Bayes Estimate

In [1]:

```
get_bayes_estimate = function(original_sample,dist_name,known_params)
{
    if (dist_name=='binomial')
    {
        # Conjugate prior distribution of p: Beta
        prior_alpha = known_params$prior_alpha # alpha of prior distribution
        prior_beta = known_params$prior_beta # beta of prior distribution
        # Posterior distribution of p: Beta
        no_of_trials = known_params$no_of_trials #
        posterior_alpha = prior_alpha + sum(original_sample) # alpha of posterior distribution
        posterior_beta = prior_beta + length(original_sample)*no_of_trials - sum(original_sample) # beta if post
        # Bayes estimate of p - Mean of posterior distribution
        estimated_p = posterior_alpha / (posterior_alpha + posterior_beta)
        estimate = list(p=estimated_p,posterior_alpha=posterior_alpha,posterior_beta=posterior_beta)
    }
    else if (dist_name == 'bernoulli')
    {
        # Conjugate prior distribution of p: Beta
        prior_alpha = known_params$prior_alpha # alpha of prior distribution
        prior_beta = known_params$prior_beta # beta of prior distribution
        # Posterior distribution of p: Beta
        posterior_alpha = prior_alpha + sum(original_sample) # alpha of posterior distribution
        posterior_beta = prior_beta + length(original_sample) - sum(original_sample) # beta if posterior distrib
        # Bayes estimate of p
        estimated_p = posterior_alpha / (posterior_alpha + posterior_beta) # expectation of posterior distributi
        estimate = list(p=estimated_p,posterior_alpha=posterior_alpha,posterior_beta=posterior_beta)
    }
    else if (dist_name=='geometric')
    {
        # Conjugate prior distribution of p: Beta
        prior_alpha = known_params$prior_alpha # alpha of prior distribution
        prior_beta = known_params$prior_beta # beta of prior distribution
        # Posterior distribution of p: Beta
        posterior_alpha = prior_alpha + length(original_sample)
        posterior_beta = prior_beta + sum(original_sample)
        # Bayes estimate of p
        estimated_p = posterior_alpha / (posterior_alpha + posterior_beta) # expectation of posterior distributi
        estimate = list(p=estimated_p,posterior_alpha=posterior_alpha,posterior_beta=posterior_beta)
```

```
            }
            else if (dist_name == 'poisson')
            {
                # Conjugate prior distribution of lambda: Gamma
```

```
        prior_alpha = known_params$prior_alpha # alpha of prior distribution
        prior_beta = known_params$prior_beta # beta of prior distribution
        # Posterior distribution of lambda: Gamma
        posterior_alpha = prior_alpha + sum(original_sample)
        posterior_beta = prior_beta + length(original_sample)
        # Bayes estimate
        estimated_lambda = posterior_alpha / posterior_beta
        estimate = list(lambda=estimated_lambda,posterior_alpha=posterior_alpha,posterior_beta=posterior_beta)
    }
    else if(dist_name == 'exponential')
    {
        # Conjugate prior distribution of lambda: Gamma
        prior_alpha = known_params$prior_alpha # alpha of prior distribution
        prior_beta = known_params$prior_beta # beta of prior distribution
        # Posterior distribution of lambda: Gamma
        posterior_alpha = prior_alpha + length(original_sample)
        posterior_beta = prior_beta + sum(original_sample)
        # Bayes estimate
        estimated_lambda = posterior_alpha / posterior_beta
        estimate = list(lambda=estimated_lambda,posterior_alpha=posterior_alpha,posterior_beta=posterior_beta)
    }
    else if(dist_name == 'gamma')
    {
        # Conjugate prior distribution of beta: Gamma
        prior_alpha = known_params$prior_alpha # alpha of prior distribution
        prior_beta = known_params$prior_beta # beta of prior distribution
        # Posterior distribution of beta: Gamma
        known_alpha = known_params$known_alpha
        posterior_alpha = prior_alpha + (length(original_sample))*known_alpha
        posterior_beta = prior_beta + sum(original_sample)
        # Bayes estimate
        estimated_beta = posterior_alpha / posterior_beta
        estimate = list(beta = estimated_beta,posterior_alpha=posterior_alpha,posterior_beta=posterior_beta)
    }
    else if(dist_name == 'normal')
    {
        # Conjugate prior distribution of mu: normal
        prior_mu = known_params$prior_mu
        prior_var = known_params$prior_var
        # Posterior distribution of mu: normal
        known_var = known_params$known_var
        posterior_mu =  (1/(1/prior_var + length(original_sample)/known_var))*(prior_mu/prior_var + sum(original
        posterior_var = 1/(1/prior_var + length(original_sample)/known_var)
        # Bayes estimate
        estimated_mu = posterior_mu
        estimate = list(mu=estimated_mu,posterior_mu=posterior_mu,posterior_var=posterior_var)
    }
    else if(dist_name == 'uniform')
    {
        # Conjugate prior distribution of mu: pareto
        prior_alpha = known_params$prior_alpha # alpha of prior distribution
        prior_beta = known_params$prior_beta # beta of prior distribution
        # Posterior distribution of mu: pareto
        posterior_alpha = prior_alpha + length(original_sample)
        max_x = max(original_sample)
        posterior_beta = max(prior_beta,max_x)
        # Bayes estimate
        estimated_b = posterior_alpha*posterior_beta/(posterior_alpha-1)
        estimate = list(b=estimated_b,posterior_alpha=posterior_alpha,posterior_beta=posterior_beta)
    }
    else if(dist_name == 'multinomial')
    {
        # Conjugate prior distribution of p: Dirichlet
        prior_alpha = known_params$prior_alpha
        # Posterior distribution of p: Dirichlet
        posterior_alpha = prior_alpha
        for (i in (1:length(original_sample[1,])))
        {
            posterior_alpha = posterior_alpha + original_sample[,i]
        }
        # Bayes estimate
        estimated_prob = c()
        post_sum = sum(posterior_alpha)
        for (i in (1:length(posterior_alpha)))
        {
            temp_prob = posterior_alpha[i]/post_sum
            estimated_prob = c(estimated_prob,temp_prob)
        }
        estimate = list(prob=estimated_prob,posterior_alpha=posterior_alpha)
    }
    return (estimate)
}
```

In [2]:

```
# Binomial distribution
original_sample = rbinom(1000,10,0.3)
known_params = list(no_of_trials=10,prior_alpha=1,prior_beta=1)
estimate = get_bayes_estimate(original_sample,'binomial',known_params)
estimate
options(repr.plot.width = 4, repr.plot.height=3)
x = seq(0,1, by=0.01)
plot(x,dbeta(x,estimate$posterior_alpha, estimate$posterior_beta), ylab="density", xlab="quantiles", type ="l",
```

$p

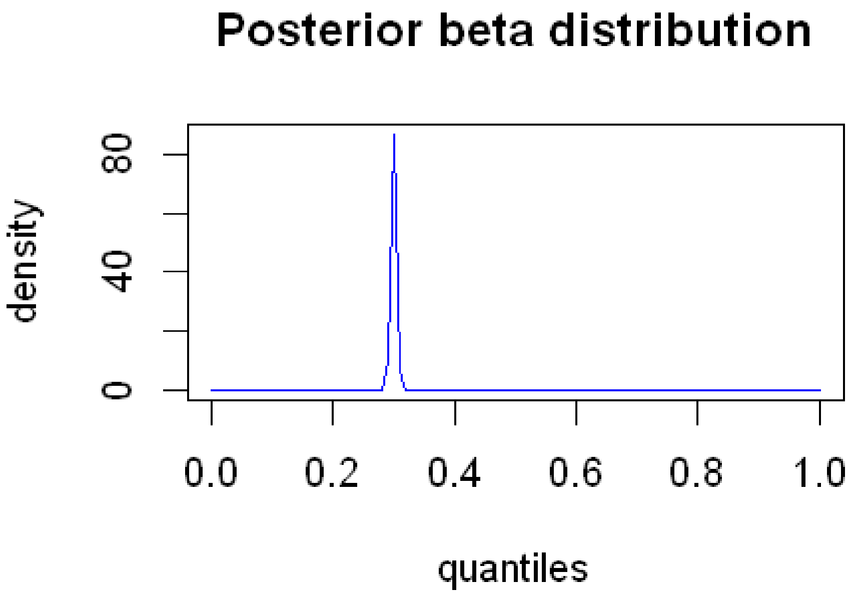
0.299540091981604

$posterior_alpha

2996

$posterior_beta

7006



In [3]:
```
# Bernoulli distribution - Binomial with number of trials = 1
original_sample = rbinom(n=1000,size=1,prob=0.6)
known_params = list(prior_alpha=1,prior_beta=1)
estimate = get_bayes_estimate(original_sample,'bernoulli',known_params)
estimate
options(repr.plot.width = 4, repr.plot.height=3)
x = seq(0,1, by=0.01)
plot(x,dbeta(x, estimate$posterior_alpha, estimate$posterior_beta), ylab="density", xlab="quantiles", type ="l",
```

$p

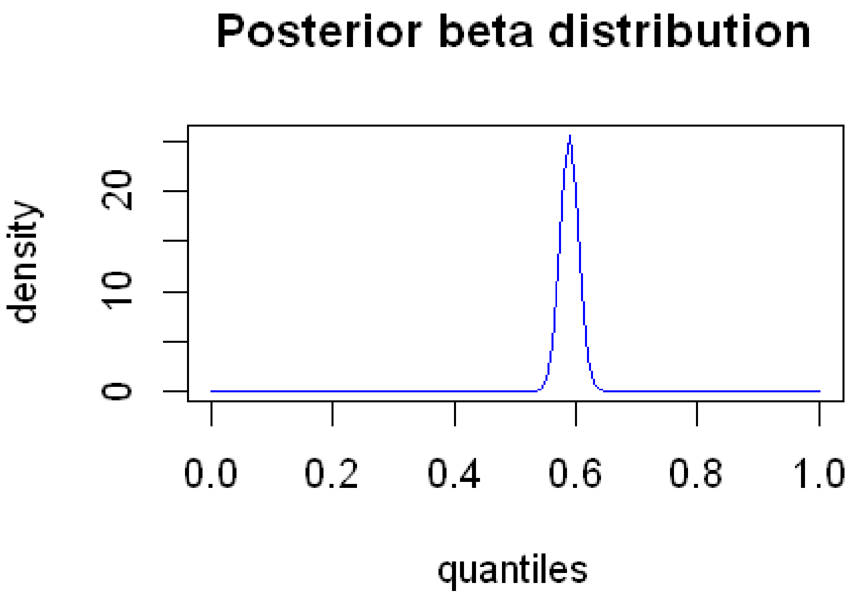
0.588822355289421

$posterior_alpha

590

$posterior_beta

412



In [4]:

```
# Geometric distribution
original_sample = rgeom(n=1000,prob=0.2)
known_params = list(prior_alpha=1,prior_beta=1)
estimate = get_bayes_estimate(original_sample,'geometric',known_params)
estimate
s = rbeta(10000,estimate$posterior_alpha,estimate$posterior_beta)
options(repr.plot.width = 4, repr.plot.height=3)
x = seq(0,1, by=0.01)
plot(x,dbeta(x, estimate$posterior_alpha, estimate$posterior_beta), ylab="density", xlab="quantiles", type ="l",
```

$p

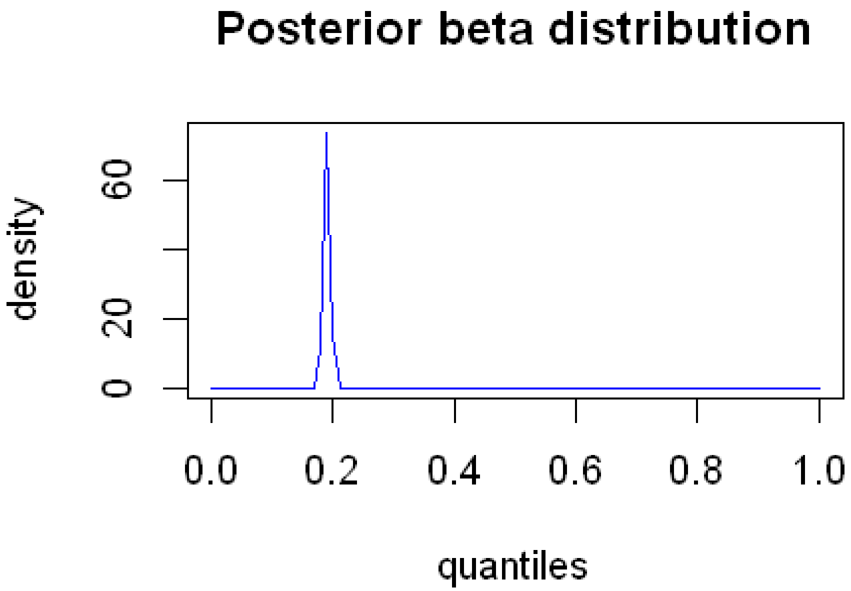
0.190376569037657

$posterior_alpha

1001

$posterior_beta

4257



In [5]:

```
# Poisson distribution
original_sample = rpois(n=1000,lambda=3)
known_params = list(prior_alpha=1,prior_beta=1)
estimate = get_bayes_estimate(original_sample,'poisson',known_params)
estimate
options(repr.plot.width = 4, repr.plot.height=3)
x = seq(0,5, by=0.01)
plot(x,dgamma(x, estimate$posterior_alpha, estimate$posterior_beta), ylab="density", xlab="quantiles", type ="l'
```

$lambda

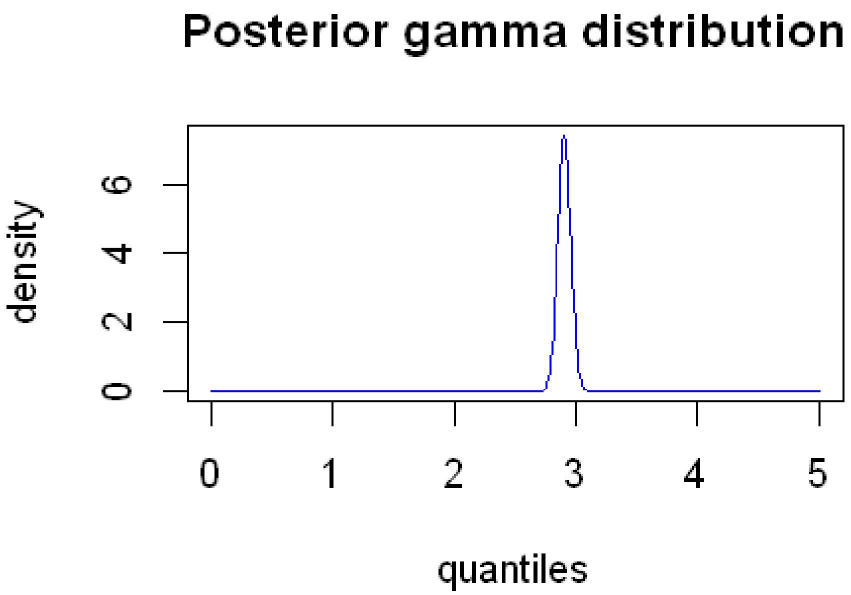
2.9020979020979

$posterior_alpha

2905

$posterior_beta

1001



In [6]:

```
# Normal distribution
original_sample = rnorm(n=1000,mean=3,sd=4)
known_params = list(known_var=4,prior_mu=1,prior_var=4)
estimate = get_bayes_estimate(original_sample,'normal',known_params)
estimate
options(repr.plot.width = 4, repr.plot.height=3)
x = seq(0,6, by=0.01)
plot(x,dnorm(x, estimate$posterior_mu,(estimate$posterior_var)^0.5), ylab="density", xlab="quantiles", type ="l'
```

$mu

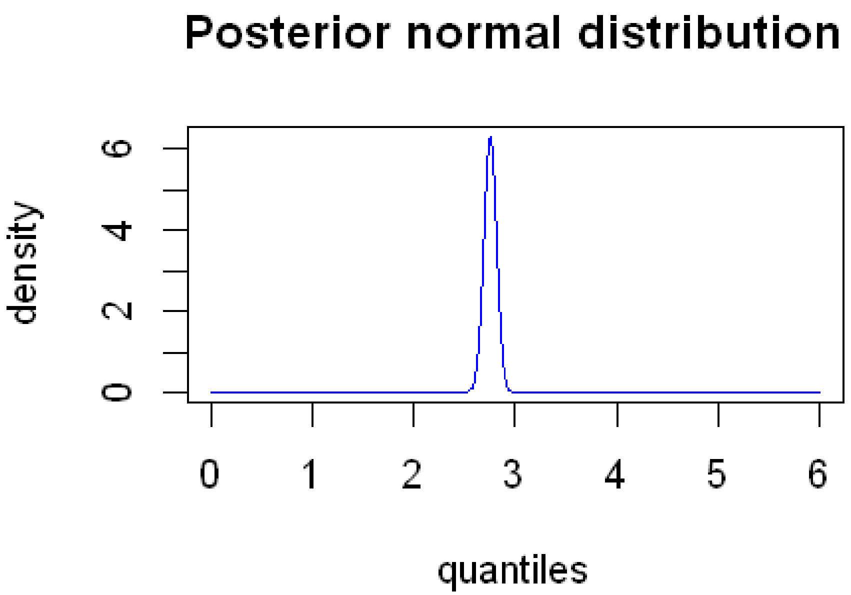
2.74905985387666

$posterior_mu

2.74905985387666

$posterior_var

0.003996003996004



In [7]:
```
# Exponential distribution
original_sample = rexp(n=10000,rate=1.5)
known_params = list(prior_alpha=1,prior_beta=1)
estimate = get_bayes_estimate(original_sample,'exponential',known_params)
estimate
options(repr.plot.width = 4, repr.plot.height=3)
x = seq(0,3, by=0.01)
plot(x,dgamma(x, estimate$posterior_alpha, estimate$posterior_beta), ylab="density", xlab="quantiles", type ="l'
```

$lambda

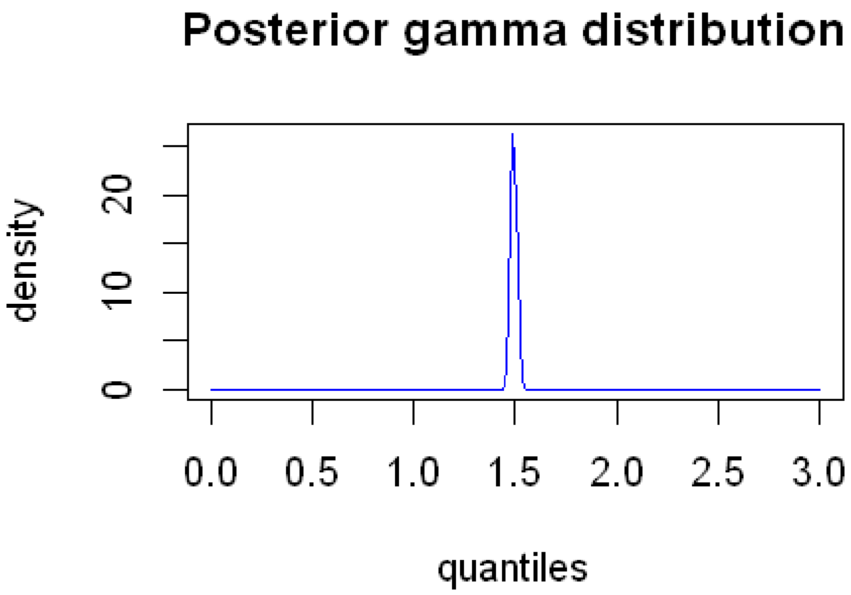
1.4928205098389

$posterior_alpha

10001

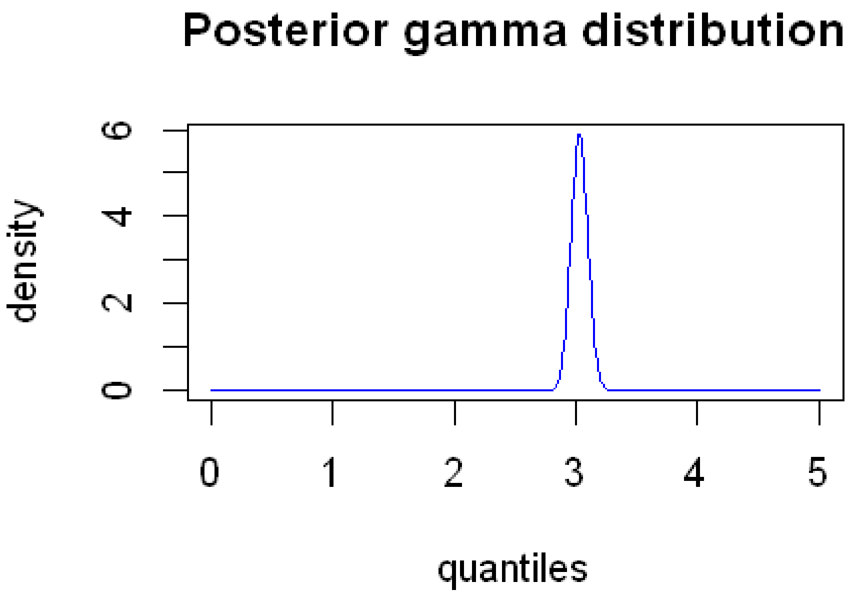
$posterior_beta

6699.39884539723



In [8]:
```
# Gamma distribution
original_sample = rgamma(n=1000,2,3)
known_params = list(known_alpha=2,prior_alpha=1,prior_beta=1)
estimate = get_bayes_estimate(original_sample,'gamma',known_params)
estimate
options(repr.plot.width = 4, repr.plot.height=3)
x = seq(0,5, by=0.01)
plot(x,dgamma(x, estimate$posterior_alpha, estimate$posterior_beta), ylab="density", xlab="quantiles", type ="l'
```

```
$beta
3.02762645062504
$posterior_alpha
2001
$posterior_beta
660.913766157281
```



In [10]:

```
library(rmutil)
```

In [11]:

```
# Uniform distribution
original_sample = runif(1000,min=0,max=4)
known_params = list(prior_alpha=1,prior_beta=1)
estimate = get_bayes_estimate(original_sample,'uniform',known_params)
estimate
options(repr.plot.width = 4, repr.plot.height=3)
x = rpareto(1000,estimate$posterior_alpha,estimate$posterior_beta)
plot(density(x),main='Posterior pareto distribution')
```

$b

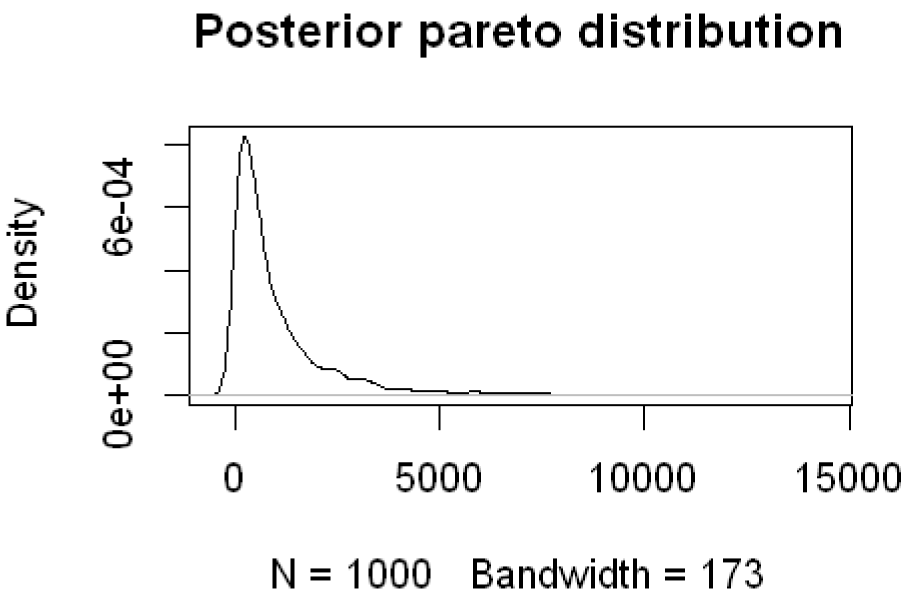
3.99645483011845

$posterior_alpha

1001

$posterior_beta

3.9924623677507



In [13]:

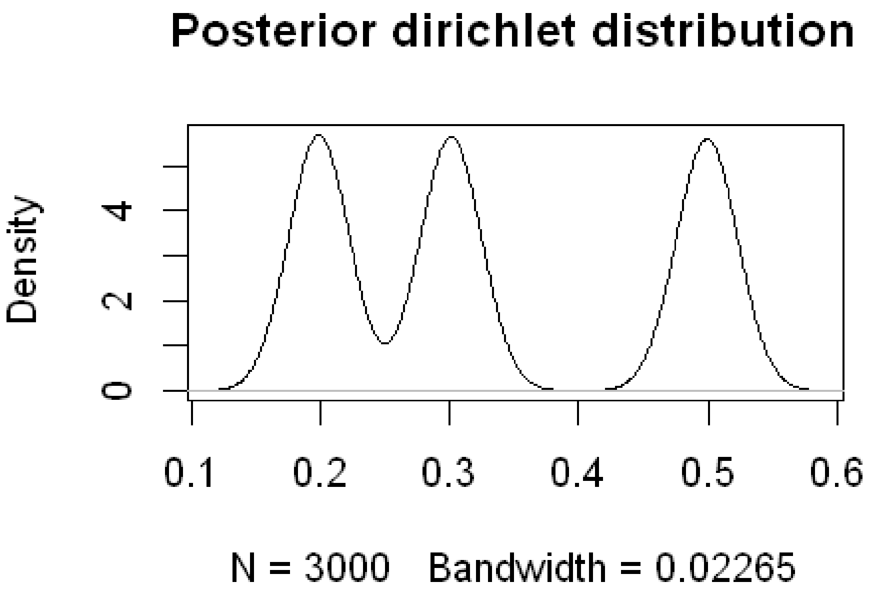
```
library('DirichletReg')
```

In [14]:

```
# Multinomial distribution
# As posterior dirichlet has 3 paremeters, each data point becomes 3-dimensional vector.
# Hence, density plot becomes 4-dimensional (density + 3 dimensions of a data point).
# So, it is plotted in a different way.
original_sample = rmultinom(1000,5,c(0.2,0.3,0.5))
known_params = list(prior_alpha=c(1,1,1))
estimate = get_bayes_estimate(original_sample,'multinomial',known_params)
estimate
s = rdirichlet(1000,estimate$posterior_alpha)
options(repr.plot.width = 4, repr.plot.height=3)
plot(density(s),main='Posterior dirichlet distribution')
```

```
$prob
0.198880671597042   0.30161902858285   0.499500299820108

$posterior_alpha
995   1509   2499
```



In [ ]: