GSOC 2024: tests for dirichletprocess

Import packages

```
library(tidyverse) # data manipulation
```

```
Warning: package 'ggplot2' was built under R version 4.3.1
```

```
Warning: package 'tidyr' was built under R version 4.3.1
```

```
Warning: package 'readr' was built under R version 4.3.1
```

```
Warning: package 'dplyr' was built under R version 4.3.1
```

```
Warning: package 'stringr' was built under R version 4.3.1
```

```
Warning: package 'lubridate' was built under R version 4.3.1
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
library(dirichletprocess) # dirichletprocess
library(distr) # simulate mixture distribution
```

```
Warning: package 'distr' was built under R version 4.3.1
```

```
Loading required package: startupmsg
Utilities for Start-Up Messages (version 0.9.6)
For more information see ?"startupmsg", NEWS("startupmsg")

Loading required package: sfsmisc

Attaching package: 'sfsmisc'

The following object is masked from 'package:dplyr':

    last

Object Oriented Implementation of Distributions (version 2.9.3)
Attention: Arithmetics on distribution objects are understood as operations on corresponding
Some functions from package 'stats' are intentionally masked ---see distrMASK().
Note that global options are controlled by distroptions() ---c.f. ?"distroptions".
For more information see ?"distr", NEWS("distr"), as well as
  http://distr.r-forge.r-project.org/
Package "distrDoc" provides a vignette to this package as well as to several extension packag


Attaching package: 'distr'

The following objects are masked from 'package:dplyr':

    location, n

The following objects are masked from 'package:stats':

    df, qqplot, sd
```

```r
library(extraDistr) # beta prime dist
```

```
Warning: package 'extraDistr' was built under R version 4.3.1
```

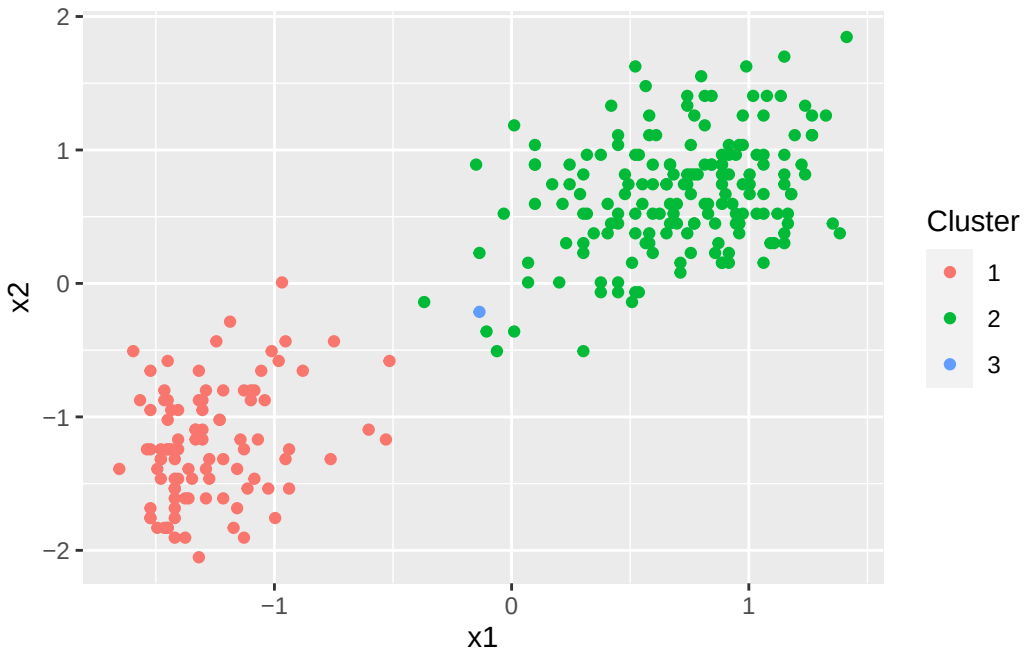
```
Attaching package: 'extraDistr'

The following object is masked from 'package:purrr':

    rdunif
```

EASY

```
data("faithful")
glimpse(faithful)
```

```
Rows: 272
Columns: 2
$ eruptions <dbl> 3.600, 1.800, 3.333, 2.283, 4.533, 2.883, 4.700, 3.600, 1.95~
$ waiting   <dbl> 79, 54, 74, 62, 85, 55, 88, 85, 51, 85, 54, 84, 78, 47, 83, ~
```

```
faithful_trans <- scale(faithful)
dp_faithful <- DirichletProcessMvnormal(faithful_trans)
dp_fit_faithful <- Fit(dp_faithful,its = 100)
plot(dp_fit_faithful)
```

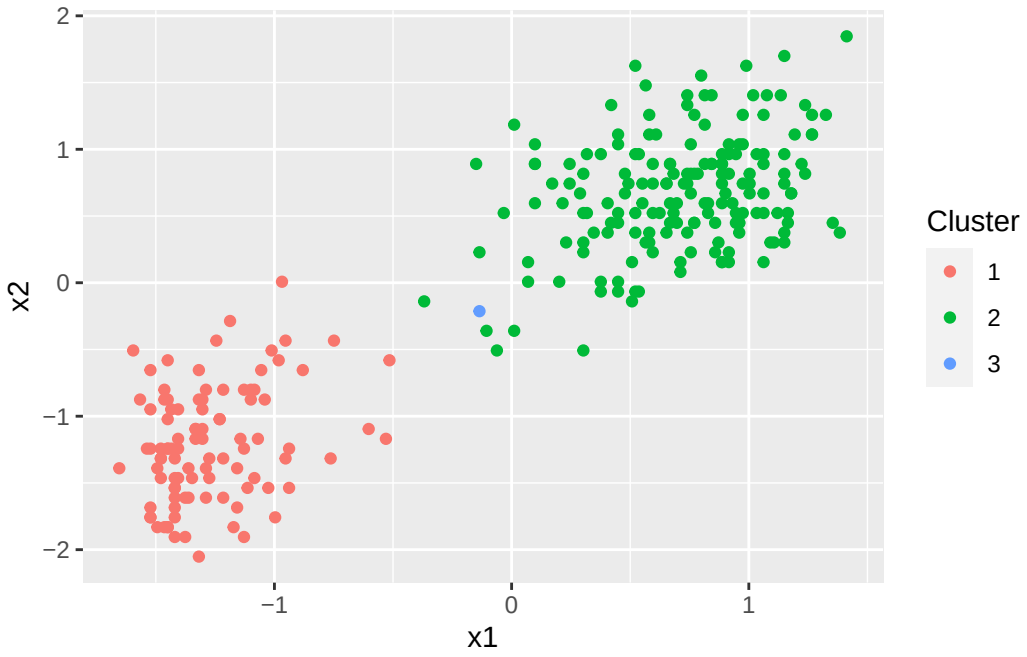


```
data("iris")
glimpse(iris)
```

```
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
$ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s~
```

```
iris_trans <- scale(iris %>% select(-Species))
```

```
dp_iris <- DirichletProcessMvnormal(iris_trans,numInitialClusters = length(unique(iris$Spe
dp_fit_iris <- Fit(dp_iris,its = 1000)
plot(dp_fit_faithful) # only two clusters found by dp when there are actually three cluste
```



MEDIUM

This is the function for sampling a mixture distribution of two lognormal distributions.

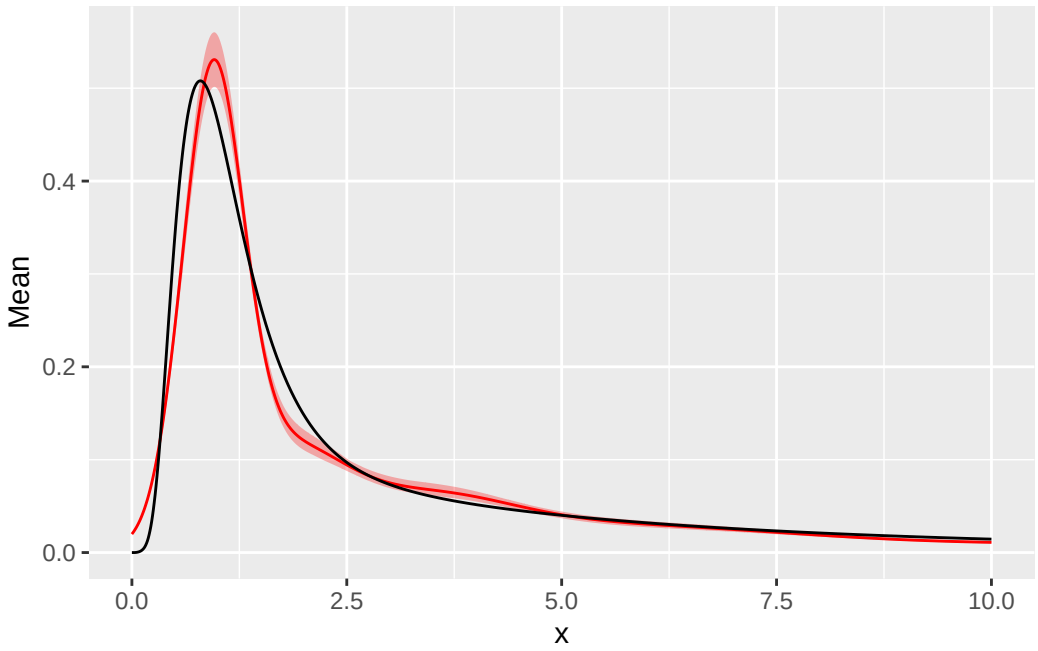
```
sim_log_mixture <- function(n=1000,meanlogs=c(0,1.5),sdlogs=c(0.5,1)){
  tmp_sampler <- r(UnivarMixingDistribution(Lnorm(meanlog=meanlogs[1],sdlog=sdlogs[1]),
                           Lnorm(meanlog=meanlogs[2],sdlog=sdlogs[2])))
  tmp_sampler(n)
}
```

Using the above function, we sampled $n = 1,000$ from the lognormal mixture and fitted a simple, Gaussian mixture distribution. From the fitted model, we took 100 samples from the posterior distribution along with 95% credible intervals and then compared the resulting empirical distribution (red) with the true distribution (black). Based on the plot, the model fitted the data reasonably well.

```
set.seed(1)
df_lognormal <- sim_log_mixture()
dp_lognormal <- DirichletProcessGaussian(df_lognormal)
dp_fit_lognormal <- Fit(dp_lognormal,its=100)
dp_fit_lognormal
```

```
Dirichlet process object run for 100 iterations.

  Mixing distribution          normal
  Base measure parameters  0, 1, 1, 1
  Alpha Prior parameters         2, 4
  Conjugacy                 conjugate
  Sample size                    1000

  Mean number of clusters        8.74
  Median alpha                   0.90
```

```
xx <- seq(0,10,by=0.01)
dp_posteriro_lognormal <- PosteriorFrame(dp_fit_lognormal,xx,ndraws=100,ci_size=0.05) %>%
  as.data.frame()
trueFrame <- data.frame(x = xx,
                        y = 0.5*dlnorm(xx,meanlog = 0,sdlog = 0.5) + 0.5*dlnorm(xx,meanlog
ggplot(data=dp_posteriro_lognormal,aes(x=x,y=Mean)) +
  geom_line(col='red') +
  geom_ribbon(aes(x=x,ymin=X2.5.,ymax=X97.5.),alpha=0.3,fill='red',colour=NA) +
  geom_line(data=trueFrame,aes(x=x,y=y))
```



This was a simple experiment to investigate the effect of alpha priors on the number of clusters, time needed for model convergence, and alpha chains. We denote the first argument of the alpha prior by *alpha location* and the second argument by *alpha scale*. Fixing alpha scale, we saw that the number of clusters and time needed increased in alpha location. On the other hand, they decreased in alpha scale. As for the alpha chains, when the alpha location was extremely low, the alpha chains consisted of zeros, regardless of alpha locations. Now as alpha location increased, the alpha chain values increased as well. When the alpha location was not so low, the alpha chains decreased in alpha scale. One can associate that alpha chains and time needed are positively correlated (i.e., the higher the values in alpha chains, the longer the model convergence time required).

```
fit_dp <- function(alpha){
  start.time <- Sys.time()
  tmp_fit <- Fit(DirichletProcessGaussian(df_lognormal,alphaPriors = alpha),its=100)
  end.time <- Sys.time()
  return(list(alpha=alpha,
              num_cluster=tmp_fit$numberClusters,
              alphachain = tmp_fit$alphaChain,
              time_taken = end.time-start.time))
}

alpha_grid <- expand.grid(alpha_location=c(0.000001,1,10000),alpha_scale=c(0.000001,1,1000
  mutate(num_cluster=0,
```

```
          time=0)
alphaChains <- c()
for(i in 1:nrow(alpha_grid)){
  tmp_fit <- fit_dp(c(alpha_grid$alpha_location[i],alpha_grid$alpha_scale[i]))
  alpha_grid$num_cluster[i] <- tmp_fit$num_cluster
  alpha_grid$time[i] <- tmp_fit$time_taken
  alphaChains[[i]] <- tmp_fit$alphachain
}

alpha_grid
```

```
  alpha_location alpha_scale num_cluster      time
1          1e-06       1e-06           1  2.760010
2          1e+00       1e-06         996 15.153502
3          1e+04       1e-06        1000 15.001850
4          1e-06       1e+00           1  2.724498
5          1e+00       1e+00           5  2.883467
6          1e+04       1e+00         954 14.852421
7          1e-06       1e+04           1  2.815432
8          1e+00       1e+04           3  2.776042
9          1e+04       1e+04           8  2.906381
```

HARD

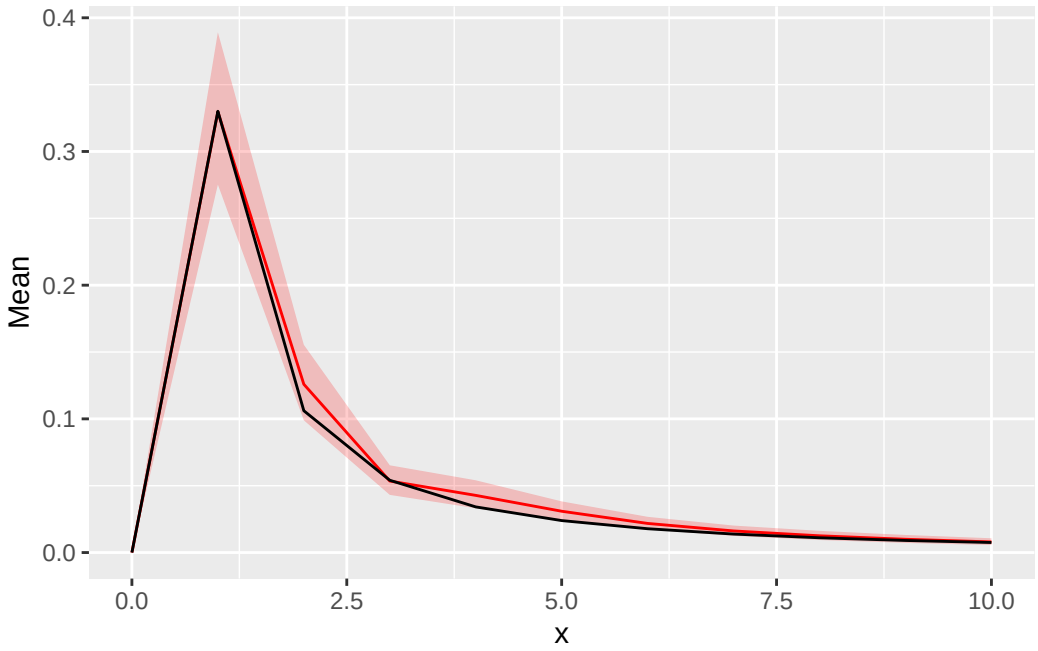
I chose to implement a custom mixture model for the beta prime distribution.

```
Likelihood.betaprime <- function(mdobj, x, theta){
  return(as.numeric(dbetapr(x,theta[[1]],theta[[2]])))
}

PriorDraw.betaprime  <- function(mdobj, n=1){
  theta <- list()
  theta[[1]] = array(rexp(n, mdobj$priorParameters[1]), dim=c(1,1, n))
  theta[[2]] = array(rexp(n, mdobj$priorParameters[2]), dim=c(1,1, n))
  return(theta)
}


PriorDensity.betaprime  <- function(mdobj, theta){
priorParameters <- mdobj$priorParameters
```

```
thetaDensity <- dexp(theta[[1]], priorParameters[1])
thetaDensity <- thetaDensity * dexp(theta[[2]], priorParameters[2])
return(as.numeric(thetaDensity))
}

MhParameterProposal.betaprime  <- function(mdobj, oldParams){
  mhStepSize <- mdobj$mhStepSize
  newParams <- oldParams
  newParams[[1]] <- abs(oldParams[[1]] + mhStepSize[1]*rnorm(1,0.1,0.1))
  newParams[[2]] <- abs(oldParams[[2]] + mhStepSize[2]*rnorm(1,0.1,0.1))
  return(newParams)
}

gomMd <- MixingDistribution(distribution = "betaprime",
                            priorParameters = c(0.1,0.1),
                            conjugate = "nonconjugate",
                            mhStepSize = c(0.1,0.1))
set.seed(1)
y <- c(rbetapr(100, 2,1), rbetapr(100, 3,5))
dp <- DirichletProcessCreate(y, gomMd)
dp_init <- Initialise(dp)
```

```
Accept Ratio:  0.762
```

```
dp_fit <- Fit(dp_init, 1000)
xx <- 0:10
pf <- PosteriorFrame(dp_fit, xx, 1000,ci_size = 0.10)
trueFrame <- data.frame(x= xx,
                        y= 0.5*dbetapr(xx, 2,1) + 0.5*dbetapr(xx, 3,5))

ggplot() +
  geom_ribbon(data=pf,
              aes(x=x, ymin=X5., ymax=X95.),
              colour=NA,
              fill="red",
              alpha=0.2) +
  geom_line(data=pf, aes(x=x, y=Mean), colour="red") +
  geom_line(data=trueFrame, aes(x=x, y=y))
```


0.4
0.3
0.2
0.1
0.0
Mean
0.0
2.5
5.0
7.5
10.0
x