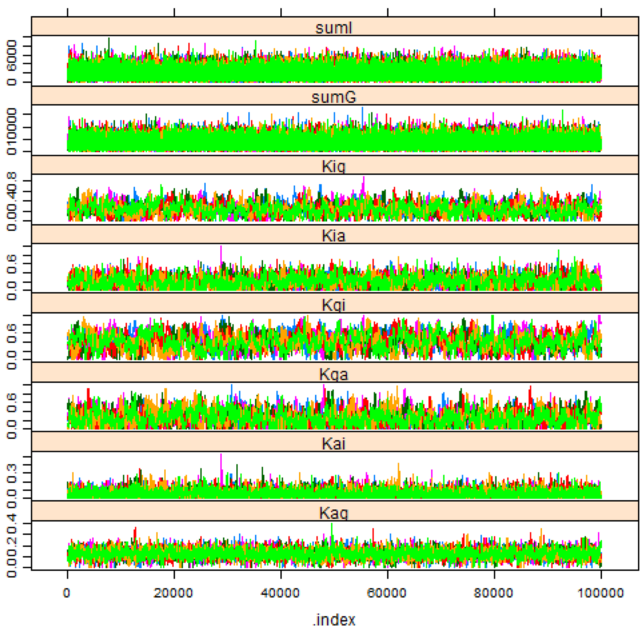
```
# windows() # dev.off()
gelman.diag(chains, autoburnin = FALSE)  #This is R-hat; the burnin period is manually specified above,
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## Kag        1.00       1.00
## Kai        1.00       1.00
## Kga        1.01       1.02
## Kgi        1.01       1.01
## Kia        1.01       1.02
## Kig        1.00       1.01
## sumG       1.00       1.00
## sumI       1.00       1.00
##
## Multivariate psrf
##
## 1.01
```

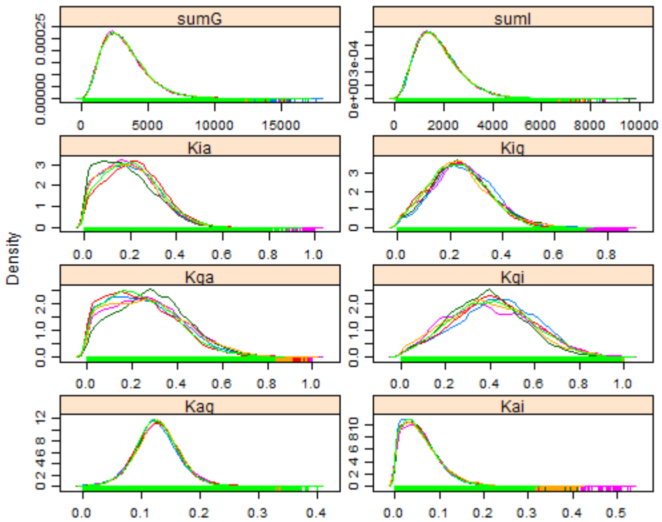
```
effectiveSize(chains)  #Sample size adjusted for autocorrelation
```

```
##     Kag     Kai     Kga     Kgi     Kia     Kig    sumG    sumI
##  3893.6  5491.8  1355.9   771.1  2402.7  1078.5 42181.0 45149.5
```

```
xyplot(chains)  #Needs at least two parameters; else throws an error.
```




```
densityplot(chains)
```




```
# gelman.plot(chains) print(rbind(paste('estimated mu: ',
# condensed$statistics['mu0', 'Mean']), paste('observed mean:', mean(y,
# na.rm=T))))
elapsed
```

```
## Time difference of 10 mins
```

Cohort: 1982

```
cohortYear <- 1982
```

```
require(rjags)



if (Sys.info()["nodename"] == "MICKEY") pathDirectory <- "F:/Users/wibeasley/Documents/Consulting/EmosaM
# pathDirectory <-
# 'F:/Users/wibeasley/Documents/Consulting/EmosaMcmc/Dev/EMOSA/OneShot_Only1984Diffusion'
if (Sys.info()["nodename"] == "MERKANEZ-PC") pathDirectory <- "F:/Users/wibeasley/Documents/SSuccess/Int

# pathModel <- file.path(pathDirectory,
# 'DiffusionOnly/DiffusionGauss.bugs') pathModel <-
# file.path(pathDirectory, 'DiffusionOnly/DiffusionLogit.bugs')
pathModel <- file.path(pathDirectory, "DiffusionOnly/DiffusionBeta.bugs")
pathData <- file.path(pathDirectory, "Data/SummaryBirthYearByTime.csv")
# curve(dbeta(x, 1,1)) curve(dbeta(x, 10,10)) curve(dlogis(x, location =
# .25, scale = 1), xlim=c(-5, 5))


ds <- read.csv(pathData, stringsAsFactors = FALSE)
ds <- ds[ds$byear == cohortYear, ]  #Select only the desired cohort
ds <- ds[order(ds$time), ]  #Sort, just, to make sure values will be passed to JAGS in the correct order

pg <- ds$ProportionGoers
pi <- ds$ProportionIrregulars
pa <- ds$ProportionAbsentees

# Proportion of Goers, of Irregulars, or Nongoers (or absentees) {Check
# these with data; I may have messed up the order} For the 1984 cohort pg
# <- c(0.401088929, 0.340290381, 0.249546279, 0.218693285, 0.180580762,
# 0.167876588, 0.157894737, 0.158802178, 0.161524501) pi <- c(0.233212341,
# 0.256805808, 0.288566243, 0.305807623, 0.27676951, 0.270417423,
# 0.229582577, 0.250453721, 0.237749546) pa <- c(0.36569873, 0.402903811,
# 0.461887477, 0.475499093, 0.542649728, 0.561705989, 0.612522686,
# 0.590744102, 0.600725953)
timeCount <- length(pg)
if (length(pi) != timeCount) stop("The proportions have a different number of time points.")
if (length(pa) != timeCount) stop("The proportions have a different number of time points.")
mean(c(pg, pi, pa))
```

```
## [1] 0.3333
```

```
jagsData <- list(pg = pg, pi = pi, pa = pa, timeCount = timeCount)

# parameters <- c('mu')
parametersToTrack <- c("Kgi", "Kga", "Kig", "Kia", "Kag", "Kai", "sumG", "sumI")
# parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia', 'Kag', 'Kai', 'sumG',
# 'sumI', 'sumA') parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia',
# 'Kag', 'Kai', 'sigmaG', 'sigmaI') inits <- function(){
# list(Kgi=rnorm(1), Kga=rnorm(1), Kig=rnorm(1), Kia=rnorm(1),
# Kag=rnorm(1), Kai=rnorm(1)) }

countChains <- 6  #3 #6
countIterations <- 1e+05

startTime <- Sys.time()

jagsModel <- jags.model(file = pathModel, data = jagsData, n.chains = countChains)  #, inits=inits)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 185
##
## Initializing model
```

```
# print(jagsModel) update(jagsModel, 1000) #modifies the original object
# and returns NULL
dic <- dic.samples(jagsModel, n.iter = countIterations)
dic
```

```
## Mean deviance:  -92.4
## penalty 8.04
## Penalized deviance: -84.3
```

```
# mcarray <- jags.samples(model=jagsModel, c('mu'),
# n.iter=countIterations) #If I understand correctly, the following line
# is similar, but better
chains <- coda.samples(jagsModel, variable.names = parametersToTrack, n.iter = countIterations)  # updat
elapsed <- Sys.time() - startTime
(condensed <- summary(chains))
```

```
##
## Iterations = 101001:201000
## Thinning interval = 1
## Number of chains = 6
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD Naive SE Time-series SE
## Kag  1.05e-01 5.65e-02 7.29e-05       0.001414
## Kai  5.58e-02 3.41e-02 4.40e-05       0.000523
## Kga  2.06e-01 1.41e-01 1.82e-04       0.002688
## Kgi  6.71e-01 1.32e-01 1.70e-04       0.003229
## Kia  2.20e-01 1.41e-01 1.82e-04       0.004504
## Kig  4.47e-01 1.65e-01 2.13e-04       0.006000
## sumG 7.23e+02 3.60e+02 4.64e-01       1.799472
## sumI 2.71e+03 1.39e+03 1.79e+00       7.575576
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%      75%    97.5%
## Kag  1.19e-02 6.39e-02 9.95e-02 1.39e-01    0.230
## Kai  4.41e-03 3.03e-02 5.17e-02 7.59e-02    0.134
## Kga  1.01e-02 9.36e-02 1.86e-01 2.94e-01    0.528
## Kgi  3.99e-01 5.87e-01 6.75e-01 7.60e-01    0.920
## Kia  1.22e-02 1.07e-01 2.02e-01 3.12e-01    0.531
## Kig  1.08e-01 3.36e-01 4.55e-01 5.65e-01    0.754
## sumG 1.99e+02 4.59e+02 6.64e+02 9.23e+02 1578.505
## sumI 7.15e+02 1.70e+03 2.48e+03 3.47e+03 6040.636
```

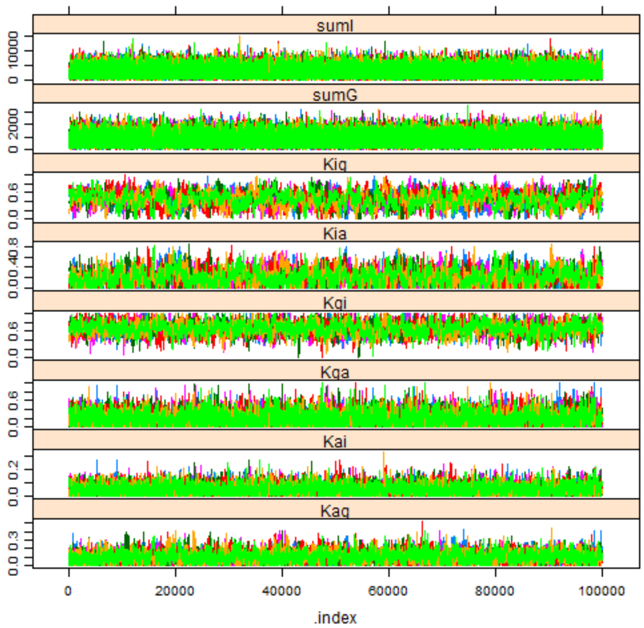
```
# windows() # dev.off()
gelman.diag(chains, autoburnin = FALSE)  #This is R-hat; the burnin period is manually specified above,
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## Kag           1       1.00
## Kai           1       1.00
## Kga           1       1.00
## Kgi           1       1.00
## Kia           1       1.01
## Kig           1       1.01
## sumG          1       1.00
## sumI          1       1.00
##
## Multivariate psrf
##
## 1
```

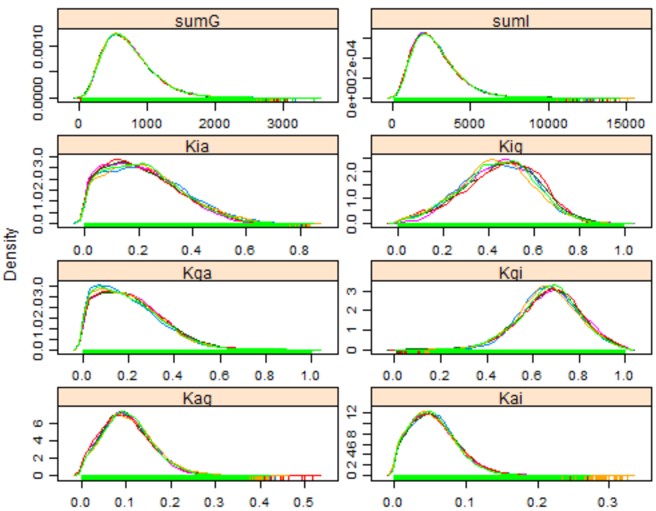
```
effectiveSize(chains)  #Sample size adjusted for autocorrelation
```

```
##      Kag     Kai     Kga     Kgi     Kia     Kig    sumG    sumI
##   3676.9  6295.0  5322.9  2272.7  1557.8   892.2 70069.6 47044.0
```

```
xyplot(chains)  #Needs at least two parameters; else throws an error.
```




```
densityplot(chains)
```




```
# gelman.plot(chains) print(rbind(paste('estimated mu: ',
# condensed$statistics['mu0', 'Mean']), paste('observed mean:', mean(y,
# na.rm=T))))
elapsed
```

```
## Time difference of 10.32 mins
```

Cohort: 1983

```
cohortYear <- 1983
```

```
require(rjags)



if (Sys.info()["nodename"] == "MICKEY") pathDirectory <- "F:/Users/wibeasley/Documents/Consulting/EmosaM
# pathDirectory <-
# 'F:/Users/wibeasley/Documents/Consulting/EmosaMcmc/Dev/EMOSA/OneShot_Only1984Diffusion'
if (Sys.info()["nodename"] == "MERKANEZ-PC") pathDirectory <- "F:/Users/wibeasley/Documents/SSuccess/Int

# pathModel <- file.path(pathDirectory,
# 'DiffusionOnly/DiffusionGauss.bugs') pathModel <-
# file.path(pathDirectory, 'DiffusionOnly/DiffusionLogit.bugs')
pathModel <- file.path(pathDirectory, "DiffusionOnly/DiffusionBeta.bugs")
pathData <- file.path(pathDirectory, "Data/SummaryBirthYearByTime.csv")
# curve(dbeta(x, 1,1)) curve(dbeta(x, 10,10)) curve(dlogis(x, location =
# .25, scale = 1), xlim=c(-5, 5))


ds <- read.csv(pathData, stringsAsFactors = FALSE)
ds <- ds[ds$byear == cohortYear, ]  #Select only the desired cohort
ds <- ds[order(ds$time), ]  #Sort, just, to make sure values will be passed to JAGS in the correct order

pg <- ds$ProportionGoers
pi <- ds$ProportionIrregulars
pa <- ds$ProportionAbsentees

# Proportion of Goers, of Irregulars, or Nongoers (or absentees) {Check
# these with data; I may have messed up the order} For the 1984 cohort pg
# <- c(0.401088929, 0.340290381, 0.249546279, 0.218693285, 0.180580762,
# 0.167876588, 0.157894737, 0.158802178, 0.161524501) pi <- c(0.233212341,
# 0.256805808, 0.288566243, 0.305807623, 0.27676951, 0.270417423,
# 0.229582577, 0.250453721, 0.237749546) pa <- c(0.36569873, 0.402903811,
# 0.461887477, 0.475499093, 0.542649728, 0.561705989, 0.612522686,
# 0.590744102, 0.600725953)
timeCount <- length(pg)
if (length(pi) != timeCount) stop("The proportions have a different number of time points.")
if (length(pa) != timeCount) stop("The proportions have a different number of time points.")
mean(c(pg, pi, pa))
```

```
## [1] 0.3333
```

```
jagsData <- list(pg = pg, pi = pi, pa = pa, timeCount = timeCount)

# parameters <- c('mu')
parametersToTrack <- c("Kgi", "Kga", "Kig", "Kia", "Kag", "Kai", "sumG", "sumI")
# parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia', 'Kag', 'Kai', 'sumG',
# 'sumI', 'sumA') parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia',
# 'Kag', 'Kai', 'sigmaG', 'sigmaI') inits <- function(){
# list(Kgi=rnorm(1), Kga=rnorm(1), Kig=rnorm(1), Kia=rnorm(1),
# Kag=rnorm(1), Kai=rnorm(1)) }

countChains <- 6  #3 #6
countIterations <- 1e+05

startTime <- Sys.time()

jagsModel <- jags.model(file = pathModel, data = jagsData, n.chains = countChains)  #, inits=inits)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 183
##
## Initializing model
```

```
# print(jagsModel) update(jagsModel, 1000) #modifies the original object
# and returns NULL
dic <- dic.samples(jagsModel, n.iter = countIterations)
dic
```

```
## Mean deviance:  -106
## penalty 8.7
## Penalized deviance: -97
```

```
# mcarray <- jags.samples(model=jagsModel, c('mu'),
# n.iter=countIterations) #If I understand correctly, the following line
# is similar, but better
chains <- coda.samples(jagsModel, variable.names = parametersToTrack, n.iter = countIterations)  # updat
elapsed <- Sys.time() - startTime
(condensed <- summary(chains))
```

```
##
## Iterations = 101001:201000
## Thinning interval = 1
## Number of chains = 6
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean       SD Naive SE Time-series SE
## Kag  7.33e-02 1.96e-02 2.53e-05       4.24e-04
## Kai  9.27e-02 5.60e-02 7.24e-05       8.11e-04
## Kga  6.83e-02 5.77e-02 7.45e-05       1.49e-03
## Kgi  3.06e-01 6.15e-02 7.94e-05       1.58e-03
## Kia  3.55e-01 1.31e-01 1.69e-04       2.47e-03
## Kig  8.71e-02 5.28e-02 6.82e-05       1.23e-03
## sumG 9.86e+03 5.25e+03 6.78e+00       3.71e+01
## sumI 1.09e+03 5.63e+02 7.27e-01       4.62e+00
##
## 2. Quantiles for each variable:
##
##           2.5%      25%      50%      75%    97.5%
## Kag  2.91e-02 6.16e-02 7.50e-02 8.70e-02 1.07e-01
## Kai  7.14e-03 5.06e-02 8.69e-02 1.26e-01 2.19e-01
## Kga  2.25e-03 2.41e-02 5.38e-02 9.70e-02 2.18e-01
## Kgi  1.56e-01 2.74e-01 3.15e-01 3.48e-01 4.02e-01
## Kia  1.11e-01 2.65e-01 3.50e-01 4.39e-01 6.27e-01
## Kig  7.11e-03 4.79e-02 8.04e-02 1.18e-01 2.09e-01
## sumG 2.41e+03 6.01e+03 8.93e+03 1.27e+04 2.25e+04
## sumI 2.84e+02 6.73e+02 9.88e+02 1.39e+03 2.44e+03
```

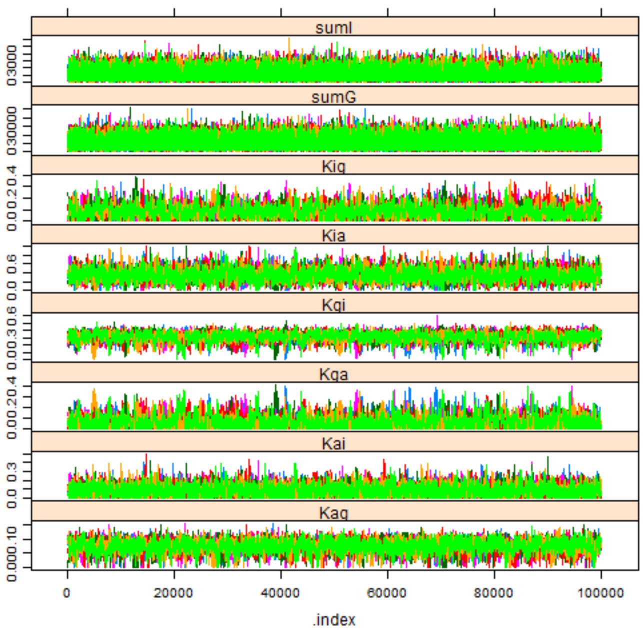
```
# windows() # dev.off()
gelman.diag(chains, autoburnin = FALSE)  #This is R-hat; the burnin period is manually specified above,
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## Kag           1          1
## Kai           1          1
## Kga           1          1
## Kgi           1          1
## Kia           1          1
## Kig           1          1
## sumG          1          1
## sumI          1          1
##
## Multivariate psrf
##
## 1
```

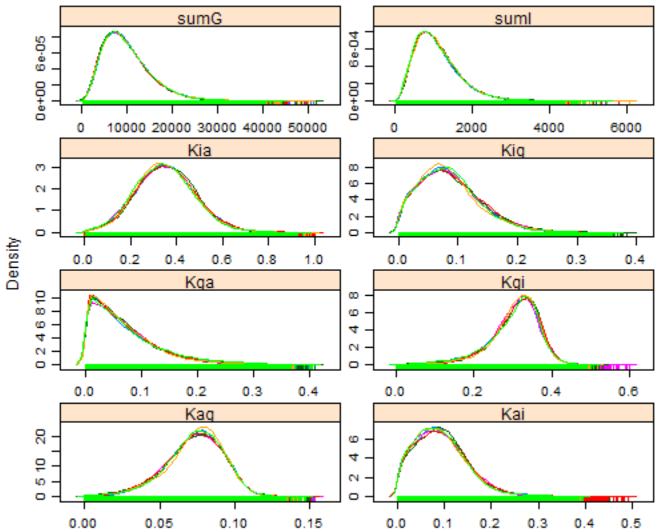
```
effectiveSize(chains)  #Sample size adjusted for autocorrelation
```

```
##   Kag   Kai   Kga   Kgi   Kia   Kig  sumG  sumI
##  3461  5351  2291  2456  3902  2552 39536 31796
```

```
xyplot(chains)  #Needs at least two parameters; else throws an error.
```




```
densityplot(chains)
```




```
# gelman.plot(chains) print(rbind(paste('estimated mu: ',
# condensed$statistics['mu0', 'Mean']), paste('observed mean:', mean(y,
# na.rm=T))))
elapsed
```

```
## Time difference of 9.871 mins
```

Cohort: 1984

```
cohortYear <- 1984
```

```
require(rjags)



if (Sys.info()["nodename"] == "MICKEY") pathDirectory <- "F:/Users/wibeasley/Documents/Consulting/EmosaM
# pathDirectory <-
# 'F:/Users/wibeasley/Documents/Consulting/EmosaMcmc/Dev/EMOSA/OneShot_Only1984Diffusion'
if (Sys.info()["nodename"] == "MERKANEZ-PC") pathDirectory <- "F:/Users/wibeasley/Documents/SSuccess/Int

# pathModel <- file.path(pathDirectory,
# 'DiffusionOnly/DiffusionGauss.bugs') pathModel <-
# file.path(pathDirectory, 'DiffusionOnly/DiffusionLogit.bugs')
pathModel <- file.path(pathDirectory, "DiffusionOnly/DiffusionBeta.bugs")
pathData <- file.path(pathDirectory, "Data/SummaryBirthYearByTime.csv")
# curve(dbeta(x, 1,1)) curve(dbeta(x, 10,10)) curve(dlogis(x, location =
# .25, scale = 1), xlim=c(-5, 5))


ds <- read.csv(pathData, stringsAsFactors = FALSE)
ds <- ds[ds$byear == cohortYear, ]  #Select only the desired cohort
ds <- ds[order(ds$time), ]  #Sort, just, to make sure values will be passed to JAGS in the correct order

pg <- ds$ProportionGoers
pi <- ds$ProportionIrregulars
pa <- ds$ProportionAbsentees

# Proportion of Goers, of Irregulars, or Nongoers (or absentees) {Check
# these with data; I may have messed up the order} For the 1984 cohort pg
# <- c(0.401088929, 0.340290381, 0.249546279, 0.218693285, 0.180580762,
# 0.167876588, 0.157894737, 0.158802178, 0.161524501) pi <- c(0.233212341,
# 0.256805808, 0.288566243, 0.305807623, 0.27676951, 0.270417423,
# 0.229582577, 0.250453721, 0.237749546) pa <- c(0.36569873, 0.402903811,
# 0.461887477, 0.475499093, 0.542649728, 0.561705989, 0.612522686,
# 0.590744102, 0.600725953)
timeCount <- length(pg)
if (length(pi) != timeCount) stop("The proportions have a different number of time points.")
if (length(pa) != timeCount) stop("The proportions have a different number of time points.")
mean(c(pg, pi, pa))
```

```
## [1] 0.3333
```

```
jagsData <- list(pg = pg, pi = pi, pa = pa, timeCount = timeCount)

# parameters <- c('mu')
parametersToTrack <- c("Kgi", "Kga", "Kig", "Kia", "Kag", "Kai", "sumG", "sumI")
# parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia', 'Kag', 'Kai', 'sumG',
# 'sumI', 'sumA') parametersToTrack <- c('Kgi', 'Kga', 'Kig', 'Kia',
# 'Kag', 'Kai', 'sigmaG', 'sigmaI') inits <- function(){
# list(Kgi=rnorm(1), Kga=rnorm(1), Kig=rnorm(1), Kia=rnorm(1),
# Kag=rnorm(1), Kai=rnorm(1)) }

countChains <- 6  #3 #6
countIterations <- 1e+05

startTime <- Sys.time()

jagsModel <- jags.model(file = pathModel, data = jagsData, n.chains = countChains)  #, inits=inits)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 185
##
## Initializing model
```

```
# print(jagsModel) update(jagsModel, 1000) #modifies the original object
# and returns NULL
dic <- dic.samples(jagsModel, n.iter = countIterations)
dic
```

```
## Mean deviance:  -83.2
## penalty 7.72
## Penalized deviance: -75.5
```

```
# mcarray <- jags.samples(model=jagsModel, c('mu'),
# n.iter=countIterations) #If I understand correctly, the following line
# is similar, but better
chains <- coda.samples(jagsModel, variable.names = parametersToTrack, n.iter = countIterations)  # updat
elapsed <- Sys.time() - startTime
(condensed <- summary(chains))
```

```
##
## Iterations = 101001:201000
## Thinning interval = 1
## Number of chains = 6
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD Naive SE Time-series SE
## Kag    0.0509   0.0239 3.09e-05       0.000163
## Kai    0.0918   0.0591 7.63e-05       0.000660
## Kga    0.0974   0.0661 8.54e-05       0.000588
## Kgi    0.2125   0.0621 8.02e-05       0.000576
## Kia    0.2954   0.1420 1.83e-04       0.001657
## Kig    0.0646   0.0525 6.77e-05       0.000455
## sumG 971.8325 506.7934 6.54e-01       1.745397
## sumI 647.0774 326.0862 4.21e-01       1.137730
##
## 2. Quantiles for each variable:
##
##          2.5%      25%      50%      75%    97.5%
## Kag  5.89e-03   0.0340   0.0514 6.71e-02 9.77e-02
## Kai  6.25e-03   0.0469   0.0837 1.26e-01 2.29e-01
## Kga  4.97e-03   0.0450   0.0873 1.38e-01 2.49e-01
## Kgi  8.01e-02   0.1732   0.2162 2.55e-01 3.25e-01
## Kia  5.36e-02   0.1924   0.2846 3.85e-01 6.03e-01
## Kig  2.11e-03   0.0229   0.0520 9.42e-02 1.92e-01
## sumG 2.52e+02 601.5754 883.0996 1.24e+03 2.20e+03
## sumI 1.74e+02 408.1151 592.9147 8.27e+02 1.43e+03
```

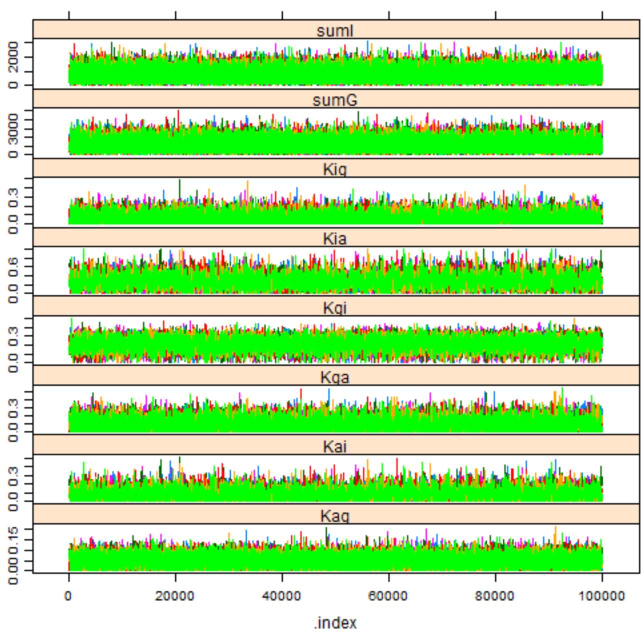
```
# windows() # dev.off()
gelman.diag(chains, autoburnin = FALSE)  #This is R-hat; the burnin period is manually specified above,
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## Kag           1          1
## Kai           1          1
## Kga           1          1
## Kgi           1          1
## Kia           1          1
## Kig           1          1
## sumG          1          1
## sumI          1          1
##
## Multivariate psrf
##
## 1
```

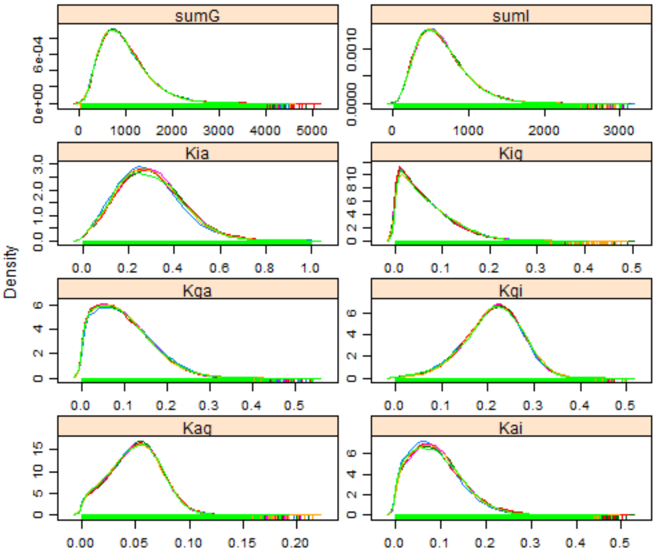
```
effectiveSize(chains)  #Sample size adjusted for autocorrelation
```

```
##   Kag   Kai   Kga   Kgi   Kia   Kig  sumG  sumI
## 21849  7708 12423 10730  5976 13241 91591 96004
```

```
xyplot(chains)  #Needs at least two parameters; else throws an error.
```




```
densityplot(chains)
```




```
# gelman.plot(chains) print(rbind(paste('estimated mu: ',
# condensed$statistics['mu0', 'Mean']), paste('observed mean:', mean(y,
# na.rm=T))))
elapsed
```

```
## Time difference of 10.54 mins
```