Functional ANOVA Example

November 13, 2018

Simulate Three-Group One-Way Functional ANOVA Data

Three-group one-way functional ANOVA data were simulated using methods described in the paper "Fast Function-on-Scalar Regression with Penalized Basis Expansions" from Reiss et al:

$$
\begin{aligned}
y_i(t) &= \mu(t) + \beta_{gp(i)}(t) + \epsilon_i(t) \\
\mu(t) &= 0.4 \arctan(10x - 5) + 0.6 \\
\beta_1(t) &= -0.5e^{-10t} - 0.04\sin(8t) - 0.3t + 0.5 \\
\beta_2(t) &= -(t - 0.5)^2 - 0.15\sin(13t) \\
\beta_3(t) &= -\beta_1(t) - \beta_2(t)
\end{aligned}
$$

where $t = m/200$ for $m = 0, \ldots, 200$ and the error functions, $\epsilon_i(t)$, were simulated from a mean-zero Gaussian process with coveriance $V(s,t) = \sigma_1 0.15^{|s-t|} + \sigma_2 \delta_{st}$ where $\delta_{st} = 1$ if $s = t$ and 0 otherwise. For the simulations, N=10 curves were simulated for each of the 3 groups, $\sigma_1 = 0.15$, and $\sigma_2 = 0.05$. So, we have 6030 observations from 30 curves with 10 curves per group.

```
################################
## Create the "True" Dataset ##
################################

## Model
## y_i(t) = mu(t) + b_gp(i)(t) + e_i(t) for t in [0,1]

## Values
t = 0:200/200     ## Evaluation time points
g = 3             ## Number of groups
N = 10            ## Number of curves per group, N = 10 or 60

## Coefficients from the paper
mu.t = 0.4*atan(10*t - 5) + 0.6
b1.t = -0.5*exp(-10*t) - 0.04*sin(8*t) - 0.3*t + 0.5
b2.t = -(t - 0.5)^2 - 0.15*sin(13*t)
b3.t = -b1.t - b2.t

## Mean functions by group
mean_1 = mu.t + b1.t
mean_2 = mu.t + b2.t
mean_3 = mu.t + b3.t

## Create the "true" dataset
true.data = data.frame(Y_True = c(mean_1, mean_2, mean_3),
                       Time = rep(t, 3),
                       Group = as.factor(c(rep(1, length(t)),
                                           rep(2, length(t)),
                                           rep(3, length(t)))) )
```

Note that the X variable, Time, was centered and scaled.

```
############################
## Generate Simulated Data ##
############################

## Create a stacked dataframe for the simulated data
dat = data.frame(Y_True = c(rep(mean_1, N),
                            rep(mean_2, N),
                            rep(mean_3, N)),
                 Time = rep(t, N*g),
                 Group = as.factor( c(rep(1, N*length(t)),
                                      rep(2, N*length(t)),
                                      rep(3, N*length(t))) ),
                 ID = factor( rep(1:(N*g),
                                  length(t))[ order(rep(1:(N*g), length(t))) ] ) )
rm(mean_1, mean_2, mean_3)

## Define values
sigma_1 = 0.15    ## Simulations used 0.05 and 0.15
sigma_2 = 0.05    ## 0.05 for all simulations

## Calculate the covariance matrix
grid = expand.grid(t, t)
cov.matrix = apply(X = grid, MARGIN = 1,
                   FUN = function(x){ ((sigma_1^2)*0.15^(abs(x[1] - x[2]) )) +
                       (sigma_2^2)*as.numeric(x[1] == x[2]) } )
cov.matrix = matrix(data = cov.matrix, ncol = length(t), nrow = length(t))
rm(grid)

## Generate errors
seed = 87654321
set.seed(seed)
errors = mvrnorm(n = N*g, mu = rep(0, length(t)), Sigma = cov.matrix)

## Add to the dataframe
dat$Errors = c( errors[1:(N*g),] )

## Calculate simulated Y's
dat$Y_Sim = (dat$Y_True + dat$Errors)
rm(cov.matrix, errors, sigma_1, sigma_2)

## Pre-smooth data
dat$Y_Smooth = rep(NA, N*g*length(t))
for(i in 1:(N*g)){
  start = (i-1)*length(t) + 1
  end = i*length(t)
  dat$Y_Smooth[start:end] = with(dat,
                                 smooth.spline(Time[start:end], Y_Sim[start:end],
                                               nknots = 20))$y
}
rm(i, start, end, t, mu.t)

## Center and scale the time variable
dat$Time_Standardized = scale(dat$Time)
```

```
## Calculate higher-order time terms
dat$Time_Stan_2 = dat$Time_Standardized^2
dat$Time_Stan_3 = dat$Time_Standardized^3
dat$Time_Stan_4 = dat$Time_Standardized^4
dat$Time_Stan_5 = dat$Time_Standardized^5
```

Figure 1: The true effect functions along with the simulated data.

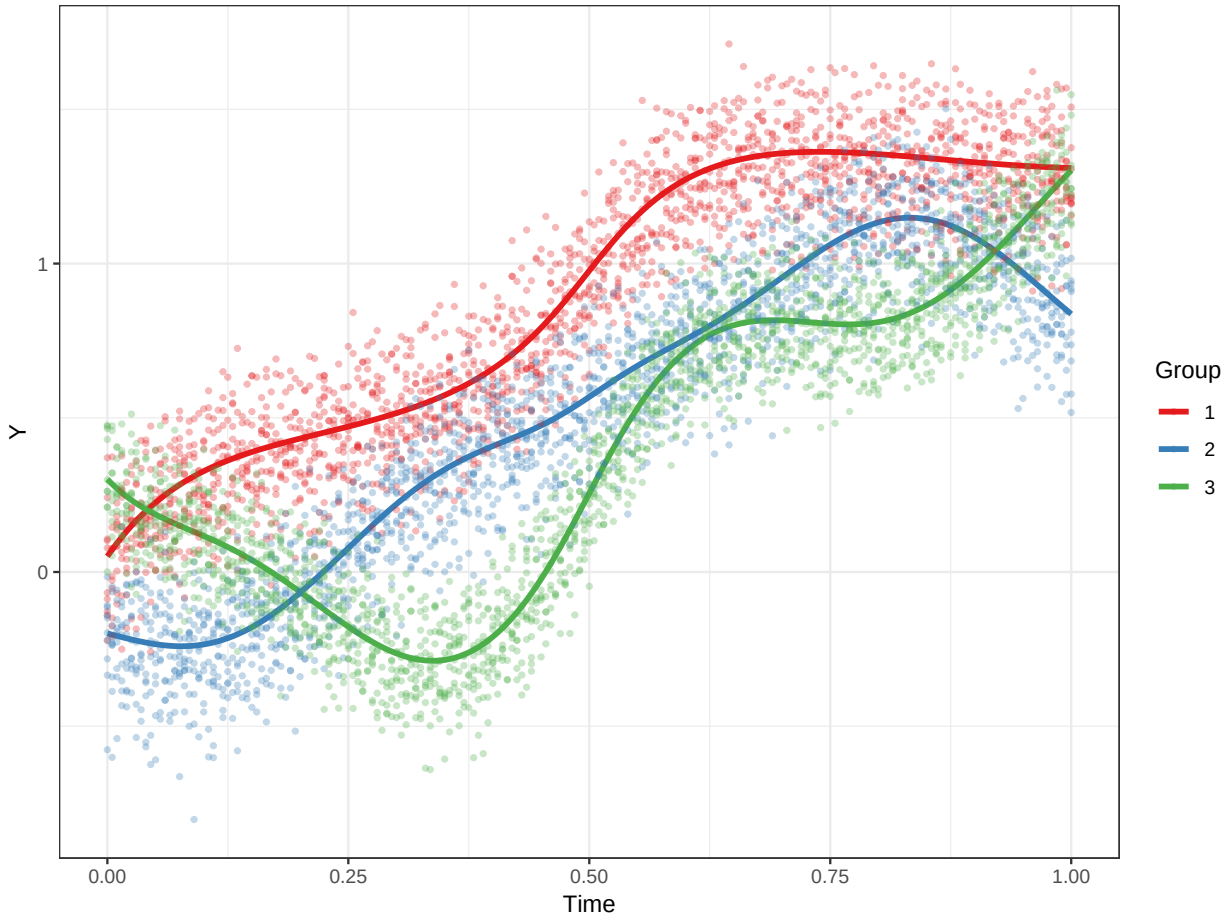


Figure 2: The smoothed subject-specific curves from the simulated data.

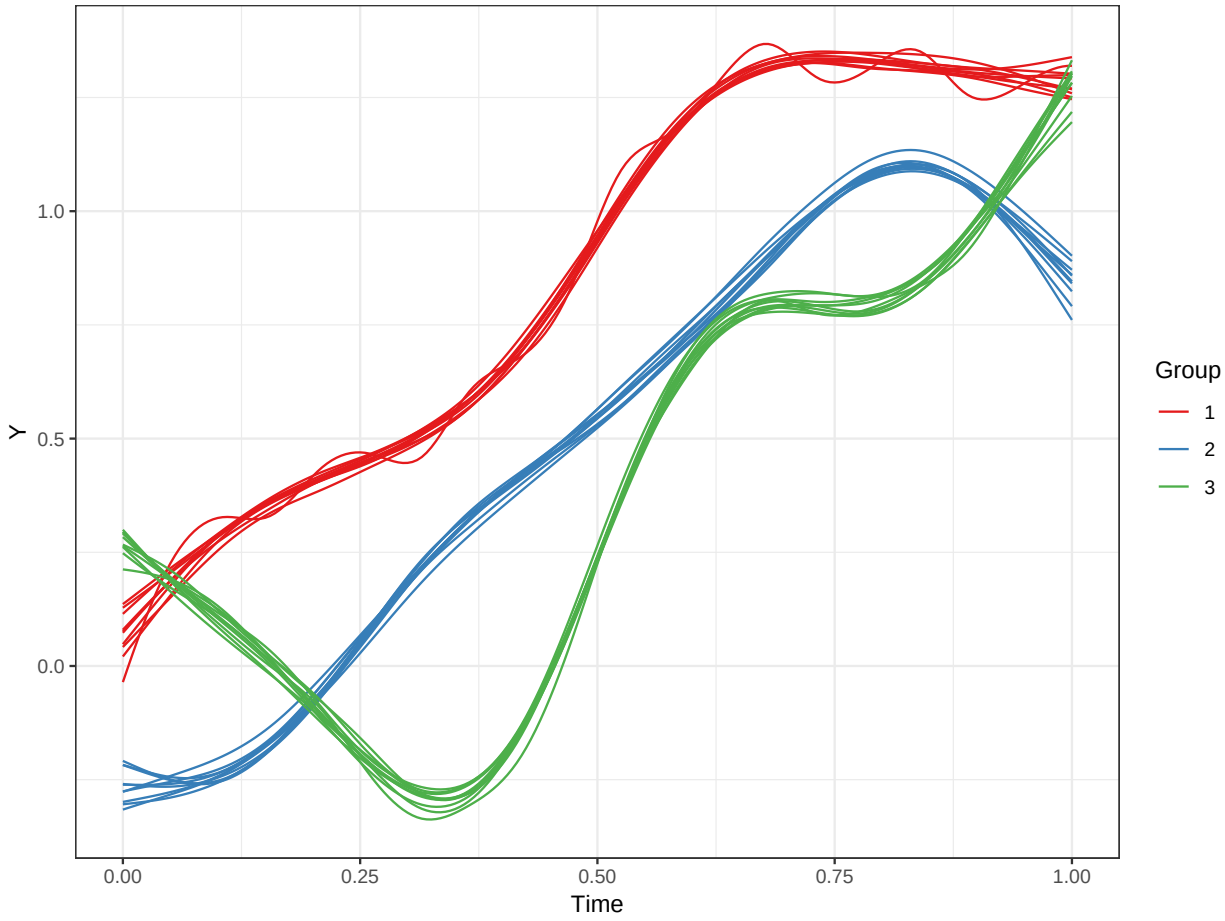
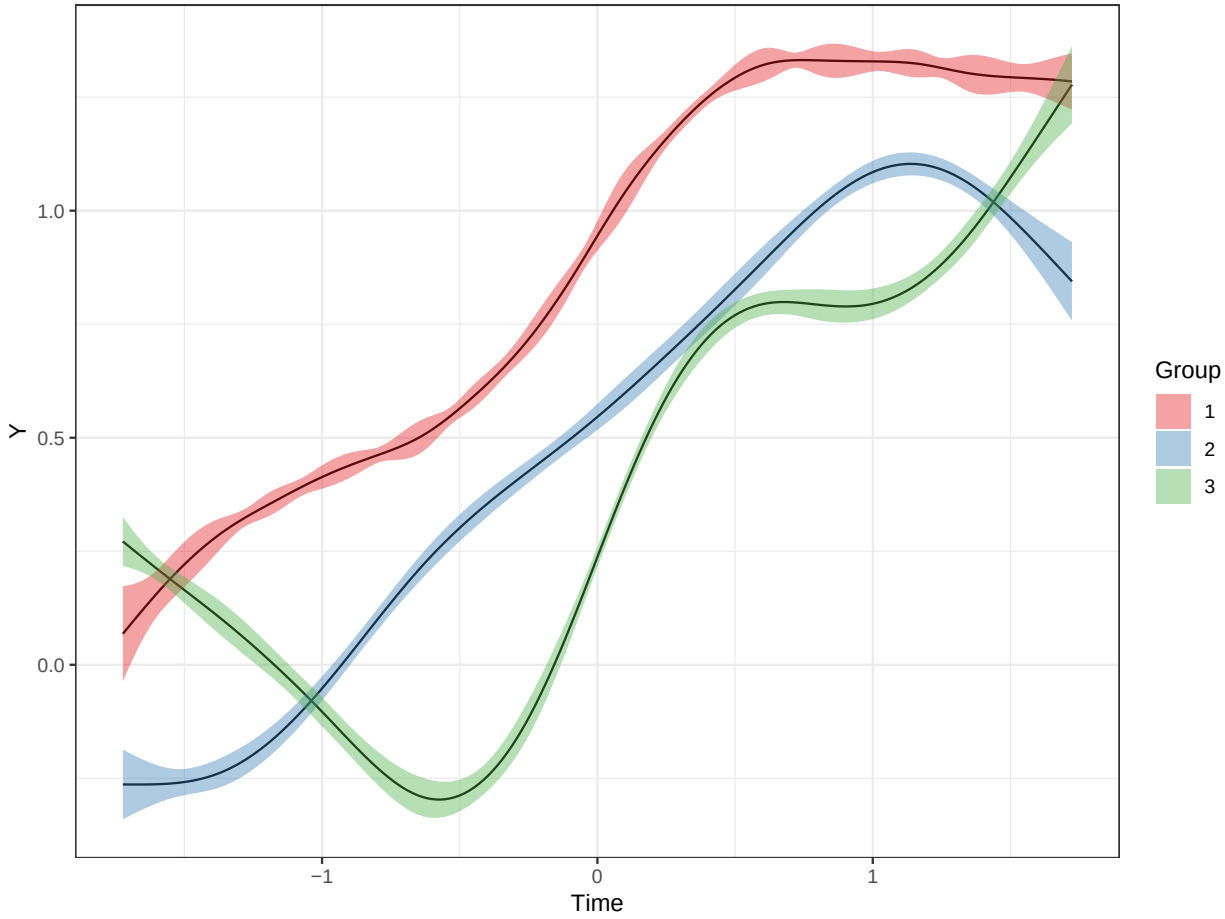


Figure 3: The group-level mean functions for the smoothed simulated data ± 2 point-wise standard deviations.

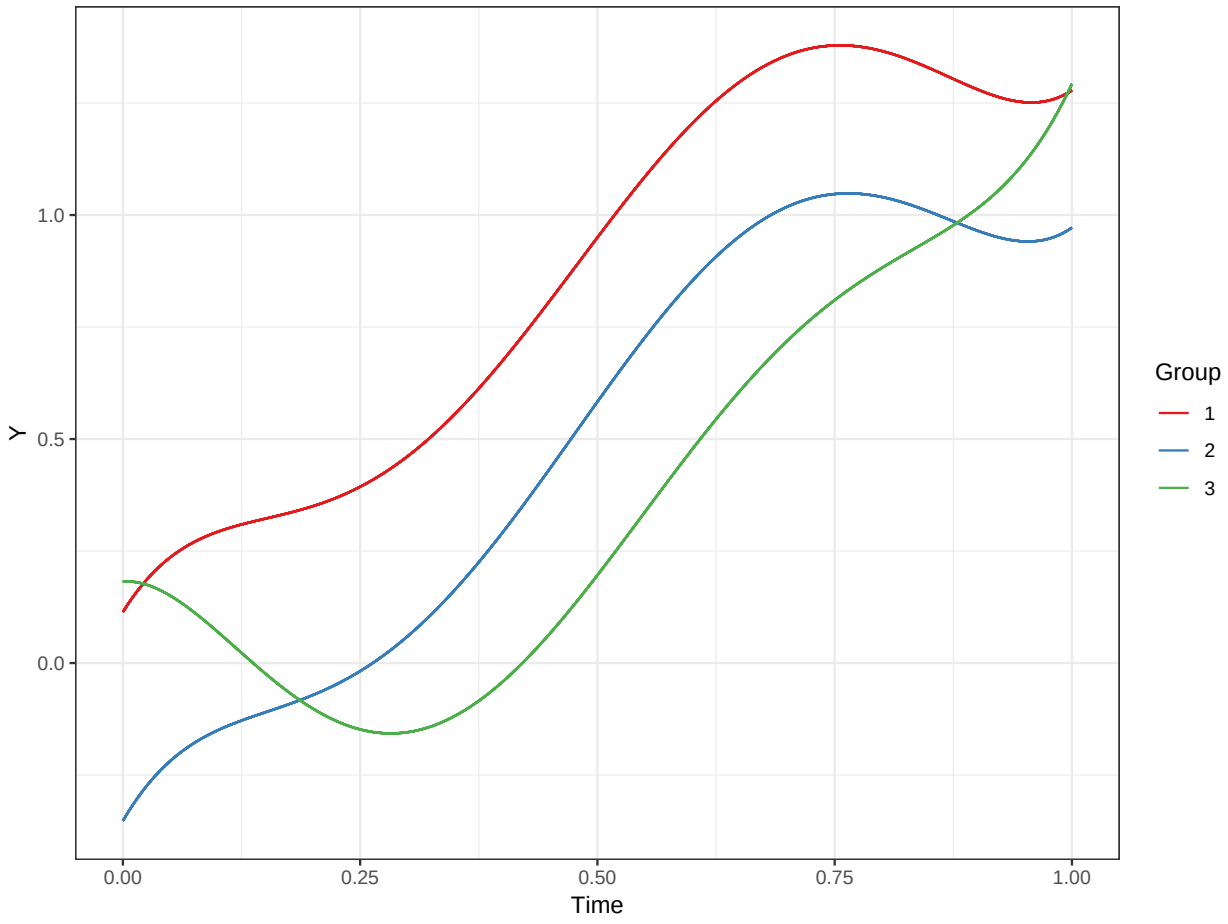


Analyze Using lm()

Fit a 5th-order linear model with time-by-group and Time^2-by-group interactions using lm():

```
## Fit the model
mod = lm(Y_Smooth ~ Group + Time_Standardized + Time_Stan_2 + Time_Stan_3 +
           Time_Stan_4 + Time_Stan_5 +
           Group*Time_Standardized + Group*Time_Stan_2, data = dat)

## Calculate the fitted values
dat$Y_Smooth_Fitted_lm = fitted(mod)
```

Figure 4: The group-level mean functions from lm() using a 5th-order linear model with time-by-group and Time^2-by-group interactions.

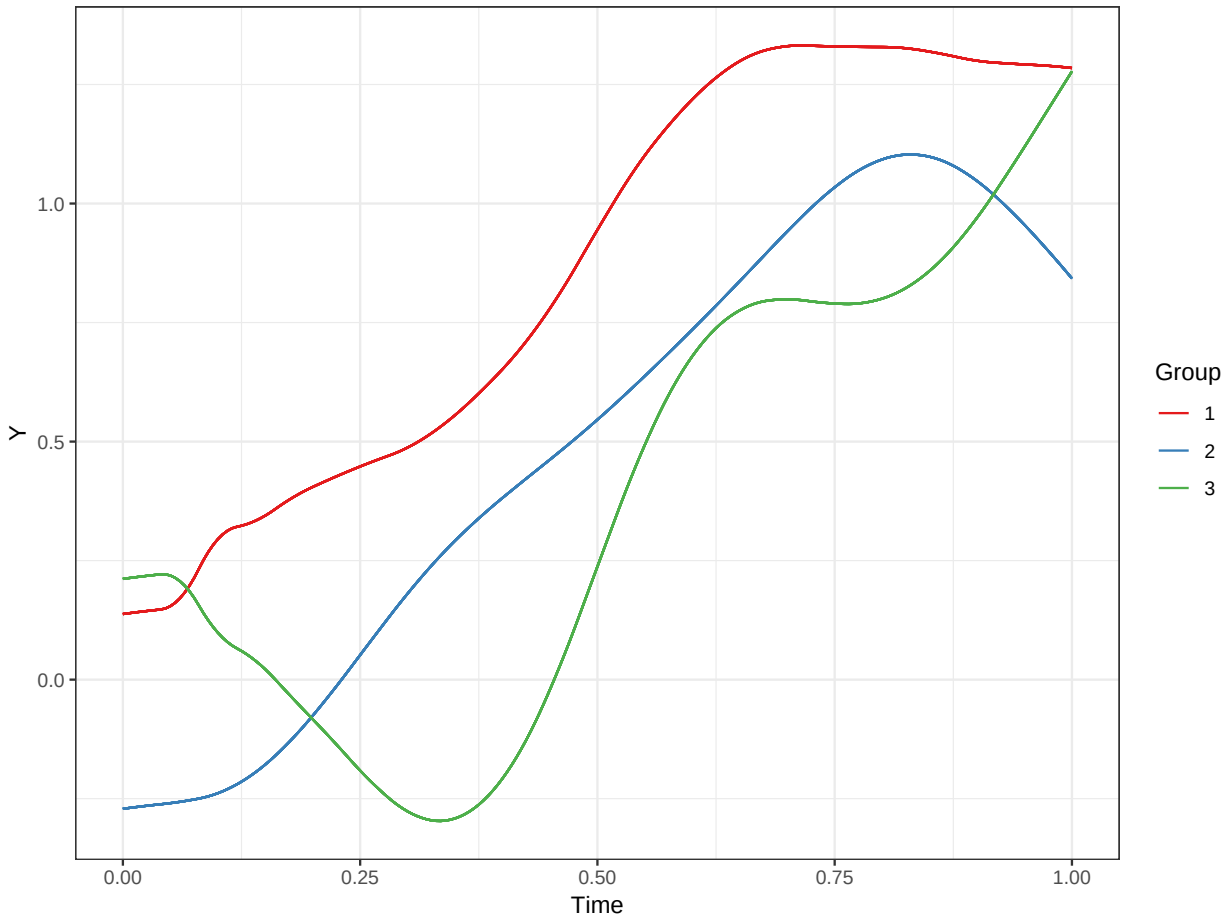


Analyze Using Penalized Splines and lme()

Fit a penalized spline using lme():

```
## Penalized basis functions
K = 24
qtiles = seq(0, 1, length = K + 2)[-c(1, K + 2)]
knots = quantile(dat$Time_Standardized, qtiles)
random.basis = cbind(sapply(knots,
                            function(k){
                              I((dat$Time_Standardized - k)^2)*(dat$Time_Standardized - k > 0)
                              }
                            )
                     )
rm(qtiles)

## Define the fixed basis
fixed.basis = cbind(1,
                    as.numeric(dat$Group == "2"),
                    as.numeric(dat$Group == "3"),
                    dat$Time_Standardized,
                    dat$Time_Stan_2)

## Partition the random.basis matrix
random.basis.1 = as.numeric(dat$Group == "1")*random.basis
random.basis.2 = as.numeric(dat$Group == "2")*random.basis
random.basis.3 = as.numeric(dat$Group == "3")*random.basis

## Define Y
Y = dat$Y_Smooth

## Fit the model
group.1 = group.2 = group.3 = rep(1, nrow(dat))
mod = lme(Y ~ fixed.basis - 1, random = list(group.1 = pdIdent(~ random.basis.1 - 1),
                                             group.2 = pdIdent(~ random.basis.2 - 1),
                                             group.3 = pdIdent(~ random.basis.3 - 1)) )
rm(group.1, group.2, group.3)

## Extract the model coefficients
coefs = c( unlist(mod$coefficients$fixed), unlist(mod$coefficients$random) )

## Extract the lambda estimates
#var.e2 = mod$sigma^2
#var.u2 = getVarCov(mod)[1,1]

## Calculate the fitted values
dat$Y_Smooth_Fitted_lme = as.vector( cbind(fixed.basis,
                                           random.basis.1,
                                           random.basis.2,
                                           random.basis.3) %*% coefs )

## Remove R objects
rm(fixed.basis, random.basis, coefs, knots, Y)
```

Figure 5: The group-level mean functions using penalized splines and lme().



Analyze Using JAGS

Fit a 5th-order linear model with time-by-group and Time^2-by-group interactions using JAGS:

```
## Define values
n = nrow(dat)
Y = dat$Y_Smooth
X = c(dat$Time_Standardized)
X_2 = c(dat$Time_Stan_2)
X_3 = c(dat$Time_Stan_3)
X_4 = c(dat$Time_Stan_4)
X_5 = c(dat$Time_Stan_5)
G.2 = as.numeric(dat$Group == "2")
G.3 = as.numeric(dat$Group == "3")

## Create the model
model_string <- "model{

# Likelihood
for(i in 1:n){
Y[i]   ~ dnorm(mu[i], inv.var)
mu[i] <- beta[1] + beta[2]*X[i] + beta[3]*X_2[i] + beta[4]*X_3[i] + beta[5]*X_4[i] + beta[6]*X_5[i] + b
}

# Prior for beta
for(j in 1:12){
beta[j] ~ dnorm(0,0.0001)
}

# Prior for the inverse variance
inv.var   ~ dgamma(0.01, 0.01)
sigma     <- 1/sqrt(inv.var)

}"

## Compile the model
model = jags.model(textConnection(model_string),
                   data = list(Y = Y,
                               n = n,
                               X = X,
                               X_2 = X_2,
                               X_3 = X_3,
                               X_4 = X_4,
                               X_5 = X_5,
                               G.2 = G.2,
                               G.3 = G.3),
                   n.chains = 3,
                   n.adapt  = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 6030
##    Unobserved stochastic nodes: 13
```

```
##    Total graph size: 50880
##
## Initializing model
```

```
## Burn-in samples
update(model,
       1000,
       progress.bar = "none")

## Draw additional samples
samp = coda.samples(model,
                    variable.names = c("beta","sigma"),
                    thin = 3,
                    n.iter = 5000,
                    progress.bar = "none")
```

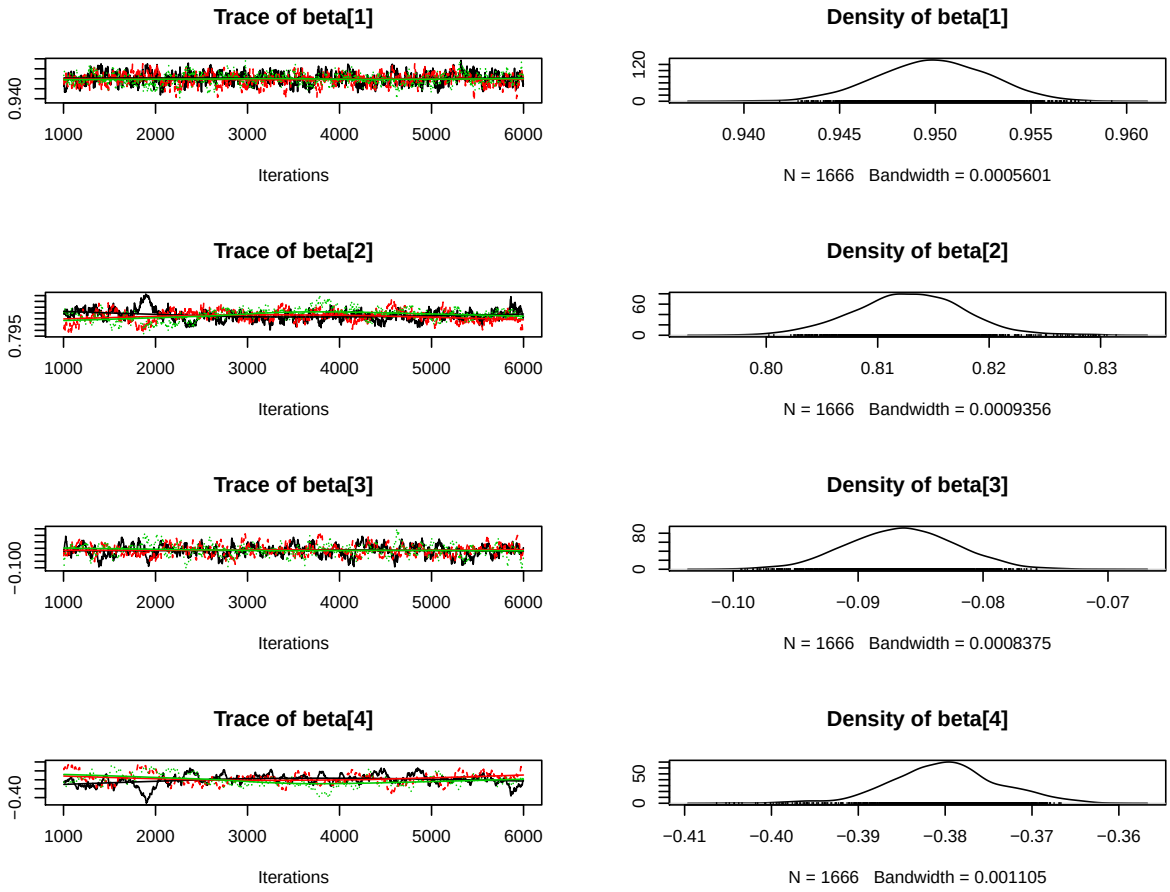
Investigate the fit of the model and extract the fitted values:

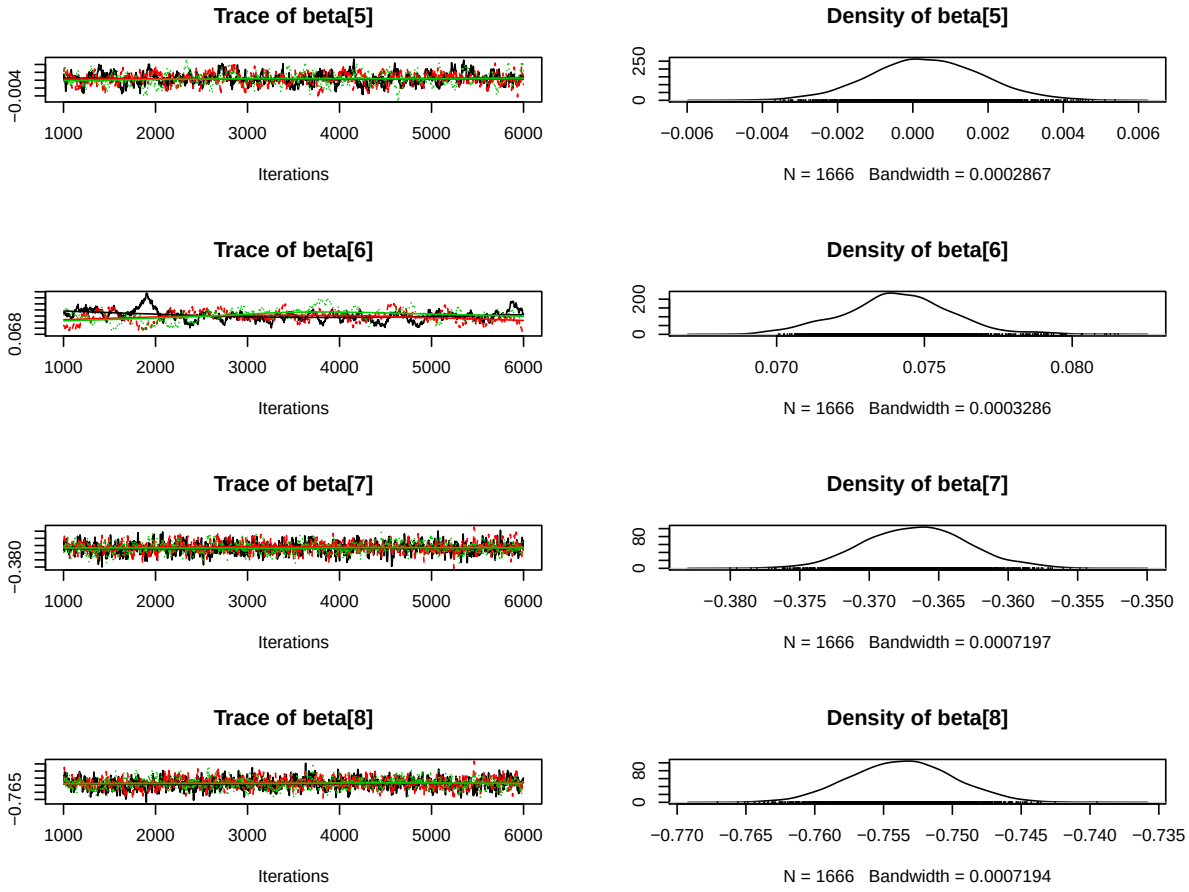
```
## Investigate the model
summary(samp)
```

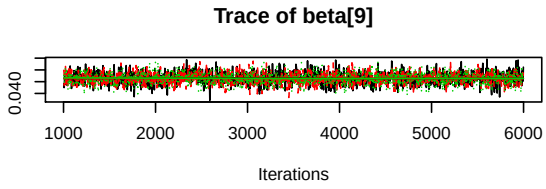
```
##
## Iterations = 1003:5998
## Thinning interval = 3
## Number of chains = 3
## Sample size per chain = 1666
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## beta[1]   0.9498982 0.0029020 4.105e-05      1.292e-04
## beta[2]   0.8128099 0.0049792 7.043e-05      5.054e-04
## beta[3]  -0.0864888 0.0043397 6.139e-05      2.794e-04
## beta[4]  -0.3799715 0.0063077 8.922e-05      7.626e-04
## beta[5]   0.0003549 0.0014858 2.102e-05      7.885e-05
## beta[6]   0.0740752 0.0018439 2.608e-05      2.142e-04
## beta[7]  -0.3665793 0.0037292 5.275e-05      1.126e-04
## beta[8]  -0.7536554 0.0037278 5.273e-05      1.114e-04
## beta[9]   0.0464071 0.0024524 3.469e-05      4.309e-05
## beta[10] -0.0154831 0.0024778 3.505e-05      4.508e-05
## beta[11] -0.0066518 0.0028241 3.995e-05      8.498e-05
## beta[12]  0.2676462 0.0028334 4.008e-05      8.939e-05
## sigma     0.0785762 0.0007096 1.004e-05      1.025e-05
##
## 2. Quantiles for each variable:
##
##               2.5%        25%        50%       75%     97.5%
## beta[1]   0.944181  0.9479650  0.9499019  0.951916  0.955420
## beta[2]   0.802991  0.8096110  0.8128233  0.816107  0.822659
## beta[3]  -0.094905 -0.0894462 -0.0864700 -0.083608 -0.078123
## beta[4]  -0.392670 -0.3838866 -0.3799998 -0.376212 -0.367346
## beta[5]  -0.002596 -0.0006482  0.0003426  0.001362  0.003295
## beta[6]   0.070404  0.0729362  0.0740730  0.075218  0.077851
## beta[7]  -0.373652 -0.3691649 -0.3665483 -0.364094 -0.358944
## beta[8]  -0.760821 -0.7562073 -0.7536427 -0.751145 -0.746443
```

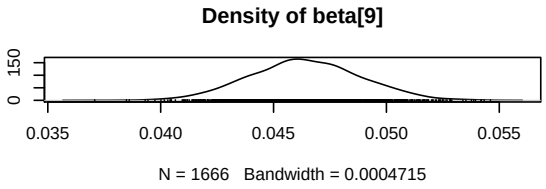
```
## beta[9]   0.041685  0.0447501  0.0463635  0.048024  0.051257
## beta[10] -0.020404 -0.0171321 -0.0154066 -0.013837 -0.010696
## beta[11] -0.012160 -0.0085226 -0.0066474 -0.004767 -0.001147
## beta[12]  0.262254  0.2656973  0.2676661  0.269595  0.273115
## sigma     0.077229  0.0780838  0.0785661  0.079050  0.079953
```

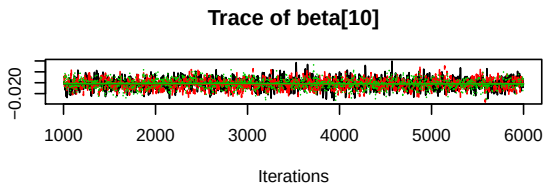
```
plot(samp)
```

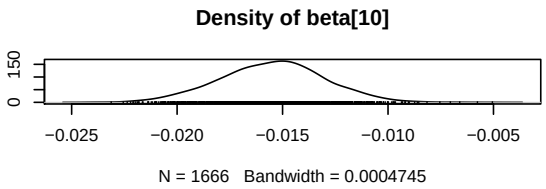


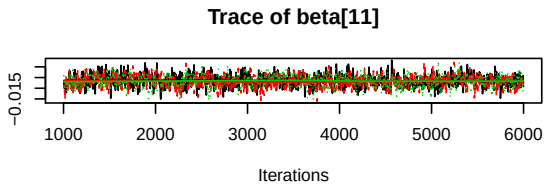

Trace of beta[5]
−0.004
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[5]
0
250
−0.006
−0.004
−0.002
0.000
0.002
0.004
0.006
N = 1666 Bandwidth = 0.0002867
Trace of beta[6]
0.068
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[6]
0
200
0.070
0.075
0.080
N = 1666 Bandwidth = 0.0003286
Trace of beta[7]
−0.380
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[7]
0
80
−0.380
−0.375
−0.370
−0.365
−0.360
−0.355
−0.350
N = 1666 Bandwidth = 0.0007197
Trace of beta[8]
−0.765
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[8]
0
80
−0.770
−0.765
−0.760
−0.755
−0.750
−0.745
−0.740
−0.735
N = 1666 Bandwidth = 0.0007194

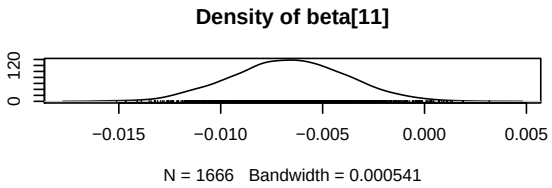

Trace of beta[9]
0.040
1000
2000
3000
4000
5000
6000
Iterations

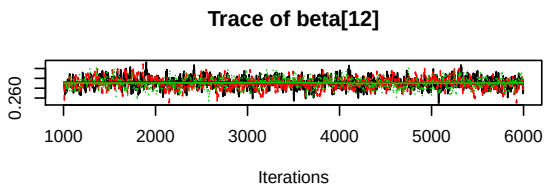

Density of beta[9]
0
150
0.035
0.040
0.045
0.050
0.055
N = 1666 Bandwidth = 0.0004715

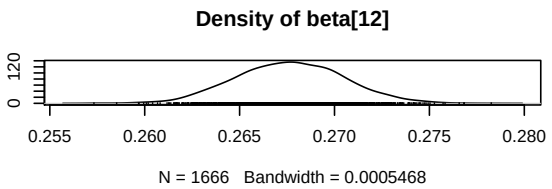

Trace of beta[10]
−0.020
1000
2000
3000
4000
5000
6000
Iterations

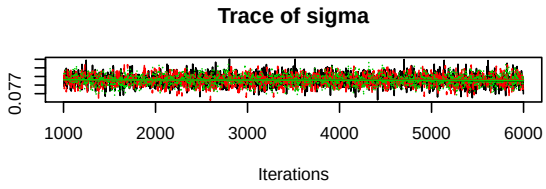

Density of beta[10]
0
150
−0.025
−0.020
−0.015
−0.010
−0.005
N = 1666 Bandwidth = 0.0004745

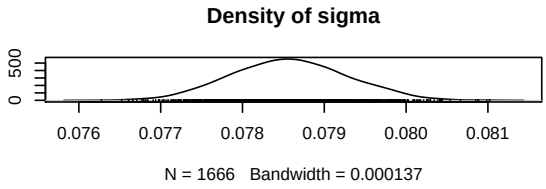

Trace of beta[11]
−0.015
1000
2000
3000
4000
5000
6000
Iterations


Density of beta[11]
0
120
−0.015
−0.010
−0.005
0.000
0.005
N = 1666 Bandwidth = 0.000541


Trace of beta[12]
0.260
1000
2000
3000
4000
5000
6000
Iterations


Density of beta[12]
0
120
0.255
0.260
0.265
0.270
0.275
0.280
N = 1666 Bandwidth = 0.0005468





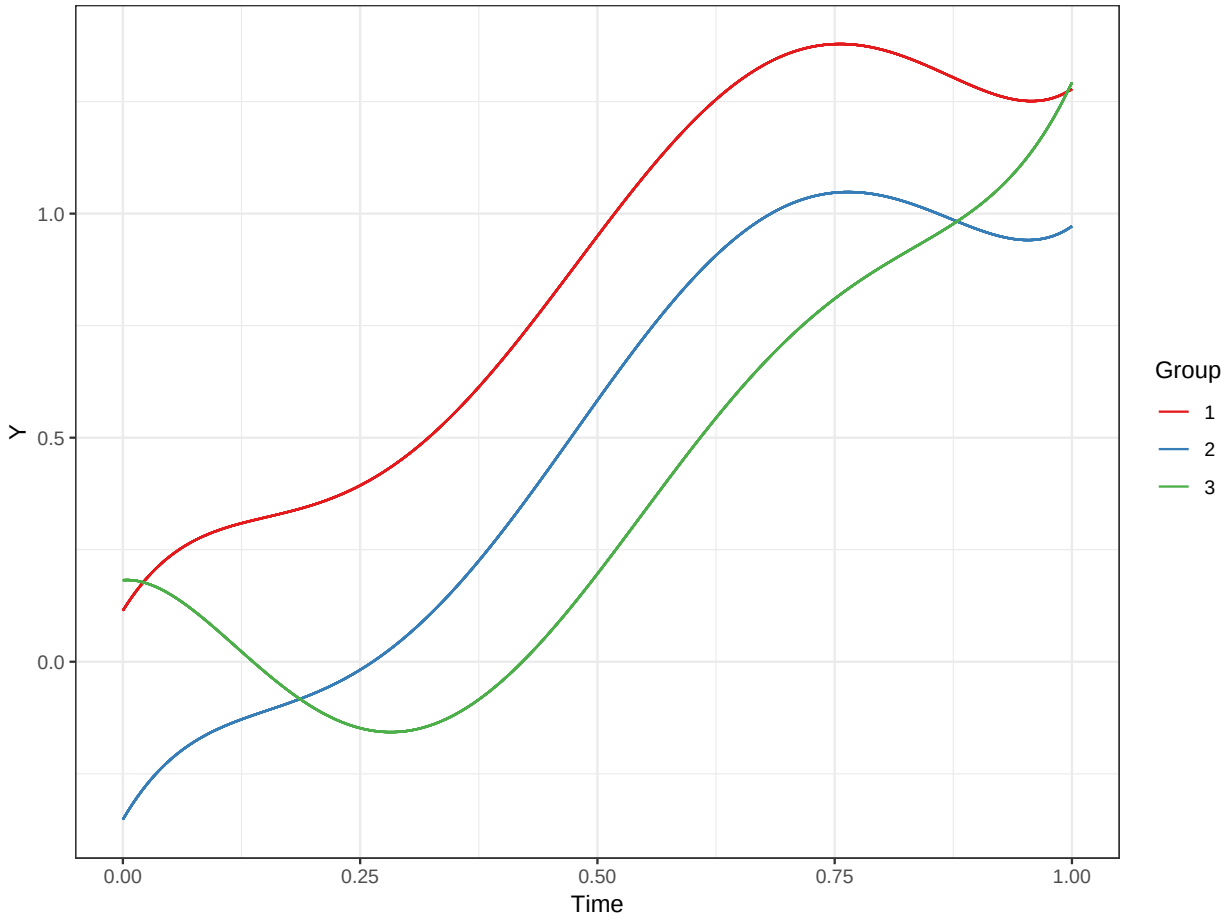

```
## Gelman-Rubin convergence diagnostic
gelman.diag(x = samp,
            confidence = 0.95,
            transform = FALSE,
            autoburnin = FALSE)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta[1]        1.01       1.02
## beta[2]        1.01       1.04
## beta[3]        1.00       1.01
## beta[4]        1.02       1.06
## beta[5]        1.00       1.00
## beta[6]        1.02       1.06
## beta[7]        1.01       1.02
## beta[8]        1.00       1.01
## beta[9]        1.00       1.00
## beta[10]       1.00       1.00
## beta[11]       1.00       1.01
## beta[12]       1.00       1.01
## sigma          1.00       1.00
##
## Multivariate psrf
```

```
##
## 1.02
```

```
## Save the coefficients from the model
coefs = data.frame( summary(samp)$statistics )

## Extract the beta estimates
betas = coefs$Mean[ row.names(coefs) %in% paste0( paste0("beta[",1:12), "]") ]

## Calculate the fitted values
dat$Y_Smooth_Fitted_JAGS_5 = c(t(betas) %*% t(cbind(1, X, X_2, X_3, X_4, X_5,
                                                    G.2, G.3, X*G.2, X*G.3,
                                                    X_2*G.2, X_2*G.3)) )

## Remove R objects
rm(model, samp, coefs, model_string, n, X_3, X_4, X_5, betas)
```

Figure 6: The group-level mean functions from JAGS using a 5th-order linear model with time-by-group and Time^2-by-group interactions.



Analyze Using Penalized Splines and JAGS

Fit a penalized spline using JAGS:

```
# Define values
n = nrow(dat)

## Create the model
model_string <- "model{

# Likelihood
for(i in 1:n){
Y[i]   ~ dnorm(mu[i], inv.var)
mu[i] <- beta[1] + beta[2]*G.2[i] + beta[3]*G.3[i] + beta[4]*X[i] + beta[5]*X_2[i] + alpha[1]*random.bas
}

# Prior for beta
for(j in 1:5){
beta[j] ~ dnorm(0,0.0001)
}

# Prior for alpha
for(k in 1:(3*K)){
alpha[k] ~ dnorm(0,0.0001)
}

# Prior for the inverse variance
inv.var    ~ dgamma(0.01, 0.01)
sigma     <- 1/sqrt(inv.var)

}"

## Compile the model
model = jags.model(textConnection(model_string),
                   data = list(Y = Y,
                               n = n,
                               X = X,
                               X_2 = X_2,
                               G.2 = G.2,
                               G.3 = G.3,
                               K = K,
                               random.basis.1 = random.basis.1,
                               random.basis.2 = random.basis.2,
                               random.basis.3 = random.basis.3),
                   n.chains = 3,
                   n.adapt  = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 6030
##    Unobserved stochastic nodes: 78
##    Total graph size: 472577
##
```

```
## Initializing model
```

```
## Burn-in samples
update(model,
       1000,
       progress.bar = "none")

## Draw additional samples
samp = coda.samples(model,
                    variable.names = c("beta","alpha","sigma"),
                    thin = 3,
                    n.iter = 5000,
                    progress.bar = "none")
```

Investigate the fit of the model and extract the fitted values:

```
## Investigate the model
summary(samp)
```

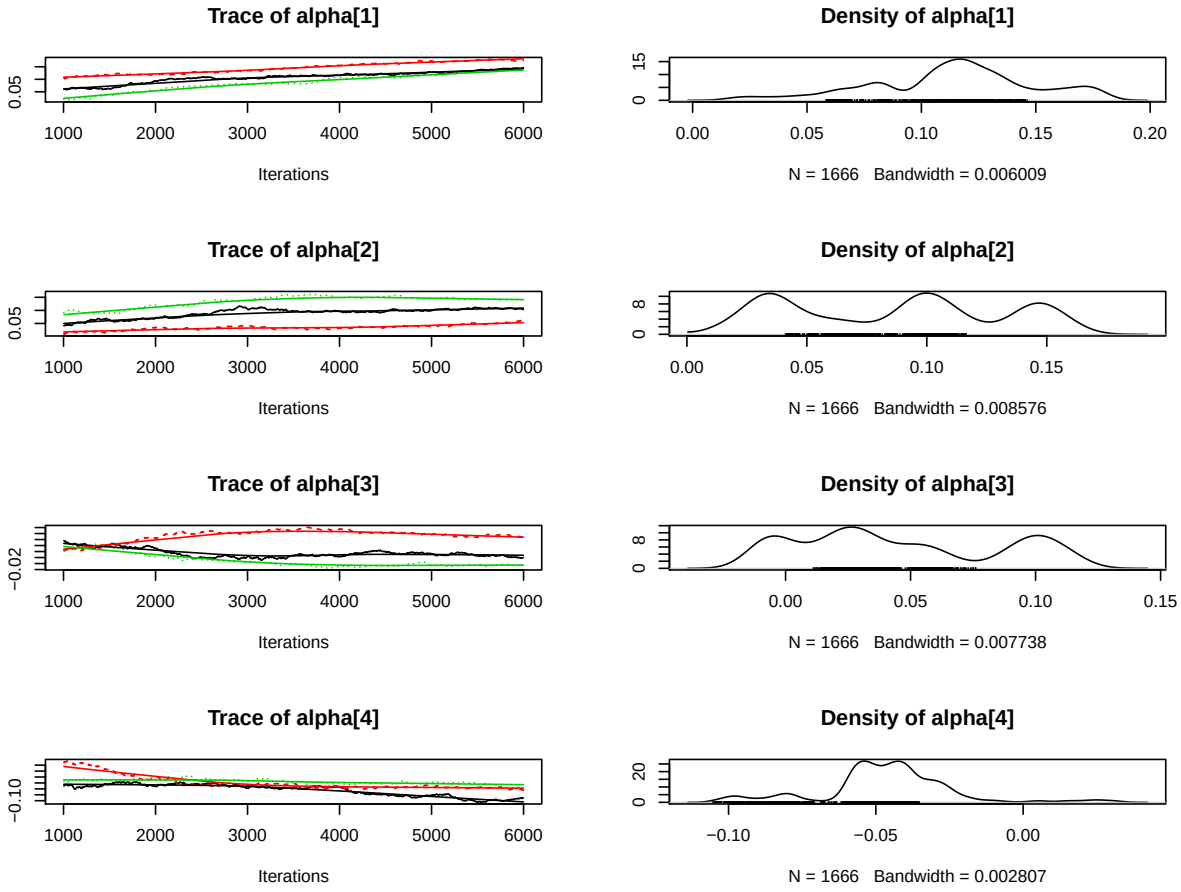
```
##
## Iterations = 1003:5998
## Thinning interval = 3
## Number of chains = 3
## Sample size per chain = 1666
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## alpha[1]   0.113424  0.035434 5.012e-04      1.126e-02
## alpha[2]   0.086439  0.044434 6.285e-04      6.730e-03
## alpha[3]   0.045598  0.040094 5.671e-04      7.453e-03
## alpha[4]  -0.046597  0.022348 3.161e-04      7.201e-03
## alpha[5]  -0.037036  0.047944 6.782e-04      2.263e-02
## alpha[6]  -0.043044  0.072531 1.026e-03      1.157e-02
## alpha[7]  -0.170157  0.106705 1.509e-03      2.543e-02
## alpha[8]   0.006867  0.042062 5.950e-04      2.098e-02
## alpha[9]   0.042399  0.101395 1.434e-03      1.946e-02
## alpha[10]  0.009517  0.085220 1.205e-03      4.108e-02
## alpha[11]  0.133479  0.128535 1.818e-03      5.122e-02
## alpha[12] -0.025174  0.092888 1.314e-03      1.676e-02
## alpha[13] -0.245378  0.111154 1.572e-03      2.437e-02
## alpha[14] -0.417179  0.130136 1.841e-03      4.159e-02
## alpha[15] -0.431264  0.361252 5.110e-03      1.919e-01
## alpha[16] -0.484419  0.602309 8.520e-03      4.351e-02
## alpha[17]  0.483253  0.608466 8.607e-03      1.813e-01
## alpha[18]  0.498610  0.509885 7.212e-03      9.834e-02
## alpha[19]  1.752580  0.408013 5.771e-03      1.422e-01
## alpha[20] -2.689976  0.599519 8.480e-03      1.834e-01
## alpha[21]  1.009785  1.341639 1.898e-02      3.689e-01
## alpha[22]  1.145701  2.144042 3.033e-02      6.092e-01
## alpha[23] -1.358492  2.307997 3.265e-02      3.692e-01
## alpha[24]  0.742816  2.916018 4.125e-02      2.179e-01
## alpha[25]  0.139015  0.030461 4.309e-04      9.534e-03
## alpha[26]  0.124552  0.036239 5.126e-04      1.101e-02
## alpha[27]  0.117285  0.054277 7.677e-04      5.613e-03
```

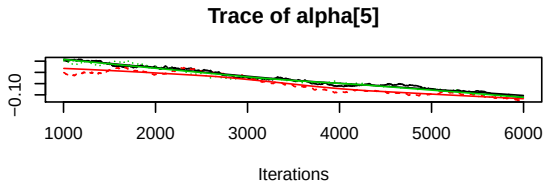
```
## alpha[28]  0.104309  0.024670 3.490e-04      1.107e-02
## alpha[29]  0.005285  0.050472 7.139e-04      1.379e-02
## alpha[30] -0.171124  0.071947 1.018e-03      2.490e-02
## alpha[31] -0.218120  0.097013 1.372e-03      3.984e-02
## alpha[32] -0.345771  0.105052 1.486e-03      2.428e-02
## alpha[33] -0.360937  0.057911 8.191e-04      1.290e-02
## alpha[34] -0.205943  0.090493 1.280e-03      3.689e-02
## alpha[35] -0.091932  0.158339 2.240e-03      8.075e-02
## alpha[36]  0.321133  0.259190 3.666e-03      1.105e-01
## alpha[37]  0.659110  0.239679 3.390e-03      1.016e-01
## alpha[38]  0.500718  0.145748 2.062e-03      4.605e-02
## alpha[39]  0.127663  0.504433 7.135e-03      2.707e-01
## alpha[40] -0.301366  0.598886 8.471e-03      3.107e-01
## alpha[41] -0.992310  0.599616 8.482e-03      2.005e-01
## alpha[42] -1.339860  0.855023 1.209e-02      4.868e-01
## alpha[43]  0.619995  1.870425 2.646e-02      1.012e+00
## alpha[44]  1.176992  1.510347 2.136e-02      4.897e-01
## alpha[45] -0.364607  4.423588 6.257e-02      4.225e+00
## alpha[46] -1.728248  6.194526 8.762e-02      3.524e+00
## alpha[47]  2.597209  6.531755 9.239e-02      2.475e+00
## alpha[48] -1.185088  6.292262 8.900e-02      1.569e+00
## alpha[49] -0.242968  0.066207 9.365e-04      2.970e-02
## alpha[50] -0.216271  0.049242 6.965e-04      1.848e-02
## alpha[51] -0.184753  0.039224 5.548e-04      7.005e-03
## alpha[52] -0.072943  0.036702 5.191e-04      9.655e-03
## alpha[53]  0.066279  0.067245 9.512e-04      2.628e-02
## alpha[54]  0.352281  0.114936 1.626e-03      4.885e-02
## alpha[55]  0.509539  0.157179 2.223e-03      6.963e-02
## alpha[56]  0.761584  0.133999 1.895e-03      5.424e-02
## alpha[57]  0.734537  0.086546 1.224e-03      3.085e-02
## alpha[58]  0.543340  0.114379 1.618e-03      5.611e-02
## alpha[59] -0.067810  0.323429 4.575e-03      1.577e-01
## alpha[60] -0.896793  0.483446 6.838e-03      2.152e-01
## alpha[61] -1.762448  0.487830 6.900e-03      1.912e-01
## alpha[62] -2.165925  0.295818 4.184e-03      9.300e-02
## alpha[63] -1.423591  0.760997 1.076e-02      4.345e-01
## alpha[64]  0.636310  1.559526 2.206e-02      7.578e-01
## alpha[65]  3.698968  1.419613 2.008e-02      5.404e-01
## alpha[66]  2.802275  2.052769 2.904e-02      1.260e+00
## alpha[67] -0.623541  3.723119 5.266e-02      1.995e+00
## alpha[68] -5.781914  3.403613 4.814e-02      1.158e+00
## alpha[69]  3.583632 11.428839 1.617e-01      8.176e+00
## alpha[70]  3.496247 14.317736 2.025e-01      1.046e+01
## alpha[71] -6.323298 12.885284 1.823e-01      5.287e+00
## alpha[72]  4.436901 11.647301 1.648e-01      3.378e+00
## beta[1]    0.603385  0.028255 3.997e-04      6.127e-03
## beta[2]   -0.469953  0.005911 8.361e-05      9.755e-04
## beta[3]   -0.153077  0.028790 4.072e-04      1.005e-02
## beta[4]    0.471725  0.036784 5.203e-04      9.786e-03
## beta[5]    0.155674  0.013893 1.965e-04      3.714e-03
## sigma      0.059433  0.010310 1.458e-04      4.154e-03
##
## 2. Quantiles for each variable:
##
```

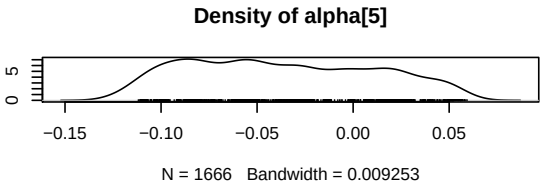
```
##                2.5%       25%       50%       75%     97.5%
## alpha[1]    0.03296   0.09203  0.116514  0.133745  0.175829
## alpha[2]    0.01875   0.03972  0.094123  0.118600  0.157035
## alpha[3]   -0.01109   0.01616  0.035290  0.089085  0.113670
## alpha[4]   -0.09800  -0.05591 -0.045901 -0.036424  0.014462
## alpha[5]   -0.11094  -0.07926 -0.041667  0.002377  0.050466
## alpha[6]   -0.20662  -0.11684 -0.015894  0.004321  0.057485
## alpha[7]   -0.33313  -0.25716 -0.175642 -0.078534  0.001868
## alpha[8]   -0.04872  -0.02036 -0.002022  0.014779  0.121385
## alpha[9]   -0.08902  -0.05391  0.029827  0.124282  0.235354
## alpha[10]  -0.12583  -0.05394  0.008170  0.059659  0.197173
## alpha[11]  -0.11790   0.04267  0.148772  0.231780  0.341037
## alpha[12]  -0.17062  -0.10314 -0.044390  0.063493  0.132635
## alpha[13]  -0.43549  -0.33129 -0.264841 -0.144158 -0.064040
## alpha[14]  -0.64200  -0.51268 -0.433671 -0.299378 -0.199632
## alpha[15]  -1.16992  -0.58171 -0.474064 -0.193054  0.218972
## alpha[16]  -1.19504  -1.03640 -0.692149  0.237375  0.542187
## alpha[17]  -0.34404  -0.06860  0.398650  0.842924  1.840094
## alpha[18]  -0.15652   0.11327  0.370219  0.844568  1.438710
## alpha[19]   0.68381   1.52281  1.850179  2.057172  2.312576
## alpha[20]  -3.56099  -3.12791 -2.859463 -2.284896 -1.340238
## alpha[21]  -2.46915   0.38772  1.158856  1.955266  2.926980
## alpha[22]  -1.75500  -0.25242  1.093440  2.069315  9.077034
## alpha[23]  -7.15805  -2.47439 -1.108872  0.117855  2.393170
## alpha[24]  -4.63105  -1.11388  0.607518  2.367671  7.122755
## alpha[25]   0.07398   0.12150  0.139165  0.158106  0.202323
## alpha[26]   0.06207   0.10222  0.119310  0.138415  0.190217
## alpha[27]   0.05388   0.06435  0.094151  0.181762  0.199548
## alpha[28]   0.05677   0.08449  0.111172  0.123682  0.136419
## alpha[29]  -0.09195  -0.03613  0.020415  0.049752  0.065395
## alpha[30]  -0.32301  -0.20107 -0.175524 -0.130878 -0.009875
## alpha[31]  -0.42118  -0.28135 -0.220389 -0.150796 -0.048176
## alpha[32]  -0.53433  -0.39434 -0.342319 -0.284706 -0.136022
## alpha[33]  -0.45914  -0.39995 -0.360325 -0.326337 -0.238094
## alpha[34]  -0.30791  -0.27431 -0.238653 -0.153408 -0.004276
## alpha[35]  -0.34397  -0.22259 -0.087218  0.036686  0.261077
## alpha[36]  -0.19424   0.13502  0.397738  0.523784  0.745076
## alpha[37]   0.08474   0.53640  0.743265  0.831003  0.972990
## alpha[38]   0.21191   0.39758  0.515744  0.615183  0.750959
## alpha[39]  -0.81418  -0.31269  0.193037  0.576472  0.846047
## alpha[40]  -1.00303  -0.82927 -0.483214  0.210809  0.890992
## alpha[41]  -2.00083  -1.31997 -1.033304 -0.762403  0.441059
## alpha[42]  -2.54375  -2.10889 -1.477469 -0.693330  0.272003
## alpha[43]  -2.95997  -0.85282  1.220280  1.900666  3.419896
## alpha[44]  -2.30139   0.19823  1.457086  2.289585  3.611935
## alpha[45]  -5.83746  -4.80369 -1.166498  4.198056  6.578248
## alpha[46] -12.01707  -7.28570 -1.373310  4.442959  6.998735
## alpha[47] -11.39655  -2.36834  2.823148  8.452786 12.750035
## alpha[48] -11.39272  -5.89636 -1.765827  2.477937 13.381794
## alpha[49]  -0.34687  -0.29396 -0.249633 -0.190874 -0.119969
## alpha[50]  -0.30787  -0.26079 -0.205339 -0.180527 -0.139564
## alpha[51]  -0.26446  -0.21910 -0.172567 -0.161340 -0.123445
## alpha[52]  -0.13998  -0.09962 -0.077183 -0.047537 -0.002488
## alpha[53]  -0.04542   0.01667  0.061326  0.122758  0.183748
```

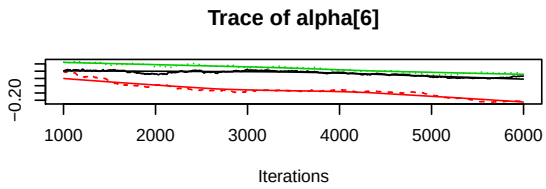
```
## alpha[54]   0.13146   0.25806  0.372822  0.440960  0.538458
## alpha[55]   0.21094   0.38407  0.530257  0.639004  0.777050
## alpha[56]   0.47896   0.67059  0.782757  0.861058  0.961860
## alpha[57]   0.49753   0.70336  0.760785  0.791526  0.838358
## alpha[58]   0.31430   0.44834  0.565505  0.640122  0.696817
## alpha[59]  -0.57566  -0.35207 -0.090923  0.199986  0.525152
## alpha[60]  -1.62899  -1.31152 -0.938242 -0.532516  0.031016
## alpha[61]  -2.45524  -2.10368 -1.879679 -1.433762 -0.686237
## alpha[62]  -2.66244  -2.32298 -2.224146 -1.980207 -1.465394
## alpha[63]  -2.45447  -2.17037 -1.520534 -0.750105 -0.108886
## alpha[64]  -2.21705  -0.66373  0.932931  1.980325  2.916107
## alpha[65]   0.21948   3.16954  3.969631  4.849347  5.330663
## alpha[66]  -0.79267   0.96037  3.119730  4.717257  5.687437
## alpha[67]  -4.97157  -3.98449 -1.754310  2.559047  6.644411
## alpha[68] -10.76595  -8.22130 -6.512795 -3.375054  2.409147
## alpha[69] -17.08578  -7.83883  7.826858 14.325017 15.502120
## alpha[70] -16.33715 -10.31884  2.660701 17.196455 26.285080
## alpha[71] -26.75810 -18.16886 -6.265612  5.234938 16.829549
## alpha[72] -23.90260  -3.53299  5.579312 13.933373 21.310008
## beta[1]     0.55155   0.58180  0.602415  0.626457  0.650265
## beta[2]    -0.47967  -0.47441 -0.470270 -0.466249 -0.457019
## beta[3]    -0.21592  -0.17185 -0.145272 -0.129417 -0.116620
## beta[4]     0.40501   0.44384  0.468786  0.499640  0.535427
## beta[5]     0.13065   0.14333  0.157266  0.166975  0.177490
## sigma       0.04705   0.05056  0.056685  0.066634  0.082238
```

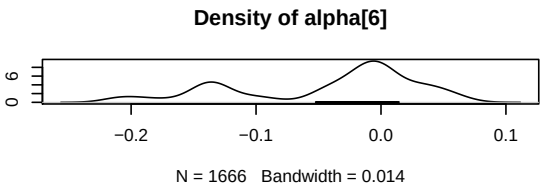
```
plot(samp)
```

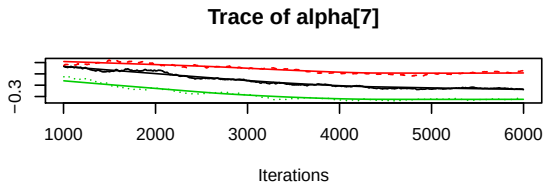

Trace of alpha[1]
0.05
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[1]
0
15
0.00
0.05
0.10
0.15
0.20
N = 1666 Bandwidth = 0.006009
Trace of alpha[2]
0.05
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[2]
0
8
0.00
0.05
0.10
0.15
N = 1666 Bandwidth = 0.008576
Trace of alpha[3]
−0.02
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[3]
0
8
0.00
0.05
0.10
0.15
N = 1666 Bandwidth = 0.007738
Trace of alpha[4]
−0.10
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[4]
0
20
−0.10
−0.05
0.00
N = 1666 Bandwidth = 0.002807

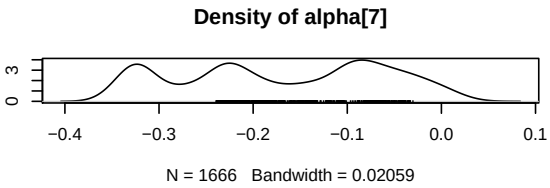

Trace of alpha[5]
−0.10
1000
2000
3000
4000
5000
6000
Iterations

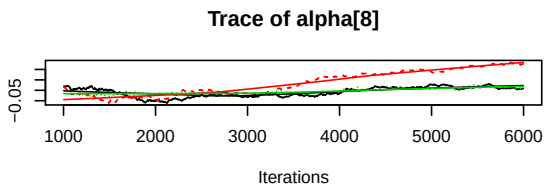

Density of alpha[5]
0
5
−0.15
−0.10
−0.05
0.00
0.05
N = 1666 Bandwidth = 0.009253

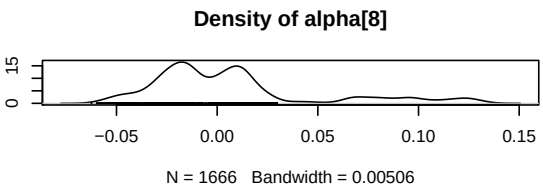

Trace of alpha[6]
−0.20
1000
2000
3000
4000
5000
6000
Iterations

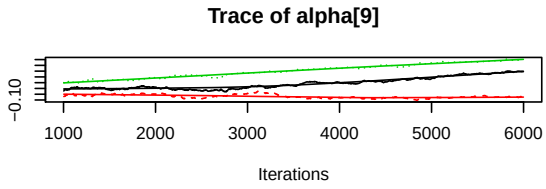

Density of alpha[6]
0
6
−0.2
−0.1
0.0
0.1
N = 1666 Bandwidth = 0.014

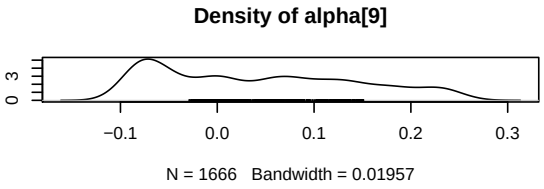

Trace of alpha[7]
−0.3
1000
2000
3000
4000
5000
6000
Iterations

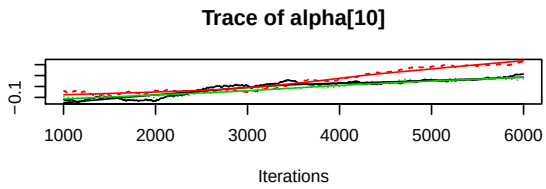

Density of alpha[7]
0
3
−0.4
−0.3
−0.2
−0.1
0.0
0.1
N = 1666 Bandwidth = 0.02059

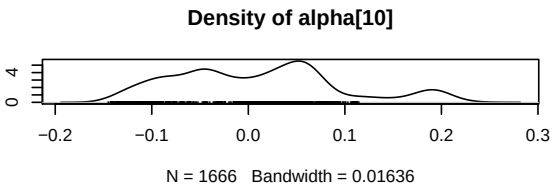

Trace of alpha[8]
−0.05
1000
2000
3000
4000
5000
6000
Iterations

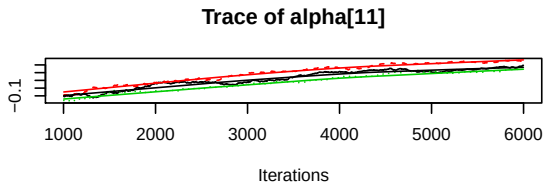

Density of alpha[8]
0
15
−0.05
0.00
0.05
0.10
0.15
N = 1666 Bandwidth = 0.00506

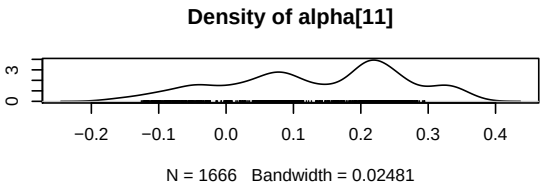

Trace of alpha[9]
−0.10
1000
2000
3000
4000
5000
6000
Iterations

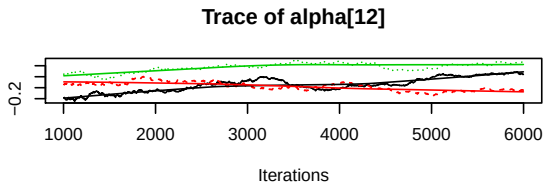

Density of alpha[9]
0 3
−0.1
0.0
0.1
0.2
0.3
N = 1666 Bandwidth = 0.01957

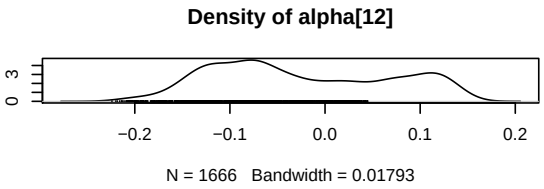

Trace of alpha[10]
−0.1
1000
2000
3000
4000
5000
6000
Iterations

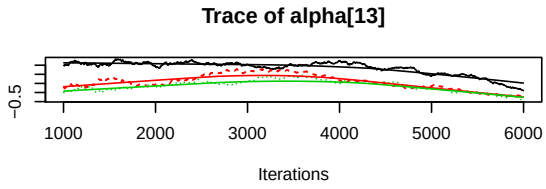

Density of alpha[10]
0 4
−0.2
−0.1
0.0
0.1
0.2
0.3
N = 1666 Bandwidth = 0.01636

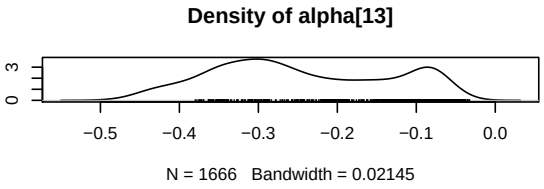

Trace of alpha[11]
−0.1
1000
2000
3000
4000
5000
6000
Iterations

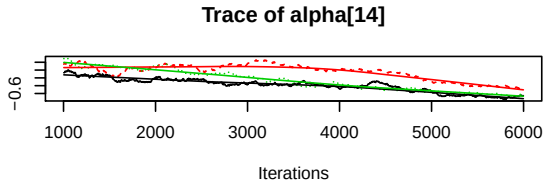

Density of alpha[11]
0 3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
N = 1666 Bandwidth = 0.02481

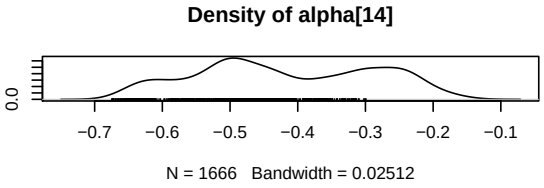

Trace of alpha[12]
−0.2
1000
2000
3000
4000
5000
6000
Iterations

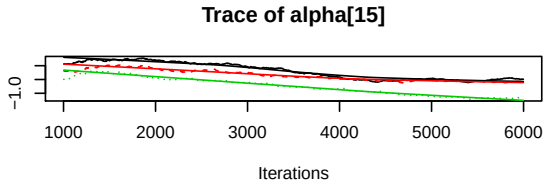

Density of alpha[12]
0 3
−0.2
−0.1
0.0
0.1
0.2
N = 1666 Bandwidth = 0.01793

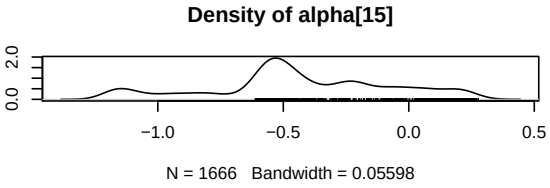

Trace of alpha[13]
−0.5
1000
2000
3000
4000
5000
6000
Iterations

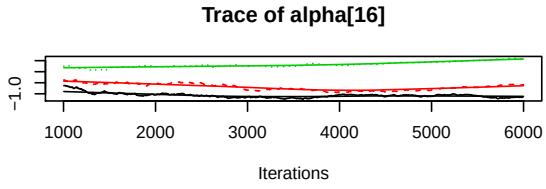

Density of alpha[13]
0
3
−0.5
−0.4
−0.3
−0.2
−0.1
0.0
N = 1666 Bandwidth = 0.02145

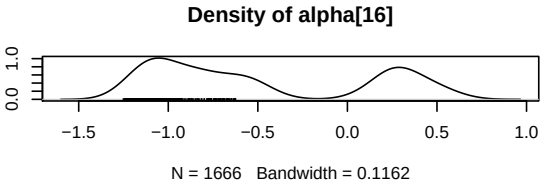

Trace of alpha[14]
−0.6
1000
2000
3000
4000
5000
6000
Iterations

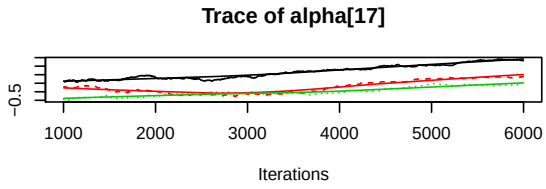

Density of alpha[14]
0.0
−0.7
−0.6
−0.5
−0.4
−0.3
−0.2
−0.1
N = 1666 Bandwidth = 0.02512

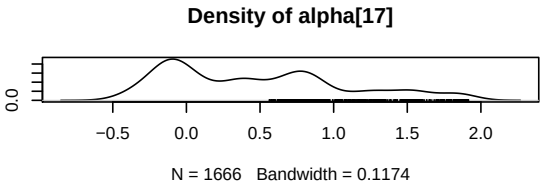

Trace of alpha[15]
−1.0
1000
2000
3000
4000
5000
6000
Iterations

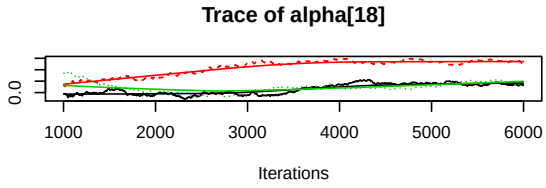

Density of alpha[15]
0.0
2.0
−1.0
−0.5
0.0
0.5
N = 1666 Bandwidth = 0.05598

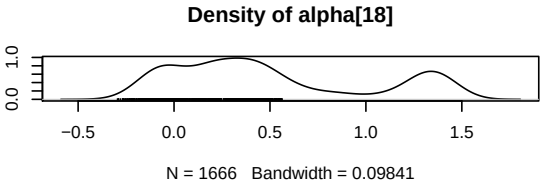

Trace of alpha[16]
−1.0
1000
2000
3000
4000
5000
6000
Iterations

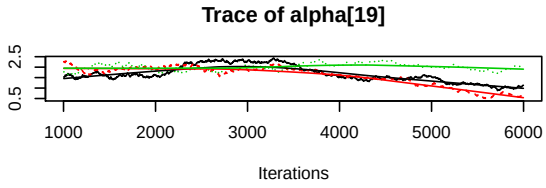

Density of alpha[16]
0.0
1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
N = 1666 Bandwidth = 0.1162

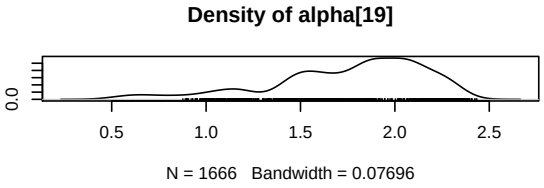

Trace of alpha[17]
−0.5
1000
2000
3000
4000
5000
6000
Iterations

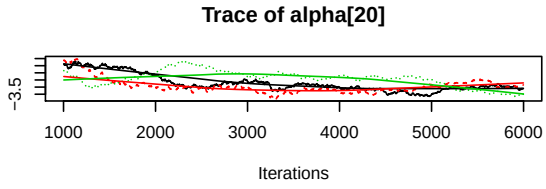

Density of alpha[17]
0.0
−0.5
0.0
0.5
1.0
1.5
2.0
N = 1666 Bandwidth = 0.1174

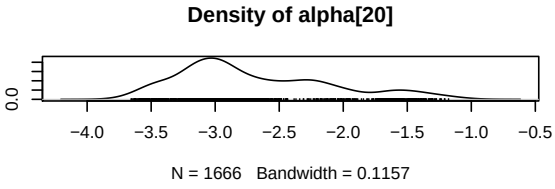

Trace of alpha[18]
0.0
1000
2000
3000
4000
5000
6000
Iterations

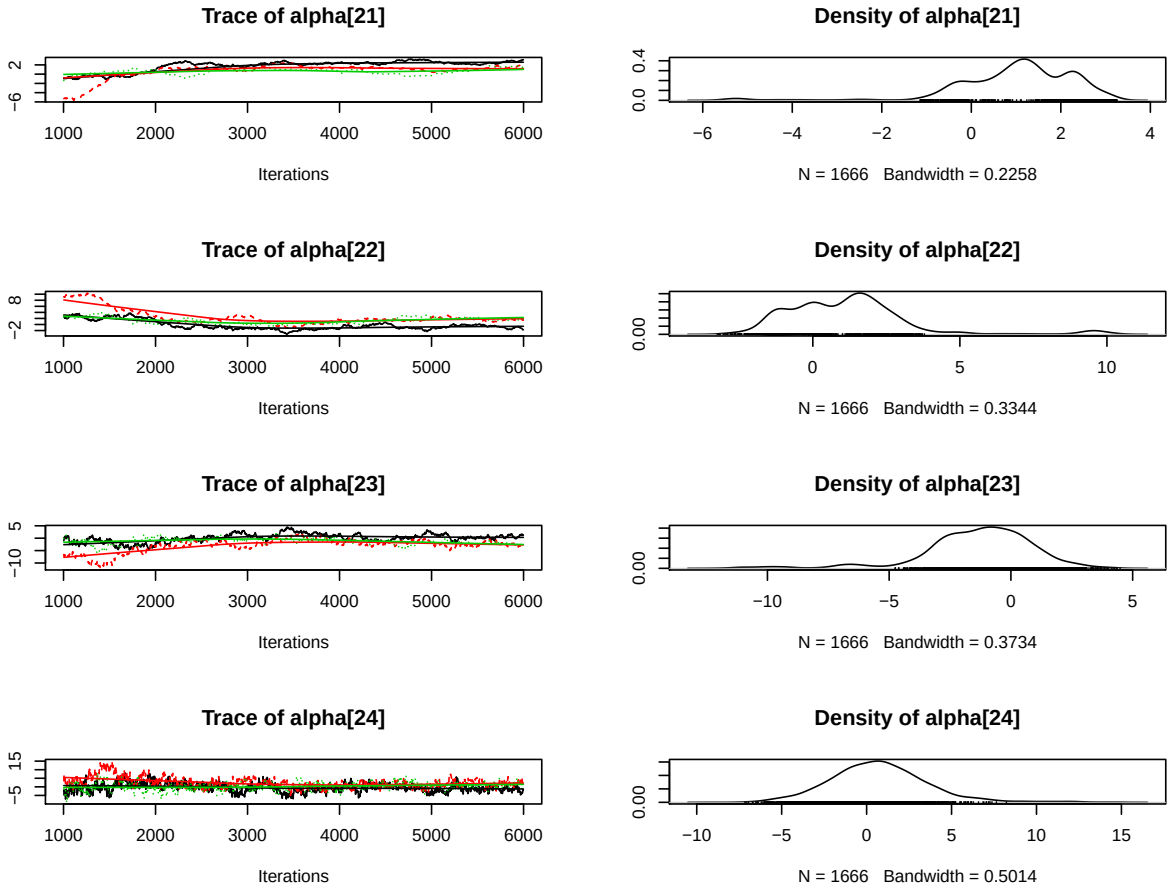

Density of alpha[18]
0.0
1.0
−0.5
0.0
0.5
1.0
1.5
N = 1666 Bandwidth = 0.09841

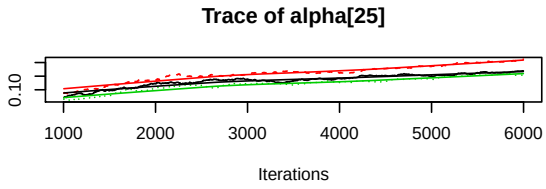

Trace of alpha[19]
0.5
2.5
1000
2000
3000
4000
5000
6000
Iterations

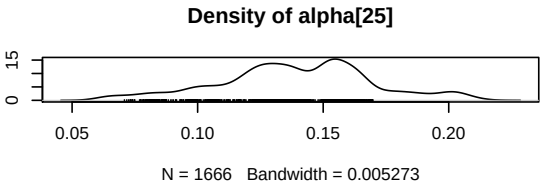

Density of alpha[19]
0.0
0.5
1.0
1.5
2.0
2.5
N = 1666 Bandwidth = 0.07696

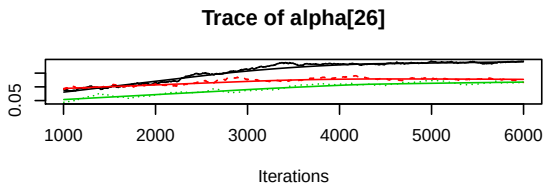

Trace of alpha[20]
−3.5
1000
2000
3000
4000
5000
6000
Iterations

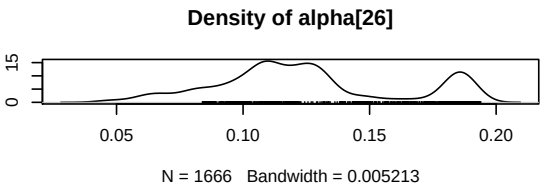

Density of alpha[20]
0.0
−4.0
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
N = 1666 Bandwidth = 0.1157

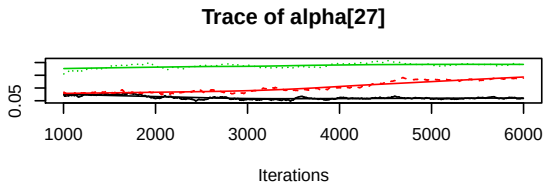

Trace of alpha[21]
Iterations
Density of alpha[21]
N = 1666 Bandwidth = 0.2258
Trace of alpha[22]
Iterations
Density of alpha[22]
N = 1666 Bandwidth = 0.3344
Trace of alpha[23]
Iterations
Density of alpha[23]
N = 1666 Bandwidth = 0.3734
Trace of alpha[24]
Iterations
Density of alpha[24]
N = 1666 Bandwidth = 0.5014

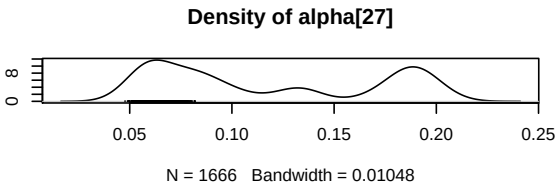

Trace of alpha[25]
0.10
1000
2000
3000
4000
5000
6000
Iterations

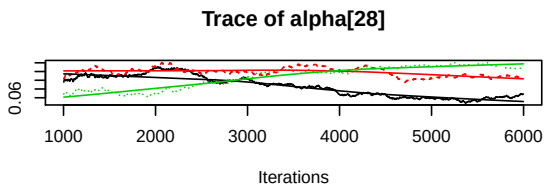

Density of alpha[25]
0
15
0.05
0.10
0.15
0.20
N = 1666 Bandwidth = 0.005273

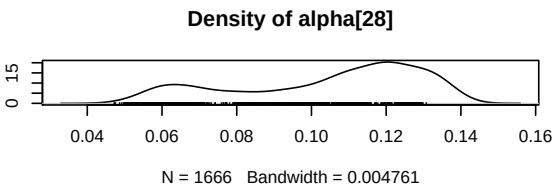

Trace of alpha[26]
0.05
1000
2000
3000
4000
5000
6000
Iterations

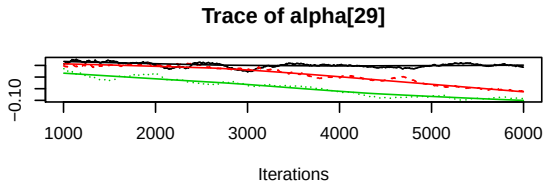

Density of alpha[26]
0
15
0.05
0.10
0.15
0.20
N = 1666 Bandwidth = 0.005213

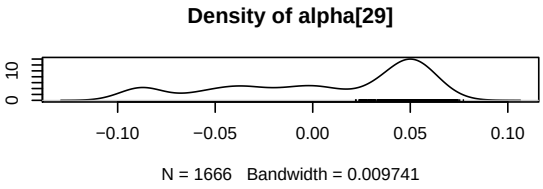

Trace of alpha[27]
0.05
1000
2000
3000
4000
5000
6000
Iterations

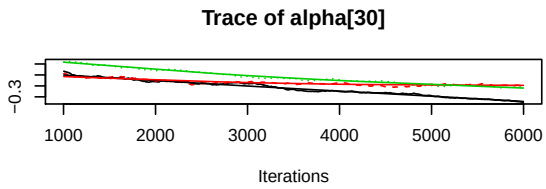

Density of alpha[27]
0
8
0.05
0.10
0.15
0.20
0.25
N = 1666 Bandwidth = 0.01048

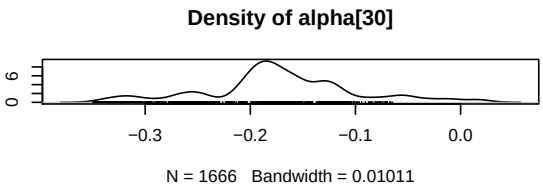

Trace of alpha[28]
0.06
1000
2000
3000
4000
5000
6000
Iterations

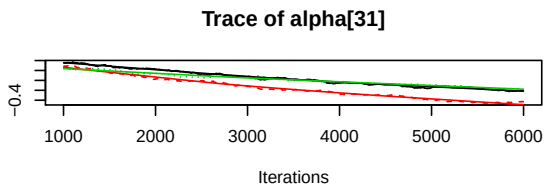

Density of alpha[28]
0
15
0.04
0.06
0.08
0.10
0.12
0.14
0.16
N = 1666 Bandwidth = 0.004761

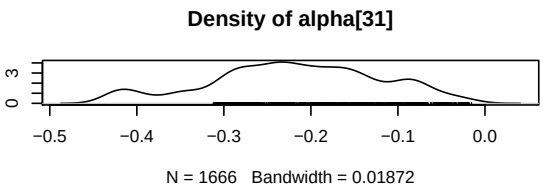

Trace of alpha[29]
−0.10
1000
2000
3000
4000
5000
6000
Iterations

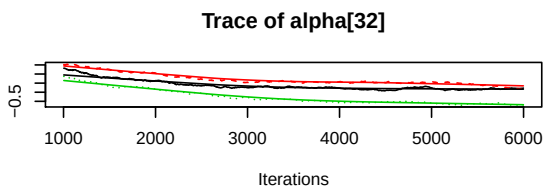

Density of alpha[29]
0
10
−0.10
−0.05
0.00
0.05
0.10
N = 1666 Bandwidth = 0.009741

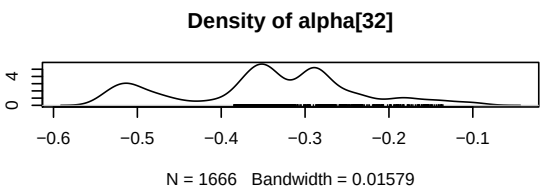

Trace of alpha[30]
−0.3
1000
2000
3000
4000
5000
6000
Iterations

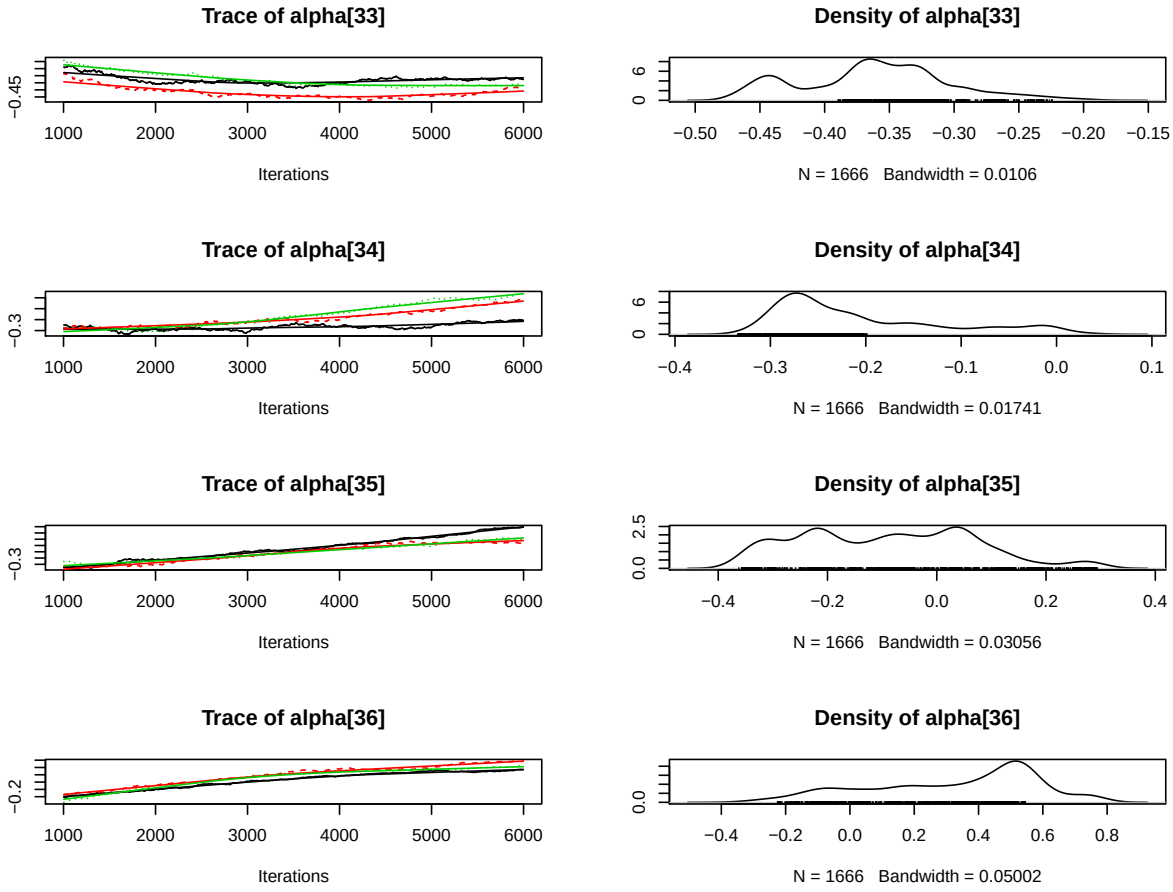

Density of alpha[30]
0
6
−0.3
−0.2
−0.1
0.0
N = 1666 Bandwidth = 0.01011

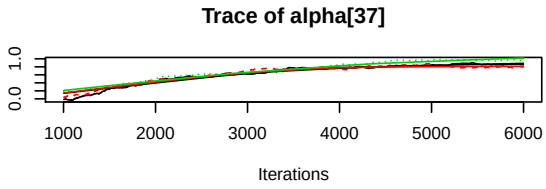

Trace of alpha[31]
−0.4
1000
2000
3000
4000
5000
6000
Iterations

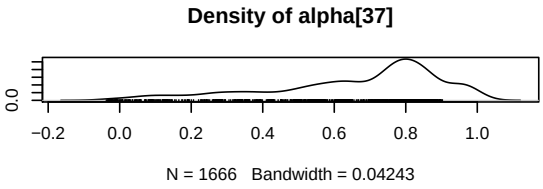

Density of alpha[31]
0
3
−0.5
−0.4
−0.3
−0.2
−0.1
0.0
N = 1666 Bandwidth = 0.01872

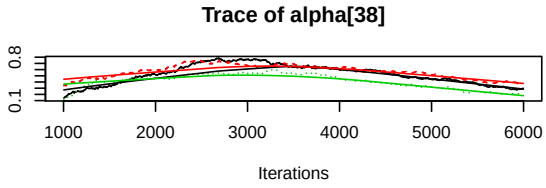

Trace of alpha[32]
−0.5
1000
2000
3000
4000
5000
6000
Iterations

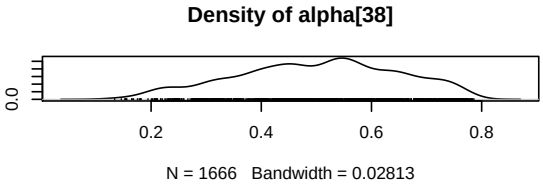

Density of alpha[32]
0
4
−0.6
−0.5
−0.4
−0.3
−0.2
−0.1
N = 1666 Bandwidth = 0.01579

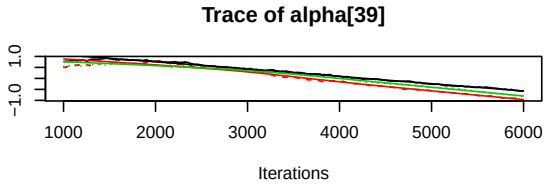

Trace of alpha[33]
−0.45
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[33]
0
6
−0.50
−0.45
−0.40
−0.35
−0.30
−0.25
−0.20
−0.15
N = 1666 Bandwidth = 0.0106
Trace of alpha[34]
−0.3
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[34]
0
6
−0.4
−0.3
−0.2
−0.1
0.0
0.1
N = 1666 Bandwidth = 0.01741
Trace of alpha[35]
−0.3
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[35]
0.0
2.5
−0.4
−0.2
0.0
0.2
0.4
N = 1666 Bandwidth = 0.03056
Trace of alpha[36]
−0.2
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[36]
0.0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
N = 1666 Bandwidth = 0.05002

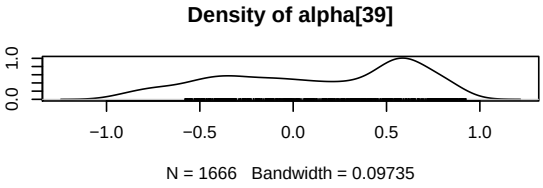

Trace of alpha[37]
0.0
1.0
1000
2000
3000
4000
5000
6000
Iterations

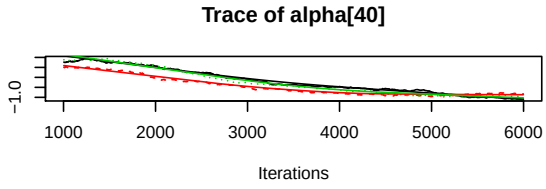

Density of alpha[37]
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
N = 1666 Bandwidth = 0.04243

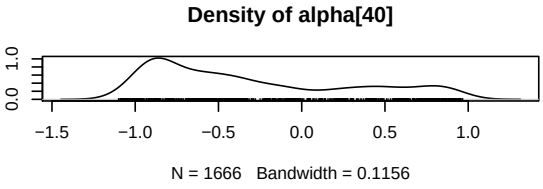

Trace of alpha[38]
0.1
0.8
1000
2000
3000
4000
5000
6000
Iterations

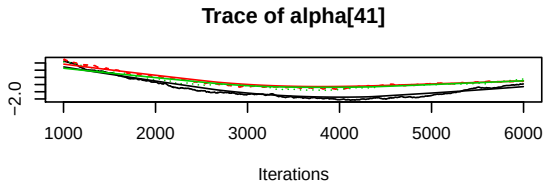

Density of alpha[38]
0.0
0.2
0.4
0.6
0.8
N = 1666 Bandwidth = 0.02813

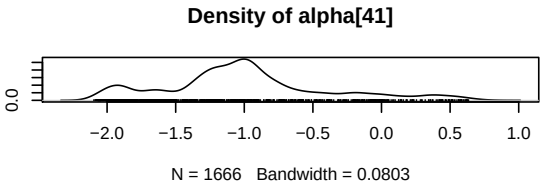

Trace of alpha[39]
−1.0
1.0
1000
2000
3000
4000
5000
6000
Iterations

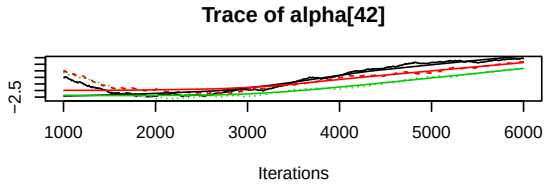

Density of alpha[39]
0.0
1.0
−1.0
−0.5
0.0
0.5
1.0
N = 1666 Bandwidth = 0.09735

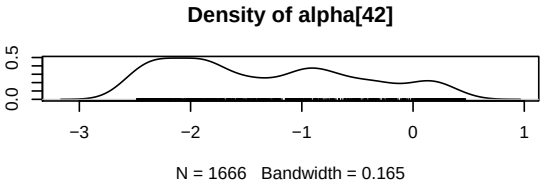

Trace of alpha[40]
−1.0
1000
2000
3000
4000
5000
6000
Iterations

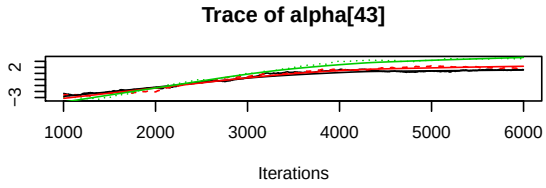

Density of alpha[40]
0.0
1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
N = 1666 Bandwidth = 0.1156

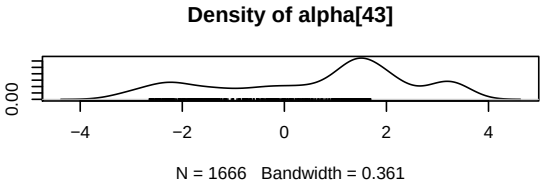

Trace of alpha[41]
−2.0
1000
2000
3000
4000
5000
6000
Iterations

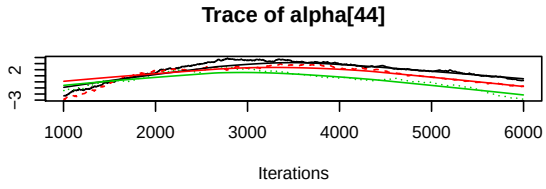

Density of alpha[41]
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
N = 1666 Bandwidth = 0.0803

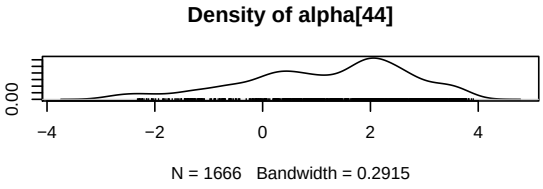

Trace of alpha[42]
−2.5
1000
2000
3000
4000
5000
6000
Iterations

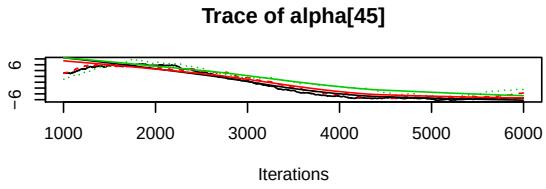

Density of alpha[42]
0.0
0.5
−3
−2
−1
0
1
N = 1666 Bandwidth = 0.165

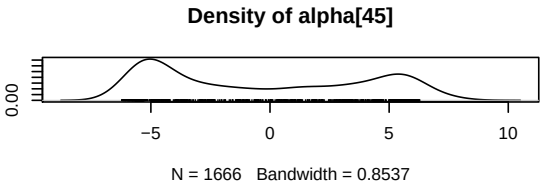

Trace of alpha[43]
−3
2
1000
2000
3000
4000
5000
6000
Iterations

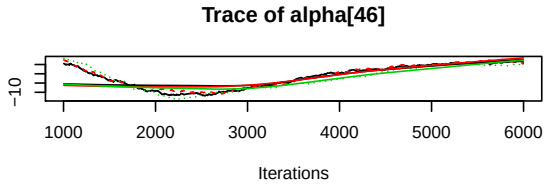

Density of alpha[43]
0.00
−4
−2
0
2
4
N = 1666 Bandwidth = 0.361

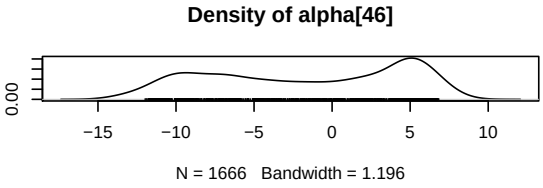

Trace of alpha[44]
−3
2
1000
2000
3000
4000
5000
6000
Iterations

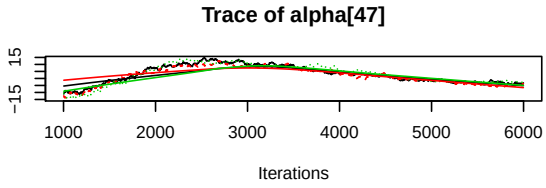

Density of alpha[44]
0.00
−4
−2
0
2
4
N = 1666 Bandwidth = 0.2915

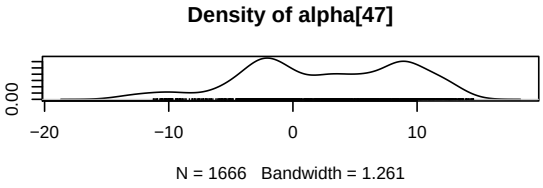

Trace of alpha[45]
−6
6
1000
2000
3000
4000
5000
6000
Iterations

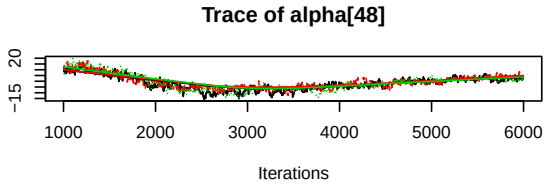

Density of alpha[45]
0.00
−5
0
5
10
N = 1666 Bandwidth = 0.8537

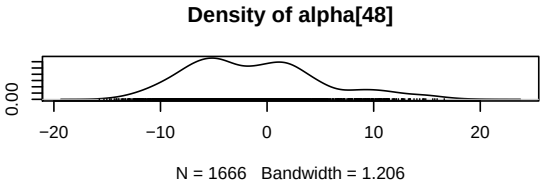

Trace of alpha[46]
−10
1000
2000
3000
4000
5000
6000
Iterations

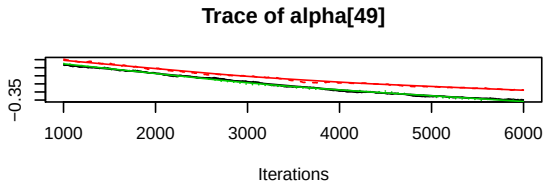

Density of alpha[46]
0.00
−15
−10
−5
0
5
10
N = 1666 Bandwidth = 1.196

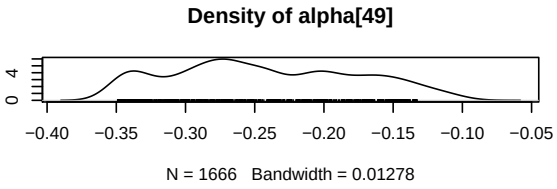

Trace of alpha[47]
−15
15
1000
2000
3000
4000
5000
6000
Iterations

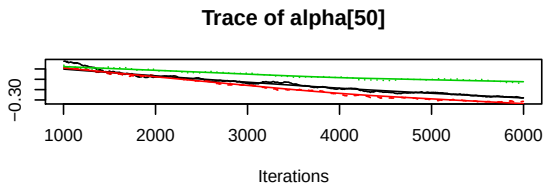

Density of alpha[47]
0.00
−20
−10
0
10
N = 1666 Bandwidth = 1.261

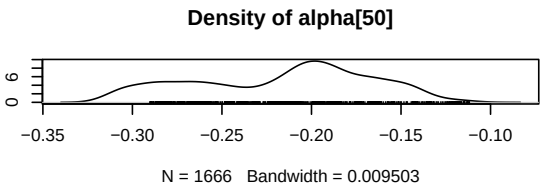

Trace of alpha[48]
−15
20
1000
2000
3000
4000
5000
6000
Iterations

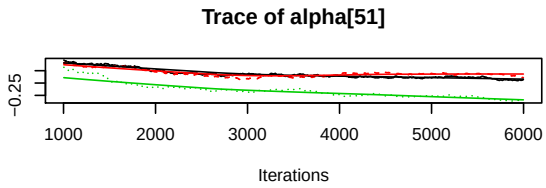

Density of alpha[48]
0.00
−20
−10
0
10
20
N = 1666 Bandwidth = 1.206

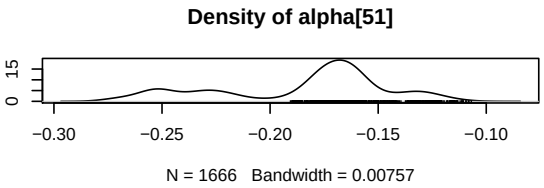

Trace of alpha[49]
−0.35
1000
2000
3000
4000
5000
6000
Iterations

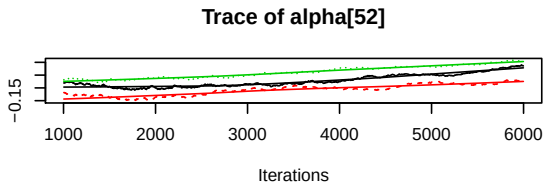

Density of alpha[49]
0
4
−0.40
−0.35
−0.30
−0.25
−0.20
−0.15
−0.10
−0.05
N = 1666 Bandwidth = 0.01278

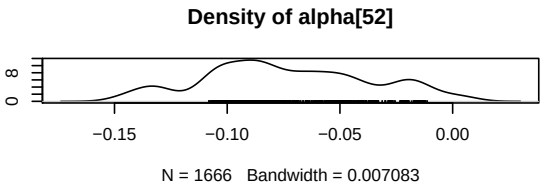

Trace of alpha[50]
−0.30
1000
2000
3000
4000
5000
6000
Iterations

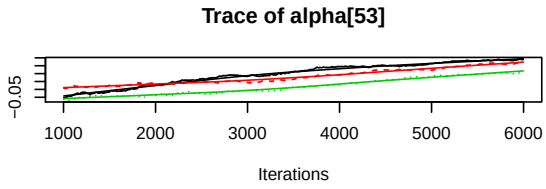

Density of alpha[50]
0
6
−0.35
−0.30
−0.25
−0.20
−0.15
−0.10
N = 1666 Bandwidth = 0.009503

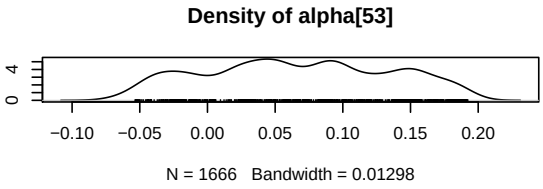

Trace of alpha[51]
−0.25
1000
2000
3000
4000
5000
6000
Iterations

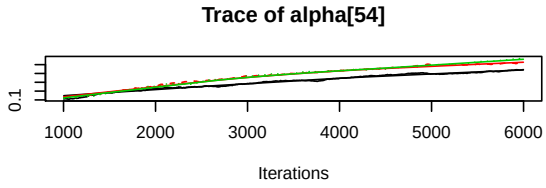

Density of alpha[51]
0
15
−0.30
−0.25
−0.20
−0.15
−0.10
N = 1666 Bandwidth = 0.00757

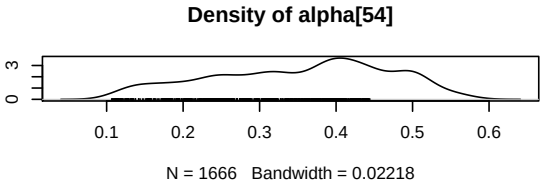

Trace of alpha[52]
−0.15
1000
2000
3000
4000
5000
6000
Iterations

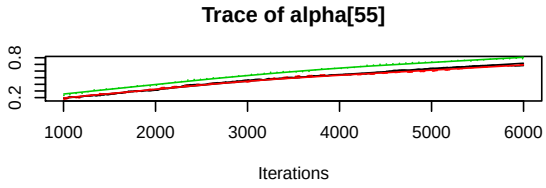

Density of alpha[52]
0
8
−0.15
−0.10
−0.05
0.00
N = 1666 Bandwidth = 0.007083

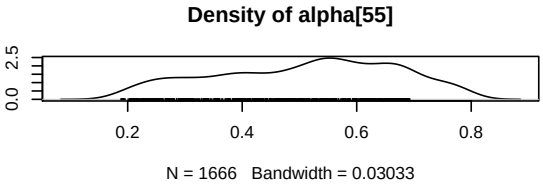

Trace of alpha[53]
−0.05
1000
2000
3000
4000
5000
6000
Iterations

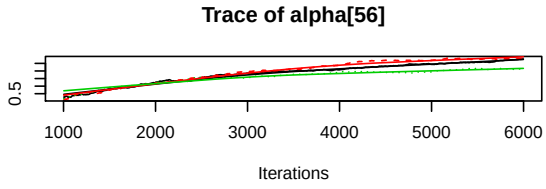

Density of alpha[53]
0
4
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
N = 1666 Bandwidth = 0.01298

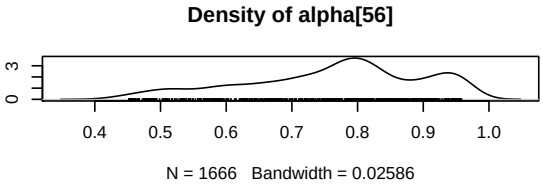

Trace of alpha[54]
0.1
1000
2000
3000
4000
5000
6000
Iterations

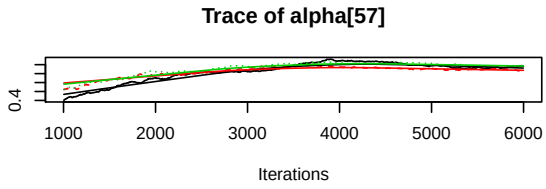

Density of alpha[54]
0
3
0.1
0.2
0.3
0.4
0.5
0.6
N = 1666 Bandwidth = 0.02218

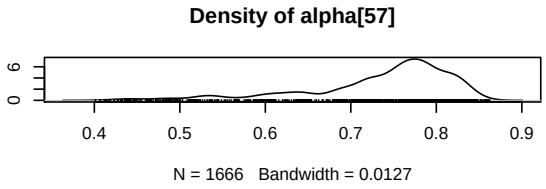

Trace of alpha[55]
0.2
0.8
1000
2000
3000
4000
5000
6000
Iterations

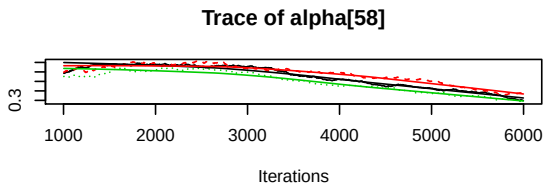

Density of alpha[55]
0.0
2.5
0.2
0.4
0.6
0.8
N = 1666 Bandwidth = 0.03033

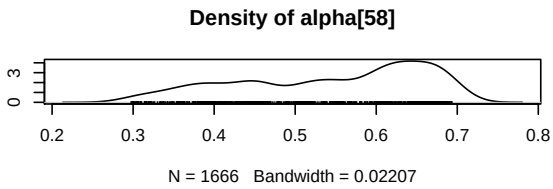

Trace of alpha[56]
0.5
1000
2000
3000
4000
5000
6000
Iterations

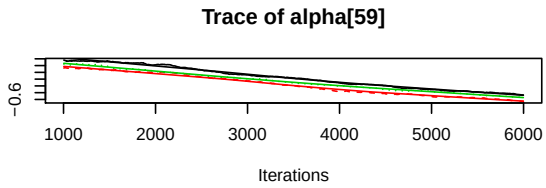

Density of alpha[56]
0
3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
N = 1666 Bandwidth = 0.02586

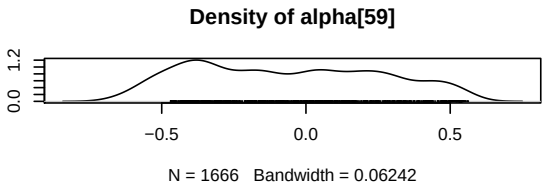

Trace of alpha[57]
0.4
1000
2000
3000
4000
5000
6000
Iterations

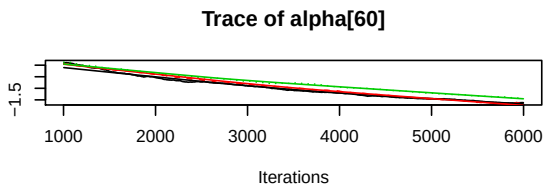

Density of alpha[57]
0
6
0.4
0.5
0.6
0.7
0.8
0.9
N = 1666 Bandwidth = 0.0127

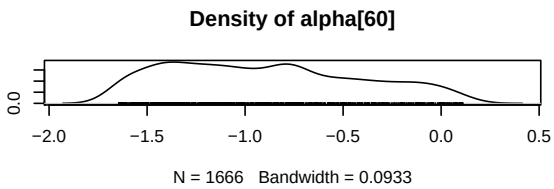

Trace of alpha[58]
0.3
1000
2000
3000
4000
5000
6000
Iterations

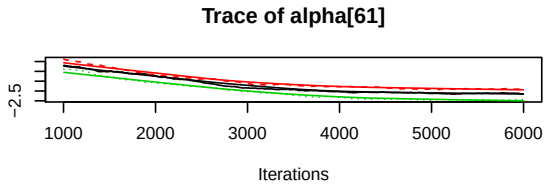

Density of alpha[58]
0
3
0.2
0.3
0.4
0.5
0.6
0.7
0.8
N = 1666 Bandwidth = 0.02207

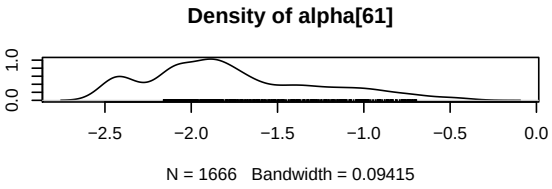

Trace of alpha[59]
−0.6
1000
2000
3000
4000
5000
6000
Iterations

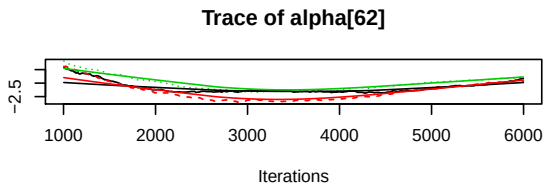

Density of alpha[59]
0.0
1.2
−0.5
0.0
0.5
N = 1666 Bandwidth = 0.06242

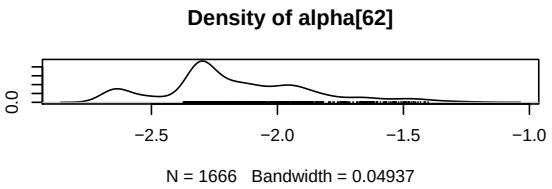

Trace of alpha[60]
−1.5
1000
2000
3000
4000
5000
6000
Iterations

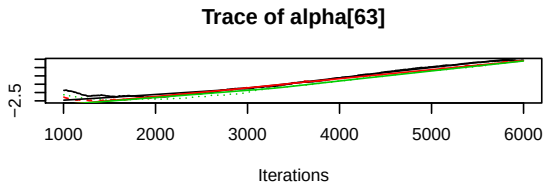

Density of alpha[60]
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
N = 1666 Bandwidth = 0.0933

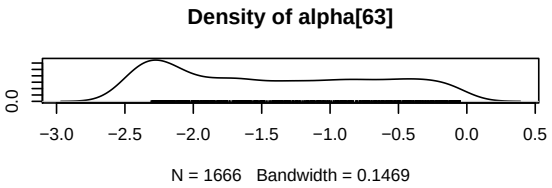

Trace of alpha[61]
−2.5
1000
2000
3000
4000
5000
6000
Iterations

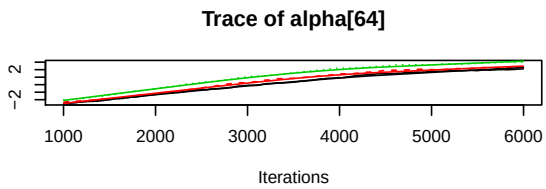

Density of alpha[61]
0.0
1.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
N = 1666 Bandwidth = 0.09415

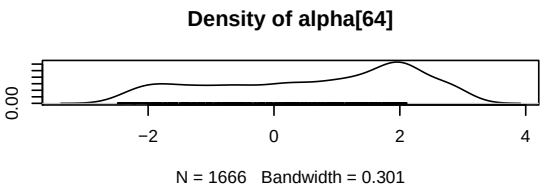

Trace of alpha[62]
−2.5
1000
2000
3000
4000
5000
6000
Iterations

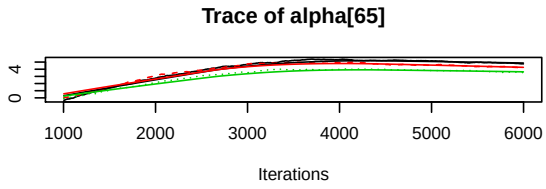

Density of alpha[62]
0.0
−2.5
−2.0
−1.5
−1.0
N = 1666 Bandwidth = 0.04937

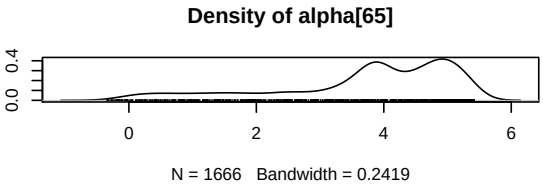

Trace of alpha[63]
−2.5
1000
2000
3000
4000
5000
6000
Iterations

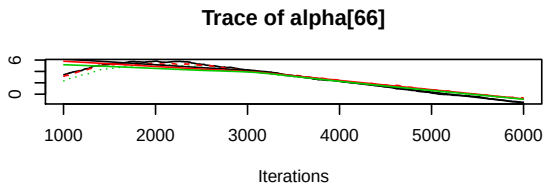

Density of alpha[63]
0.0
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
N = 1666 Bandwidth = 0.1469

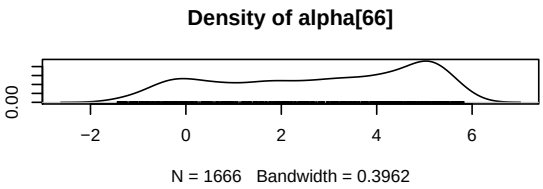

Trace of alpha[64]
−2 2
1000
2000
3000
4000
5000
6000
Iterations

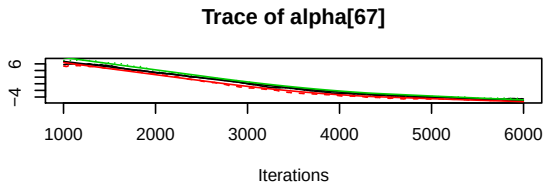

Density of alpha[64]
0.00
−2
0
2
4
N = 1666 Bandwidth = 0.301

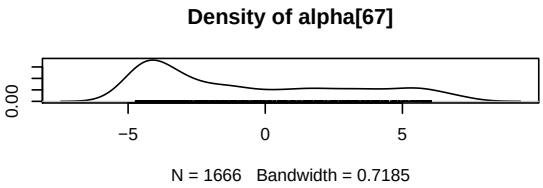

Trace of alpha[65]
0
4
1000
2000
3000
4000
5000
6000
Iterations

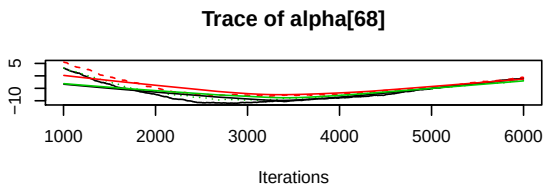

Density of alpha[65]
0.0
0.4
0
2
4
6
N = 1666 Bandwidth = 0.2419

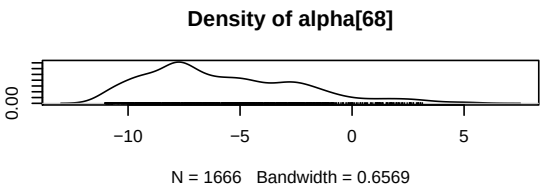

Trace of alpha[66]
0
6
1000
2000
3000
4000
5000
6000
Iterations

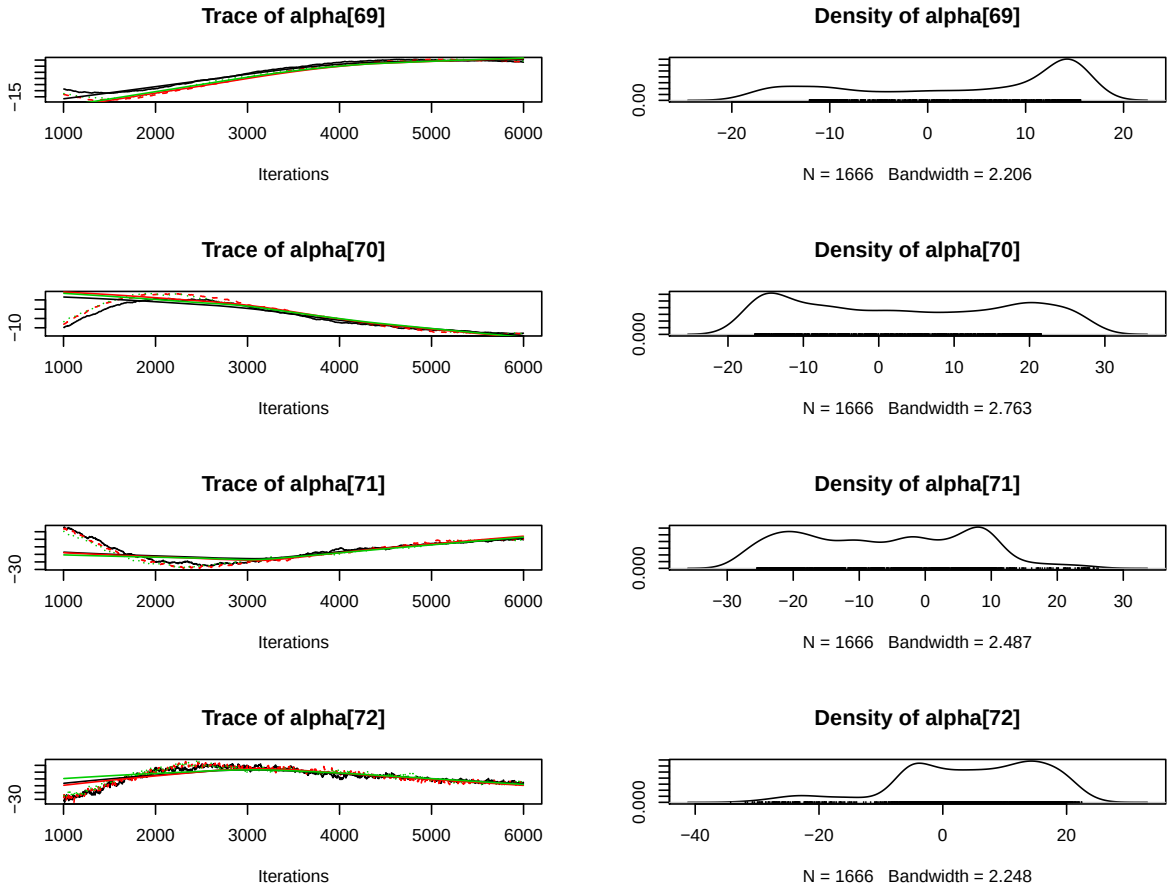

Density of alpha[66]
0.00
−2
0
2
4
6
N = 1666 Bandwidth = 0.3962

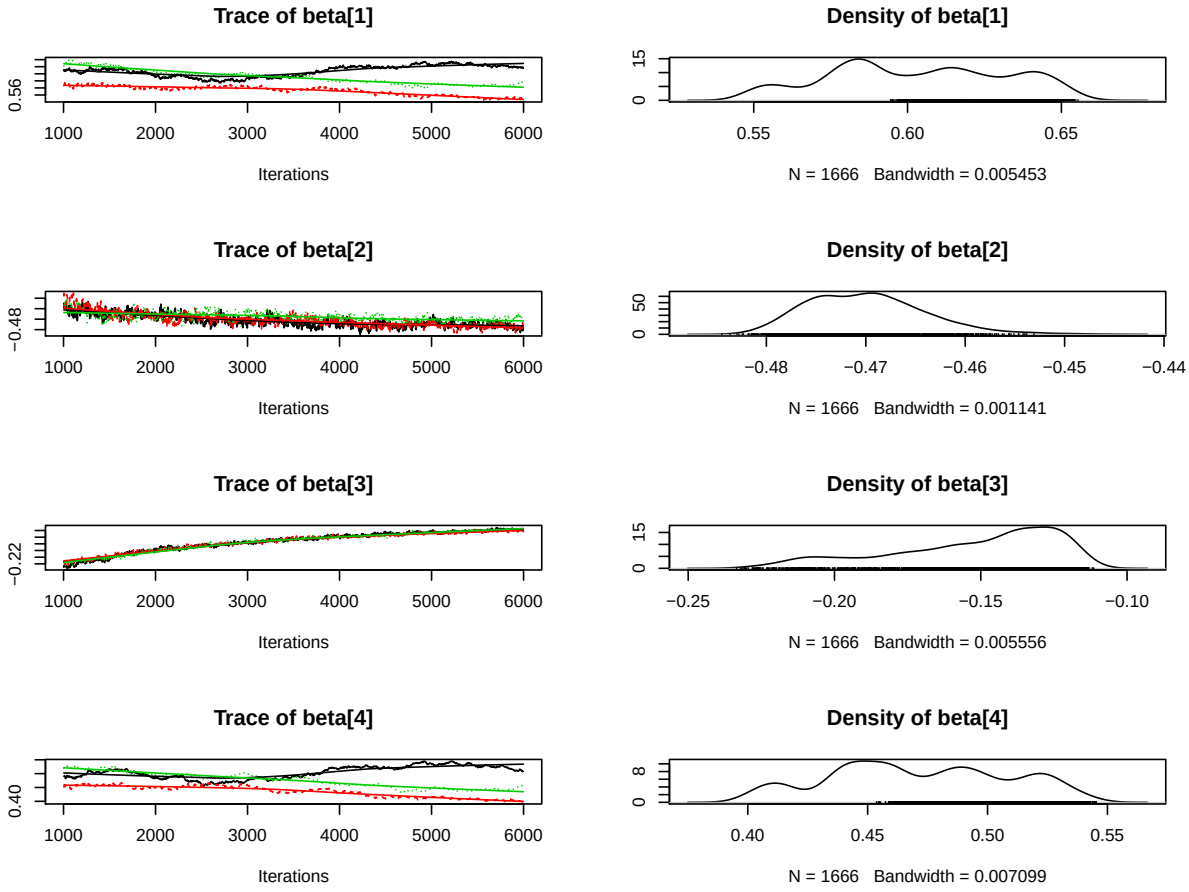

Trace of alpha[67]
−4
6
1000
2000
3000
4000
5000
6000
Iterations

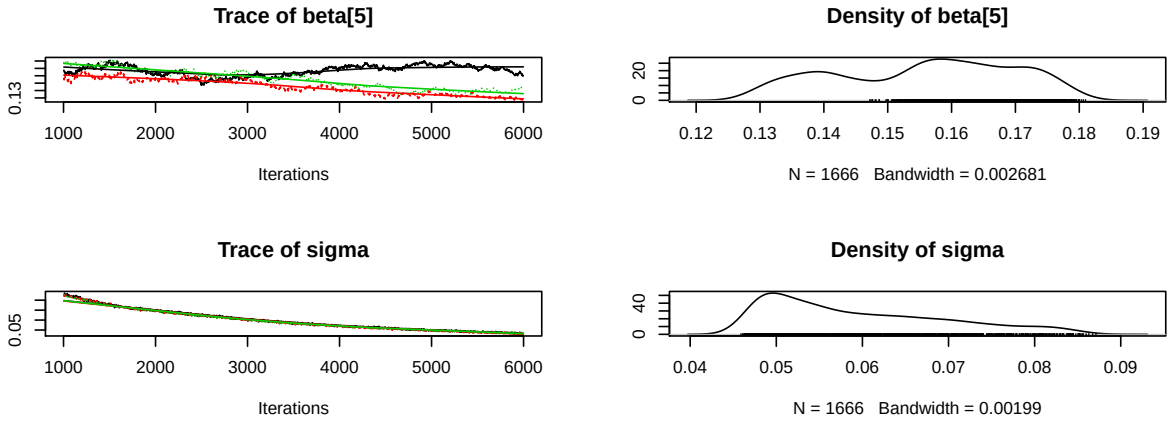

Density of alpha[67]
0.00
−5
0
5
N = 1666 Bandwidth = 0.7185


Trace of alpha[68]
−10
5
1000
2000
3000
4000
5000
6000
Iterations


Density of alpha[68]
0.00
−10
−5
0
5
N = 1666 Bandwidth = 0.6569


Trace of alpha[69]
−15
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[69]
0.00
−20
−10
0
10
20
N = 1666 Bandwidth = 2.206
Trace of alpha[70]
−10
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[70]
0.000
−20
−10
0
10
20
30
N = 1666 Bandwidth = 2.763
Trace of alpha[71]
−30
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[71]
0.000
−30
−20
−10
0
10
20
30
N = 1666 Bandwidth = 2.487
Trace of alpha[72]
−30
1000
2000
3000
4000
5000
6000
Iterations
Density of alpha[72]
0.000
−40
−20
0
20
N = 1666 Bandwidth = 2.248


Trace of beta[1]
0.56
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[1]
0
15
0.55
0.60
0.65
N = 1666 Bandwidth = 0.005453
Trace of beta[2]
−0.48
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[2]
0
50
−0.48
−0.47
−0.46
−0.45
−0.44
N = 1666 Bandwidth = 0.001141
Trace of beta[3]
−0.22
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[3]
0
15
−0.25
−0.20
−0.15
−0.10
N = 1666 Bandwidth = 0.005556
Trace of beta[4]
0.40
1000
2000
3000
4000
5000
6000
Iterations
Density of beta[4]
0
8
0.40
0.45
0.50
0.55
N = 1666 Bandwidth = 0.007099




```
## Save the coefficients from the model
coefs = data.frame( summary(samp)$statistics )

## Gelman-Rubin convergence diagnostic
gelman.diag(x = samp,
            confidence = 0.95,
            transform = FALSE,
            autoburnin = FALSE)
```

```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## alpha[1]        1.90       3.28
## alpha[2]        4.49       9.42
## alpha[3]        3.65       7.01
## alpha[4]        1.51       2.63
## alpha[5]        1.04       1.14
## alpha[6]        3.78       9.03
## alpha[7]        3.20       6.19
## alpha[8]        1.46       3.19
## alpha[9]        3.67       8.25
## alpha[10]       1.18       1.51
## alpha[11]       1.19       1.55
## alpha[12]       3.05       6.21
```

```
## alpha[13]       2.26       3.97
## alpha[14]       1.48       2.34
## alpha[15]       1.79       3.01
## alpha[16]       8.37      16.61
## alpha[17]       2.69       5.00
## alpha[18]       3.11       6.02
## alpha[19]       1.30       2.01
## alpha[20]       1.18       1.52
## alpha[21]       1.29       1.98
## alpha[22]       1.47       2.57
## alpha[23]       1.51       2.53
## alpha[24]       1.24       1.67
## alpha[25]       1.46       2.21
## alpha[26]       2.20       4.84
## alpha[27]       6.81      21.09
## alpha[28]       1.51       2.54
## alpha[29]       2.48       5.20
## alpha[30]       1.61       2.78
## alpha[31]       1.35       1.99
## alpha[32]       2.33       4.21
## alpha[33]       2.10       3.78
## alpha[34]       1.40       2.50
## alpha[35]       1.07       1.18
## alpha[36]       1.06       1.18
## alpha[37]       1.02       1.07
## alpha[38]       1.30       1.89
## alpha[39]       1.04       1.13
## alpha[40]       1.05       1.16
## alpha[41]       1.27       1.75
## alpha[42]       1.11       1.34
## alpha[43]       1.08       1.21
## alpha[44]       1.23       1.63
## alpha[45]       1.03       1.09
## alpha[46]       1.01       1.03
## alpha[47]       1.01       1.02
## alpha[48]       1.01       1.02
## alpha[49]       1.11       1.36
## alpha[50]       1.46       2.39
## alpha[51]       2.81       5.88
## alpha[52]       2.37       4.14
## alpha[53]       1.42       2.13
## alpha[54]       1.10       1.31
## alpha[55]       1.09       1.29
## alpha[56]       1.08       1.24
## alpha[57]       1.08       1.20
## alpha[58]       1.11       1.34
## alpha[59]       1.09       1.27
## alpha[60]       1.04       1.14
## alpha[61]       1.24       1.67
## alpha[62]       1.25       1.71
## alpha[63]       1.01       1.04
## alpha[64]       1.08       1.25
## alpha[65]       1.11       1.35
## alpha[66]       1.01       1.02
```

```
## alpha[67]       1.02       1.06
## alpha[68]       1.06       1.20
## alpha[69]       1.01       1.04
## alpha[70]       1.01       1.03
## alpha[71]       1.01       1.02
## alpha[72]       1.01       1.01
## beta[1]         2.69       5.03
## beta[2]         1.10       1.33
## beta[3]         1.00       1.00
## beta[4]         2.35       4.17
## beta[5]         1.74       2.99
## sigma           1.00       1.00
##
## Multivariate psrf
##
## 47.8
```

```
## Save the lambda estimates
#var.e4
#var.u4

## Extract the parameter estimates and order them c(fixed, random)
params = c( coefs$Mean[ row.names(coefs) %in% paste0( paste0("beta[",1:5),"]") ],
            coefs$Mean[ row.names(coefs) %in% paste0( paste0("alpha[",1:72),"]") ] )

## Calculate the fitted values
dat$Y_Smooth_Fitted_JAGS_Spline = c(t(params) %*% t(cbind(1,
                                                          G.2,
                                                          G.3,
                                                          X,
                                                          X_2,
                                                          random.basis.1,
                                                          random.basis.2,
                                                          random.basis.3)) )
rm(coefs, model, samp, model_string, random.basis.1, random.basis.2,
   random.basis.3, n, G.2, G.3, K, params)
```

Figure 7: The group-level mean functions using penalized splines and JAGS.

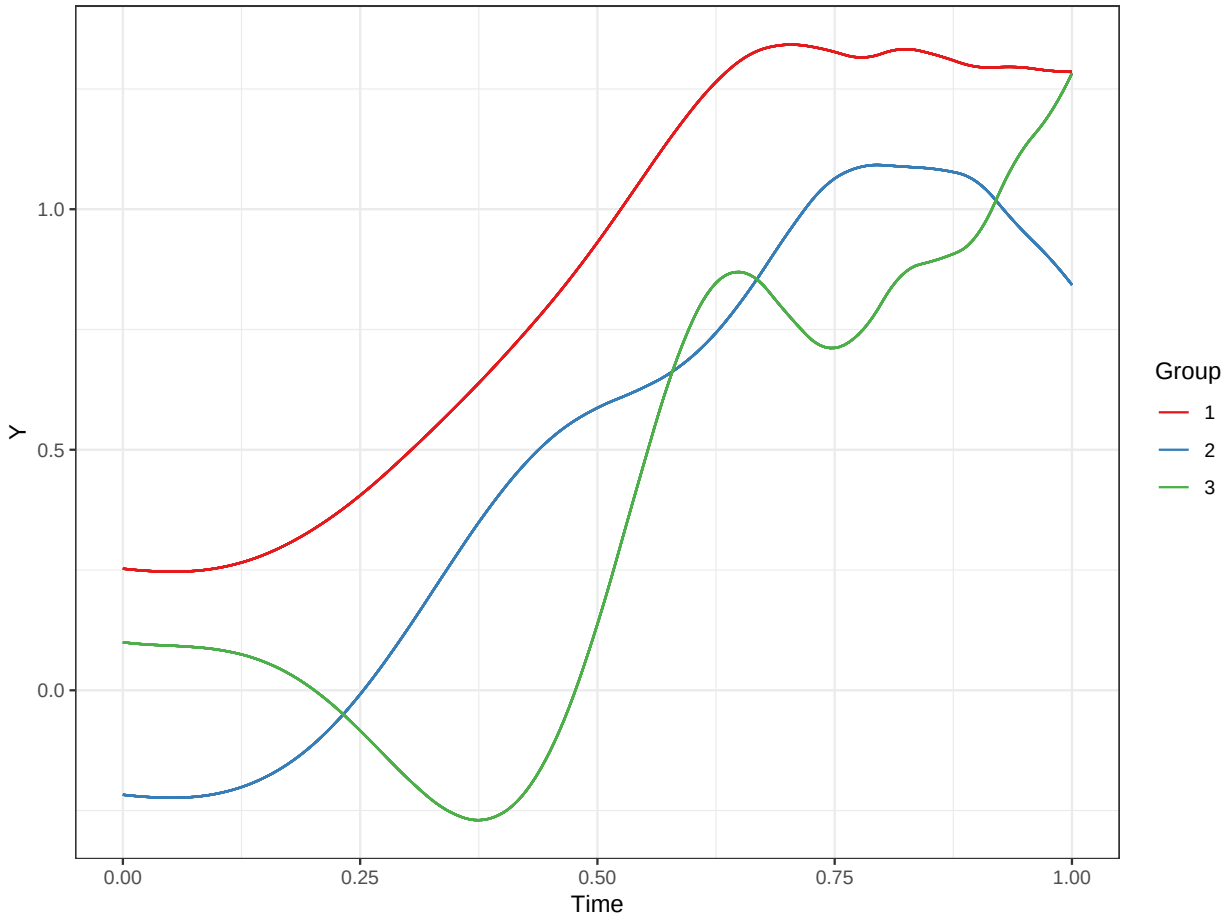
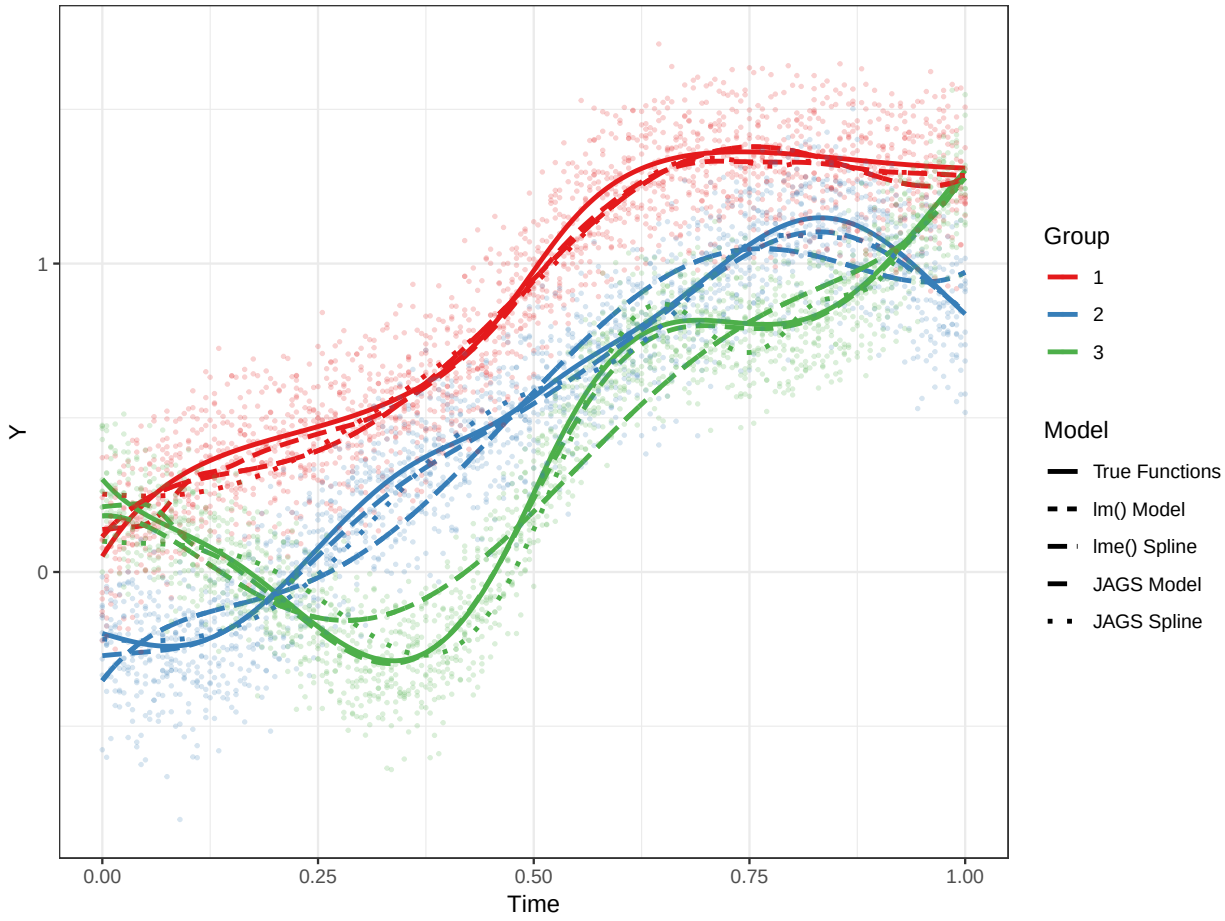

1.0
0.5
0.0
Y
0.00
0.25
0.50
0.75
1.00
Time
Group
1
2
3

Figure 8: A comparison of the fitted effect functions from each model.



Session Info

A summary of the R session used for the analysis.

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] rjags_4-8          coda_0.19-2        lme4_1.1-18-1
## [4] Matrix_1.2-15      nlme_3.1-137       RColorBrewer_1.1-2
## [7] ggplot2_3.1.0      knitr_1.20         MASS_7.3-51.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.0       pillar_1.3.0     compiler_3.5.1   nloptr_1.2.1
##  [5] plyr_1.8.4       bindr_0.1.1      tools_3.5.1      digest_0.6.18
##  [9] evaluate_0.12    tibble_1.4.2     gtable_0.2.0     lattice_0.20-38
## [13] pkgconfig_2.0.2  rlang_0.3.0.1    rstudioapi_0.8   yaml_2.2.0
## [17] bindrcpp_0.2.2   stringr_1.3.1    withr_2.1.2      dplyr_0.7.7
## [21] rprojroot_1.3-2  grid_3.5.1       tidyselect_0.2.5 glue_1.3.0
## [25] R6_2.3.0         rmarkdown_1.10   minqa_1.2.4      purrr_0.2.5
## [29] magrittr_1.5     backports_1.1.2  htmltools_0.3.6  scales_1.0.0
## [33] splines_3.5.1    assertthat_0.2.0 colorspace_1.3-2 labeling_0.3
## [37] stringi_1.2.4    lazyeval_0.2.1   munsell_0.5.0    crayon_1.3.4
```