Esercitazione 10

December 13, 2024

1 Modello Mistura

Prendiamo i dati degli stambecchi (capra Ibex), scaricati da

https://www.movebank.org/cms/webapp?gwt_fragment=page%3Dstudies%2Cpath%3Dstudy1285079529

[59]:
```
set.seed(123)
library(jpeg)
library(png)
d1 <- readPNG("/Users/gianlucamastrantonio/Dropbox (Politecnico di Torino␣
 ↪Staff)/Didattica/statistica computazionale/esercizi/capra.png") # Use␣
 ↪readPNG() for PNG images
#d1 <- readPNG("/Users/gianlucamastrantonio/Dropbox (Politecnico di Torino␣
 ↪Staff)/Didattica/statistica computazionale/esercizi/d1.png") # Use readPNG()␣
 ↪for PNG images
plot(c(0,1), c(0,1), type = "n", xlab = "", ylab = "", xaxt = "n", yaxt = "n",␣
 ↪bty = "n")

rasterImage(d1, 0, 0, 1, 1)
```

[60]:
```
load("/Users/gianlucamastrantonio/Dropbox (Politecnico di Torino Staff)/
↪Didattica/statistica computazionale/datasets/stambecco/stambecco.RData")
summary(data_subset)
```

```
Warning message in load("/Users/gianlucamastrantonio/Dropbox (Politecnico di
Torino Staff)/Didattica/statistica
computazionale/datasets/stambecco/stambecco.RData"):
"le stringhe non rappresentabili nella codifica nativa saranno tradotte in
UTF-8"
```

```
 location.long    location.lat   individual.local.identifier
 Min.   :6.612   Min.   :45.65   Length:8753
 1st Qu.:6.666   1st Qu.:45.71   Class :character
 Median :6.687   Median :45.72   Mode  :character
```

```
 Mean   :6.688   Mean   :45.72
 3rd Qu.:6.702   3rd Qu.:45.73
 Max.   :6.801   Max.   :45.78

      Date                          date_num
 Min.   :2018-07-18 01:11:00.00   Min.   :1.532e+09
 1st Qu.:2018-11-26 09:00:00.00   1st Qu.:1.543e+09
 Median :2019-04-21 18:00:00.00   Median :1.556e+09
 Mean   :2019-05-08 08:14:08.82   Mean   :1.557e+09
 3rd Qu.:2019-10-08 18:00:00.00   3rd Qu.:1.571e+09
 Max.   :2020-04-26 18:00:00.00   Max.   :1.588e+09

                individual
 guerrier_asters      :2954
 tristan_asters       :2947
 petitpied_asters     :2852
 abricot_pne          :   0
 achille_asters       :   0
 afrodite_alpimaritime:   0
 (Other)              :   0
```

il dataset contiene le posizioni a intervalli di quasi regolari della posizioni di 3 stambecchi. La colonna date_num è il numero di secondo passati da una data di riferimento, i dati sono approssimativamente presi a intervalli regolari con qualche missing

[ ]:

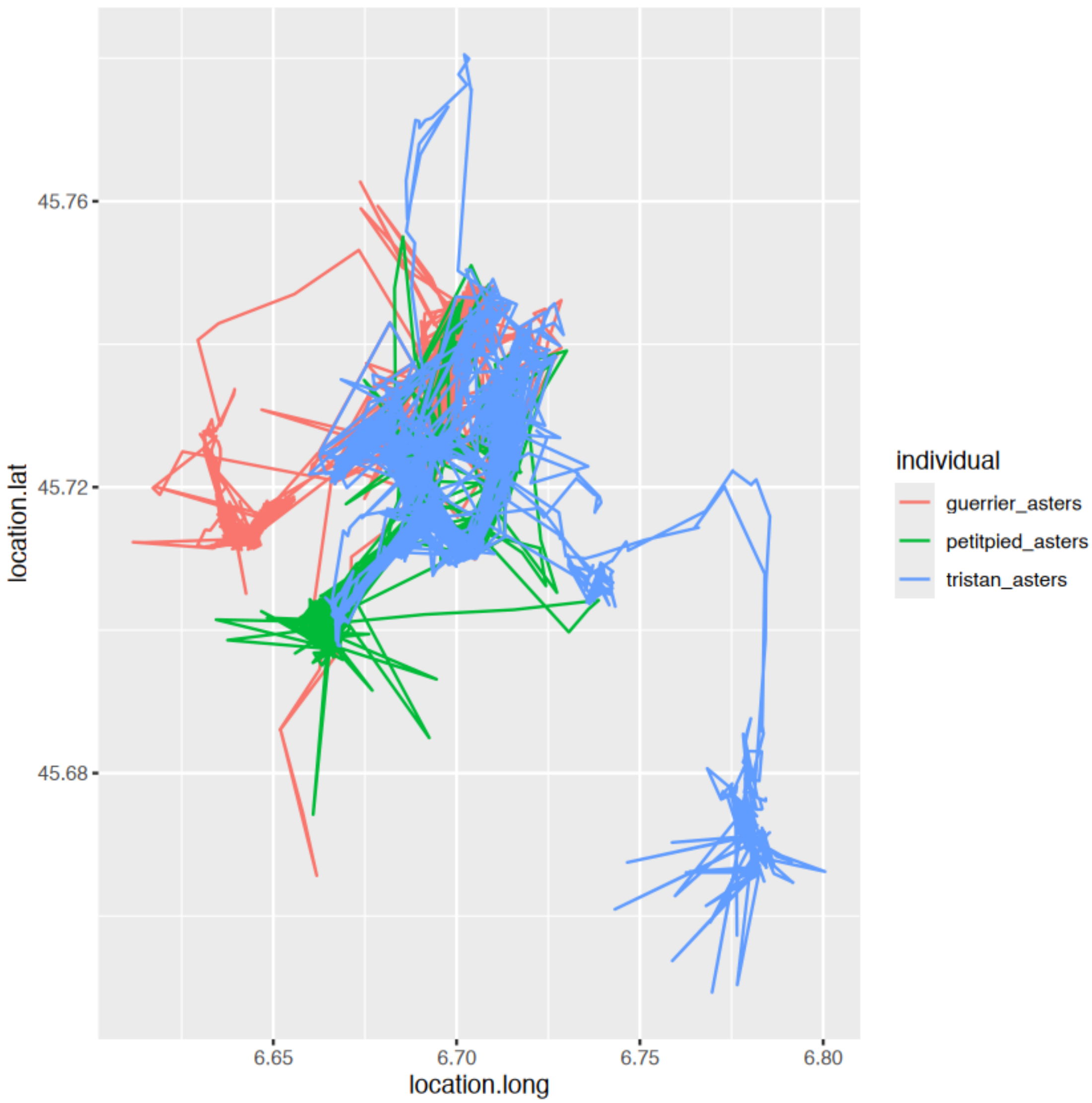
```

```

[ ]:

```

```

[61]:

```
library(ggplot2)
library(tidyverse)
data_subset %>% ggplot(aes(x = location.long, y = location.lat, color =␣
 ↪individual)) +
  geom_path()
```



I tre animali sono stati osservati in intervalli di tempo simili, e per questo possiamo creare un dataset con le righe i punti temporali, e poi due colonna per ogni coordinata dell'animale.

[62]:
```
dataset_1 <- data_subset %>% filter(individual ==␣
 ↪unique(data_subset$individual)[1])
dataset_2 <- data_subset %>% filter(individual ==␣
 ↪unique(data_subset$individual)[2])
dataset_3 <- data_subset %>% filter(individual ==␣
 ↪unique(data_subset$individual)[3])

dataset <- dataset_1 %>%
  full_join(dataset_2, by = "Date",suffix = c(".A1", ".A2")) %>%
  full_join(dataset_3, by = "Date", suffix = c(".sss", ".A3"))
```

```r
#dataset %>%  data_subset%>%
#  pivot_wider(
#    id_cols = time,
#    names_from = individual,
#    values_from = c(location.long, location.lat),
#    names_prefix = "animal_"
#  )
colnames(dataset)

dataset <- dataset[, c("Date", "location.long.A1", "location.lat.A1", "location.
↪long.A2", "location.lat.A2", "location.long", "location.lat")]

colnames(dataset) <- c("Data", "Long1", "Lat1", "Long2", "Lat2", "Long3",␣
↪"Lat3" )
dataset$time <- as.numeric(dataset$Data)/(60*60)
dataset$time <- dataset$time - min(dataset$time) # distanza in ore dalla prima␣
↪misura
summary(dataset)
```

1. 'location.long.A1' 2. 'location.lat.A1' 3. 'individual.local.identifier.A1' 4. 'Date' 5. 'date_num.A1' 6. 'individual.A1' 7. 'location.long.A2' 8. 'location.lat.A2' 9. 'individual.local.identifier.A2' 10. 'date_num.A2' 11. 'individual.A2' 12. 'location.long' 13. 'location.lat' 14. 'individual.local.identifier' 15. 'date_num' 16. 'individual'

```
      Data                             Long1            Lat1      
 Min.   :2018-07-18 01:11:00.0   Min.   :6.634   Min.   :45.67  
 1st Qu.:2018-11-29 23:15:00.0   1st Qu.:6.666   1st Qu.:45.70  
 Median :2019-04-22 04:30:00.0   Median :6.680   Median :45.71  
 Mean   :2019-05-11 08:44:52.9   Mean   :6.680   Mean   :45.71  
 3rd Qu.:2019-10-16 19:30:00.0   3rd Qu.:6.689   3rd Qu.:45.72  
 Max.   :2020-04-26 18:00:00.0   Max.   :6.739   Max.   :45.76  
                                 NA's   :276     NA's   :276    
     Long2            Lat2           Long3            Lat3      
 Min.   :6.660   Min.   :45.65   Min.   :6.612   Min.   :45.67  
 1st Qu.:6.686   1st Qu.:45.70   1st Qu.:6.644   1st Qu.:45.71  
 Median :6.695   Median :45.72   Median :6.661   Median :45.72  
 Mean   :6.712   Mean   :45.71   Mean   :6.670   Mean   :45.72  
 3rd Qu.:6.733   3rd Qu.:45.73   3rd Qu.:6.699   3rd Qu.:45.74  
 Max.   :6.801   Max.   :45.78   Max.   :6.729   Max.   :45.76  
 NA's   :181     NA's   :181     NA's   :174     NA's   :174    
      time      
 Min.   :    0  
 1st Qu.: 3238  
 Median : 6675  
 Mean   : 7136  
 3rd Qu.:10938  
 Max.   :15569  
```

Gli NA in questo caso indicano coordinate mancanti. Adesso prendiamo una sottoserie e facciamo dei plot

[63]:
```
dataset_small <- dataset[1:200, ]
summary(dataset_small)
```

```
      Data                         Long1            Lat1      
 Min.   :2018-07-18 01:11:00.0   Min.   :6.665   Min.   :45.70  
 1st Qu.:2018-07-30 22:30:00.0   1st Qu.:6.691   1st Qu.:45.71  
 Median :2018-08-12 09:00:00.0   Median :6.705   Median :45.72  
 Mean   :2018-08-12 13:32:09.2   Mean   :6.700   Mean   :45.72  
 3rd Qu.:2018-08-25 07:30:00.0   3rd Qu.:6.709   3rd Qu.:45.72  
 Max.   :2018-09-06 18:00:00.0   Max.   :6.730   Max.   :45.74  
                                                                
     Long2            Lat2           Long3            Lat3      
 Min.   :6.680   Min.   :45.71   Min.   :6.691   Min.   :45.71  
 1st Qu.:6.701   1st Qu.:45.72   1st Qu.:6.700   1st Qu.:45.73  
 Median :6.707   Median :45.73   Median :6.702   Median :45.74  
 Mean   :6.706   Mean   :45.73   Mean   :6.703   Mean   :45.74  
 3rd Qu.:6.714   3rd Qu.:45.74   3rd Qu.:6.705   3rd Qu.:45.74  
 Max.   :6.728   Max.   :45.75   Max.   :6.725   Max.   :45.75  
 NA's   :11      NA's   :11      NA's   :24      NA's   :24     
      time       
 Min.   :   0.0  
 1st Qu.: 309.3  
 Median : 607.8  
 Mean   : 612.4  
 3rd Qu.: 918.3  
 Max.   :1216.8  
```

Per esempio il path

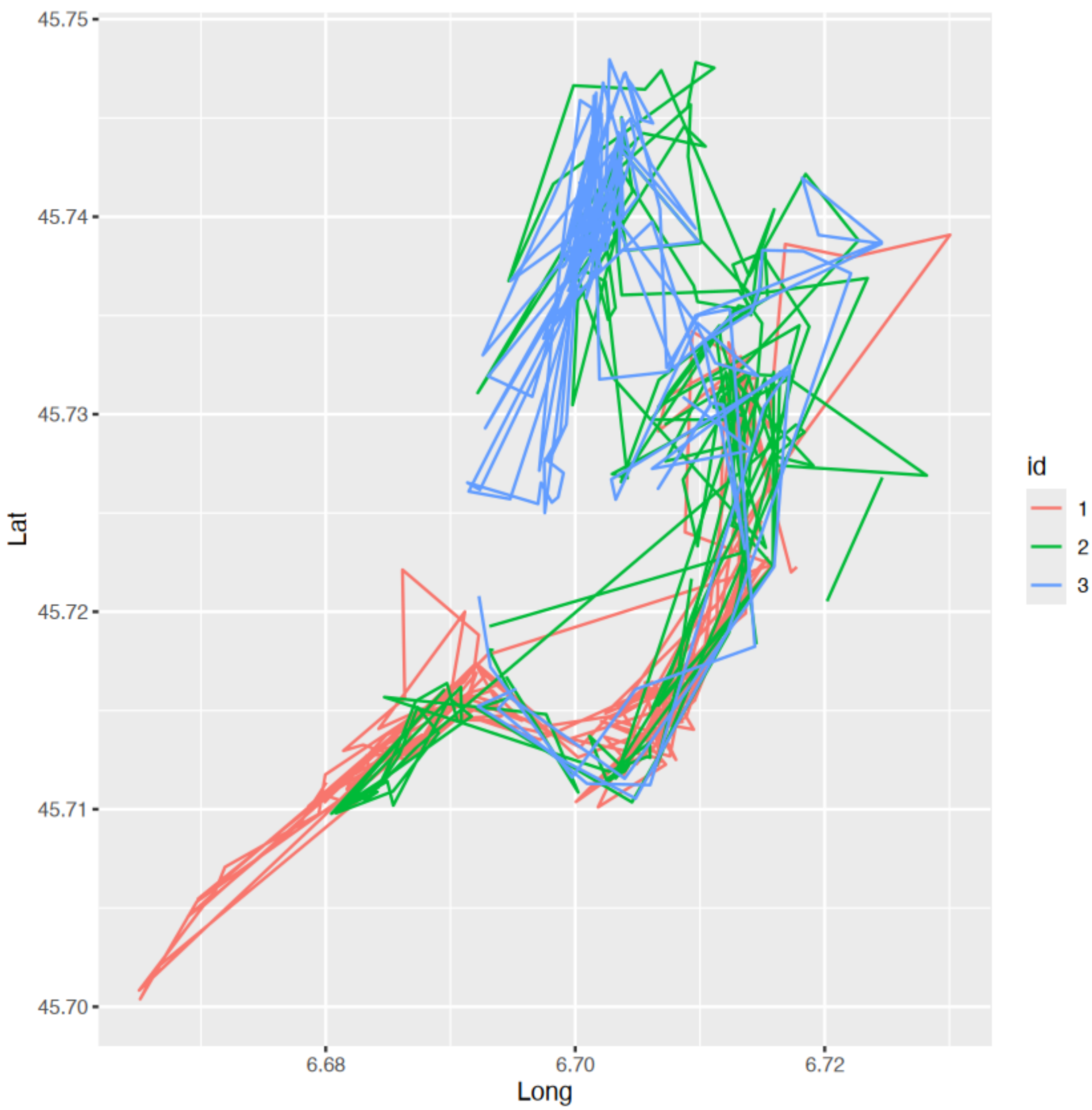
[64]:
```
dataset_plot <- data.frame(id = factor(rep(1:3, each = nrow(dataset_small))), 
 ↪time = dataset_small[, 8], Long = unlist(dataset_small[, c(2, 4, 6)]), Lat = 
 ↪unlist(dataset_small[, c(3, 5, 7)]))
summary(dataset_plot)
```

```
 id            time             Long            Lat       
 1:200   Min.   :   0.0   Min.   :6.665   Min.   :45.70  
 2:200   1st Qu.: 309.3   1st Qu.:6.698   1st Qu.:45.72  
 3:200   Median : 607.8   Median :6.704   Median :45.73  
         Mean   : 612.4   Mean   :6.703   Mean   :45.73  
         3rd Qu.: 918.3   3rd Qu.:6.710   3rd Qu.:45.74  
         Max.   :1216.8   Max.   :6.730   Max.   :45.75  
                          NA's   :35      NA's   :35     
```

[65]:

```r
# install.packages("Ecdat")
library(Ecdat)
# install.packages("tidyverse")
library(tidyverse)
# install.packages("gganimate")
library(gganimate)
# install.packages("remotes")
# remotes::install_github("R-CoderDotCom/ggcats@main")
library(ggcats)
cat_name <- c(
  "nyancat", "bongo",
  "colonel", "grumpy",
  "hipster", "lil_bub",
  "maru", "mouth",
  "pop", "pop_close",
  "pusheen", "pusheen_pc",
  "toast", "venus",
  "shironeko"
)
dataset_plot$cats <- rep(cat_name[c(1, 11, 10)], each = length(dataset_plot))

dataset_plot %>%  ggplot(aes(x = Long, y = Lat,  group = id, col=id)) +
  geom_path()
  #+
  #geom_cat(aes(cat = cats), size = 5) +
  # transition_reveal(time)
```

```
Warning message:
"Removed 22 rows containing missing values or values outside the scale
range
(`geom_path()`)."
```



1.1 Modello sulle coordinate

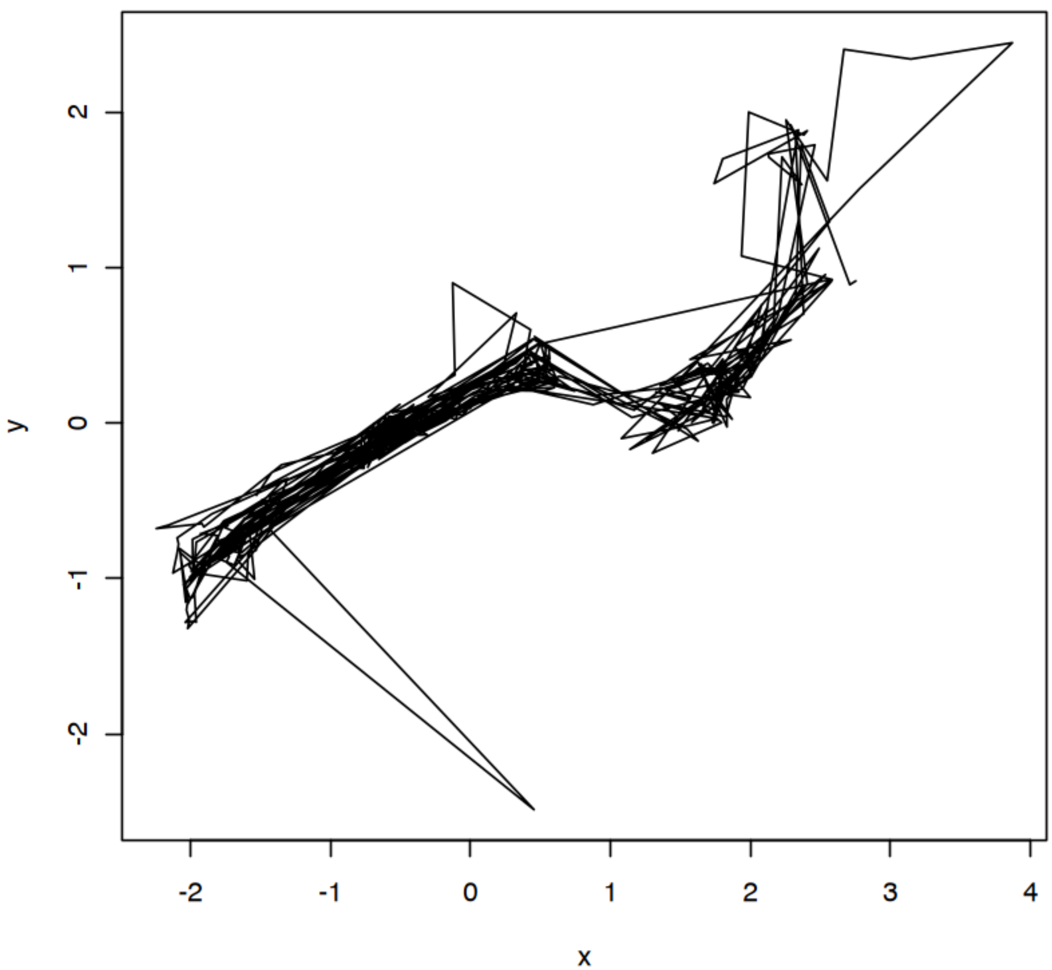
Prendete un qualsiasi dei tre animali e proviamo a stimare un modello mistura sulle coordinate. Indichiamo con

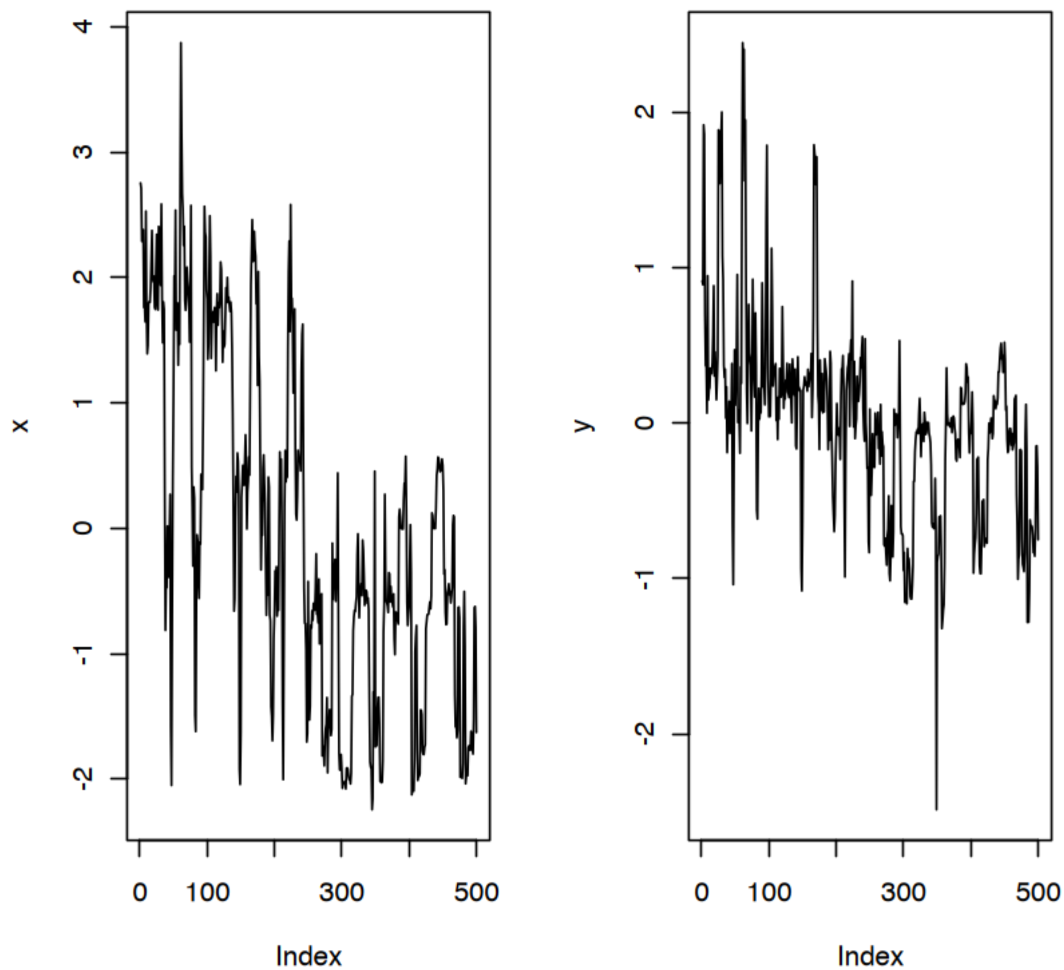
$$(x_i, y_i)$$

le coordinate dell'i-esimo tempo. Per semplicità le coordinate vengono standardizzate e prendiamo un subset dei dati. Fate attenzione che potete cambiare liberamente il valor medio delle cordinate, ma non la varianza, altrimenti state cambiando i rapporti tra le due coordinate

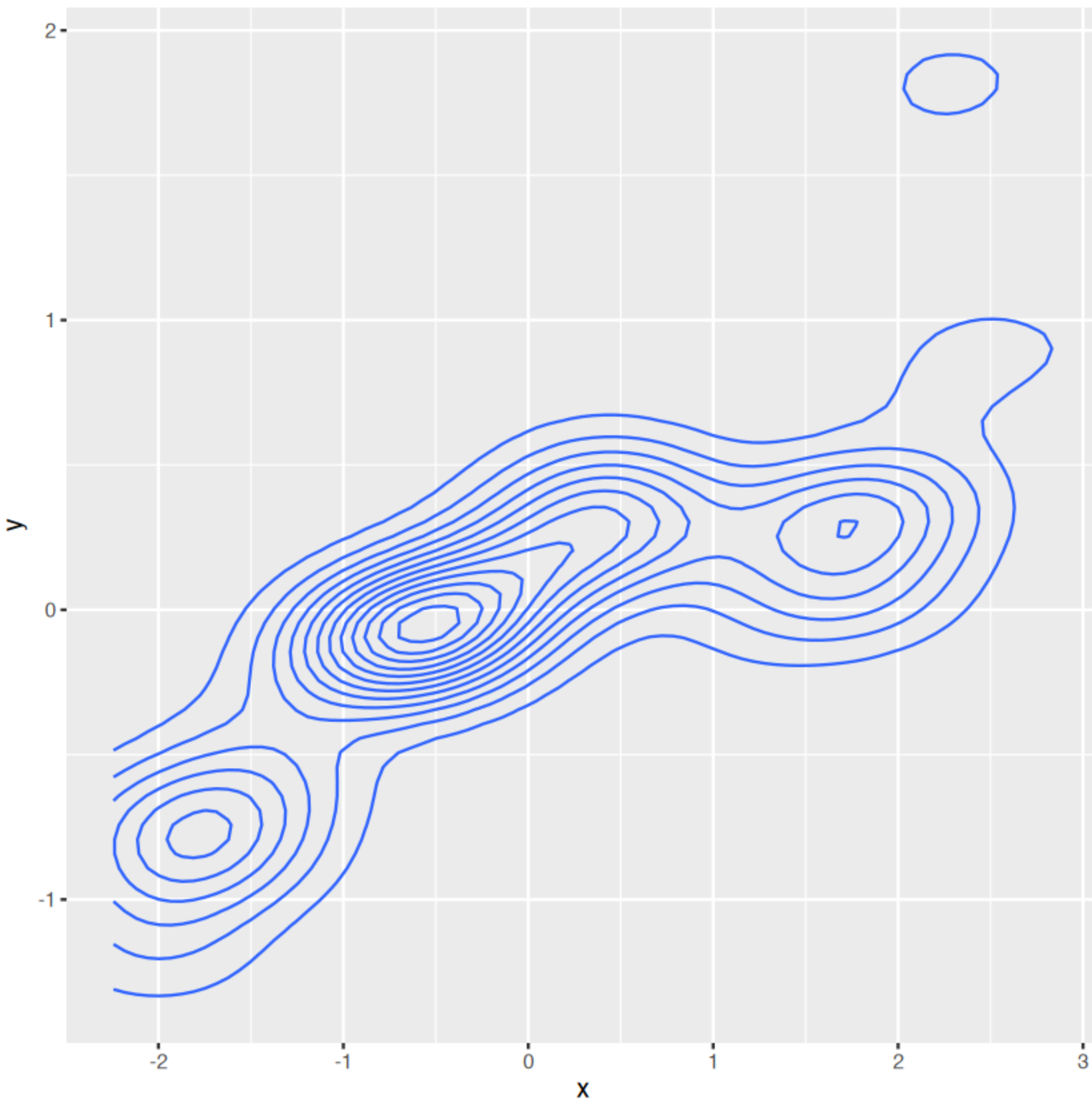
[66]:
```
ani <- 1
dataset_model <- dataset[1:500, ]
x <- dataset_model[, (ani - 1) * 2 + 2]
y <- dataset_model[, (ani - 1) * 2 + 3]
```

```
sd_xy <- 0.5 * sd(x, na.rm = T) + 0.5 * sd(y, na.rm = T)
x <- (x - mean(x, na.rm = T)) / sd_xy
y <- (y - mean(y, na.rm = T)) / sd_xy
plot(x, y, type="l")
par(mfrow=c(1,2))
plot(x, type="l")
plot(y, type = "l")
par(mfrow = c(1, 1))
```





[67]:
```
data.frame(x = x, y = y) %>% ggplot(aes(x = x, y = y)) +
  geom_density_2d()
```



chiaramente le serie temporali delle coordinate hanno una persistenza temporale, ma delle volte cambiano di valore in maniera improvvisa.

Il modello che stimiamo è

$$x_i|z_i \sim N(\mu_{x,z_i}, \sigma^2_{x,z_i}) \quad i = 1, \dots, n$$

$$y_i|z_i \sim N(\mu_{y,z_i}, \sigma^2_{y,z_i}) \quad i = 1, \dots, n$$

con

$$z_i|z_{i-1} \sim Discrete(\pi_{z_{i-1}}) \quad i = 1, \dots, n$$

I parametri sono le medie di x e y, le varianza, z_0 e la matrice di transizione

IN NIMBLE, il modello si scrive come JAGS. Differentemente da JAGS, questo viene compilato, e è possibile inserire densità che non sono presenti nel pacchetto base, scrivendo del codice.

[68]:
```
library(nimble)

mixture_model <- nimbleCode({

  z0 ~ dcat(prob_init[1:K])
  z[1] ~ dcat(prob[z0,1:K])

  x[1] ~ dnorm(mu_x[z[1]], sd = sqrt(sigma2_x[z[1]]))
  y[1] ~ dnorm(mu_y[z[1]], sd = sqrt(sigma2_y[z[1]]))
  for (i in 2:n) {
    z[i] ~ dcat(prob[z[i-1], 1:K])

    x[i] ~ dnorm(mu_x[z[i]] , sd = sqrt(sigma2_x[z[i]]))
    y[i] ~ dnorm(mu_y[z[i]] , sd = sqrt(sigma2_y[z[i]]))

  }

  prob_init[1:K] ~ ddirch(par_dir[1:K])
  for (j in 1:K)
  {
    prob[j,1:K] ~ ddirch(par_dir[1:K])
    mu_x[j] ~ dnorm(0,sd=10)
    mu_y[j] ~ dnorm(0, sd = 10)
    prec_x[j] ~ dgamma(1, 1)
    sigma2_x[j] <- 1 / prec_x[j]
    prec_y[j] ~ dgamma(1, 1)
    sigma2_y[j] <- 1 / prec_y[j]


  }
})
```

Per l'implementazione del codice assumiamo di sapere che il numero di cluster sia $K = 3$

[69]:
```
K <- 3
n <- length(x)
constants <- list(
  n = n,
  K = K,
  par_dir = rep(1, K)
)

# Data
data <- list(
  x = x,
  y = y
)
```

```r
# Initial values for the parameters
inits <- list(
  prob_init = rep(1/K, K),
  prob = matrix(1/K, nrow = K, ncol = K),
  mu_x = runif(K,-1,1),
  mu_y = runif(K,-1,1),
  prec_x = runif(K, 0.2, 2),
  prec_y = runif(K, 0.2, 2),
  sigma2_x = runif(K, 0.01, 0.02),
  sigma2_y = runif(K, 0.01, 0.02),
  #alpha = rep(0.1, K),
  z0 = 1,
  z = sample(1:K, n, replace = TRUE)
)


# Build the model
model <- nimbleModel(
  mixture_model,
  data = data,
  constants = constants,
  inits = inits
)
```

```
Defining model

Building model

Setting data and initial values

Running calculate on model
  [Note] Any error reports that follow may simply reflect missing values in
model variables.

Checking model sizes and dimensions

  [Note] This model is not fully initialized. This is not an error.
         To see which variables are not initialized, use model$initializeInfo().
         For more information on model initialization, see
help(modelInitialization).
```

Dobbiamo compilare il modello, settare che parametri vogliamo salvare, e runnare l'algoritmo.

[70]:
```r
## Compile the model
compileNimble(model)
# Configure the MCMC
```

```
mcmc <- buildMCMC(model, monitors = c("mu_x", "mu_y", "sigma2_x",  "sigma2_y", ␣
↪"z", "prob"), WAIC = TRUE, enableWAIC = T)

Cmcmc <- compileNimble(mcmc)
# Run the MCMC
mod_mix <- runMCMC(Cmcmc, niter = 5000, nburnin = 2000)
```

```
Compiling
  [Note] This may take a minute.
  [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.


Derived CmodelBaseClass created by buildModelInterface for model mixture_mo_MID_4

===== Monitors =====
thin = 1: mu_x, mu_y, prob, sigma2_x, sigma2_y, z
===== Samplers =====
conjugate sampler (10)
  - mu_x[]  (3 elements)
  - mu_y[]  (3 elements)
  - prob_init[1:3]
  - prob[1, 1:3]
  - prob[2, 1:3]
  - prob[3, 1:3]
categorical sampler (501)
  - z0
  - z[]  (500 elements)
RW sampler (6)
  - prec_x[]  (3 elements)
  - prec_y[]  (3 elements)

Compiling
  [Note] This may take a minute.
  [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.

  [Warning] To calculate WAIC, set 'WAIC = TRUE', in addition to having enabled
WAIC in building the MCMC.

running chain 1…


|-------------|-------------|-------------|-------------|
|-------------------------------------------------------|
```

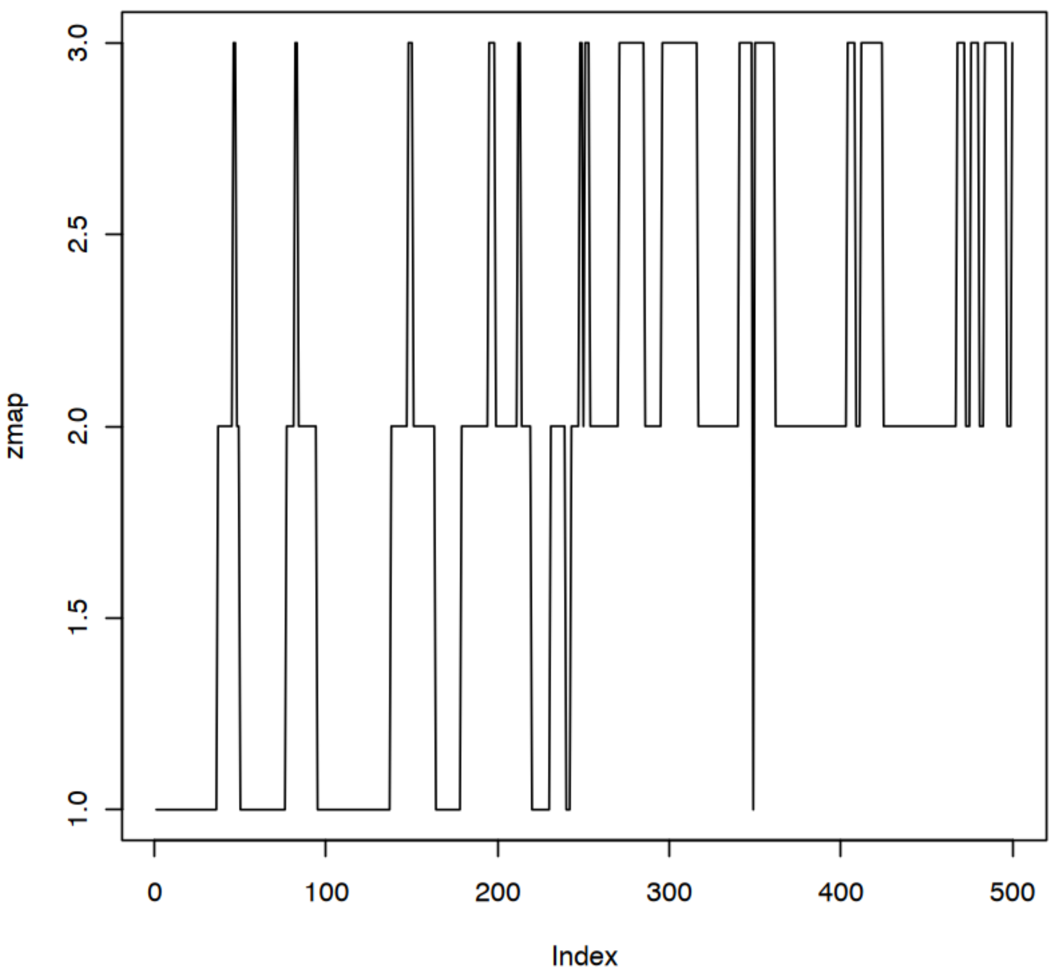
La prima cosa che possiamo fare è avere uan stima di z, ottenuta come la moda a posteriori

[71]:
```
library(coda)
findmode <- function(x) {
  TT <- table(as.vector(x))
  return(as.numeric(names(TT)[TT == max(TT)][1]))
```

```
}
z_samples <- mod_mix[, grep("^z", colnames(mod_mix))]
zmap <- apply(z_samples, 2, findmode)


#par_samples <- mod4[, -grep("^z", colnames(mod4))]
#summary(as.mcmc(par_samples))


#q1 <- apply(par_samples, 2, function(x) quantile(x, 0.0275))
#q2 <- apply(par_samples, 2, function(x) quantile(x, 1 - 0.0275))
```

e poi ne plottiamo la serie storica

[72]:
```
plot(zmap, type="l")
```



Possiamo guardare le stima della matrice di transizione (medie a posteriori)

[73]:
```
prob_samples <- mod_mix[, grep("^prob", colnames(mod_mix))]
#str(prob_samples)
prob_mean = colMeans(prob_samples)
matrix(prob_mean, ncol=K)
```

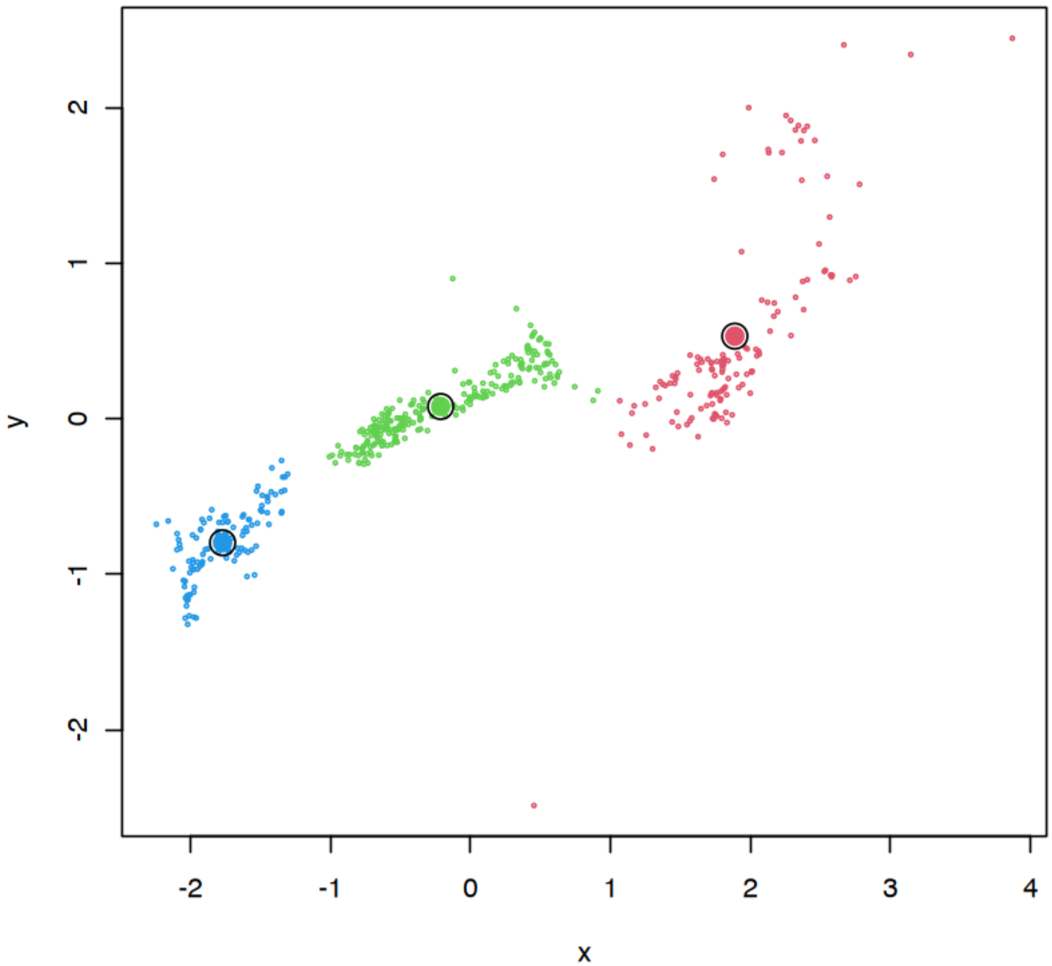
A matrix: 3 x 3 of type dbl

0.93558235	0.05004413	0.01437352
0.02405343	0.90784745	0.06809912
0.01754563	0.13637688	0.84607749

la matrice di transizione mostra che c'è molta persistenza.

Vediamo che le coordinate medie sono state ben stimate e confrontiamola con i valori delle stime di z

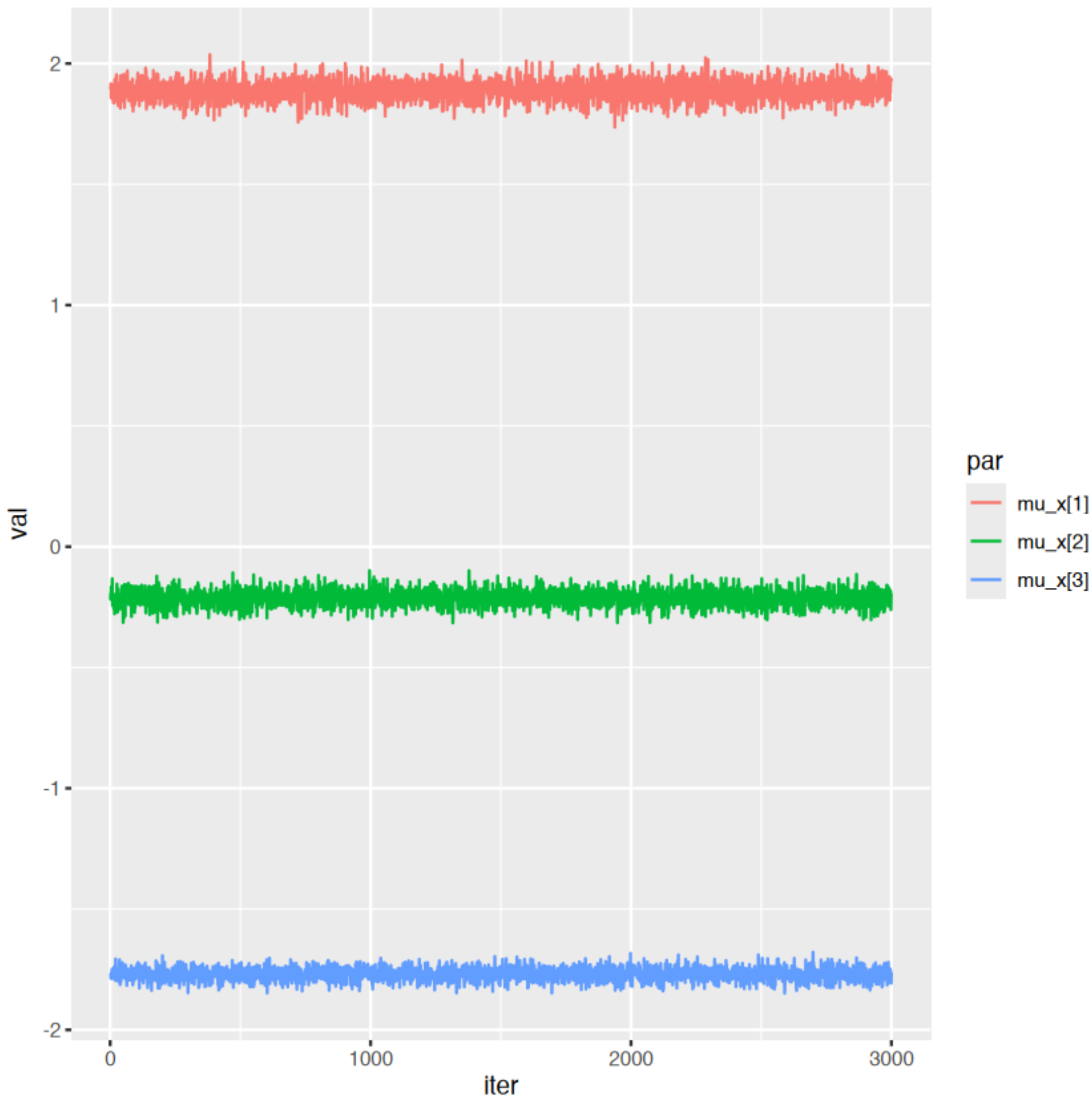
[74]:
```
mu_x_samples <- mod_mix[, grep("^mu_x", colnames(mod_mix))]
mu_y_samples <- mod_mix[, grep("^mu_y", colnames(mod_mix))]

plot(x, y, col = zmap+1, cex = 0.2)
points(colMeans(mu_x_samples), colMeans(mu_y_samples), col=2:4, pch=20, cex=2)
points(colMeans(mu_x_samples), colMeans(mu_y_samples), cex = 2)
```

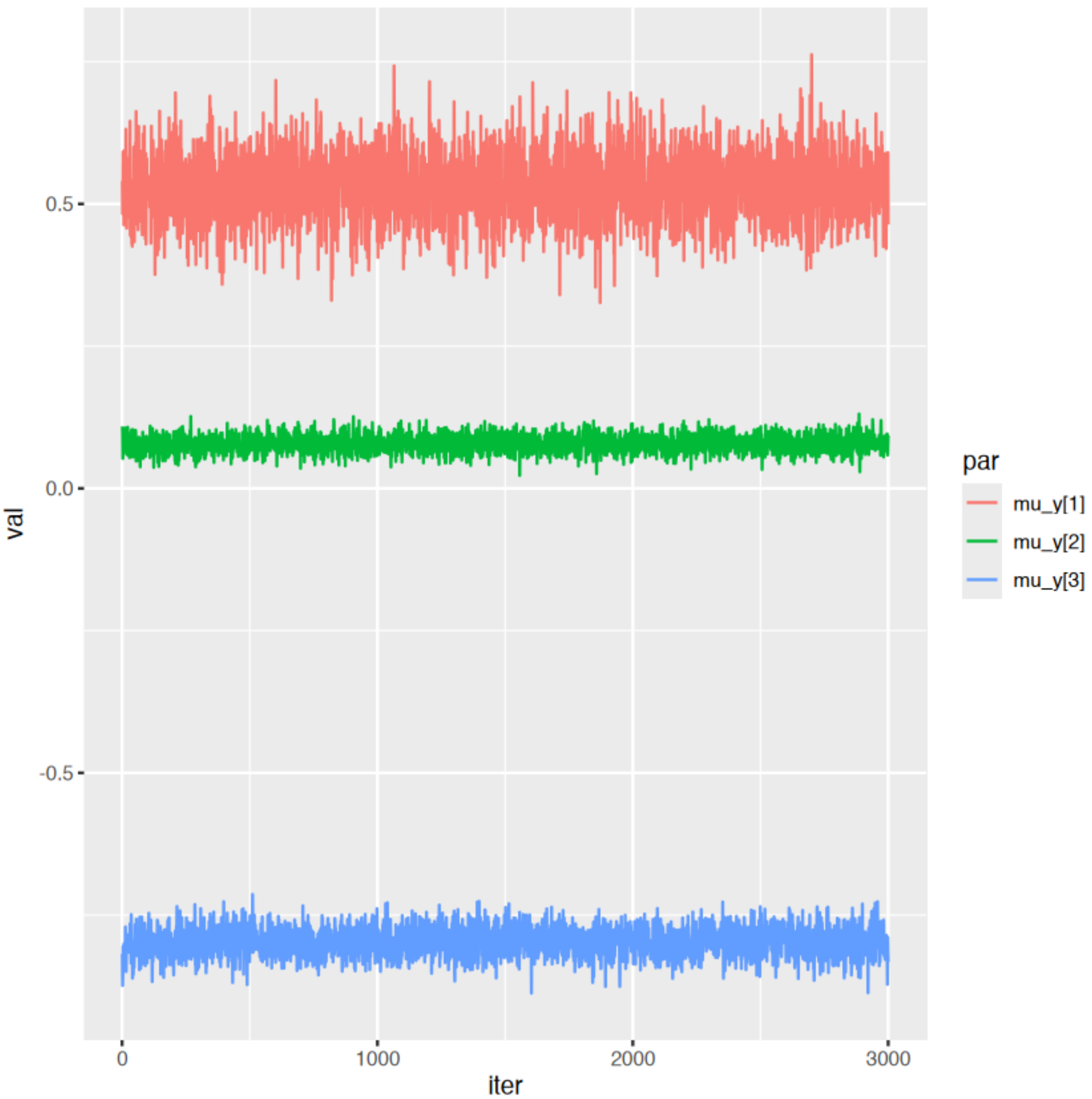


vediamo anche le catene di alcuni parametri

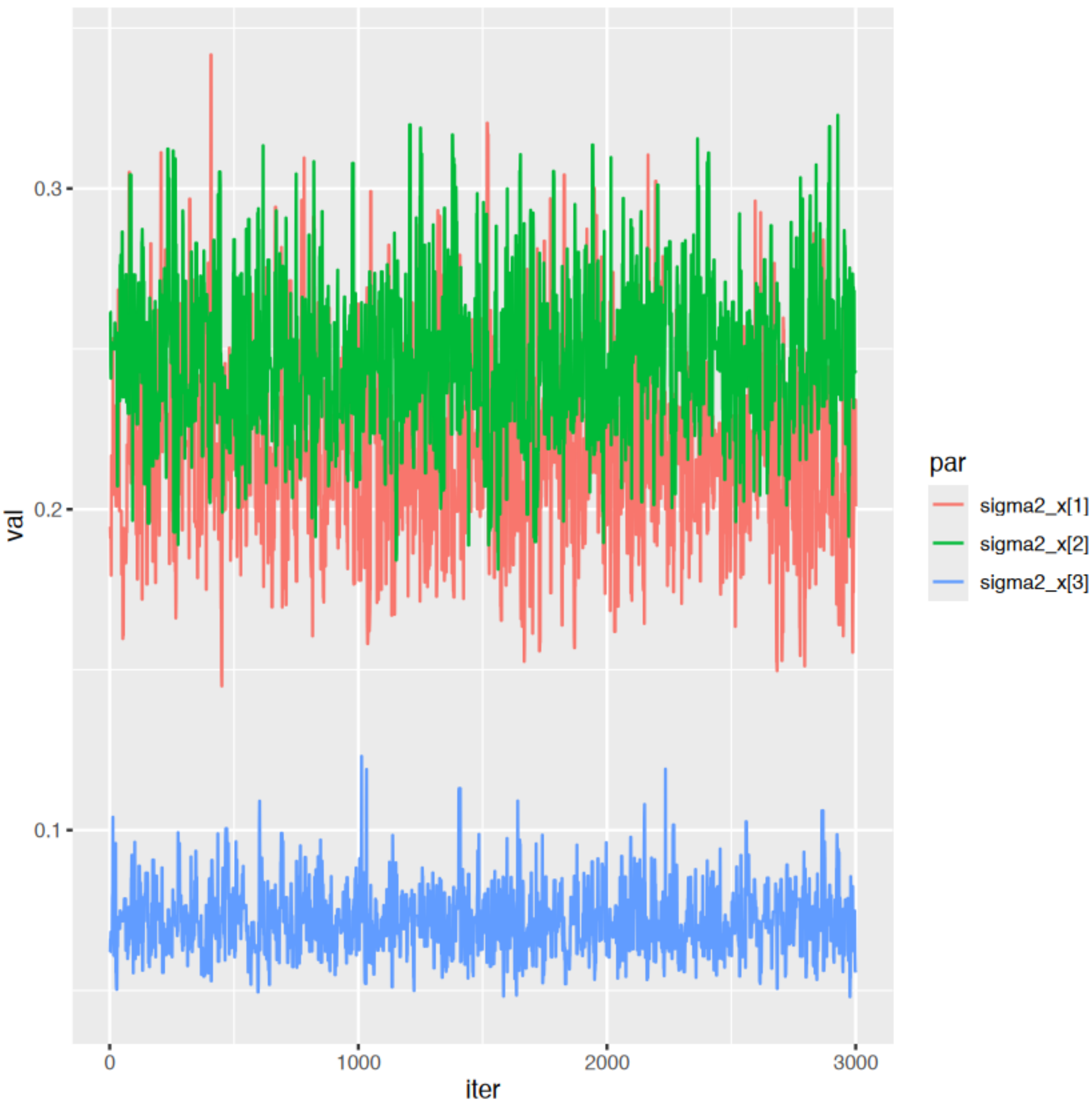
[75]:
```
mu_x_samples <- mod_mix[, grep("^mu_x", colnames(mod_mix))]
mu_x_mcmc <- mu_x_samples %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(mu_x_samples)) %>%
  pivot_longer(
    cols = 1:3,
    names_to = "par",
    values_to = "val"
  )
mu_x_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()
```

```r
mu_y_samples <- mod_mix[, grep("^mu_y", colnames(mod_mix))]
mu_y_mcmc <- mu_y_samples %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(mu_x_samples)) %>%
  pivot_longer(
    cols = 1:3,
    names_to = "par",
    values_to = "val"
  )
mu_y_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()


sigma2_x_samples <- mod_mix[, grep("^sigma2_x", colnames(mod_mix))]
sigma2_x_mcmc <- sigma2_x_samples %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(mu_x_samples)) %>%
  pivot_longer(
    cols = 1:3,
    names_to = "par",
    values_to = "val"
  )
sigma2_x_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()


sigma2_y_samples <- mod_mix[, grep("^sigma2_y", colnames(mod_mix))]
sigma2_y_mcmc <- sigma2_y_samples %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(mu_x_samples)) %>%
  pivot_longer(
    cols = 1:3,
    names_to = "par",
    values_to = "val"
  )
sigma2_y_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()
```

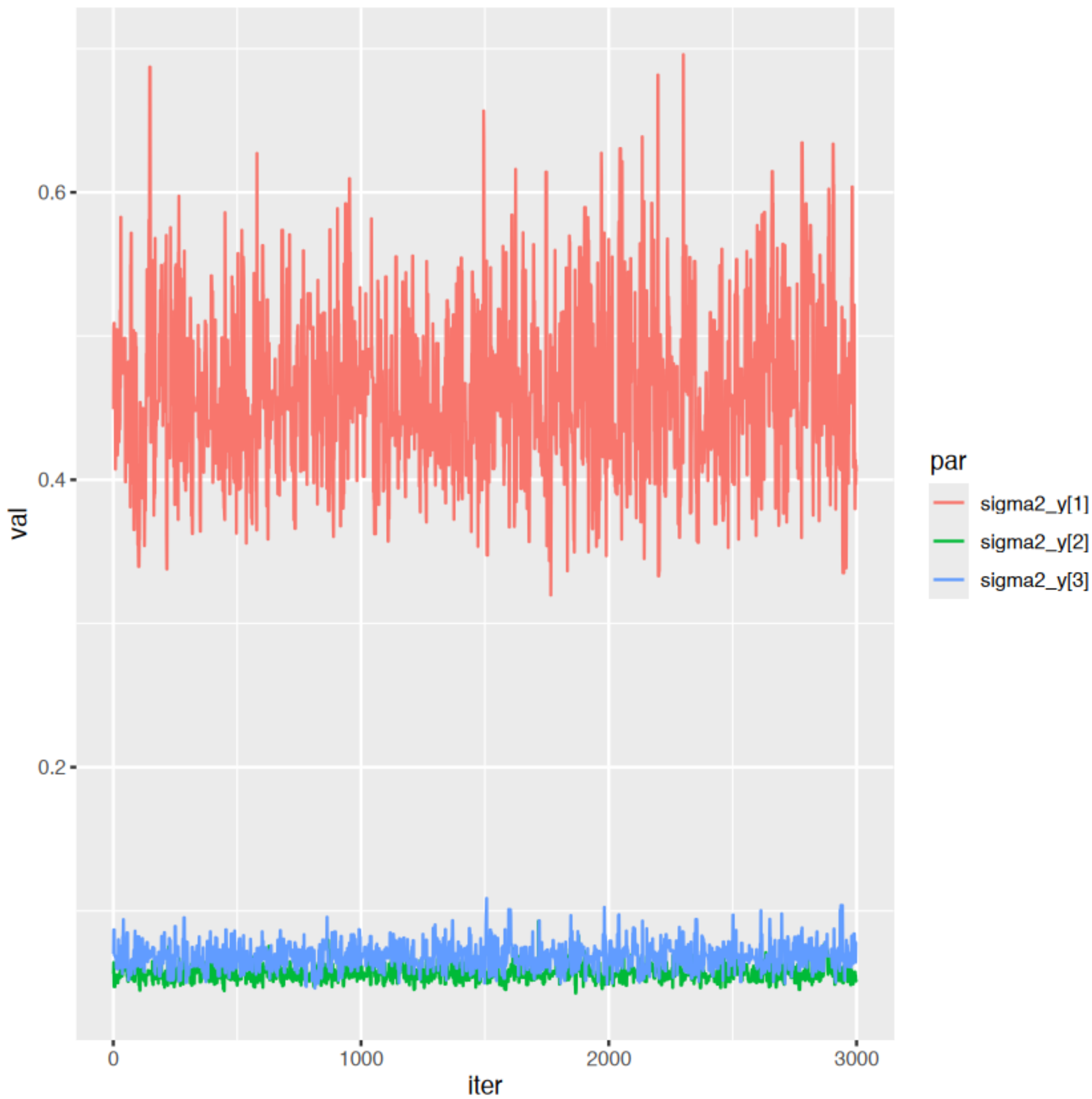

2
1
0
-1
-2
val
0
1000
2000
3000
iter
par
mu_x[1]
mu_x[2]
mu_x[3]



0.5
0.0
-0.5
val
0
1000
2000
3000
iter
par
mu_y[1]
mu_y[2]
mu_y[3]



0.3
0.2
0.1
val
0
1000
2000
3000
iter
par
sigma2_x[1]
sigma2_x[2]
sigma2_x[3]



1.2 Secondo modello

In questo secondo modello calcoliamo la step-length

$$r_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

il bearing angle

$$\psi_i = atan^*(y_{i+1} - y_i, x_{i+1} - x_i)$$

e il turning angle

$$\theta_i = \psi_i - \psi_{i-1}$$

Il primo è una proxy della velocità, il secondo è l'angolo rispetto al nord, mentre il terzo è l'angolo di movimento rispetto all'ultima direzione. La funzione $atan^*$, che viene anche chiamata come atan2, è un inversa della finzione tangent

$$\psi_i = tan^{-1}\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$$

che tiene conto del numeratore e denominatore per definire l'angolo nell'opportuno quadrante, visto che $ tan^{-1}(.) (- /2, /2)$, ma $\psi_i in[0, 2\pi)$

Modellizziamo step-length e turning angle, assumendo

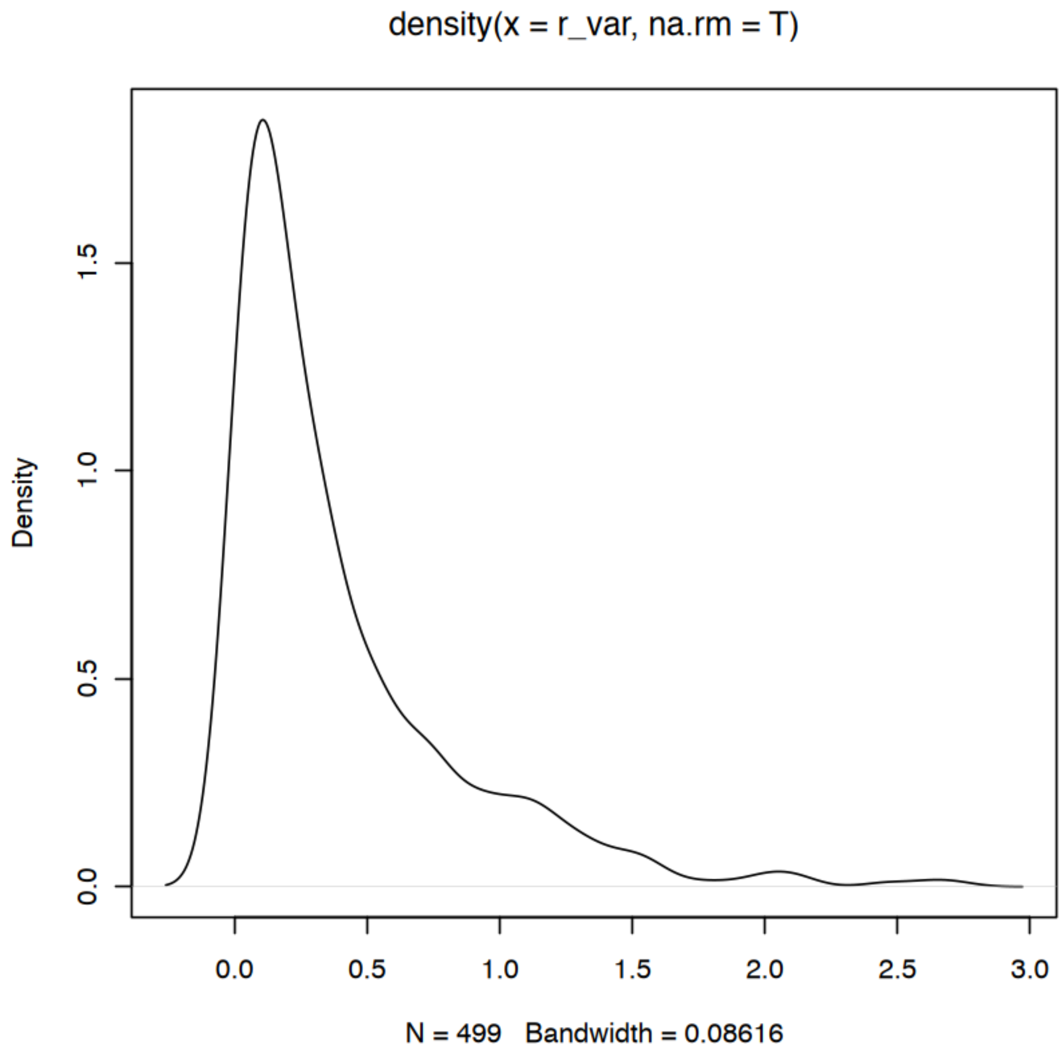
$$r_i|z_i \sim G(a_{z_i}, b_{z_i})$$

$$\theta_i \sim WC(\rho_{z_i}, \tau_{z_i})$$

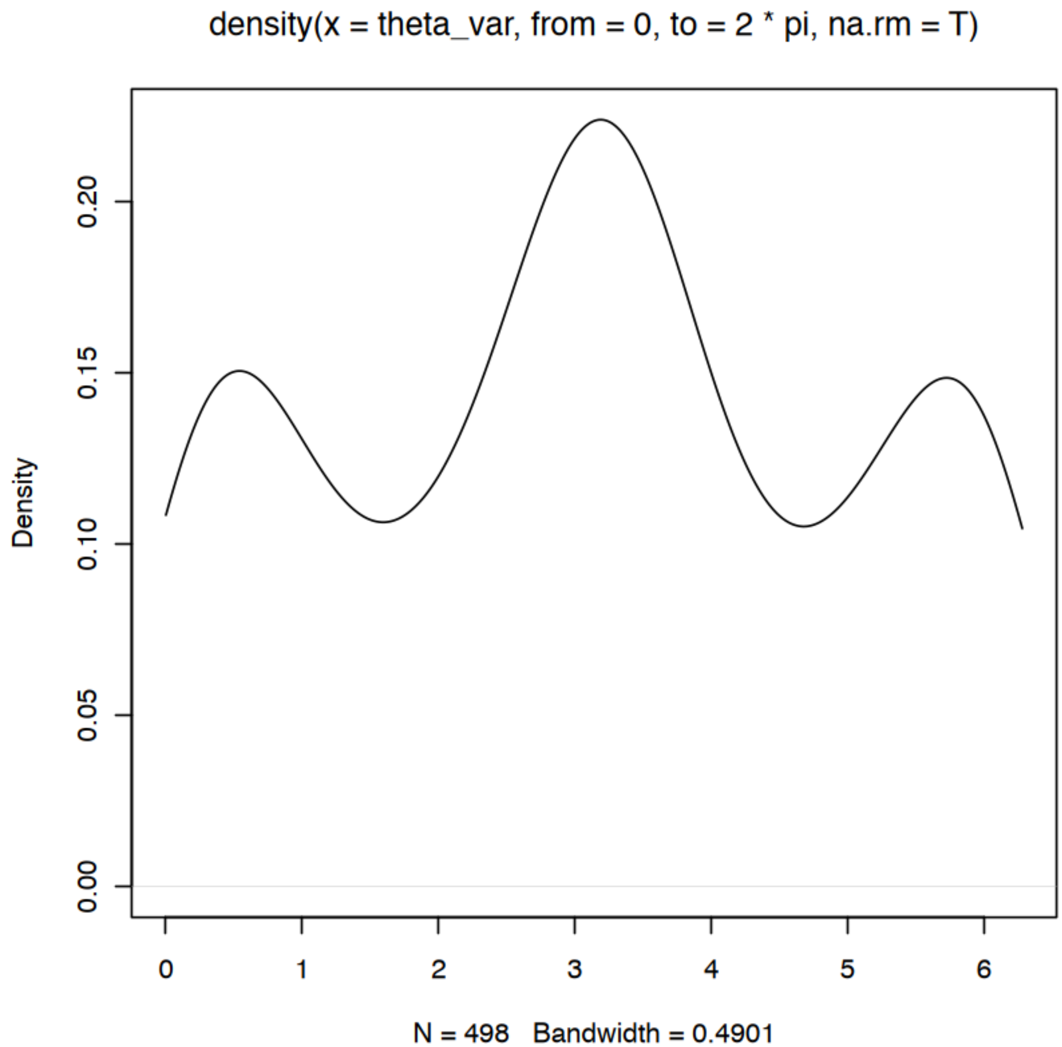
In questo modello stiamo ignorando la parte spaziale e ci curiamo solo del movimento

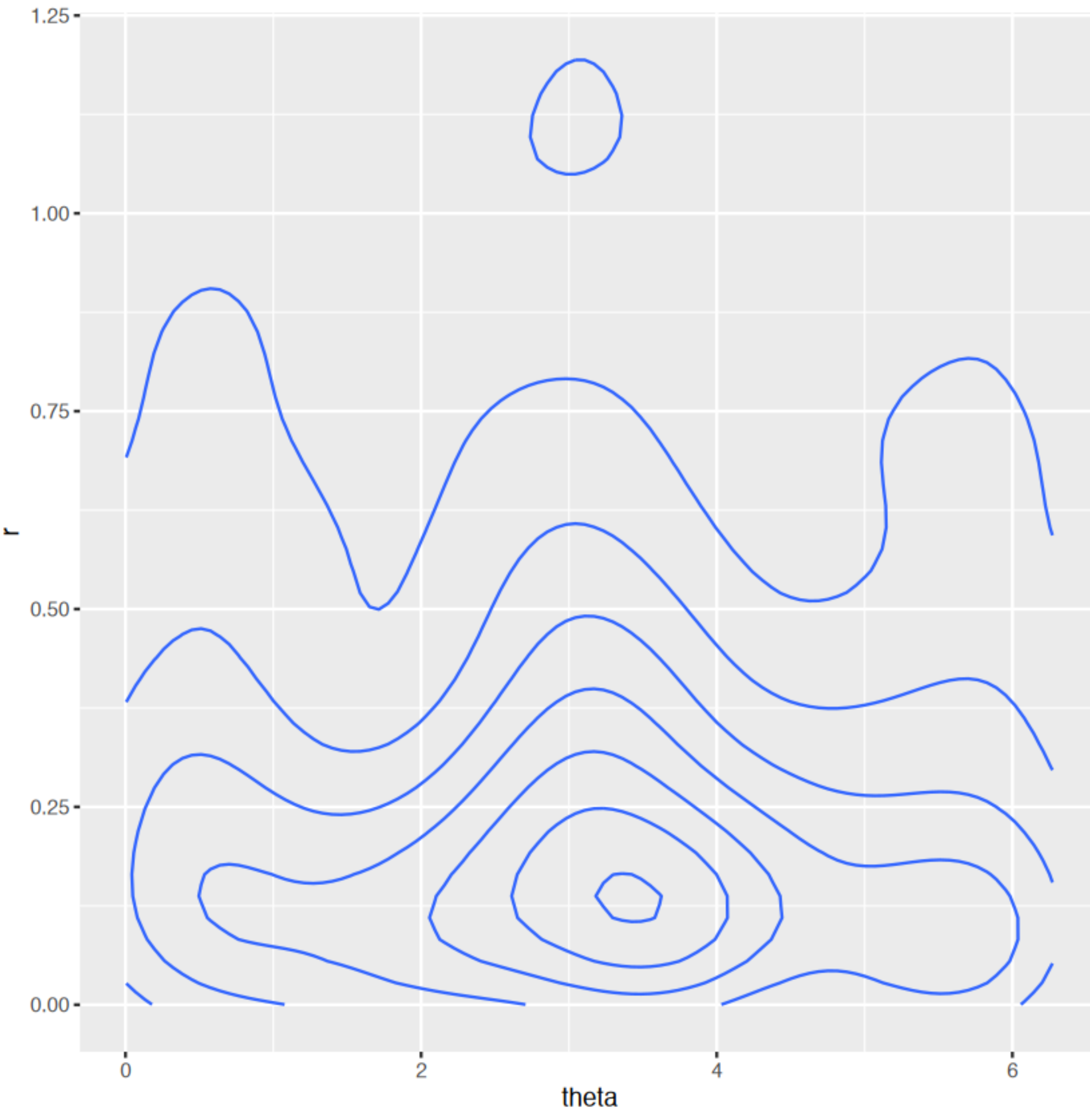
Creiamo le variabili

[76]:
```
r_var <- sqrt(diff(x)^2 + diff(y)^2)
phi_var <- atan2(diff(y), diff(x))
theta_var <- c(NA, diff(phi_var))%%(2*pi)
plot(density(r_var, na.rm=T))
plot(density(theta_var, na.rm = T, from = 0, to = 2 * pi), ylim = c(0, max(density(theta_var, na.rm = T)$y)))


data.frame(theta = theta_var, r = r_var) %>% ggplot(aes(x = theta, y = r)) +
  geom_density_2d()
```




```
Warning message:
"Removed 1 row containing non-finite outside the scale range
(`stat_density2d()`)."
```


density(x = theta_var, from = 0, to = 2 * pi, na.rm = T)
Density
0.00
0.05
0.10
0.15
0.20
0
1
2
3
4
5
6
N = 498   Bandwidth = 0.4901



La Wrapped cauchy non esiste in nimble, dobbiamo implementare la dua densità e un campionatore

[77]:
```
dwrappedcauchy <- nimbleFunction(
  run = function(x = double(0), mu = double(0, default = 0), prec = double(0,␣
↪default = 1),
                log = integer(0, default = 0)) {
    returnType(double(0))
    logProb <- log(sinh(prec)) - log(cosh(prec) - cos(x - mu)) - log(2.0 * pi)
    if (log) {
      return(logProb)
    } else {
      return(exp(logProb))
    }
```

```
  }
)
rwrappedcauchy <- nimbleFunction(
  run = function(n = integer(0), mu = double(0, default = 0), prec = double(0,␣
↪default = 1)) {
    returnType(double(0))
    if (n != 1) print("rmyexp only allows n = 1; using n = 1.")
    u <- runif(1, 0, 1)

    return(mu + prec * tan(pi * (u - 0.5)))
  }
)
```

[78]:

```
mixture_model2 <- nimbleCode({
  z0 ~ dcat(prob_init[1:K])
  z[1] ~ dcat(prob[z0, 1:K])

  r_var[1] ~ dgamma(shape= a_par[z[1]], rate=b_par[z[1]])
  theta_var[1] ~ dwrappedcauchy(rho[z[1]], prec = tau[z[1]])
  for (i in 2:n) {
    z[i] ~ dcat(prob[z[i - 1], 1:K])

    r_var[i] ~ dgamma(shape = a_par[z[i]], rate = b_par[z[i]])
    theta_var[i] ~ dwrappedcauchy(rho[z[i]], prec = tau[z[i]])
  }

  prob_init[1:K] ~ ddirch(par_dir[1:K])
  for (j in 1:K)
  {
    prob[j, 1:K] ~ ddirch(par_dir[1:K])

    a_par[j] ~ dgamma(1, 1)
    b_par[j] ~ dgamma(1, 1)
    tau[j] ~ dunif(0,1)
    rho[j] ~ dunif(0, 2 * const_pi)


  }
})
```

anche in questo caso assumiamo 3 cluster

[79]:

```
K <- 3
n <- length(r_var)
constants <- list(
  n = n,
  K = K,
```

```r
  par_dir = rep(1, K),
  const_pi = pi
)

# Data
data <- list(
  r_var = r_var,
  theta_var = theta_var
)

# Initial values for the parameters
inits <- list(
  prob_init = rep(1 / K, K),
  prob = matrix(1 / K, nrow = K, ncol = K),
  a_par = runif(K,0.5,1.5),
  b_par = runif(K,0.5,1.5),
  tau = runif(K,0.5,0.9),
  rho = runif(K,0.5,0.9),
  z0 = 1,
  z = sample(1:K, n, replace = TRUE)
)


# Build the model
model_dir <- nimbleModel(
  mixture_model2,
  data = data,
  constants = constants,
  inits = inits
)
```

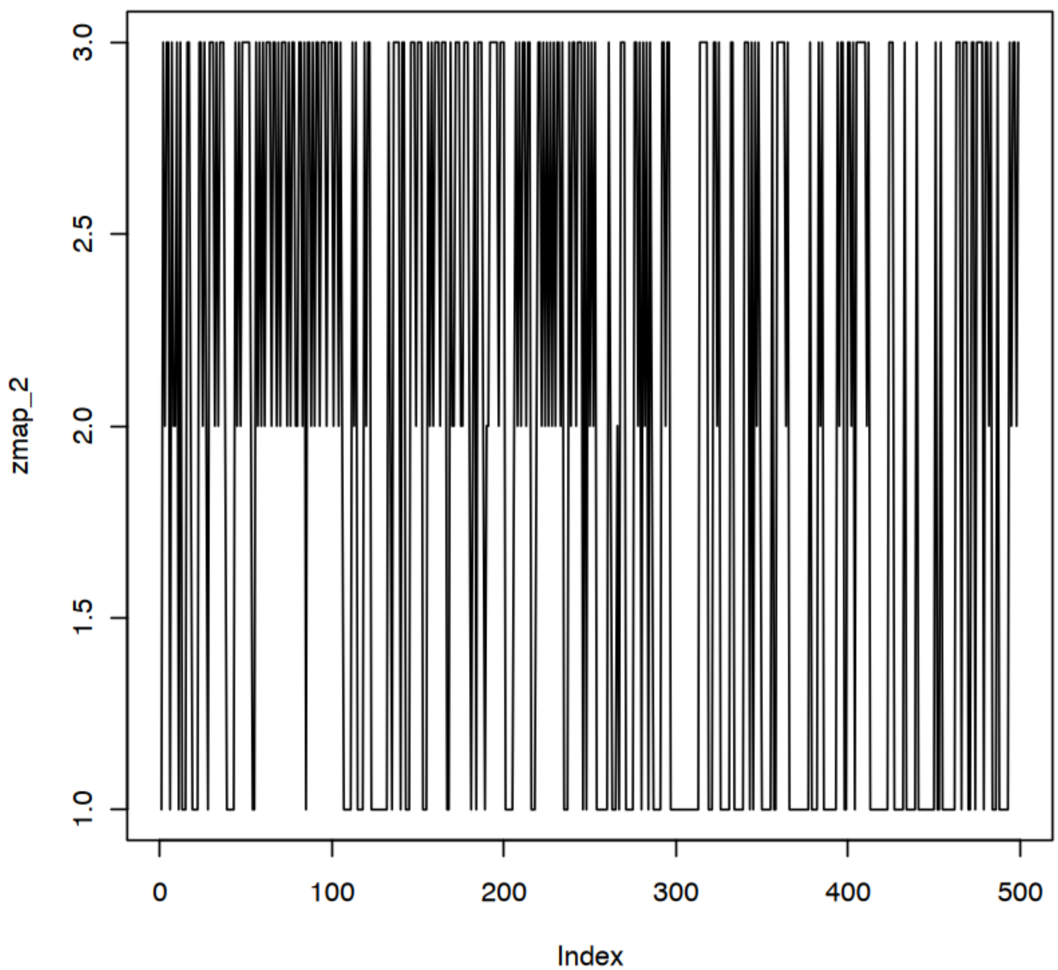
```
Defining model

Building model

Setting data and initial values

Running calculate on model
  [Note] Any error reports that follow may simply reflect missing values in
model variables.

Checking model sizes and dimensions

  [Note] This model is not fully initialized. This is not an error.
         To see which variables are not initialized, use model$initializeInfo().
         For more information on model initialization, see
help(modelInitialization).
```

[80]:
```
# Compile the model
Cmodel2 <- compileNimble(model_dir)
# Configure the MCMC
mcmc2 <- buildMCMC(model_dir, monitors = c("z", "prob", "a_par", "b_par",␣
↪"rho", "tau"), WAIC = TRUE, enableWAIC = T)

Cmcmc2 <- compileNimble(mcmc2)
# Run the MCMC
mod_dir <- runMCMC(Cmcmc2, niter = 5000, nburnin = 2000)
```

```
Compiling
  [Note] This may take a minute.
  [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.


===== Monitors =====
thin = 1: a_par, b_par, prob, rho, tau, z
===== Samplers =====
posterior_predictive sampler (1)
  - theta_var[]  (1 element)
conjugate sampler (7)
  - b_par[]  (3 elements)
  - prob_init[1:3]
  - prob[1, 1:3]
  - prob[2, 1:3]
  - prob[3, 1:3]
categorical sampler (500)
  - z0
  - z[]  (499 elements)
RW sampler (9)
  - a_par[]  (3 elements)
  - tau[]  (3 elements)
  - rho[]  (3 elements)

Compiling
  [Note] This may take a minute.
  [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.

  [Warning] To calculate WAIC, set 'WAIC = TRUE', in addition to having enabled
WAIC in building the MCMC.

running chain 1…


|-------------|-------------|-------------|-------------|
|-------------------------------------------------------|
```

Anche in questo caso vediamo le stime di zeta

[81]:
```
library(coda)
findmode <- function(x) {
  TT <- table(as.vector(x))
  return(as.numeric(names(TT)[TT == max(TT)][1]))
}
z_samples <- mod_dir[, grep("^z", colnames(mod_dir))]
zmap_2 <- apply(z_samples, 2, findmode)
plot(zmap_2, type="l")
```



Possiamo confrontare le stime dei due modelli

[82]:
```
table(zmap[-1],zmap_2)
```

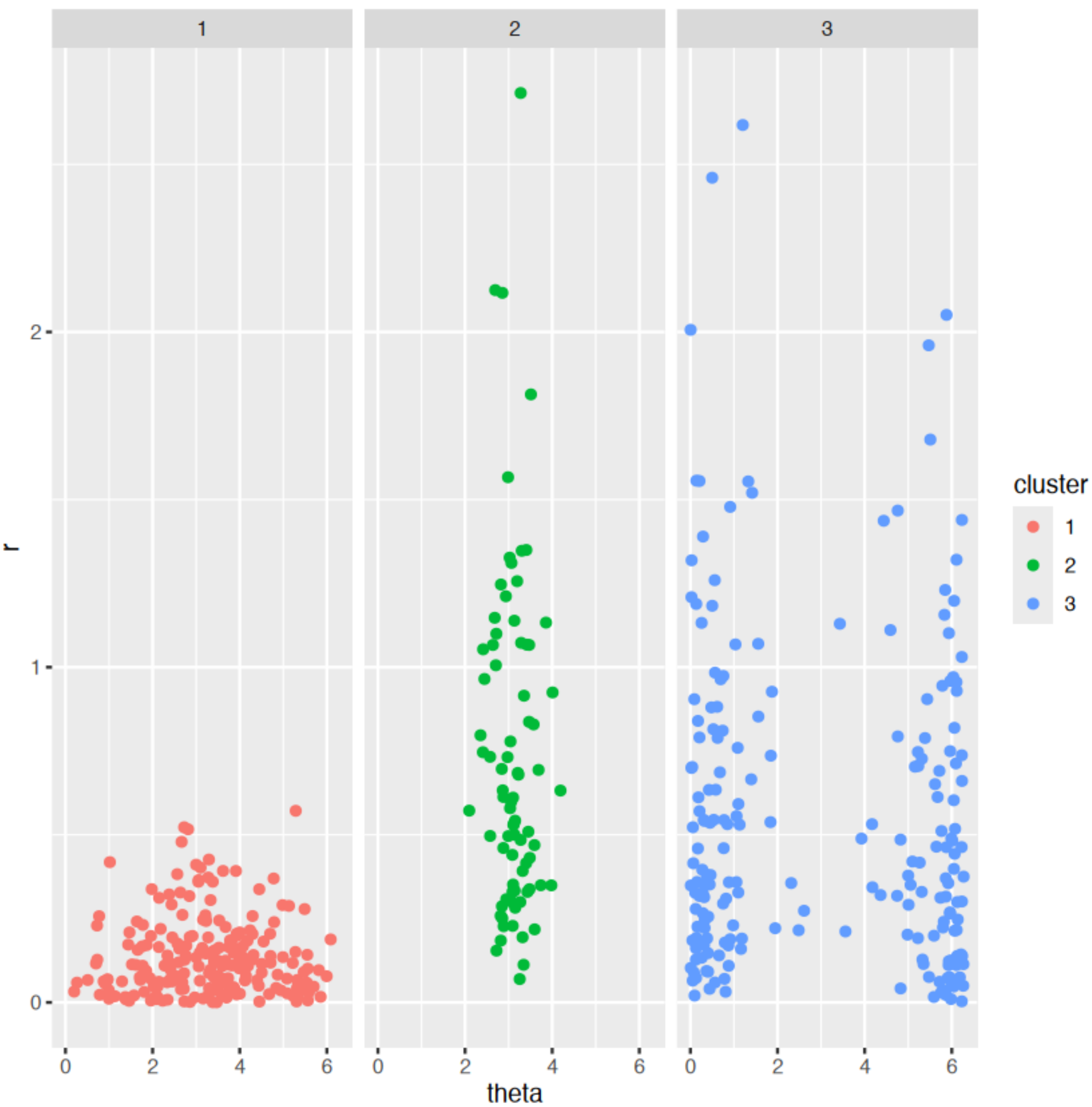
```
   zmap_2
```

	1	2	3
1	35	34	66
2	119	34	95
3	59	11	46

sebbene stimate sugli stessi dati, sembra non esserci nessuna connessione tra i due. Vediamo come si distribuiscono i dati nei tre gruppi individuati da questo modello

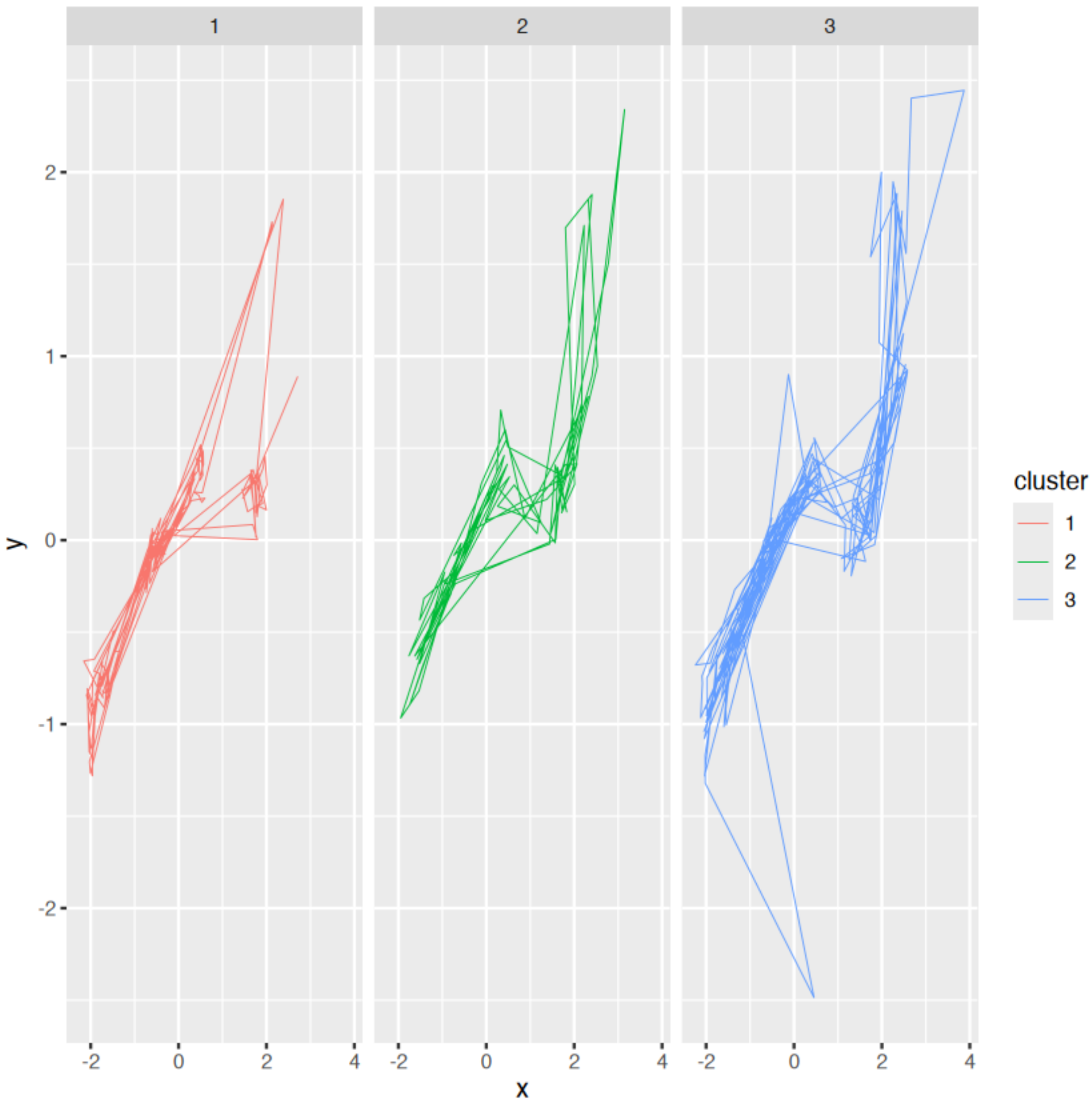
[83]:
```
data_plot <- data.frame(theta = theta_var, r = r_var, cluster = factor(zmap_2))
data_plot %>% ggplot(aes(x = theta, y = r, col = cluster)) +
  geom_point() +
  facet_wrap(~cluster)
```

```
Warning message:
"Removed 1 row containing missing values or values outside the scale
range
(`geom_point()`)."
```



e se sono connessi al path

[84]:
```
data_plot_path <- data.frame(x = x[-1], y = y[-1], cluster = factor(zmap_2))
data_plot_path %>% ggplot(aes(x = x, y = y, col = cluster)) +
  geom_path(size = 0.2) +
  facet_wrap(~cluster)
```



Vediamo la matrice di transizione

[85]:
```
prob_samples_2 <- mod_dir[, grep("^prob", colnames(mod_dir))]
prob_mean_2 <- colMeans(prob_samples_2)
```

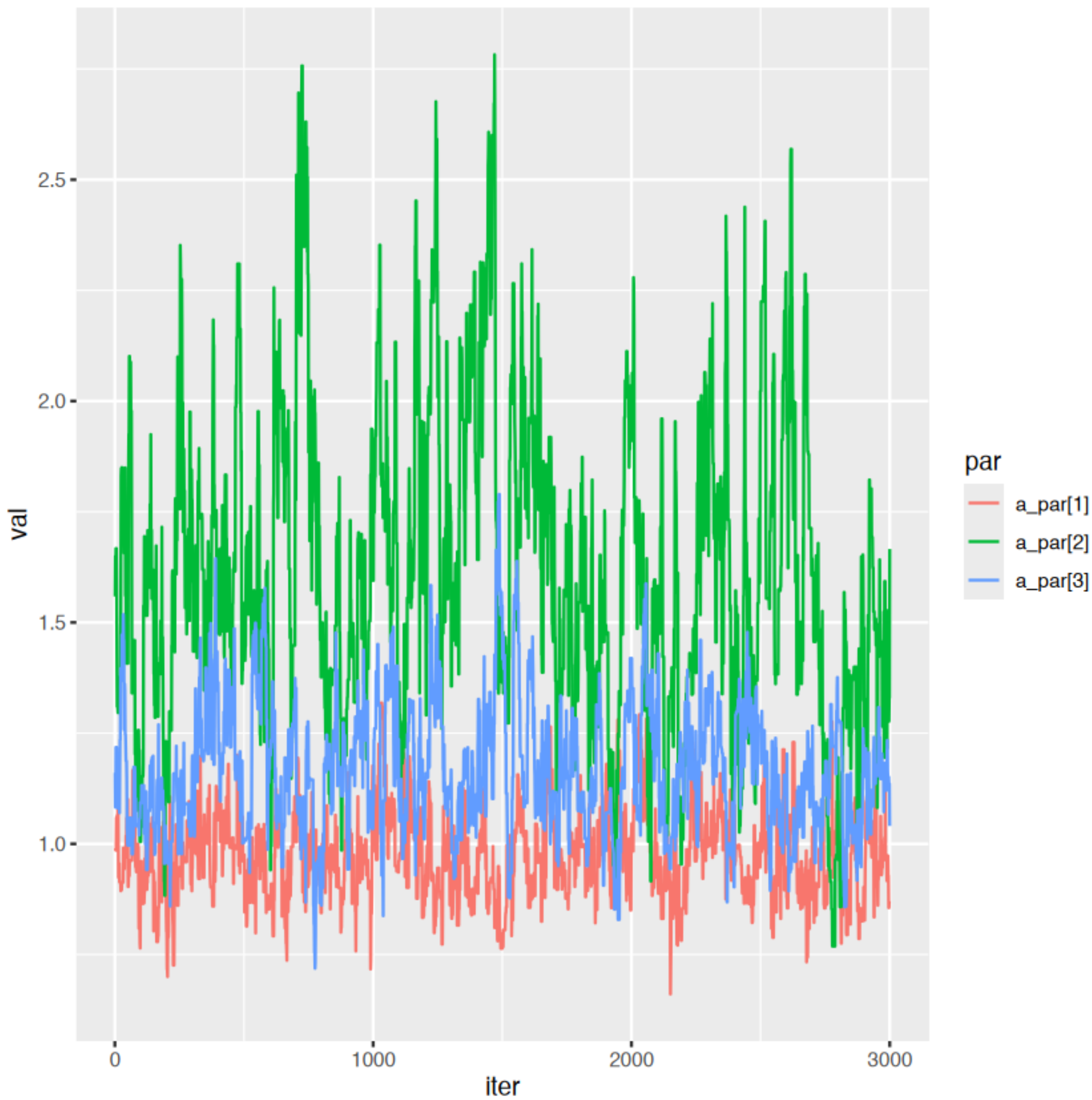
```
matrix(prob_mean_2, ncol = K)
```

A matrix: 3 x 3 of type dbl

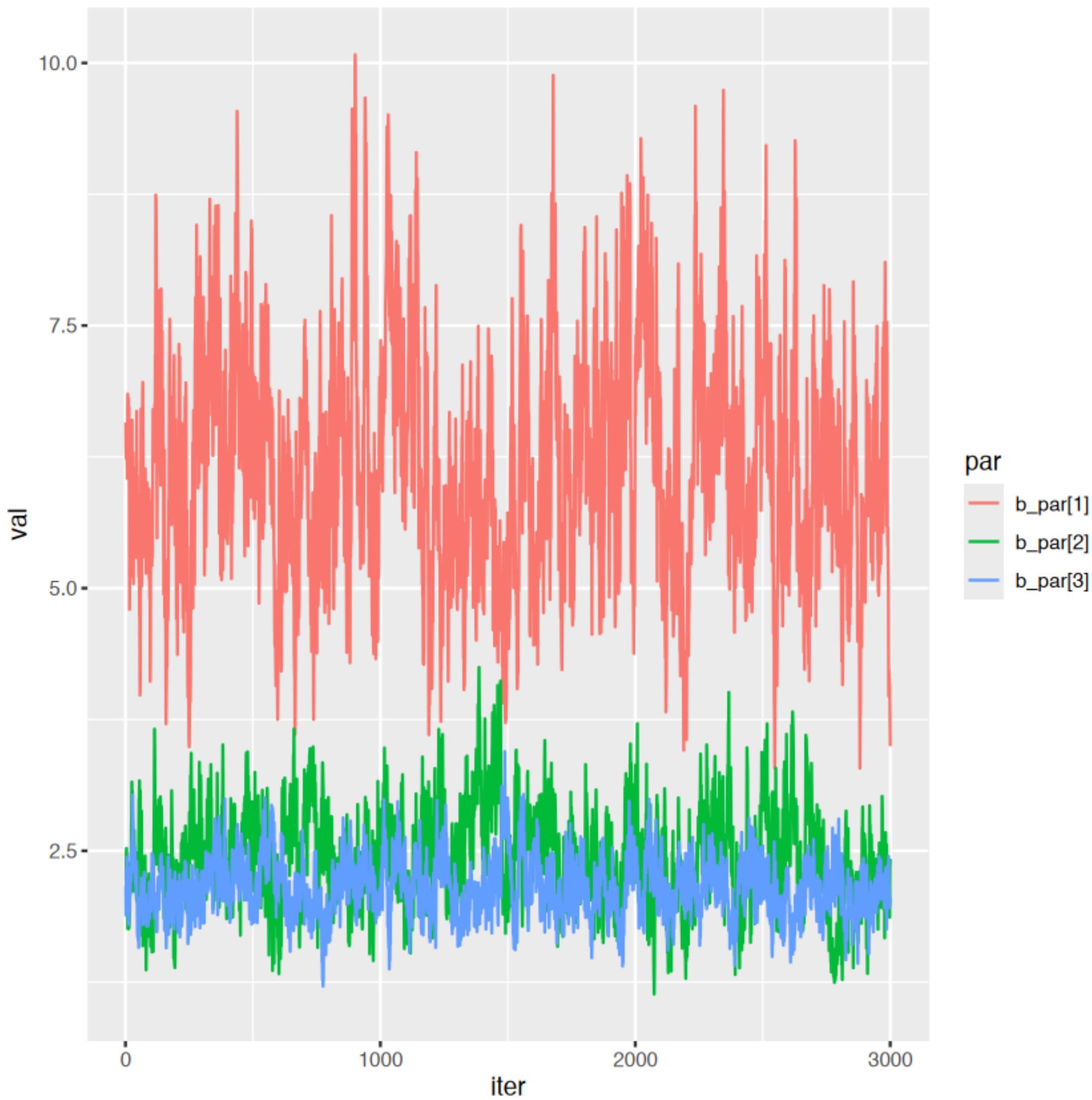
0.7135633	0.03893766	0.2474991
0.1384738	0.10296772	0.7585585
0.2224036	0.35418636	0.4234100

E poi le catene dei parametri

[86]:
```
a_samples <- mod_dir[, grep("^a_par", colnames(mod_dir))]
a_mcmc <- a_samples %>%
  as.data.frame() %>%
  mutate(iter= 1:nrow(a_samples))%>%
pivot_longer(
    cols = 1:K,
    names_to = "par",
    values_to = "val"
)
a_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col=par)) +
  geom_line()


b_samples <- mod_dir[, grep("^b_par", colnames(mod_dir))]
b_mcmc <- b_samples %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(a_samples)) %>%
  pivot_longer(
    cols = 1:K,
    names_to = "par",
    values_to = "val"
  )
b_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()
```



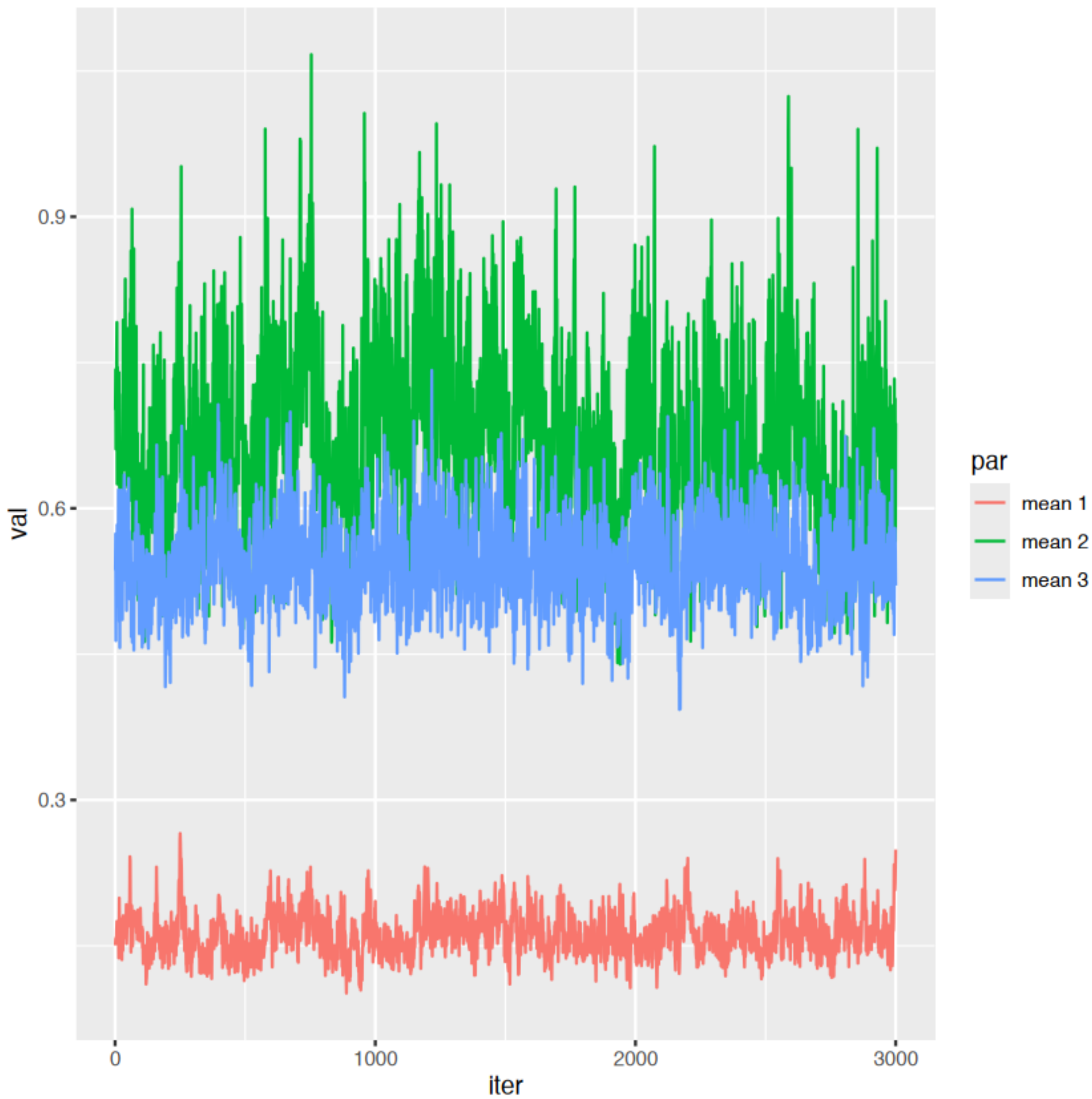
2.5
2.0
1.5
1.0
val
0
1000
2000
3000
iter
par
a_par[1]
a_par[2]
a_par[3]

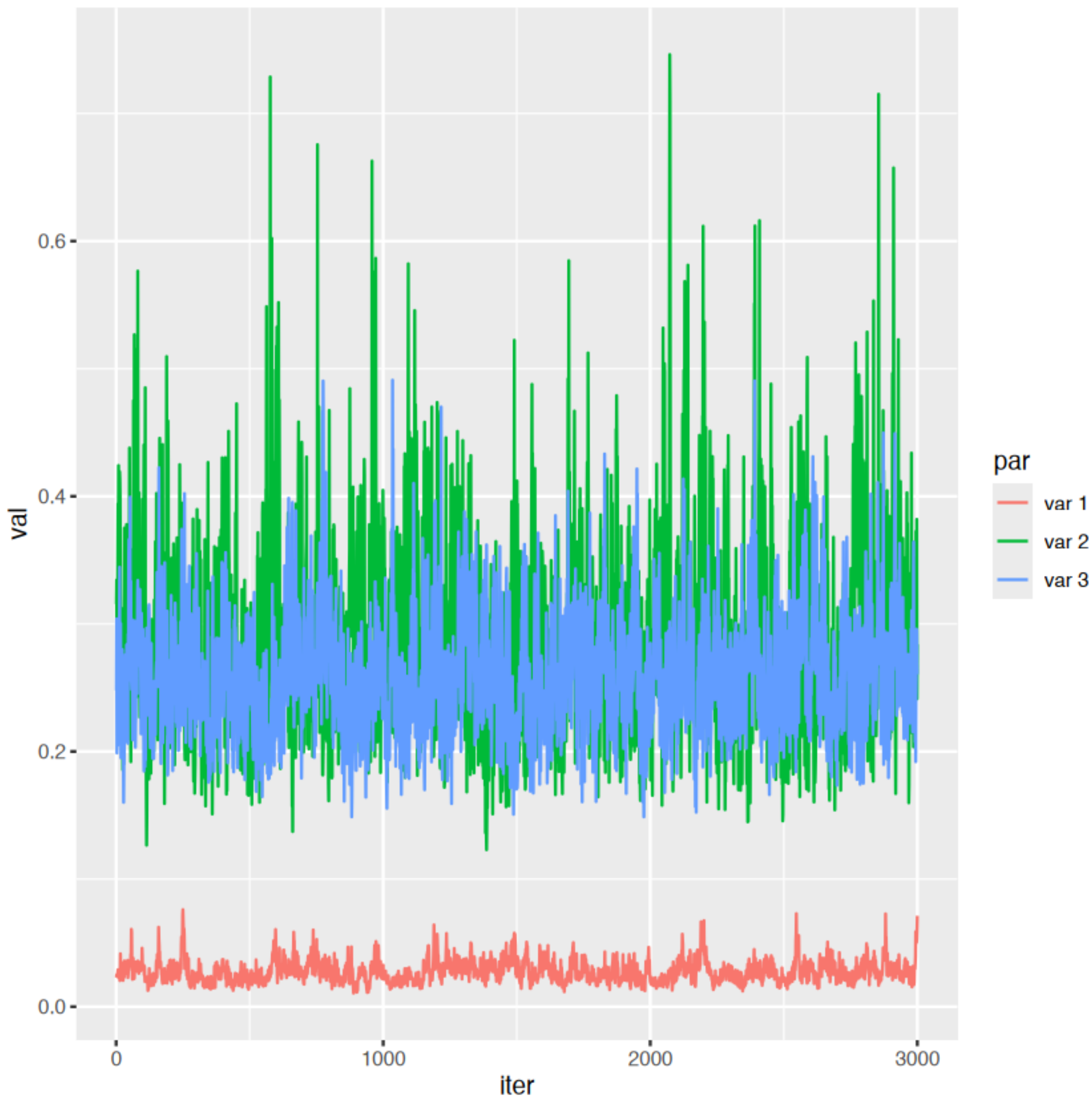


per interpretare meglio vediamo la distribuzione della media e varianza dalla velocità

[87]:
```
mean_gamma <- a_samples / b_samples
colnames(mean_gamma) = paste("mean", 1:K)

var_gamma <- a_samples / b_samples^2
colnames(var_gamma) <- paste("var", 1:K)


mean_mcmc <- mean_gamma %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(a_samples)) %>%
  pivot_longer(
```

```
    cols = 1:K,
    names_to = "par",
    values_to = "val"
  )
mean_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()


var_mcmc <- var_gamma %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(a_samples)) %>%
  pivot_longer(
    cols = 1:K,
    names_to = "par",
    values_to = "val"
  )
var_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()
```

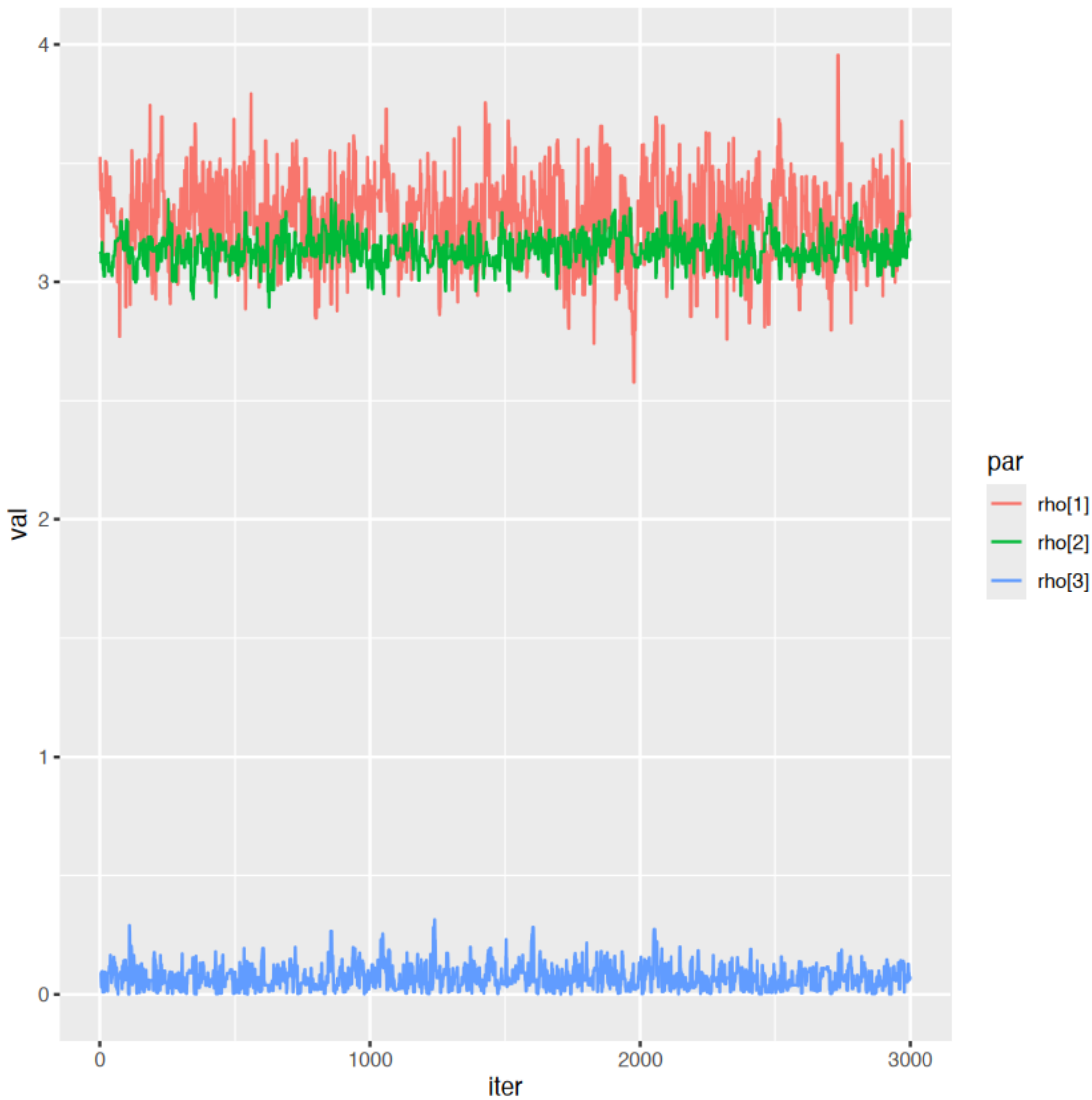

0.9
0.6
0.3
val
0
1000
2000
3000
iter
par
mean 1
mean 2
mean 3

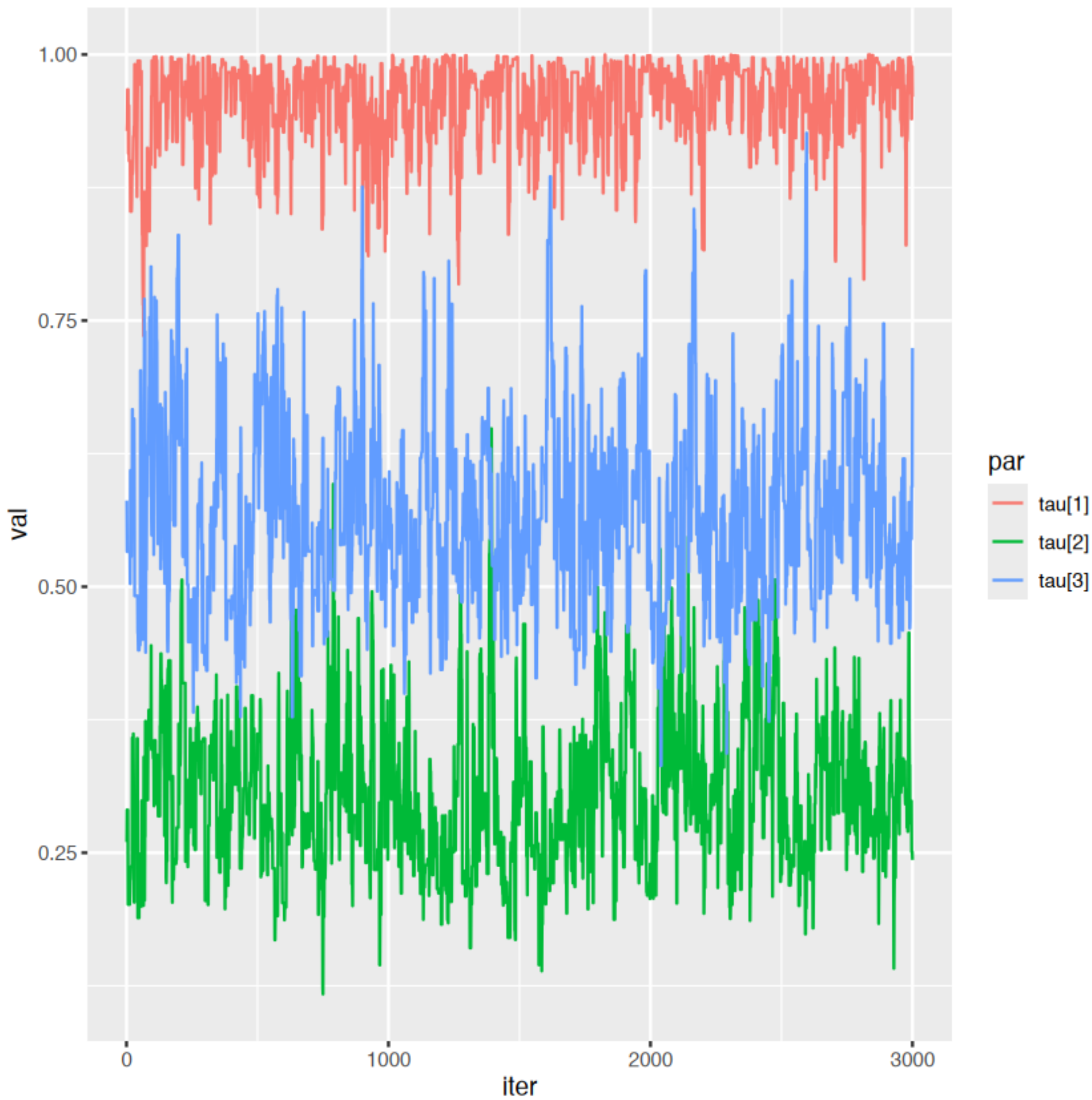


vediamo la stessa cosa per le variabili circolari

```
[88]: rho_samples <- mod_dir[, grep("^rho", colnames(mod_dir))]
rho_mcmc <- rho_samples %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(a_samples)) %>%
  pivot_longer(
    cols = 1:K,
    names_to = "par",
    values_to = "val"
  )
rho_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()
```

```r
tau_samples <- mod_dir[, grep("^tau", colnames(mod_dir))]
tau_mcmc <- tau_samples %>%
  as.data.frame() %>%
  mutate(iter = 1:nrow(a_samples)) %>%
  pivot_longer(
    cols = 1:K,
    names_to = "par",
    values_to = "val"
  )
tau_mcmc %>% ggplot(aes(x = iter, y = val, group = par, col = par)) +
  geom_line()
```





Adesso proviamo a stimare le densità predittiva della circolare e della lineare nei 3 cluster. Per esempio, per la circolare, possiamo valore la densità del gruppo k nel punto c come

$$f(\theta^* = c|z = k, \theta) = \int f(\theta^* = c|\tau_1, \tau_2, \tau_3, \rho_1, \rho_2, \rho_3, z = k, \theta) f(\tau_1, \tau_2, \tau_3, \rho_1, \rho_2, \rho_3, z = k|\theta) d\theta =$$

$$f(\theta^* = c|z = k, \theta) = \int f(\theta^* = c|\tau_k, \rho_k) f(\tau_k, \rho_k, z = k|\theta) d\theta \approx \frac{\sum_{b=1}^{B} f(\theta^* = c|\tau_k^b, \rho_k^b)}{B}$$

dove

$$f(\theta^* = c|\tau_k, \rho_k)$$

è la densità della WC con parametri dati dai k-esimi parametri. Calcoli simili si possono fare per la lineare

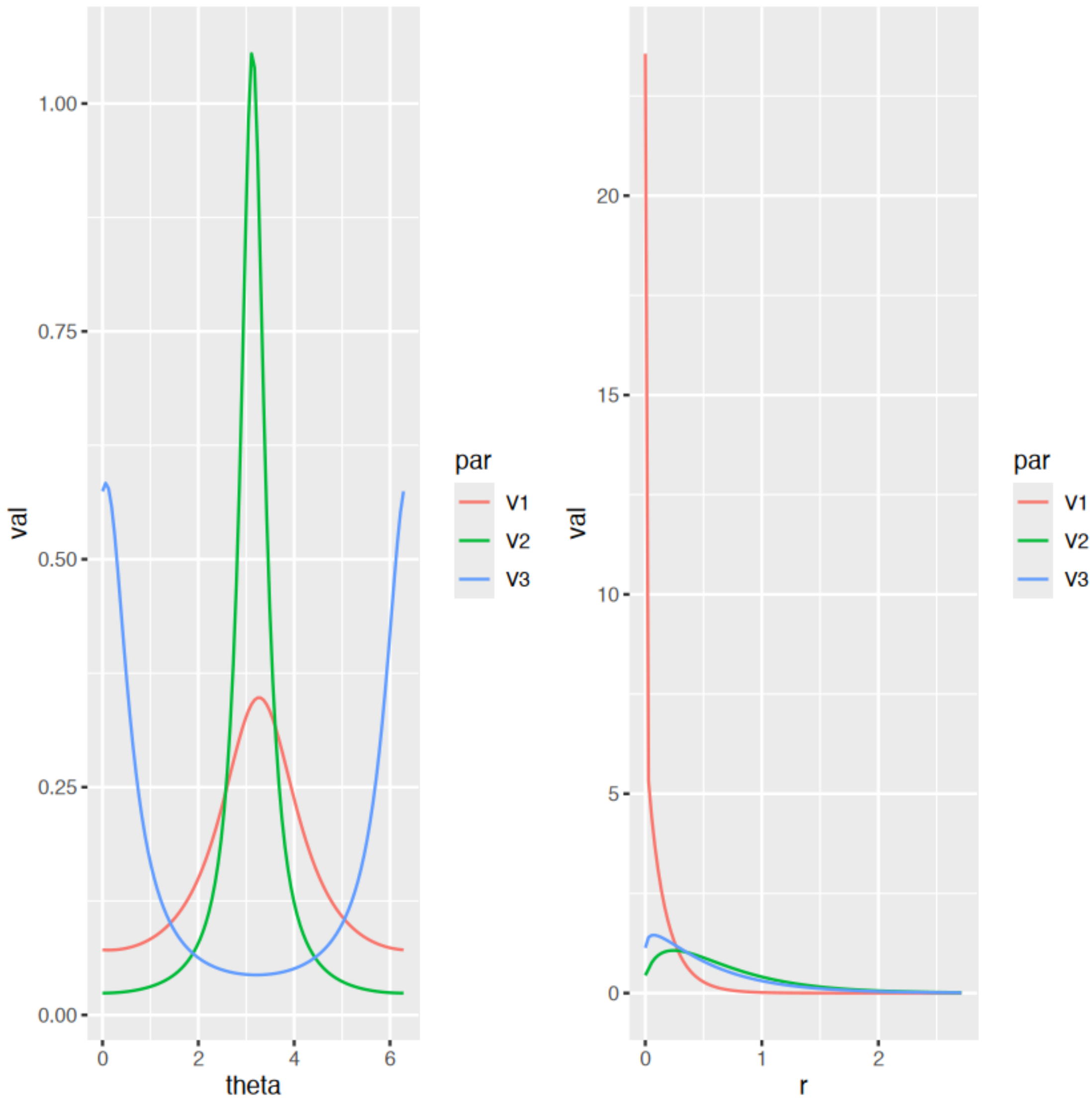
[89]:

```
nsim <- nrow(rho_samples)

n_seq <- 100
theta_seq = seq(0, 2 * pi, length.out = n_seq)
r_seq = seq(0.00000001, max(r_var), length.out = n_seq)
densi_circ <- matrix(0, ncol = K, nrow = n_seq)
densi_lin <- matrix(0, ncol = K, nrow = n_seq)
for(k in 1:K)
{
  for(iseq in 1:n_seq)
  {
    #for(isim in 1:nsim)
    #{
      #densi_circ[iseq, k] <- densi_circ[iseq, k] + exp(log(sinh(tau_samples[isim, k])) - log(cosh(tau_samples[isim, k]) - cos(theta_seq[iseq] - rho_samples[isim, k])) - log(2.0 * pi))
      #densi_lin[iseq, k] <- densi_lin[iseq, k] + dgamma(r_seq[iseq], shape = a_samples[isim, k], rate = b_samples[isim, k])
    #}
    densi_circ[iseq, k] <- mean(exp(log(sinh(tau_samples[, k])) - log(cosh(tau_samples[, k]) - cos(theta_seq[iseq] - rho_samples[, k])) - log(2.0 * pi)))
    densi_lin[iseq, k] <- mean(dgamma(r_seq[iseq], shape = a_samples[, k], rate = b_samples[, k]))
  }
}
```

[90]:

```
plot_circ <- densi_circ %>%
  as.data.frame() %>%
  mutate(theta = theta_seq) %>%
  pivot_longer(
    cols = 1:K,
    names_to = "par",
    values_to = "val"
  )
p1 <- plot_circ %>% ggplot(aes(x = theta, y = val, group = par, col = par)) +
  geom_line()

plot_lin <- densi_lin %>%
  as.data.frame() %>%
  mutate(r = r_seq) %>%
  pivot_longer(
    cols = 1:K,
    names_to = "par",
    values_to = "val"
  )
p2 <- plot_lin %>% ggplot(aes(x = r, y = val, group = par, col = par)) +
```

```
  geom_line()


library(gridExtra)
grid.arrange(p1,p2, ncol=2)
```



Il primo gruppo è quello con velocità più basse ma ha cambi di direzione di di 180 gradi. Gruppo 2 e 3 hanno le stesse velocità (alte), ma si differenziano per la direzione

possiamo anche fare stime sul piano bivariato

[91]:
```
mat_dens <- array(NA, c(n_seq, n_seq, K))
for(icirc in 1:n_seq)
{
```

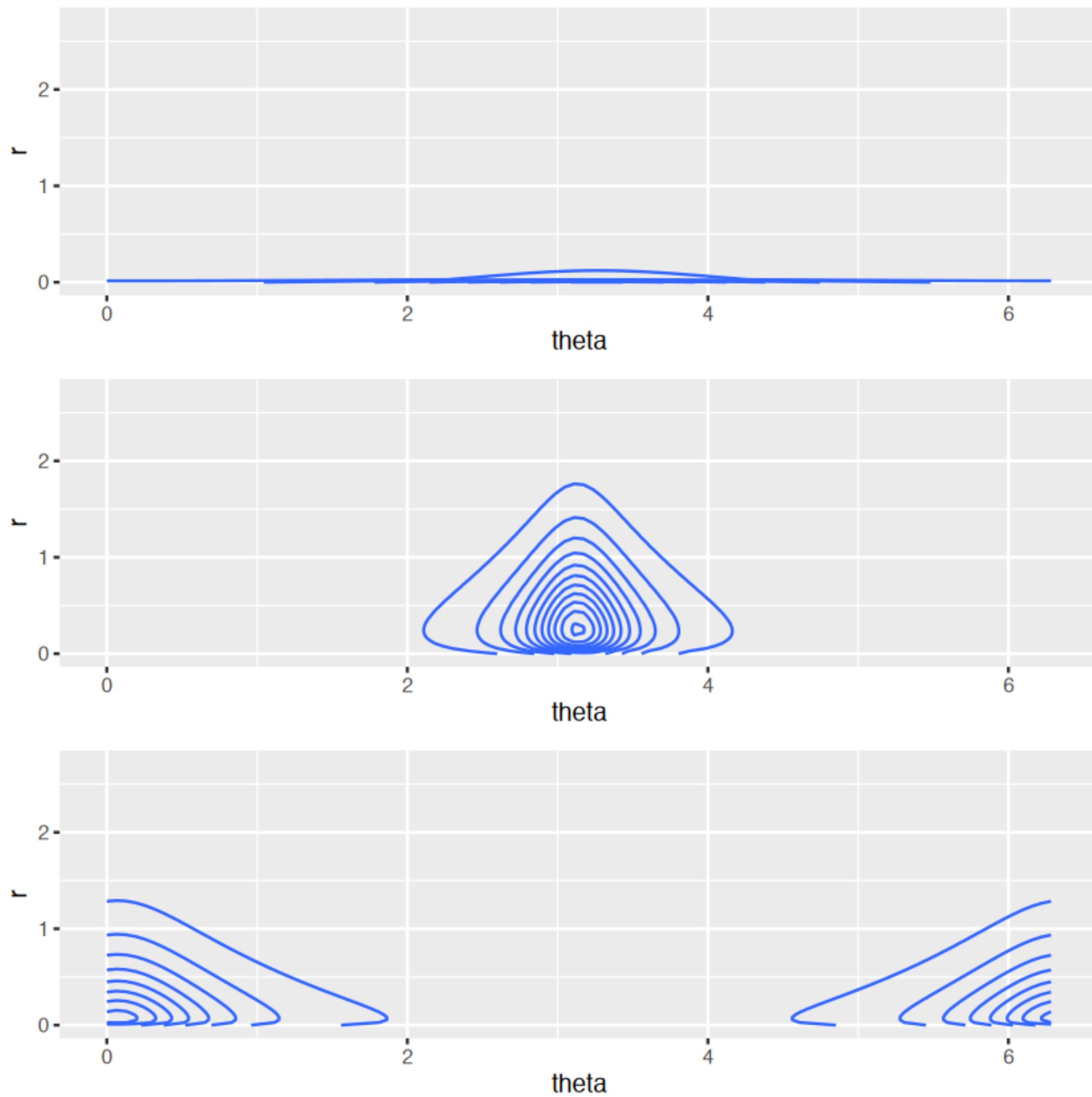
```
  for(ilin in 1:n_seq)
  {
    for(k in 1:K)
    {
      dcirc <- log(sinh(tau_samples[, k])) - log(cosh(tau_samples[, k]) -␣
↪cos(theta_seq[icirc] - rho_samples[, k])) - log(2.0 * pi)
      dlin <- dgamma(r_seq[ilin], shape = a_samples[, k], rate = b_samples[,␣
↪k], log=T)
      mat_dens[icirc, ilin,k] <- mean(exp(dcirc + dlin))
    }

  }
}
```

[ ]:

```
```

[92]:

```
data_plot <- data.frame(dens = c(mat_dens), theta = rep(rep(theta_seq, times =␣
↪n_seq), times = K), r = rep(rep(r_seq, each = n_seq), times = K), clust =␣
↪factor(rep(1:K, each =n_seq^2)))




p1 <- data_plot %>%
  filter(clust == "1") %>%
  ggplot(aes(x = theta, y = r, z = dens)) +
  geom_contour() +
  scale_colour_distiller(palette = "YlGn", direction = 1) +
  xlim(0, 2 * pi) +
  ylim(0, max(r_var))

p2 <- data_plot %>%
  filter(clust == "2") %>%
  ggplot(aes(x = theta, y = r, z = dens)) +
  geom_contour() +
  scale_colour_distiller(palette = "YlGn", direction = 1) + xlim(0, 2 * pi) +␣
↪ylim(0, max(r_var))
p3 <- data_plot %>%
  filter(clust == "3") %>%
  ggplot(aes(x = theta, y = r, z = dens)) +
  geom_contour() +
  scale_colour_distiller(palette = "YlGn", direction = 1) + xlim(0, 2 * pi) +␣
↪ylim(0, max(r_var))

grid.arrange(p1, p2,p3)
```


2
r
1
0
0
2
4
6
theta
2
r
1
0
0
2
4
6
theta
2
r
1
0
0
2
4
6
theta