IndeptGamma

Ashwath Raj

2/1/2021

This example taken from http://www.mas.ncl.ac.uk/~ndjw1/teaching/sim/metrop/indep.r We will write a metropolis-hastings independence sampler for a gamma rv based on normal candidates with the same mean and variance.

If you don't have it already, you will need the coda library:

```
#install.packages("coda")
library(coda)
```

```
## Warning: package 'coda' was built under R version 4.0.4
```

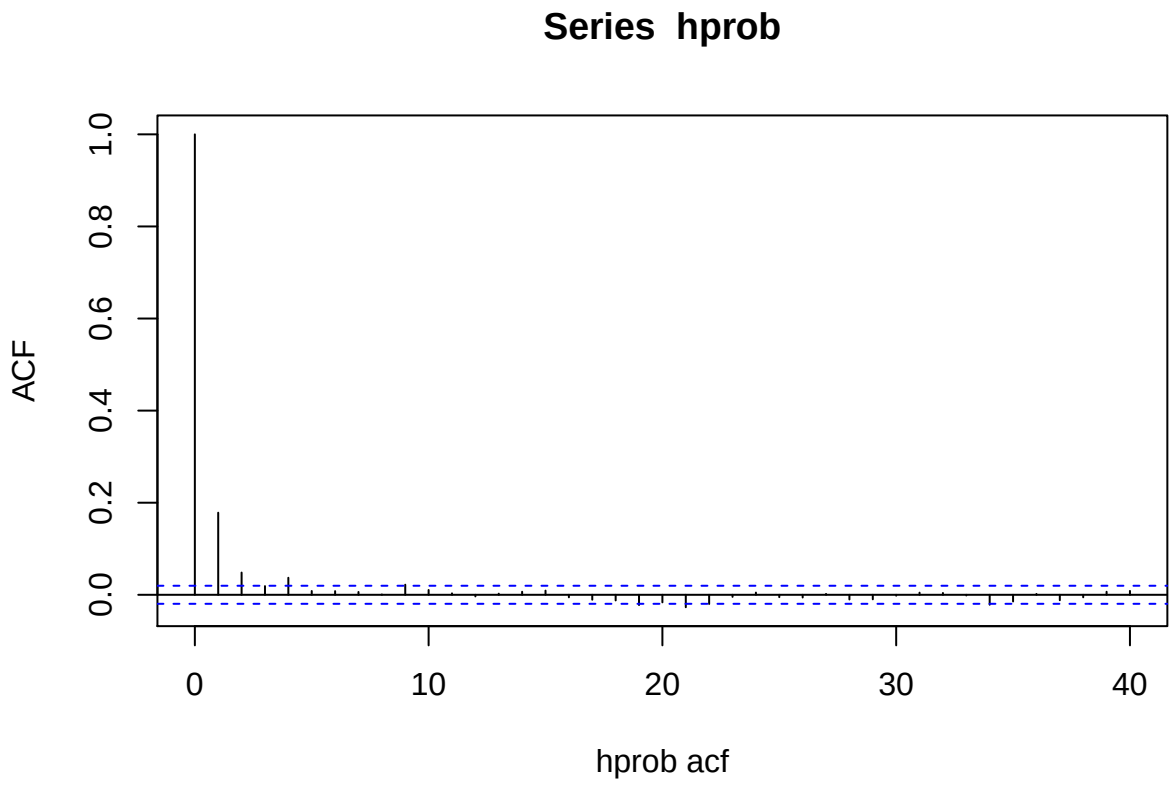
Set the seed and define a function to do this:

```
set.seed(37)

# minimizing energy:
#   q(x <-> x'), dE(x->x') = f(x')-f(x)
#   Move from x to x' with hprob min{1, exp(-dE(x,x')/T)}
#   T can be variable but decreasing: T/(T+sig)
#   Can we use h=min{1, [f(x')q(x'->x)] / [f(x)q(x->x')]} here?
#
#Attempts:
  # Hastings Ratio: change the division to multiplication

plotgamm<- function(vec3,vec4){
par(mfrow=c(2,2))
plot(ts(vec3))
plot(ts(vec3)[1:1000],type='l',main="vec3")
acf(vec3)
hist(vec3[1000:10000],30,main="vec3")
plot(ts(vec4),main="vec4")
plot(ts(vec4)[1:1000],type='l',main="vec4")
acf(vec4)
hist(vec4[1000:10000],30,main="vec2")
# curve(dgamma(x,0.1,0.01))

MCMC3<-mcmc(vec3,start=1000)
MCMC4<-mcmc(vec4,start=1000)

# combine different mcmc chain objects to an mcmc list.
Combined2<-mcmc.list(list(MCMC3,MCMC4))

# gelman functions are
gelman.plot(Combined2) # for plots
```

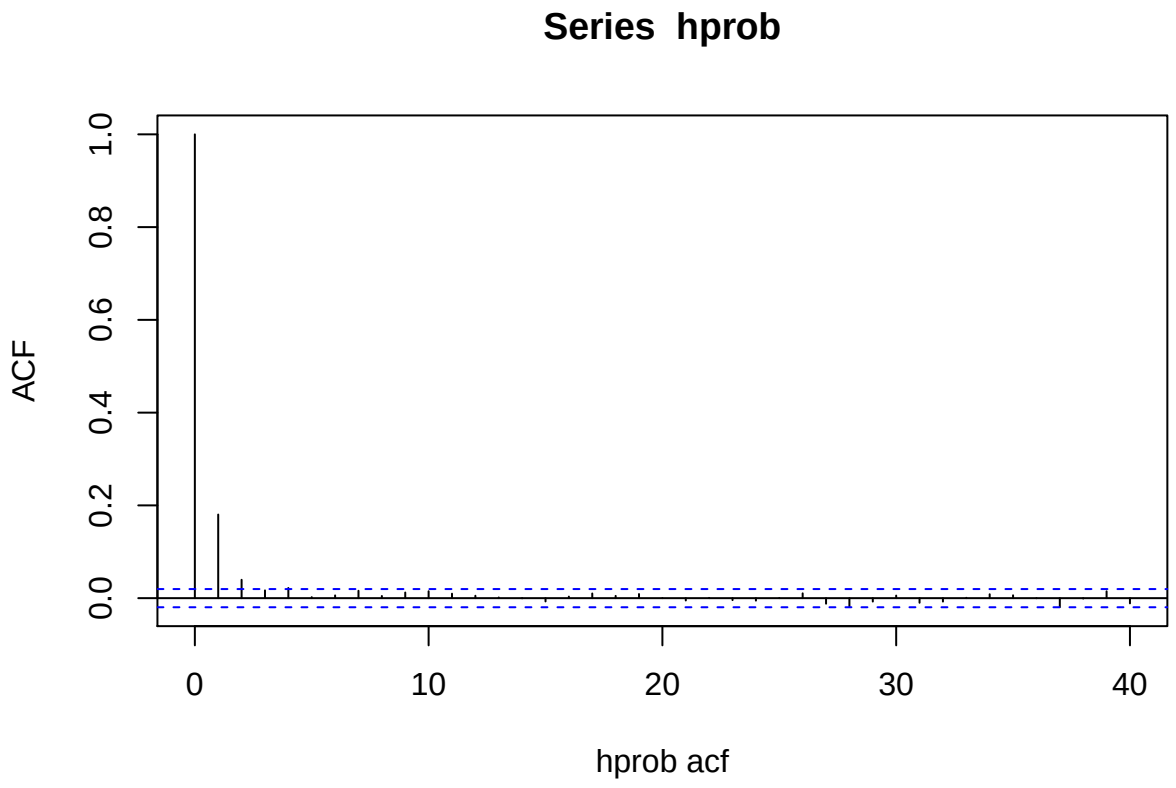
```
print(gelman.diag(Combined2)) # for diagnostic values
}

gamm<-function (n, a, b)
{
  # browser()
  mu <- a/b   # the mean of the gamma distribution
  sig <- sqrt(a/(b * b))   # the stadard deviation of the gamma distn
  vec <- vector("numeric", n)   # this is where we are going to put the random variables we generate
  x <- a/b
  hprob = 1:n

  vec[1] <- x # We arbitrarily start the MCMC process at the mean
  for (i in 2:n) {
    can <- rnorm(1, mu, sig)
    hprob[i] <- min(1, (dgamma(can, a, b)*dgamma(x,a,b))/(dnorm(can, mu, sig)*dnorm(x, mu, sig)))   # w
    u <- runif(1)
    if (u < hprob[i])
      x <- can
    vec[i] <- x
  }

  acf(hprob, xlab = "hprob acf")
  return (vec)
}

gammOG<-function (n, a, b)
{
  mu <- a/b   # the mean of the gamma distribution
  sig <- sqrt(a/(b * b))   # the stadard deviation of the gamma distn
  vec <- vector("numeric", n)   # this is where we are going to put the random variables we generate
  x <- a/b
  vec[1] <- x # We arbitrarily start the MCMC process at the mean
  for (i in 2:n) {
    can <- rnorm(1, mu, sig)
    hprob <- min(1, (dgamma(can, a, b)/dgamma(x,a,b))/(dnorm(can, mu, sig)/dnorm(x, mu, sig)))   # wher
    u <- runif(1)
    if (u < hprob)
      x <- can
    vec[i] <- x
  }
  return (vec)
}
```

Here's what happens when we use it:

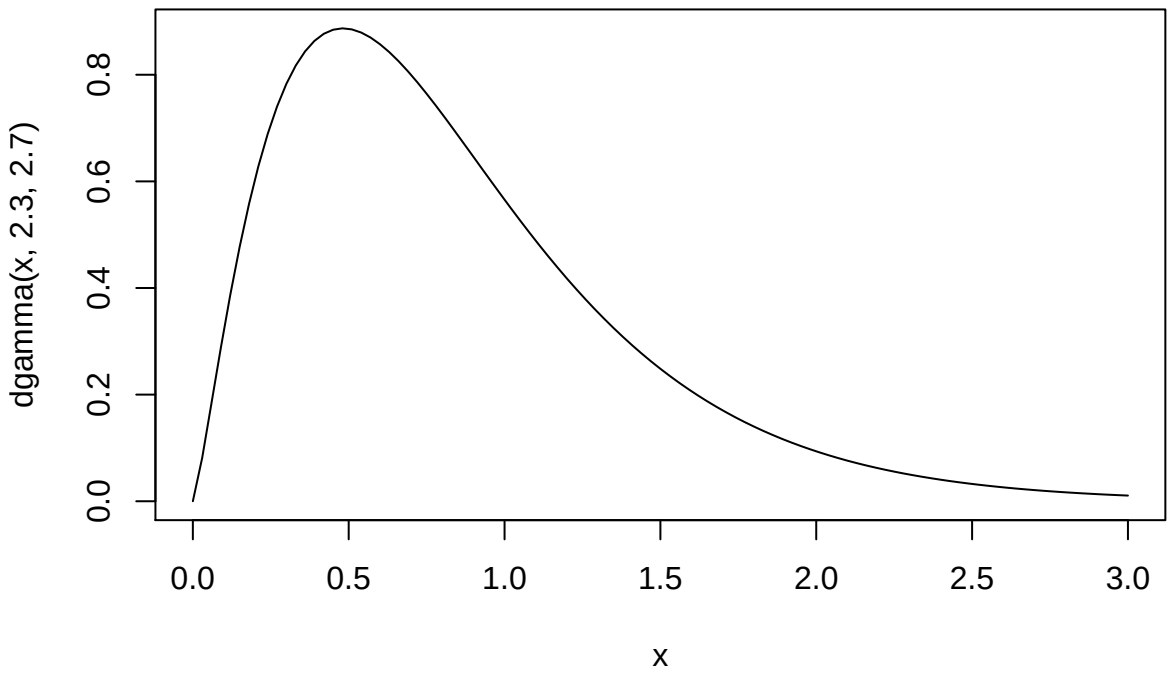
```
vec1<-gamm(10000,2.3,2.7)
```



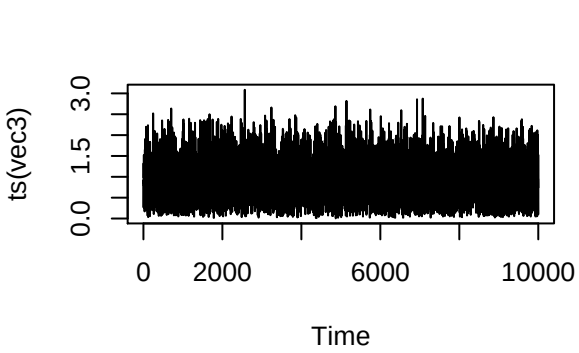

```
vec2<-gamm(10000,2.3,2.7)
```

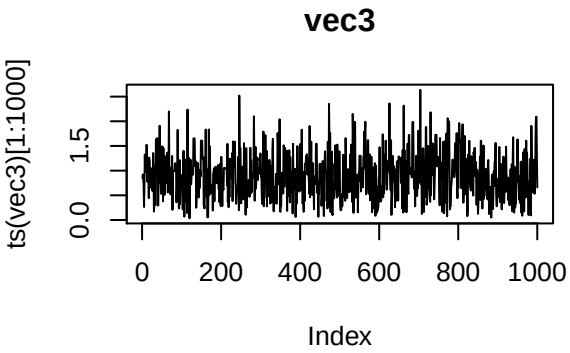


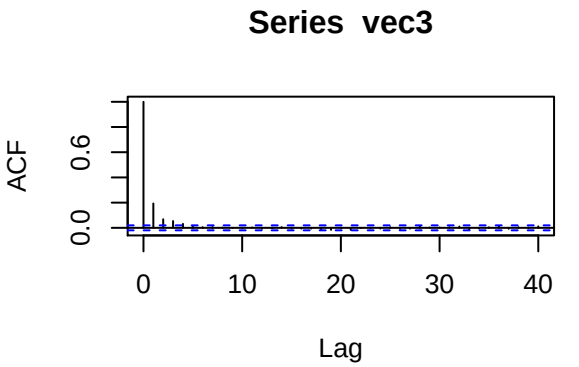

```
curve(dgamma(x,2.3,2.7),from=0,to=3)
```

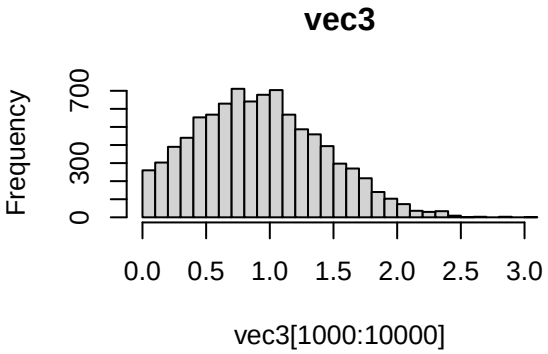


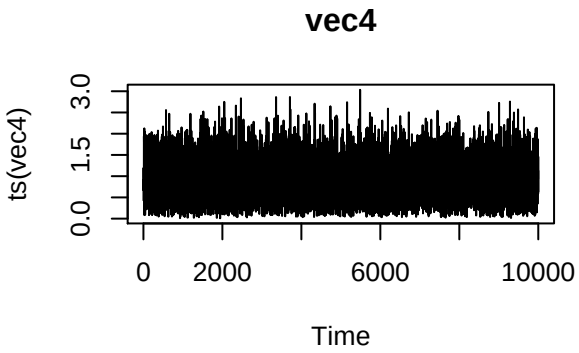

```
plotgamm(vec1,vec2)
```

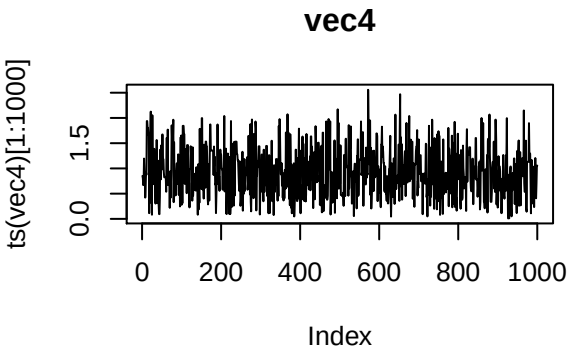

ts(vec3)
0.0
1.5
3.0
0
2000
6000
10000
Time

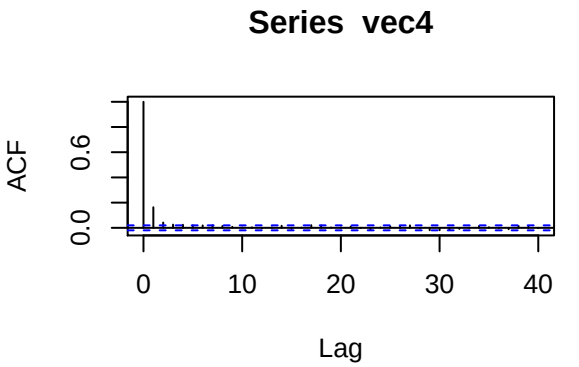

vec3
ts(vec3)[1:1000]
0.0
1.5
0
200
400
600
800
1000
Index

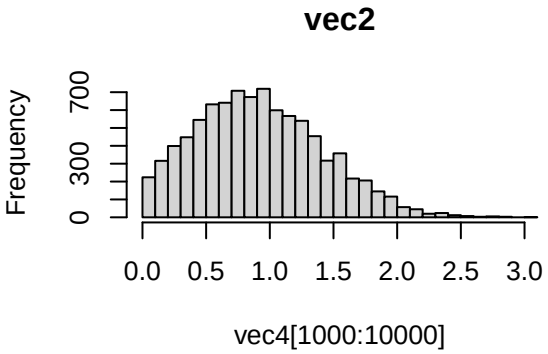

Series vec3
ACF
0.0
0.6
0
10
20
30
40
Lag

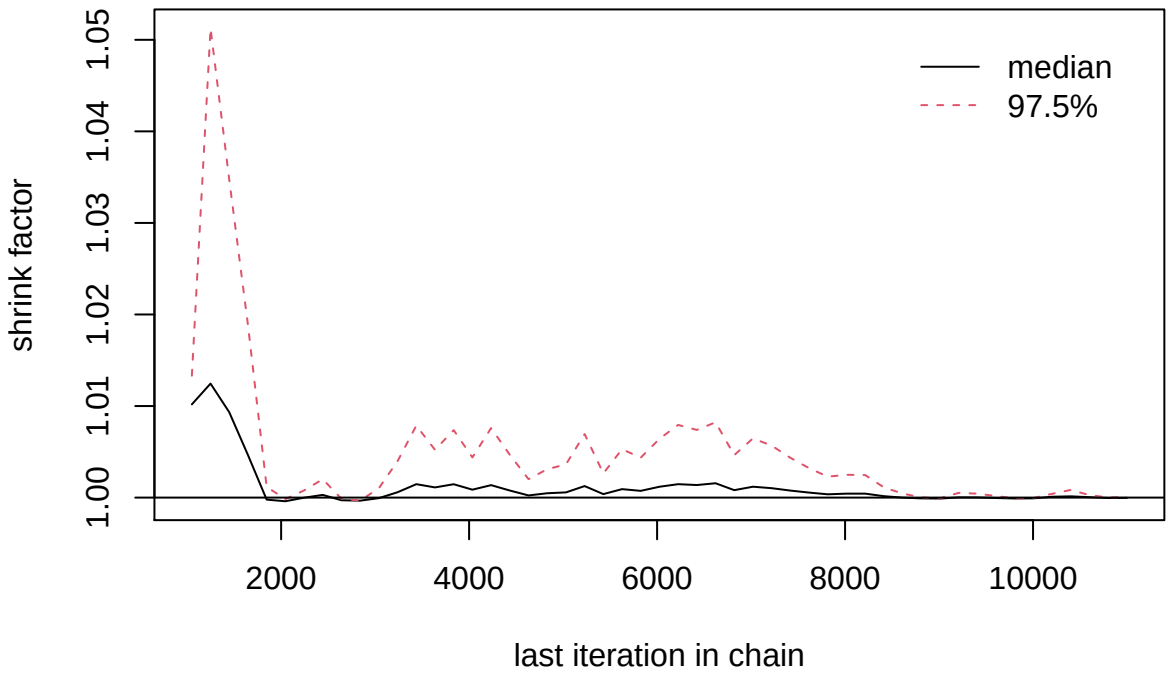

vec3
Frequency
0
300
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0
vec3[1000:10000]


vec4
ts(vec4)
0.0
1.5
3.0
0
2000
6000
10000
Time


vec4
ts(vec4)[1:1000]
0.0
1.5
0
200
400
600
800
1000
Index

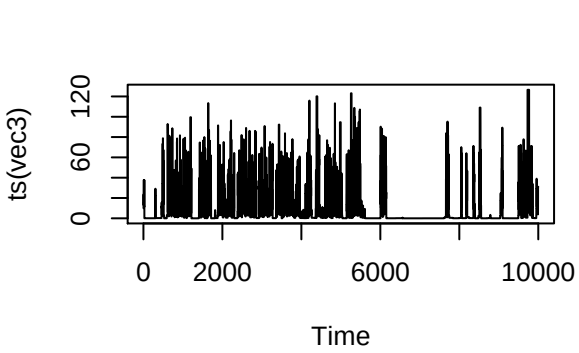

Series vec4
ACF
0.0
0.6
0
10
20
30
40
Lag

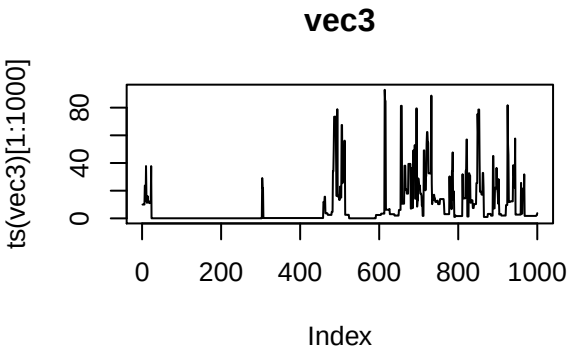

vec2
Frequency
0
300
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0
vec4[1000:10000]

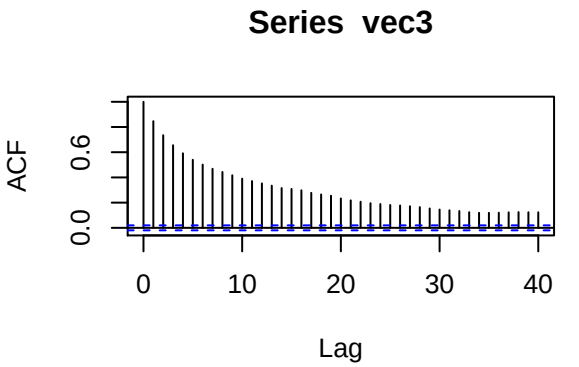


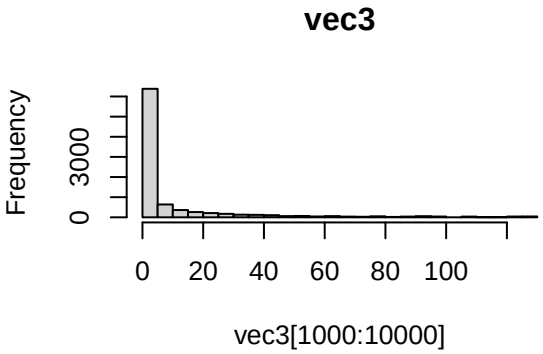

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

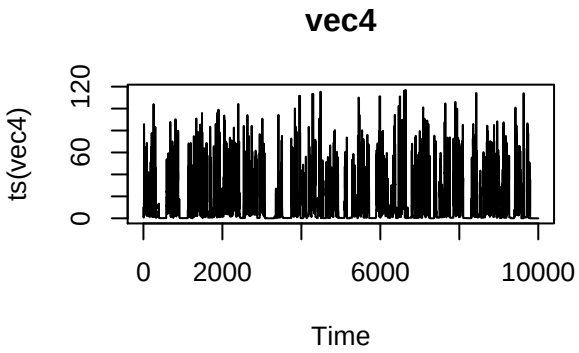
```
vec3<-gammOG(10000,0.1,0.01)
vec4<-gammOG(10000,0.1,0.01)

plotgamm(vec3,vec4)
```

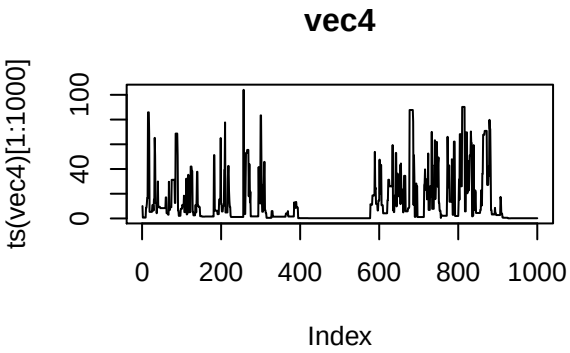

ts(vec3)
0
60
120
0
2000
6000
10000
Time

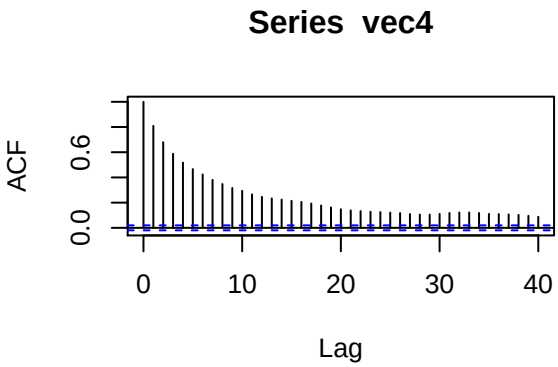

vec3
ts(vec3)[1:1000]
0
40
80
0
200
400
600
800
1000
Index

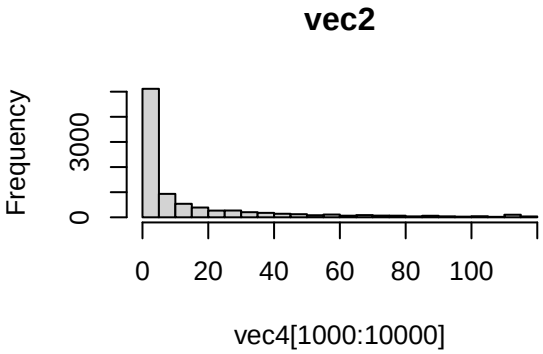

Series vec3
ACF
0.0
0.6
0
10
20
30
40
Lag

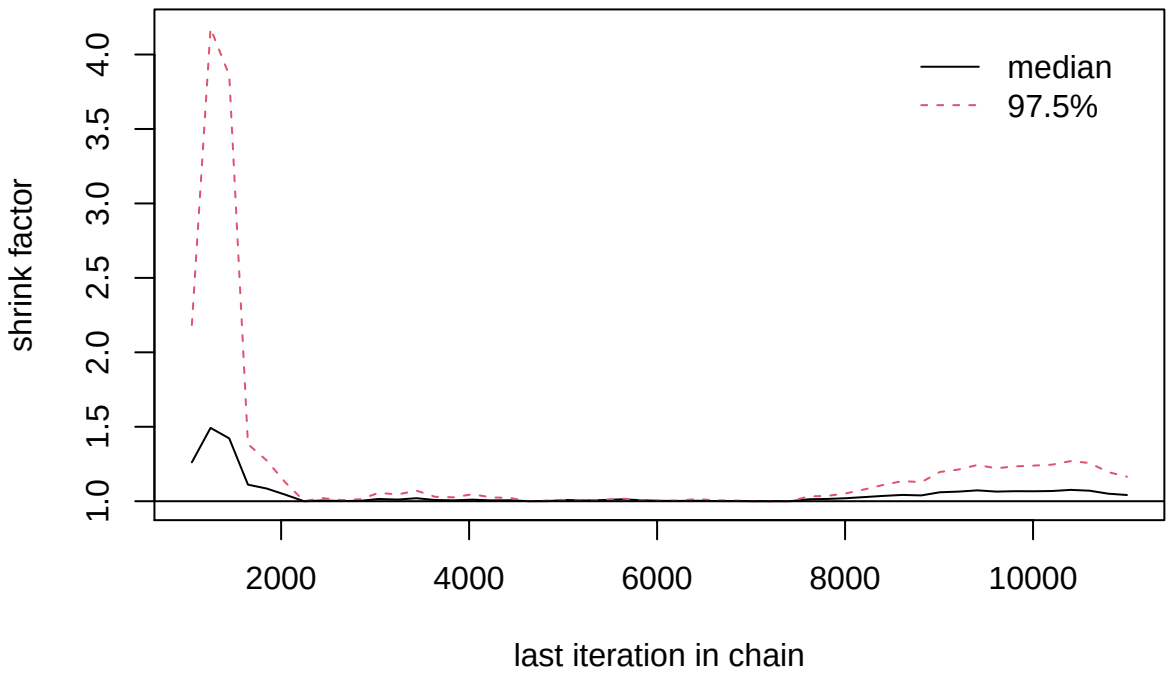

vec3
Frequency
0
3000
0
20
40
60
80
100
vec3[1000:10000]


vec4
ts(vec4)
0
60
120
0
2000
6000
10000
Time


vec4
ts(vec4)[1:1000]
0
40
100
0
200
400
600
800
1000
Index


Series vec4
ACF
0.0
0.6
0
10
20
30
40
Lag


vec2
Frequency
0
3000
0
20
40
60
80
100
vec4[1000:10000]

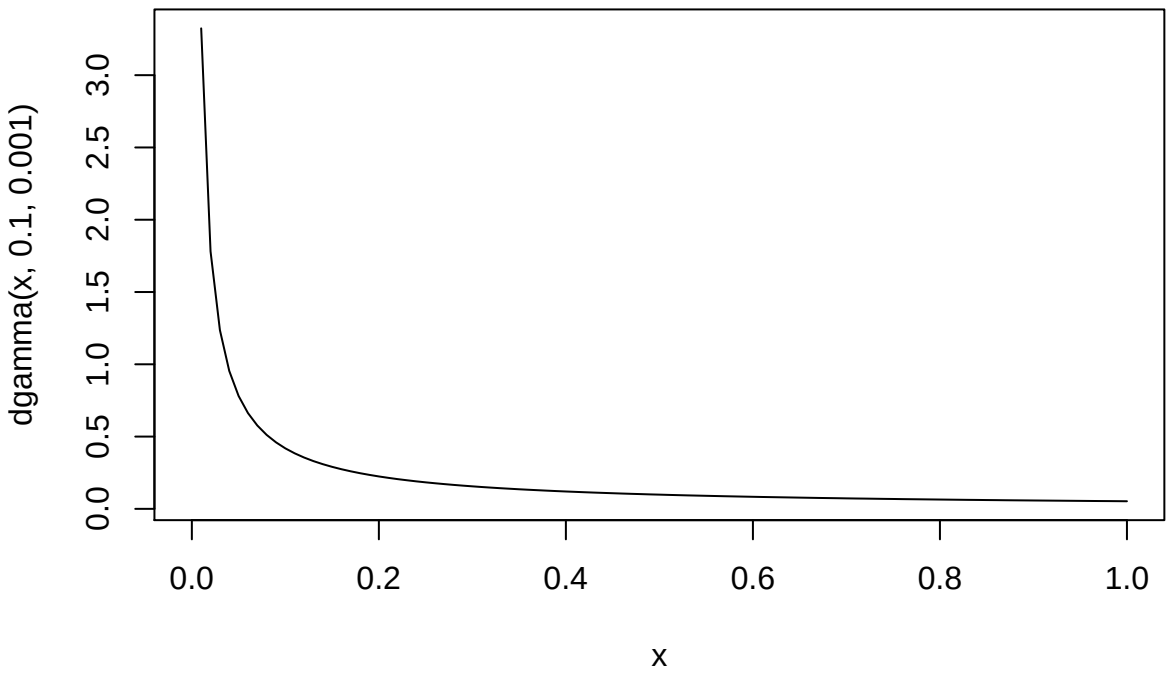



```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.04       1.16
```

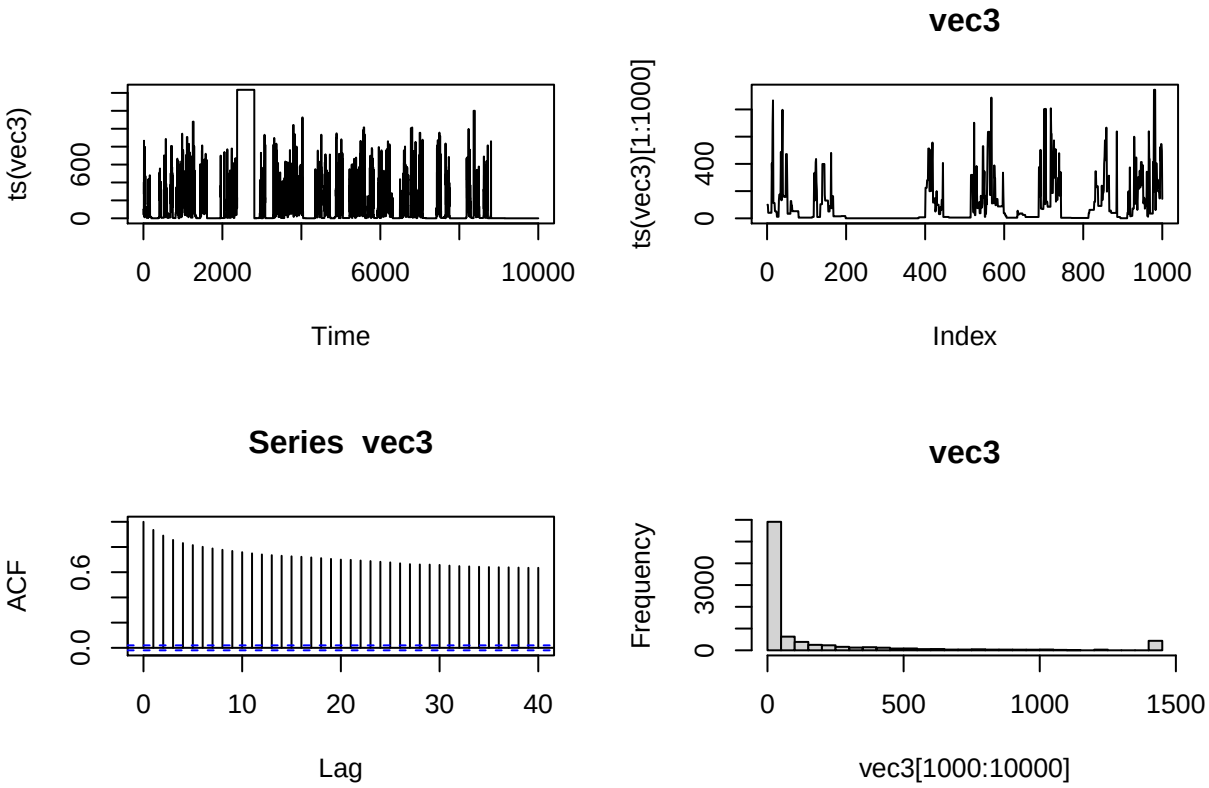
This one doesn't look so good. It seems to get stuck lfor long periods of time at low value, and even sometimes at high values.

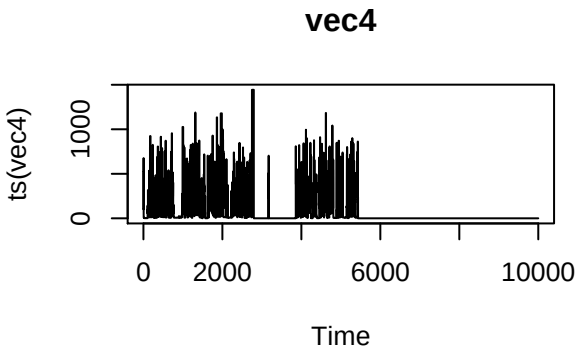
Testing an even smaller "b" (rate = 1/scale) term for original gamm

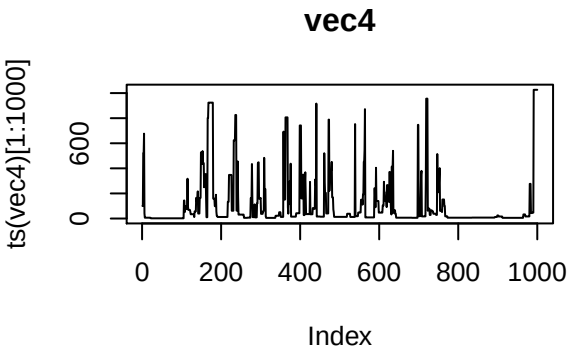
```
vec3<-gammOG(10000,0.1,0.001)
vec4<-gammOG(10000,0.1,0.001)

curve(dgamma(x,0.1,0.001))
```

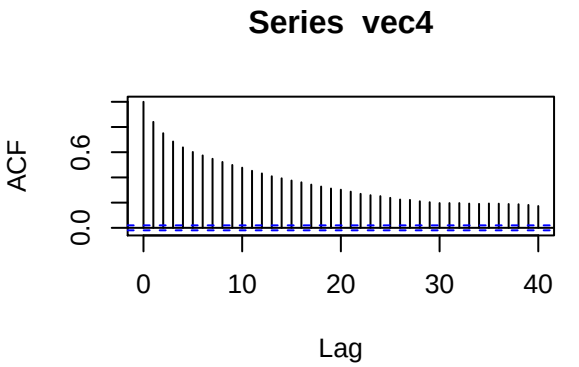


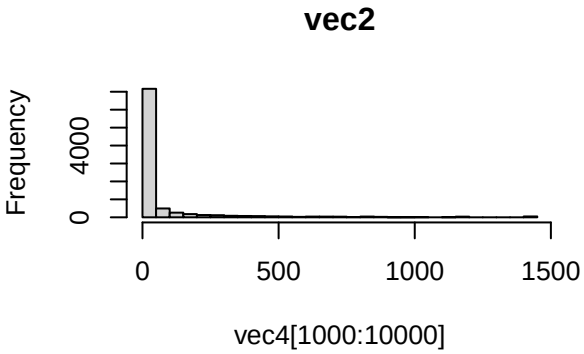

```
plotgamm(vec3,vec4)
```

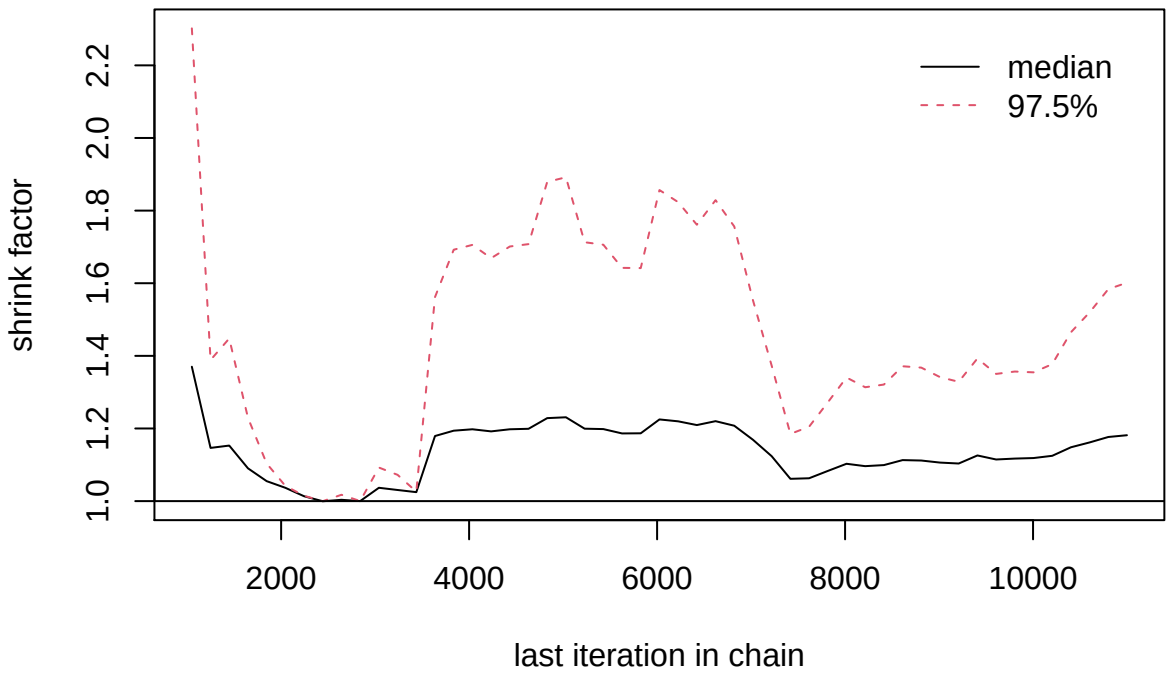

ts(vec3)
0
600
0
2000
6000
10000
Time
vec3
ts(vec3)[1:1000]
0
400
0
200
400
600
800
1000
Index
Series vec3
ACF
0.0
0.6
0
10
20
30
40
Lag
vec3
Frequency
0
3000
0
500
1000
1500
vec3[1000:10000]


vec4
ts(vec4)
0
1000
0
2000
6000
10000
Time


vec4
ts(vec4)[1:1000]
0
600
0
200
400
600
800
1000
Index


Series vec4
ACF
0.0
0.6
0
10
20
30
40
Lag

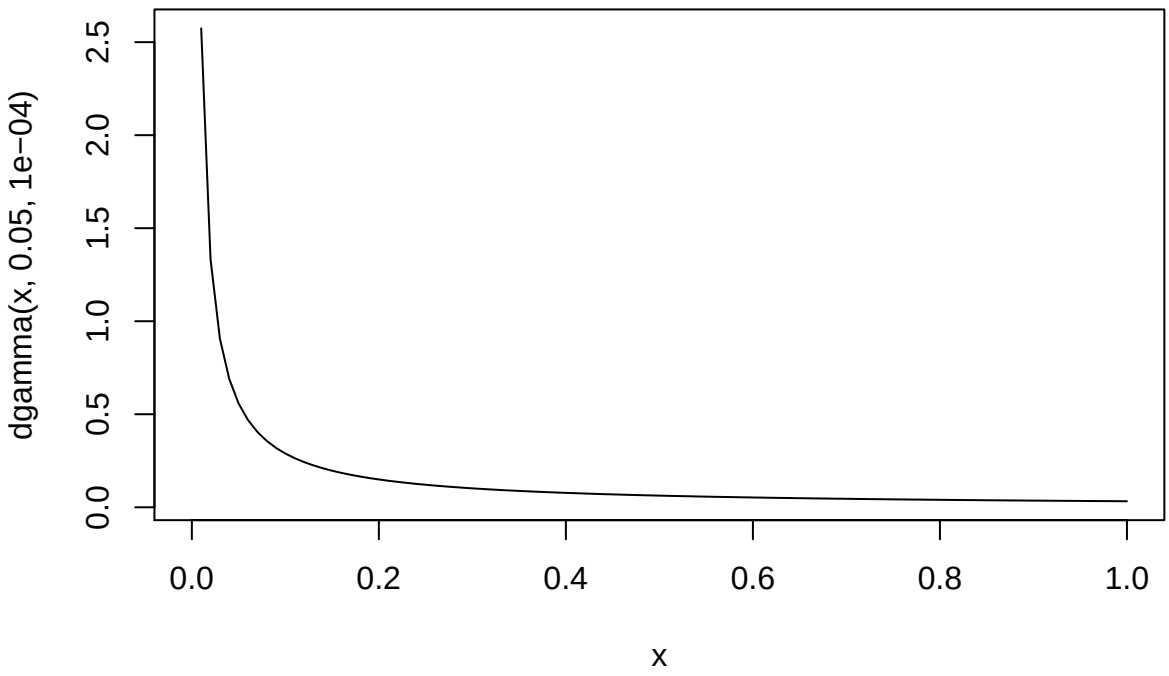

vec2
Frequency
0
4000
0
500
1000
1500
vec4[1000:10000]



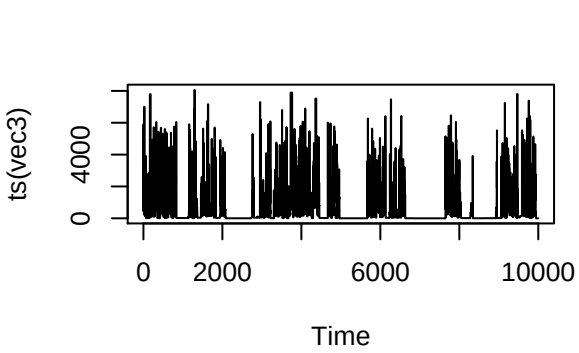

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.18        1.6
```

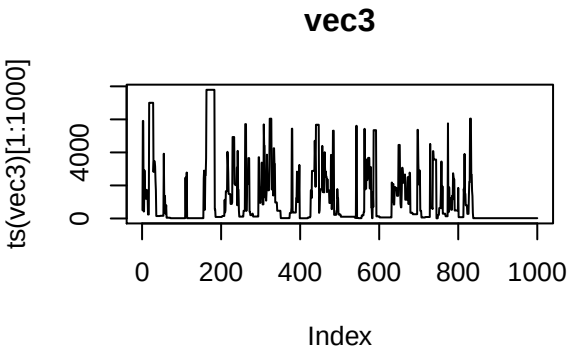
Testing an even smaller "a" (rate = 1/scale) term for original gamm

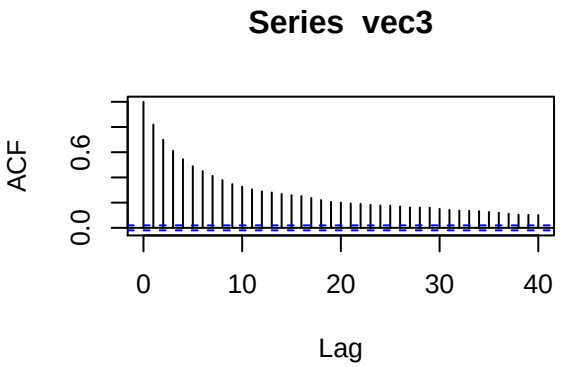
```
#Testing a smaller "a" shape term for OG
vec3<-gammOG(10000,0.05,0.0001)
vec4<-gammOG(10000,0.05,0.0001)

curve(dgamma(x,0.05,0.0001))
```

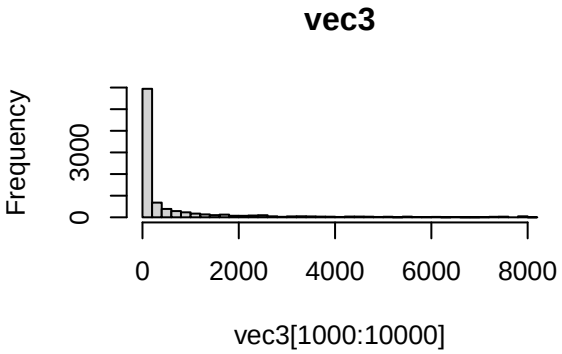


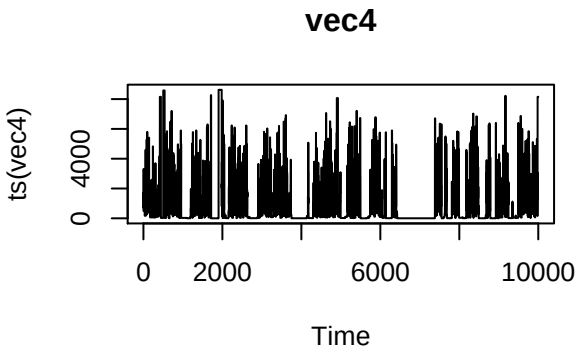

```
plotgamm(vec3,vec4)
```

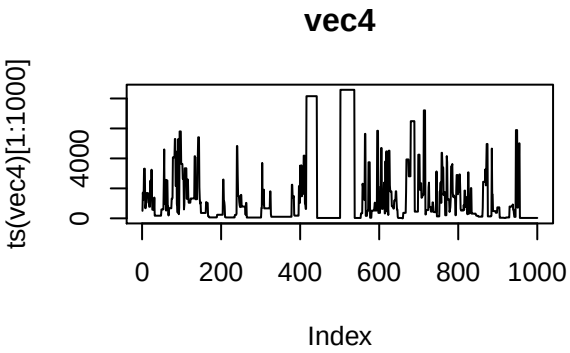

ts(vec3)
0
4000
0
2000
6000
10000
Time

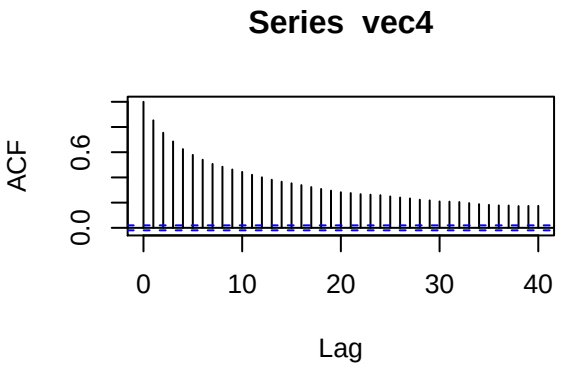

vec3
ts(vec3)[1:1000]
0
4000
0
200
400
600
800
1000
Index

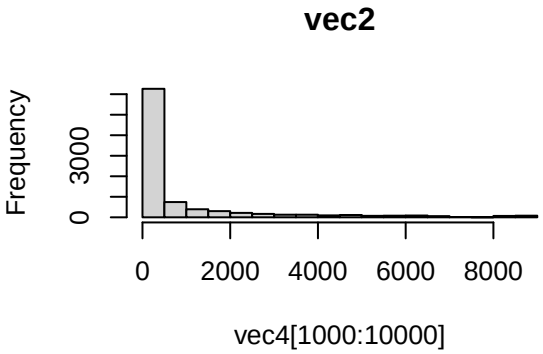

Series vec3
ACF
0.0
0.6
0
10
20
30
40
Lag

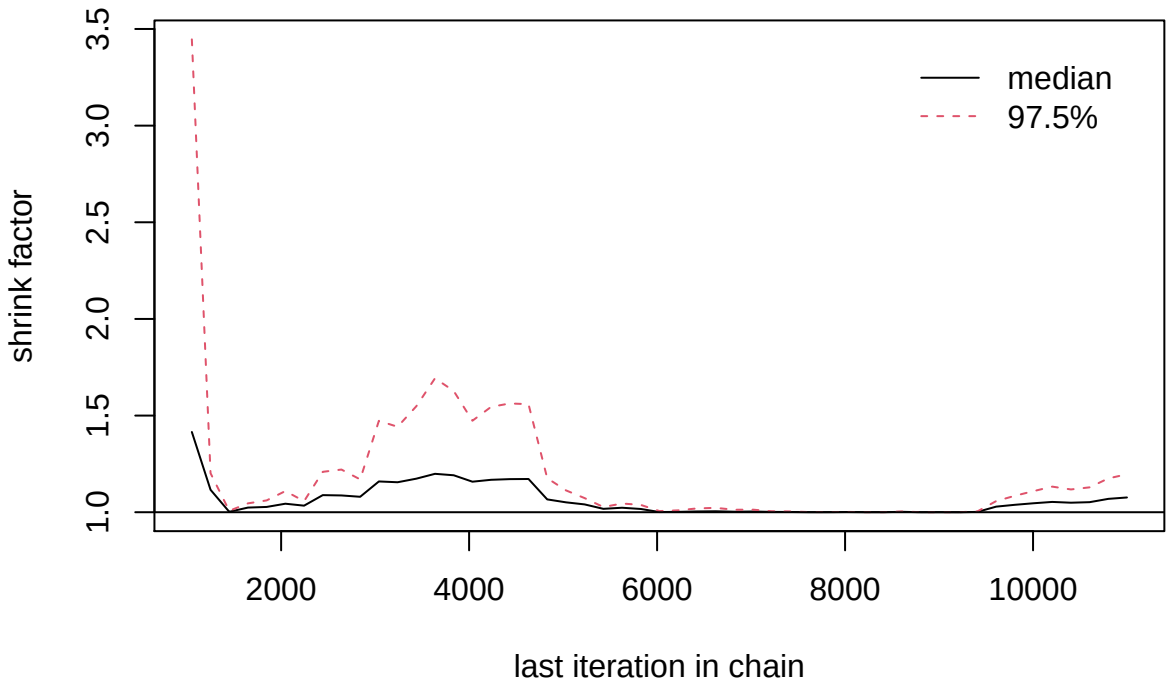

vec3
Frequency
0
3000
0
2000
4000
6000
8000
vec3[1000:10000]


vec4
ts(vec4)
0
4000
0
2000
6000
10000
Time


vec4
ts(vec4)[1:1000]
0
4000
0
200
400
600
800
1000
Index


Series vec4
ACF
0.0
0.6
0
10
20
30
40
Lag


vec2
Frequency
0
3000
0
2000
4000
6000
8000
vec4[1000:10000]




```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.08        1.2
```

Your assignment is as follows:

Try out this code for different shape and scale parameters. Notice, as above, how there is a bit of a problem with the sampler getting "stuck" at very small values for some values of a and b.
(To see what the distribution should look like use this command: curve(dgamma(x,0.1,0.01))).

Thoughts: Some ideas of what's going wrong: The default q term divides 2 divided gamma densities by 2 divided normal densities. "Can" is already normally generated, and the standard deviation term is very large (31) for small values of "b". That might be helpful for getting out of troughs. We only update "can, hprob, x, u, and vec". We have to ask, is this q symmetric (x <-> x')?

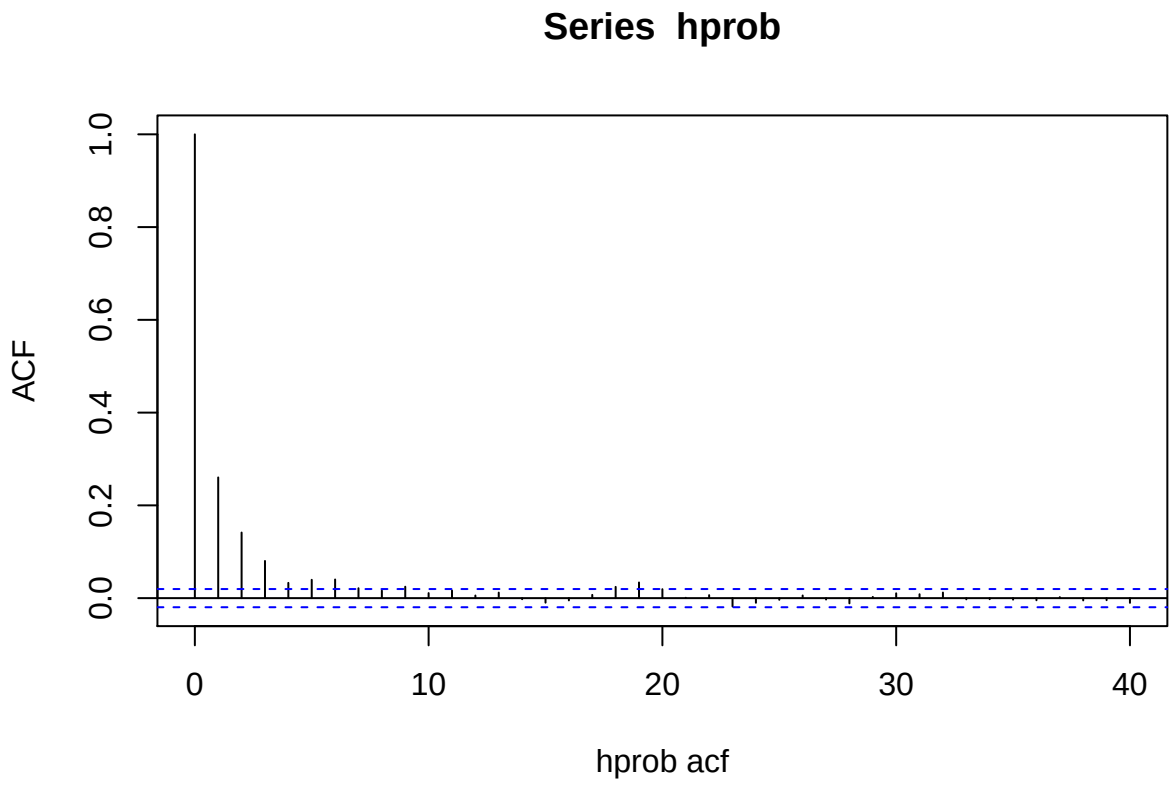
Performance improvement: The low values of a and b now perform very well, without getting stuck. The acf decreases to near zero, the histogram closely follows the dgamma curve, and the gelman point estimates are under 1.01.

1. Modify the code to keep track of acceptance probabilities (h) and plot the acf
   - Done through gamm() hprob array. It's the first acf plot.
2. When does the sampling scheme do worst? In what ways is it struggling?
   - The sampling scheme is worse with smaller "b" and incredibly poor with small "a". It primarily gets stuck at small f(x) values for long periods of time.

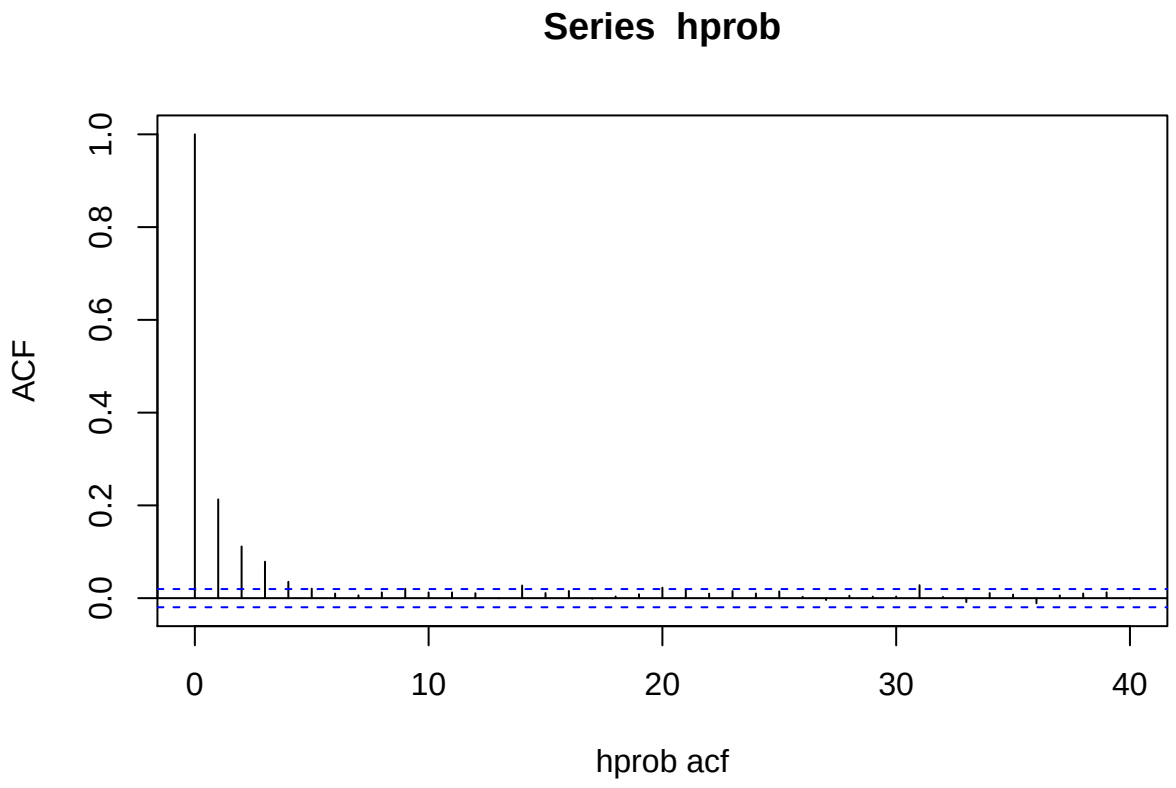
3. Modify this sampling scheme to make it more efficient. (i.e. to remove the issue of it getting stuck at small values.) -I made the gamm function hprob similar to the Hastings ratio by multiplying the two gamma "x" and "can" terms as well as the two norm terms. This modification does well on most ranges of a and b values. However it stuggles with "a" values under .01, which is still an order of magnitude lower than the original difficult value of .1.

Testing the original "a" and "b" terms for fixed gamm

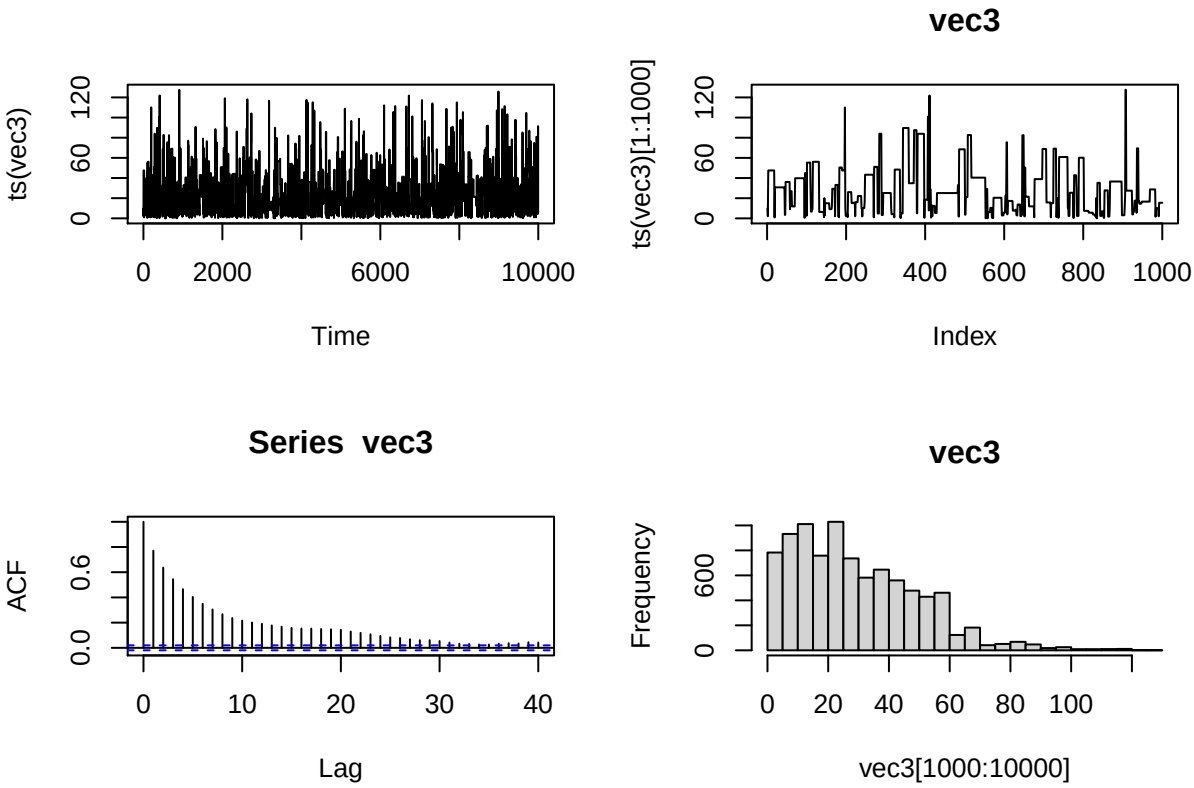
```
vec3<-gamm(10000,0.1,0.01)
```

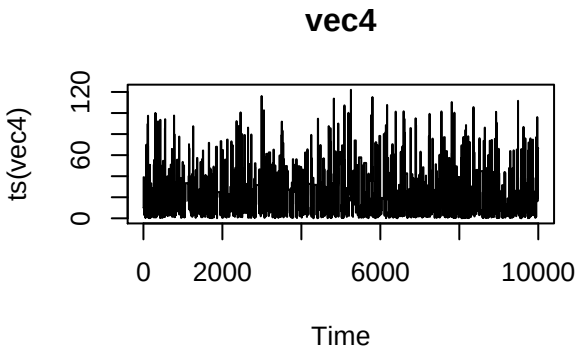


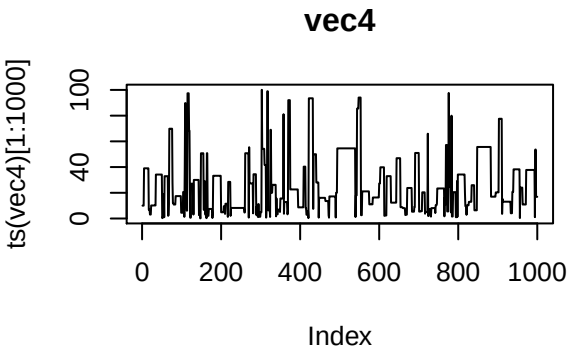

```
vec4<-gamm(10000,0.1,0.01)
```

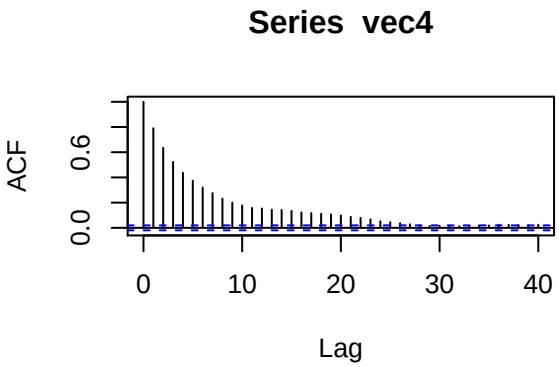


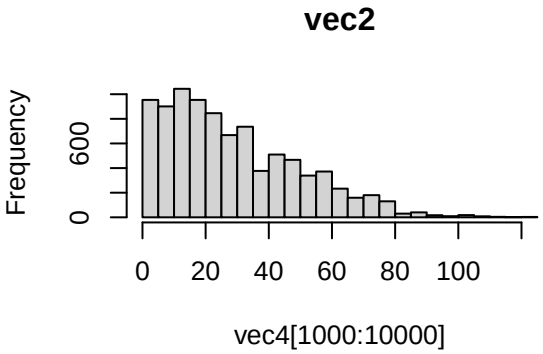

```
plotgamm(vec3,vec4)
```

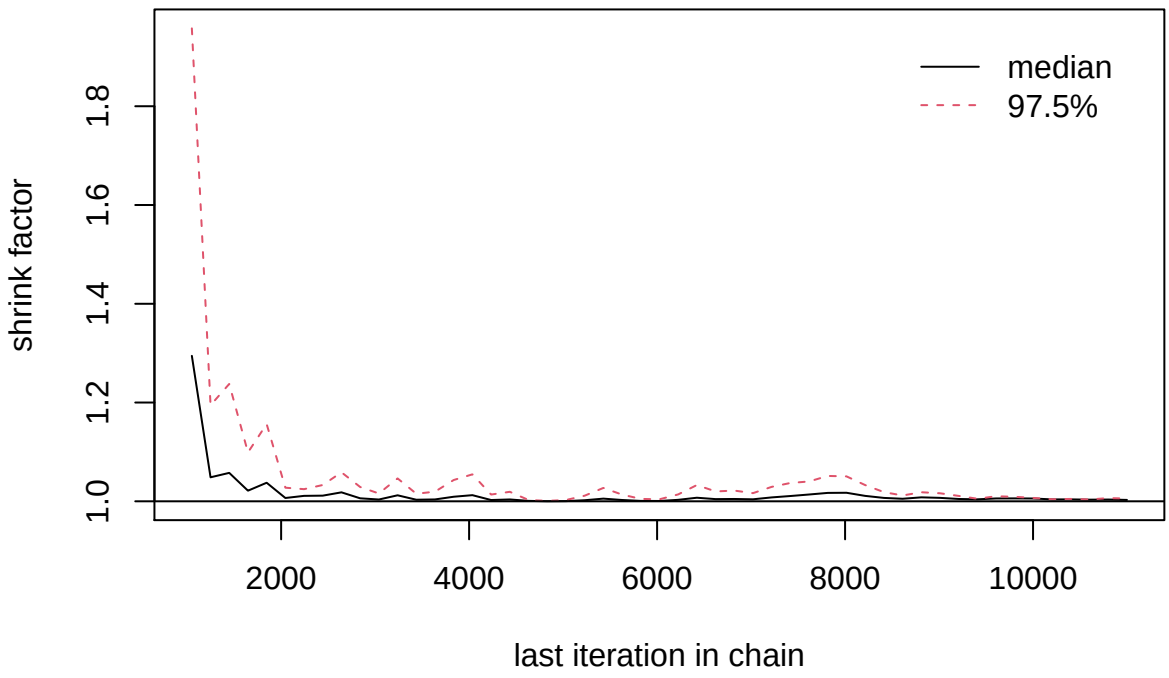

ts(vec3)
0
60
120
0
2000
6000
10000
Time
vec3
ts(vec3)[1:1000]
0
60
120
0
200
400
600
800
1000
Index
Series vec3
ACF
0.0
0.6
0
10
20
30
40
Lag
vec3
Frequency
0
600
0
20
40
60
80
100
vec3[1000:10000]


vec4
ts(vec4)
0
60
120
0
2000
6000
10000
Time


vec4
ts(vec4)[1:1000]
0
40
100
0
200
400
600
800
1000
Index


Series vec4
ACF
0.0
0.6
0
10
20
30
40
Lag

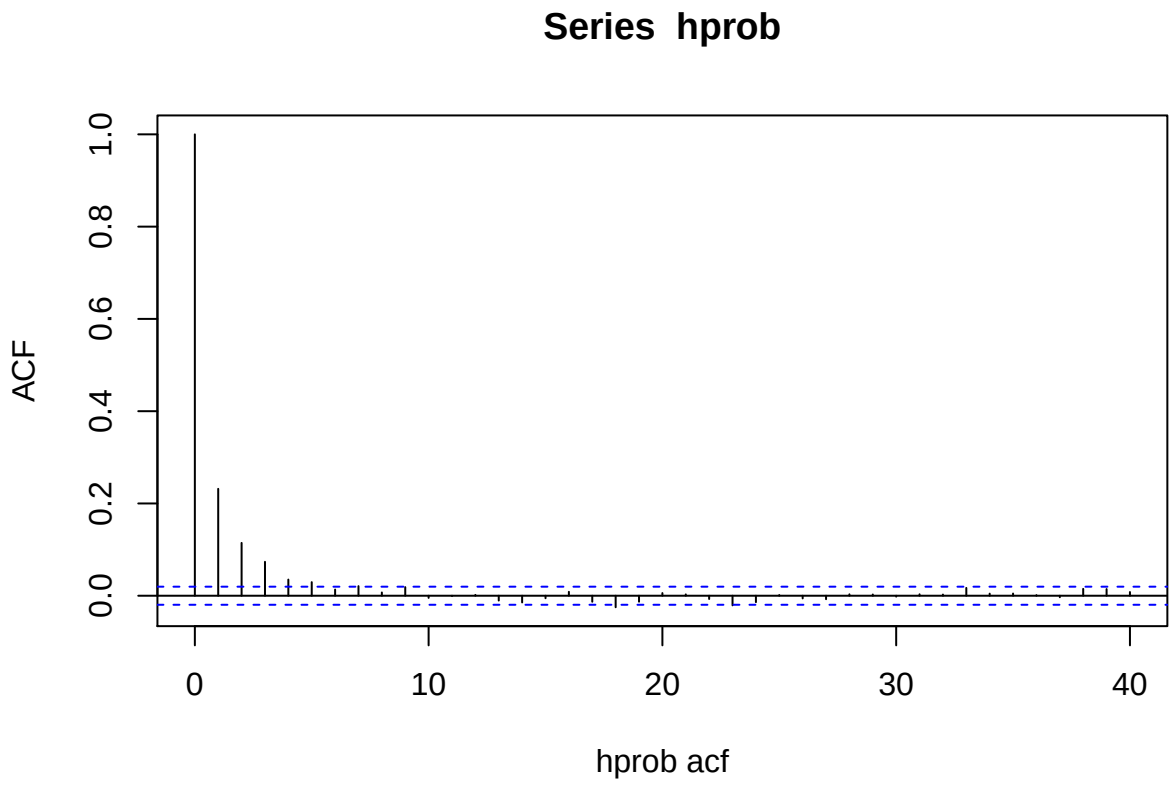

vec2
Frequency
0
600
0
20
40
60
80
100
vec4[1000:10000]



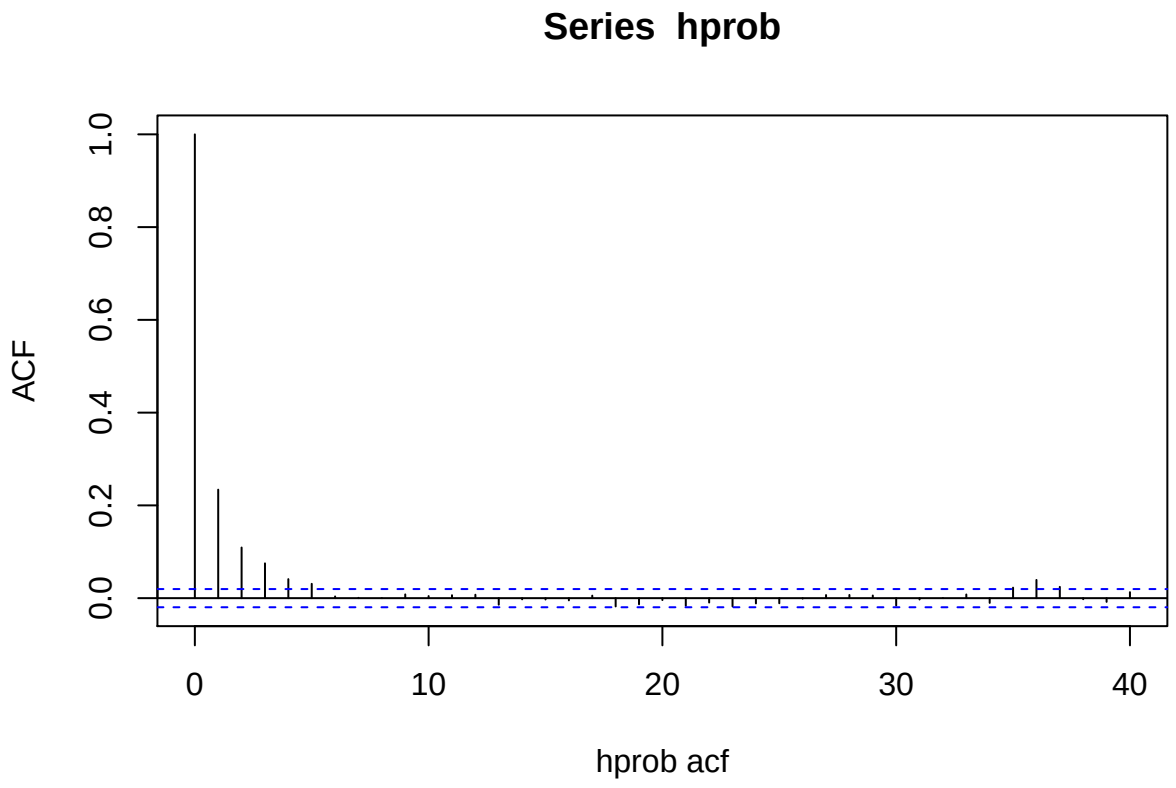

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1       1.01
```

Testing an even smaller "b" (rate = 1/scale) term for fixed gamm

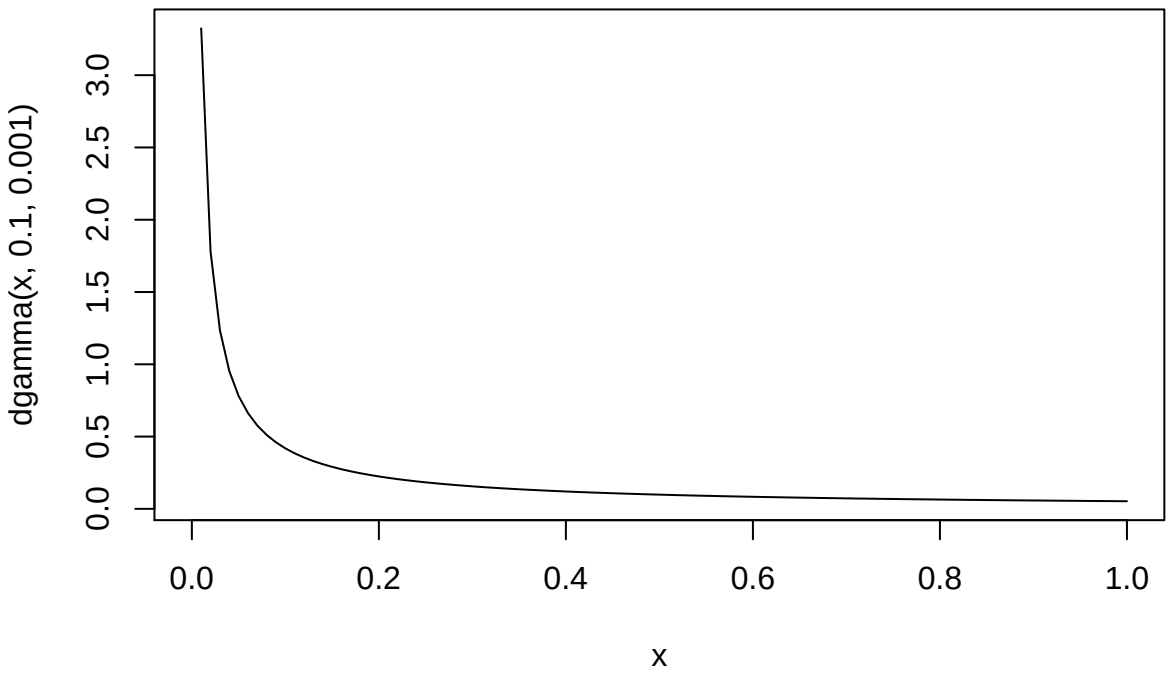
```
#Testing an even smaller "b" (rate = 1/scale) term
vec3<-gamm(10000,0.1,0.001)
```



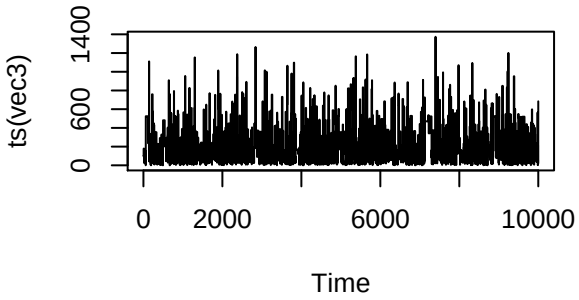

```
vec4<-gamm(10000,0.1,0.001)
```

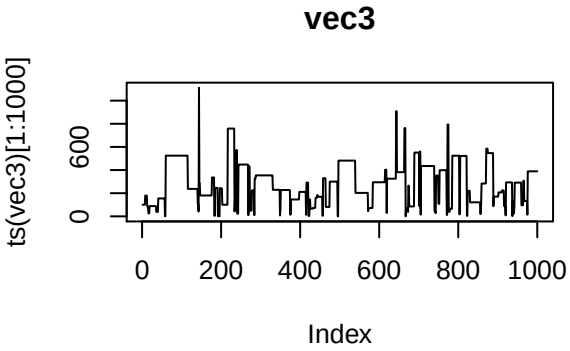


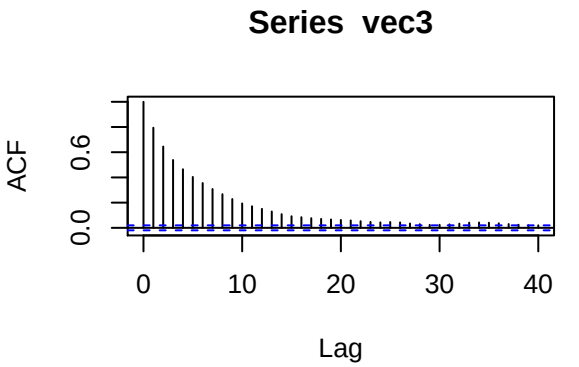

```
curve(dgamma(x,0.1,0.001))
```

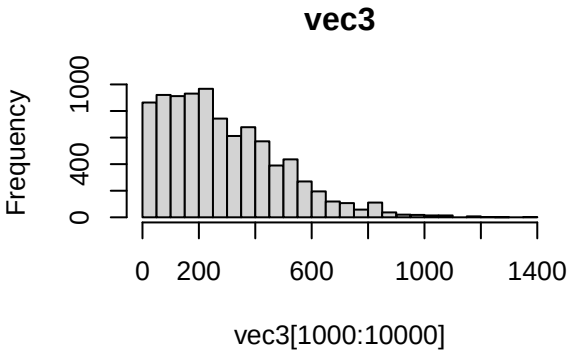


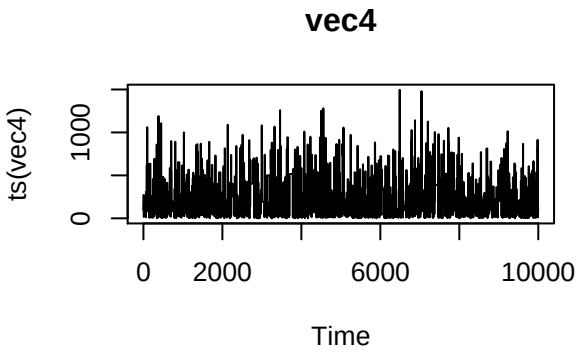

```
plotgamm(vec3,vec4)
```

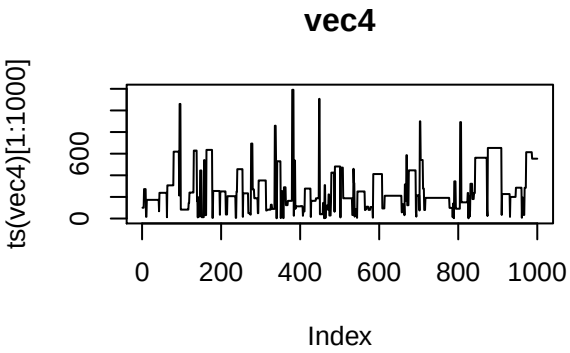

ts(vec3)
0
600
1400
0
2000
6000
10000
Time

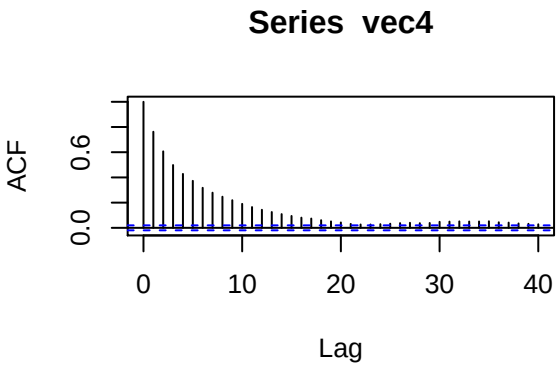

vec3
ts(vec3)[1:1000]
0
600
0
200
400
600
800
1000
Index

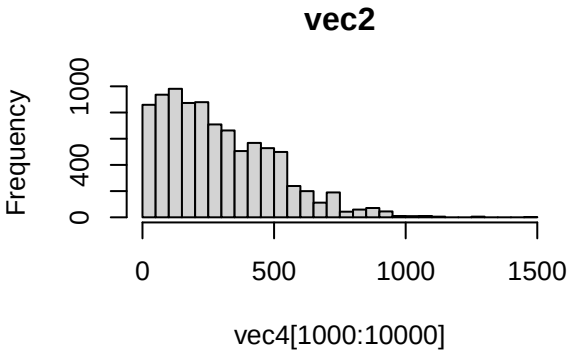

Series vec3
ACF
0.0
0.6
0
10
20
30
40
Lag

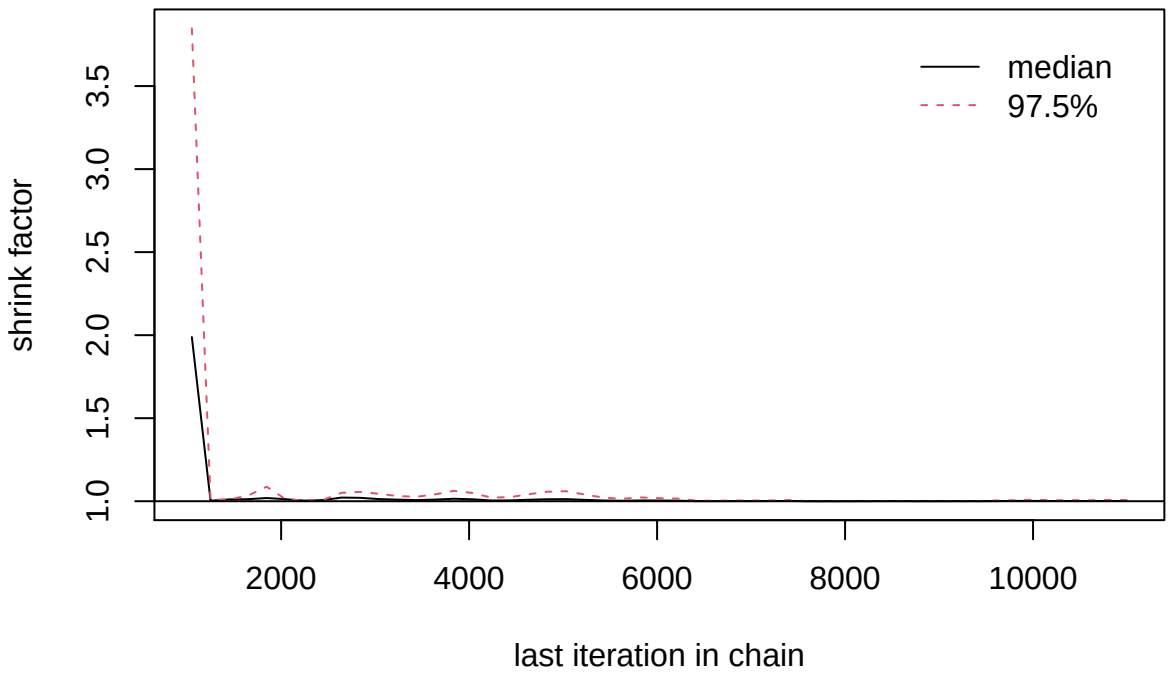

vec3
Frequency
0
400
1000
0
200
600
1000
1400
vec3[1000:10000]


vec4
ts(vec4)
0
1000
0
2000
6000
10000
Time


vec4
ts(vec4)[1:1000]
0
600
0
200
400
600
800
1000
Index


Series vec4
ACF
0.0
0.6
0
10
20
30
40
Lag

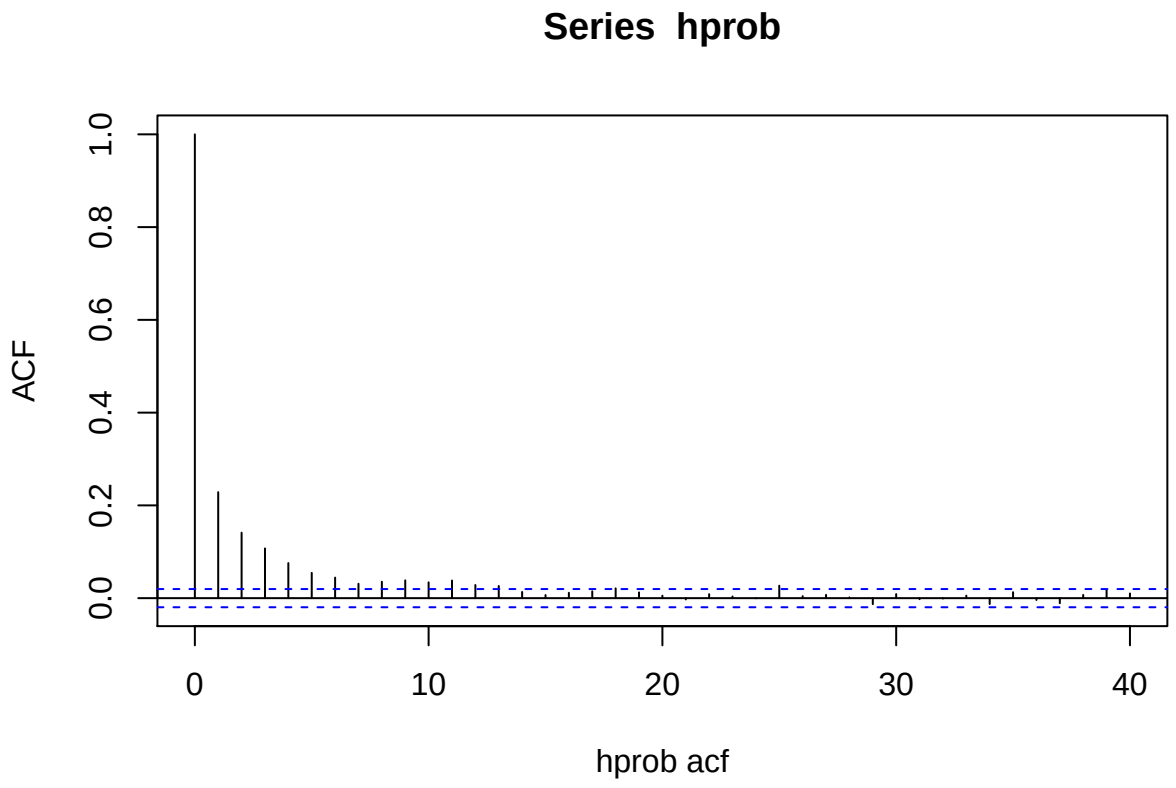

vec2
Frequency
0
400
1000
0
500
1000
1500
vec4[1000:10000]



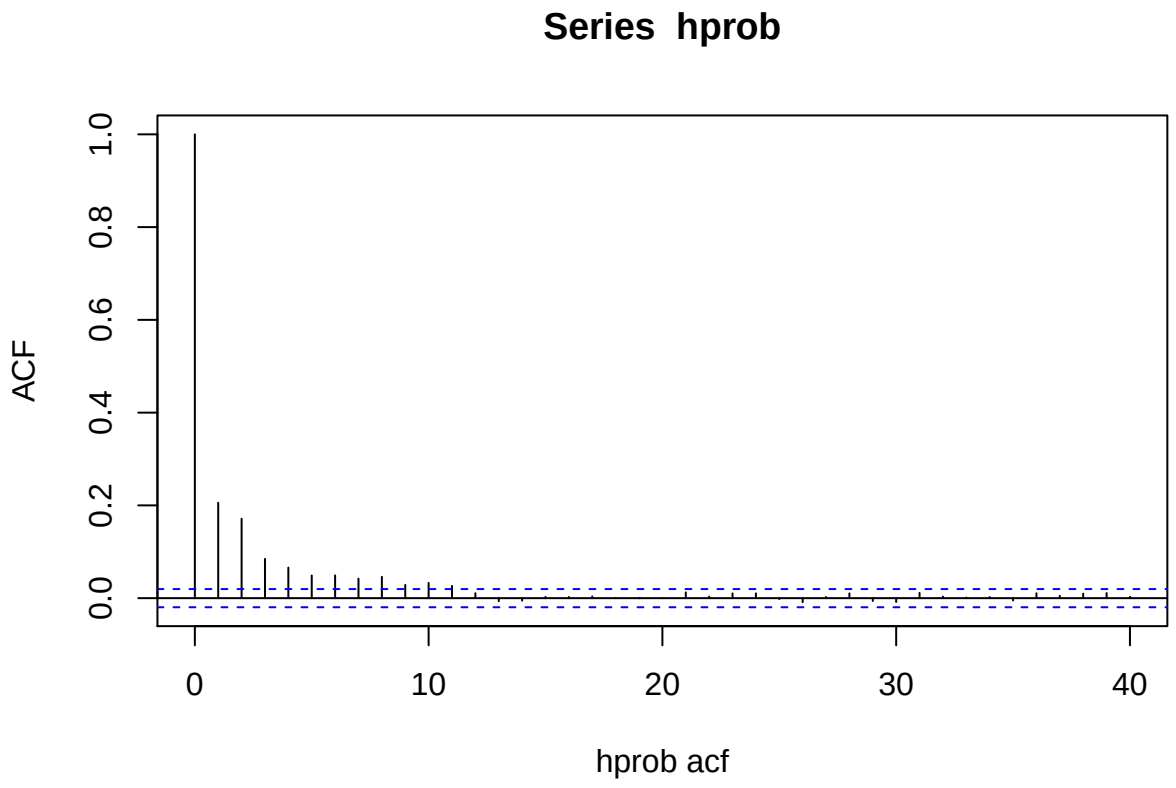

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1       1.01
```

Testing an even smaller "a" term for fixed gamm

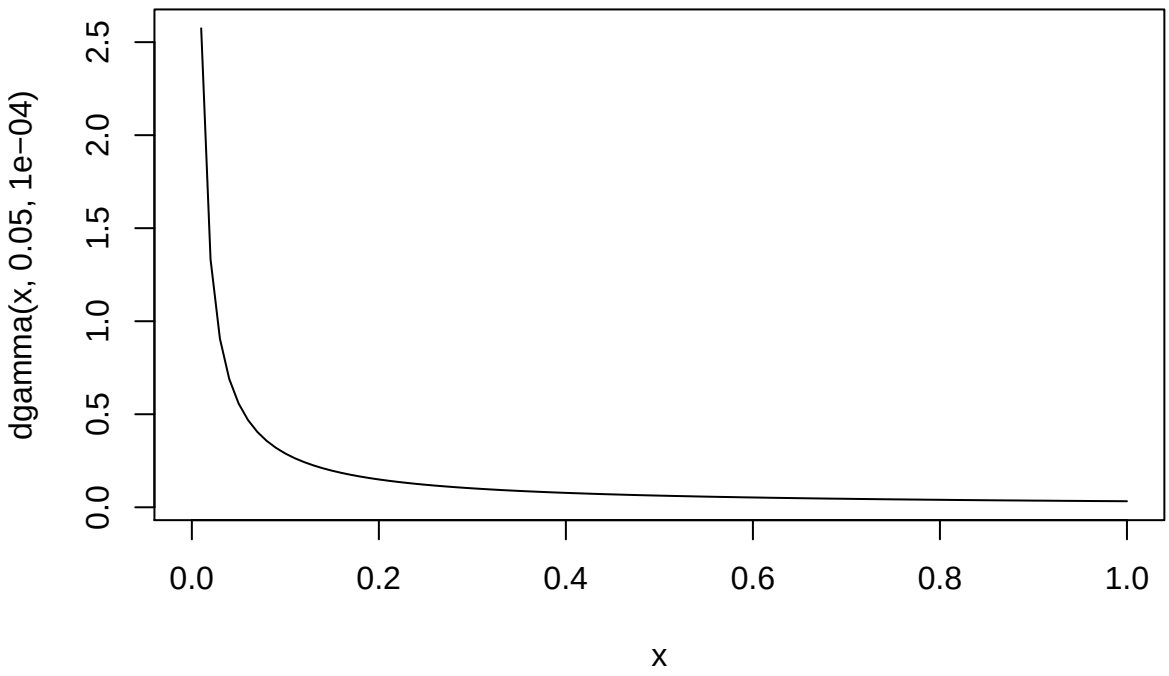
```
#Testing a smaller "a" shape term
vec3<-gamm(10000,0.05,0.0001)
```



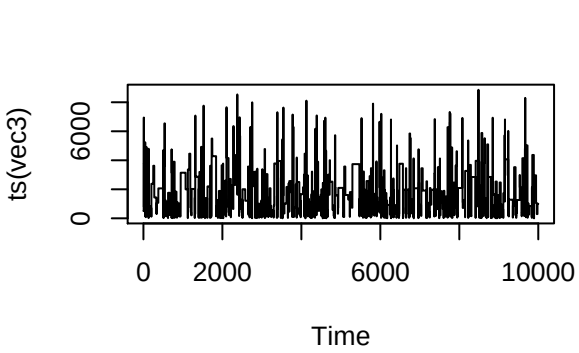

```
vec4<-gamm(10000,0.05,0.0001)
```

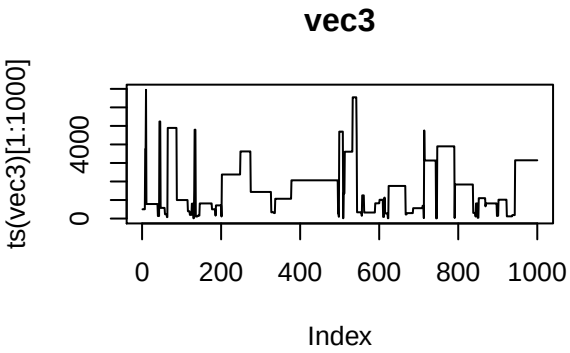


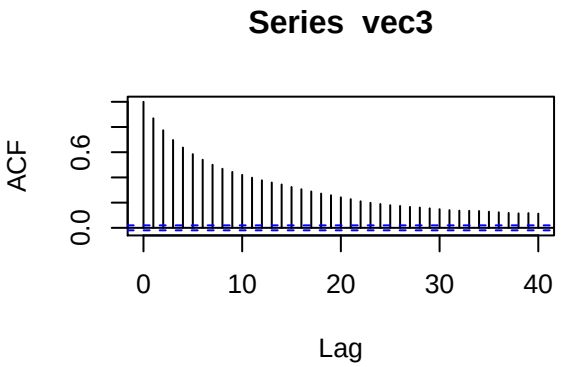

```
curve(dgamma(x,0.05,0.0001))
```

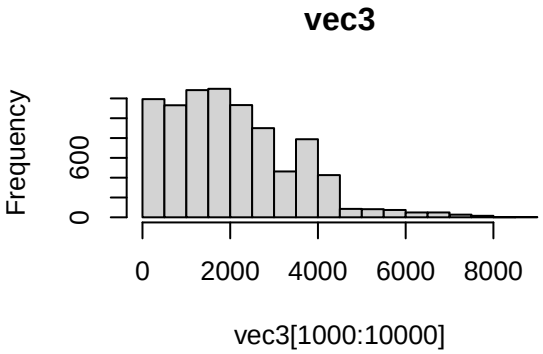


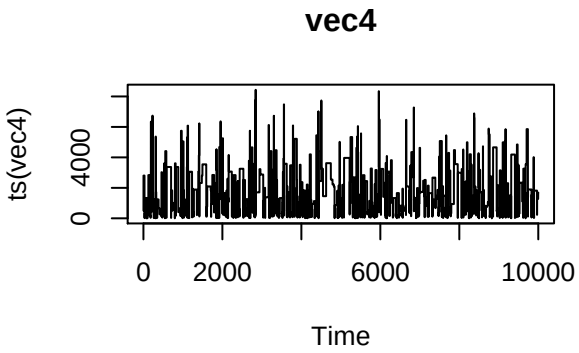

```
plotgamm(vec3,vec4)
```

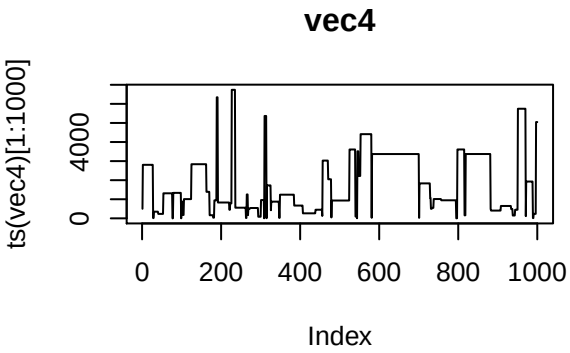

ts(vec3)
0
6000
0
2000
6000
10000
Time

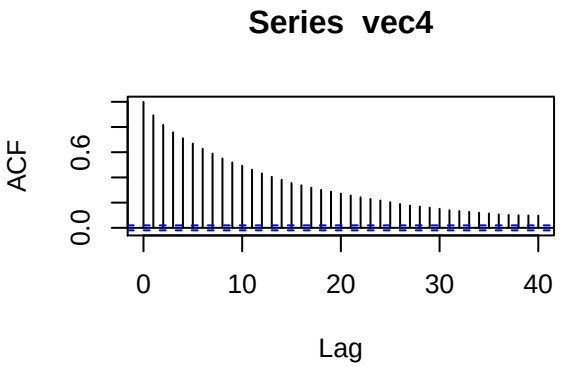

vec3
ts(vec3)[1:1000]
0
4000
0
200
400
600
800
1000
Index

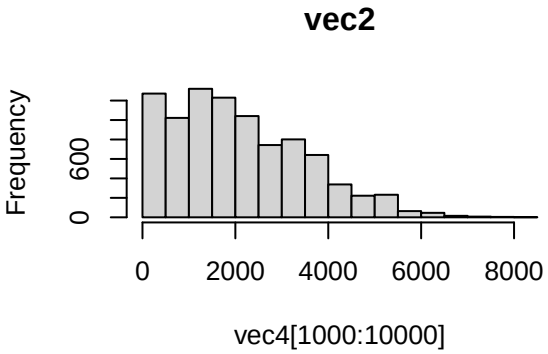

Series vec3
ACF
0.0
0.6
0
10
20
30
40
Lag

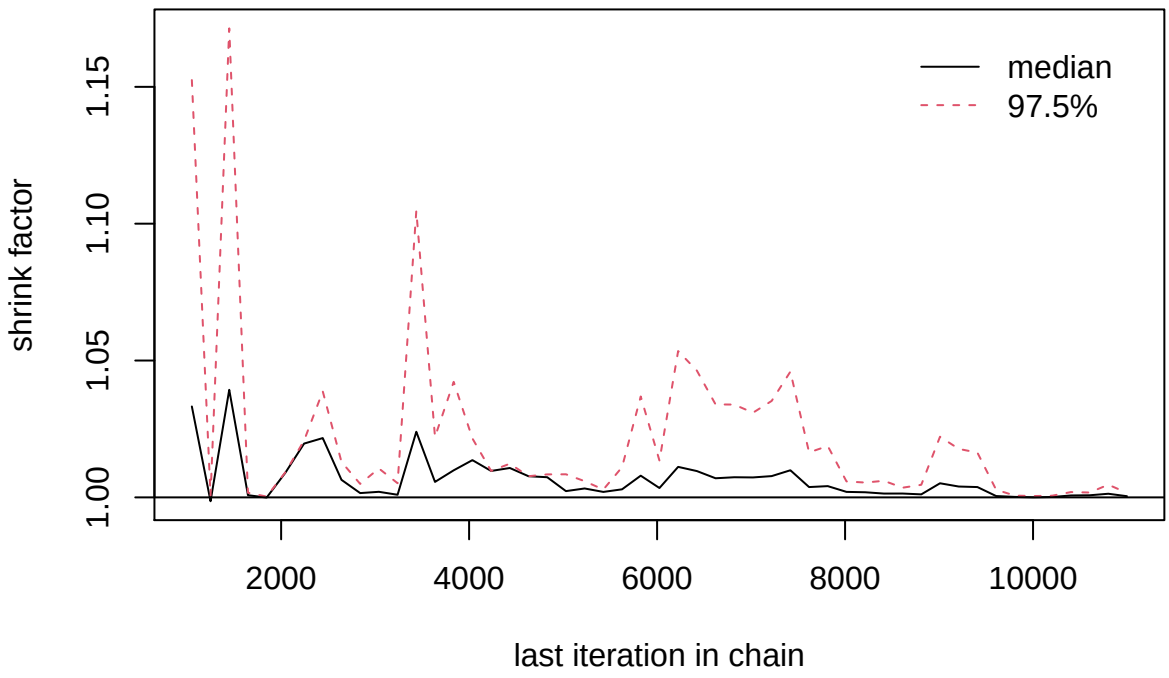

vec3
Frequency
0
600
0
2000
4000
6000
8000
vec3[1000:10000]


vec4
ts(vec4)
0
4000
0
2000
6000
10000
Time


vec4
ts(vec4)[1:1000]
0
4000
0
200
400
600
800
1000
Index


Series vec4
ACF
0.0
0.6
0
10
20
30
40
Lag


vec2
Frequency
0
600
0
2000
4000
6000
8000
vec4[1000:10000]




```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```