SPM_Project

2023-04-22

```
library(caracas)
```

```
##
## Attaching package: 'caracas'

## The following objects are masked from 'package:base':
##
##     %*%, det, diag, diag<-
```

PERIODIC ENVIRONMENTAL DRIVER

```
model <- function(parameters) {
  with(parameters, {
    (B*L/(m+v*x)-d)*x
  })
}
equilibrium <- function(diff.eq, state, parameters) {
  with(parameters, {
    solve_sys(diff.eq/state, state)
  })
}
```

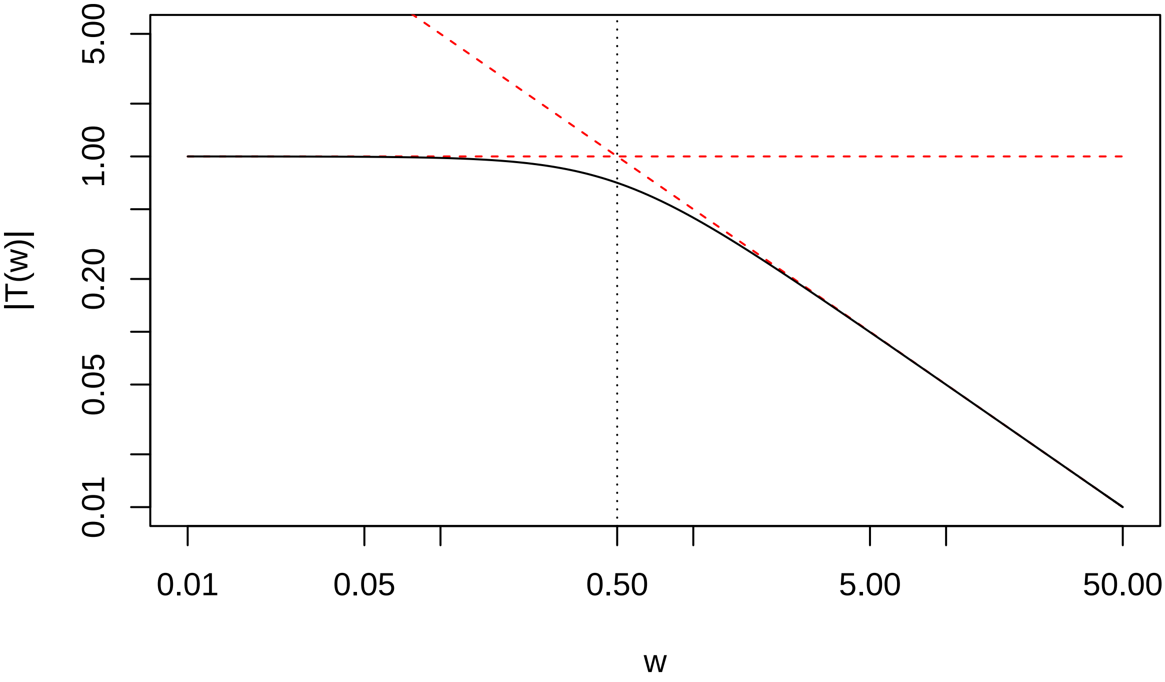
```
def_sym('Bs', 'Ls', 'ms', 'vs', 'ds', 'xs')
B = 1
L = 2
m = 1
v = 1
d = 1
```

```
x <- seq(0,1.2,0.01)
parameters.numeric <- list(B=B, L=L, m=m, v=v, d=d, x=x)
parameters.symbolic <- list(B=Bs, L=Ls, m=ms, v=vs, d=ds, x=xs)
```

```
parameters.x <- list(B=B, L=L, m=m, v=v, d=d, x=xs)
xeq <- as_expr((equilibrium(model(parameters.x), xs, parameters.x)[[1]]$xs))
```

```
f <- D(as_expr(model(parameters.x)), 'xs')
f <- as_expr(subs(eval(f), xs, xeq))

parameters.L <- list(B=B, L=Ls, m=m, v=v, d=d, x=xeq)
g <- D(as_expr(model(parameters.L)), 'Ls')
#g <- as_expr(subs(eval(g), Ls, L))
g <- eval(g)
```

```
w <- seq(1e-2, 50, 0.01)
Gain <- abs(g/(1i*w-f))

plot(w, abs(g/f)*rep(1, length(w)), ylim = c(0.01, 5), log='xy', type='l', lty=2, col="red",
     ylab = "|T(w)|")
lines(w, abs(g/w), lty=2, col="red")
lines(w, Gain, type='l')
abline(v=abs(g), lty=3)
```



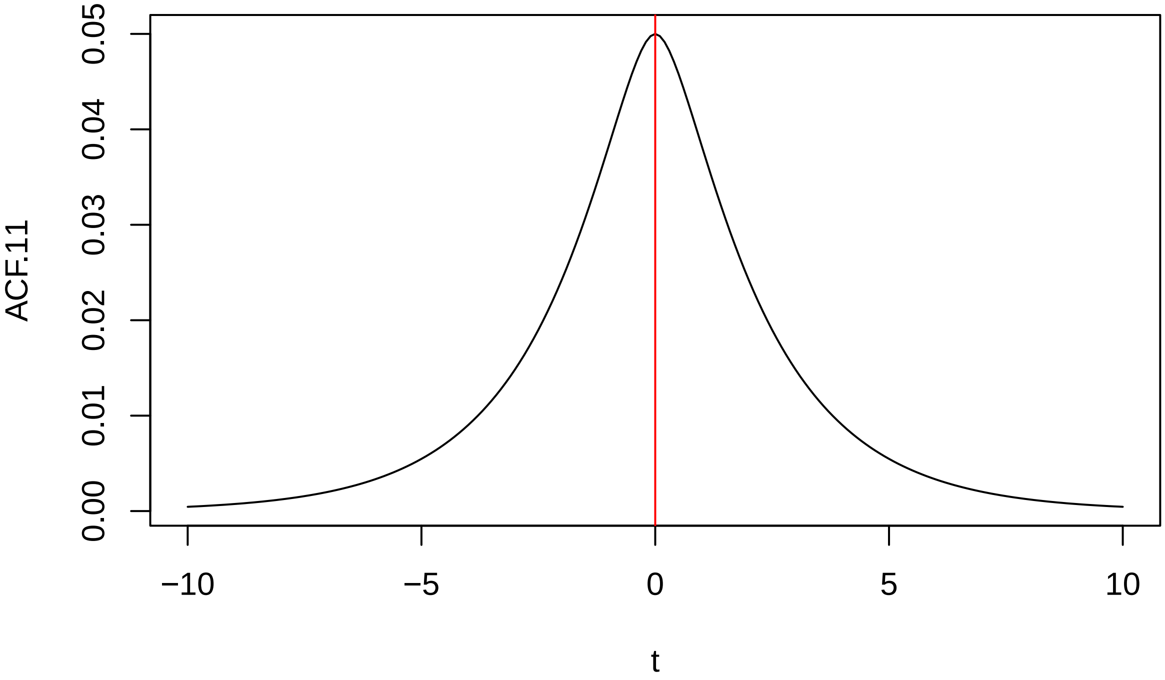
STOCHASTIC ENVIRONMENTAL DRIVER

```
# Ornstein-Uhlenbeck Driver
a <- 2
b <- 0.5

A.OU <- t(matrix(c(f, g*L, 0, -a), 2, 2))
B.OU <- t(matrix(c(0, 0, 0, b), 2, 2))
```

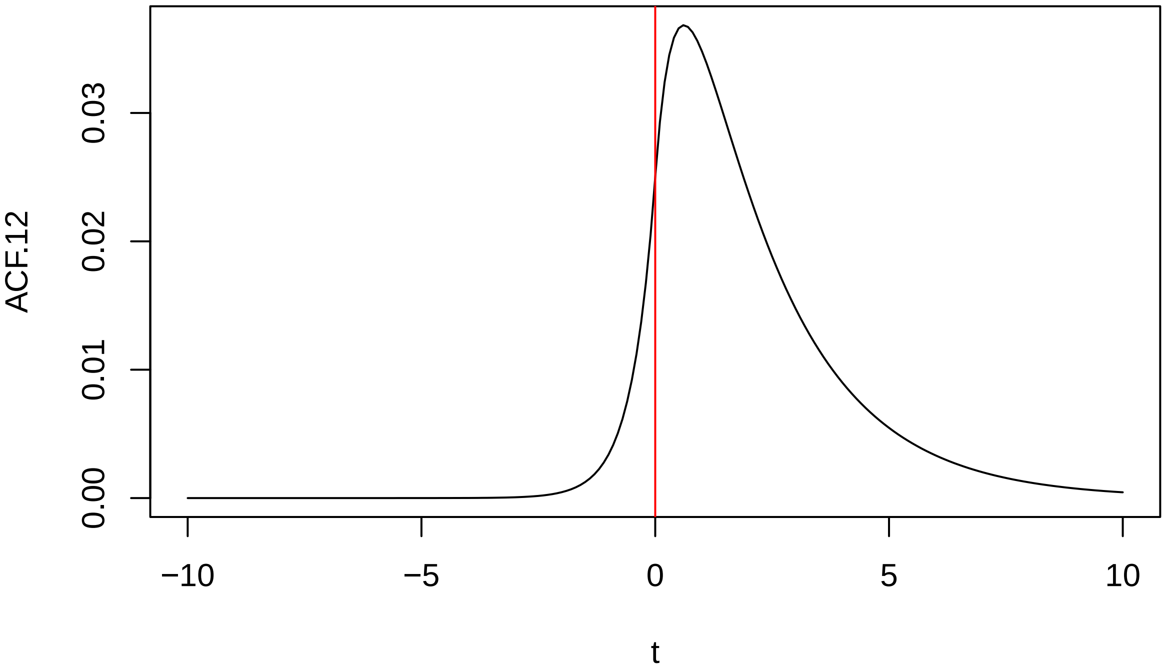
```
# Covariance Matrix

def_sym("s11", "s12", "s21", "s22")
K <- t(matrix(c("s11", "s12", "s21", "s22"), 2, 2))
K <- as_sym(K)

cov.eq <- as_sym(A.OU)%*%K + K%*%as_sym(t(A.OU)) + as_sym(B.OU)%*%as_sym(t(B.OU))

cov.elems <- solve_sys(cbind(cov.eq[1,1], cov.eq[1,2], cov.eq[2,1], cov.eq[2,2]),
                       list(s11, s12, s21, s22))[[1]]
s11 <- as_expr(cov.elems$s11)
s12 <- as_expr(cov.elems$s12)
s21 <- as_expr(cov.elems$s21)
s22 <- as_expr(cov.elems$s22)
K <- t(matrix(c(s11, s12, s21, s22), 2, 2))
```

```
# Auto-correlation Functions
eigens <- eigen(A.OU)
U <- eigens$vectors
```

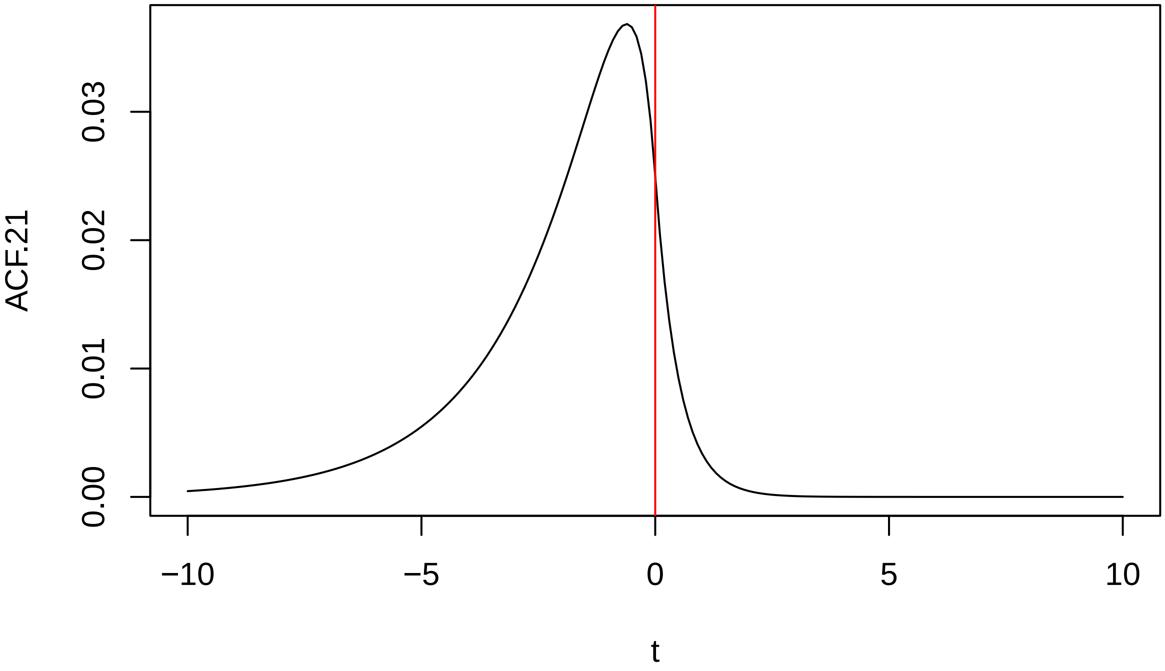
```
Lambda <- eigens$values

t.pos <- seq(0, 10, 0.1)
t.neg <- -rev(t.pos)
t <- c(t.neg, t.pos)

ACF.11.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%K)[1,1]})
ACF.11.neg <- sapply(t.neg, function(tau) {(K%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[1,1]})
ACF.11 <- c(ACF.11.neg, ACF.11.pos)

ACF.12.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%K)[1,2]})
ACF.12.neg <- sapply(t.neg, function(tau) {(K%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[1,2]})
ACF.12 <- c(ACF.12.neg, ACF.12.pos)

ACF.21.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%K)[2,1]})
ACF.21.neg <- sapply(t.neg, function(tau) {(K%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[2,1]})
ACF.21 <- c(ACF.21.neg, ACF.21.pos)

ACF.22.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%K)[2,2]})
ACF.22.neg <- sapply(t.neg, function(tau) {(K%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[2,2]})
ACF.22 <- c(ACF.22.neg, ACF.22.pos)

plot(t, ACF.11, type='l')
abline(v=0, col="red")
```



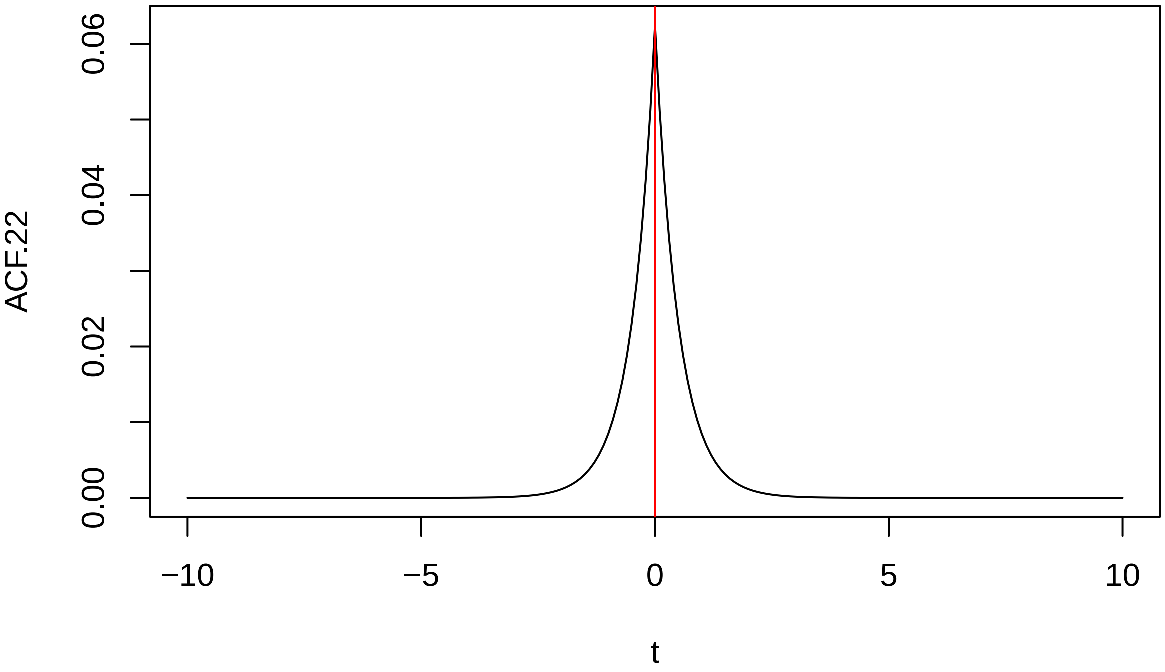
```
plot(t, ACF.12, type='l')
abline(v=0, col="red")
```



```
plot(t, ACF.21, type='l')
abline(v=0, col="red")
```

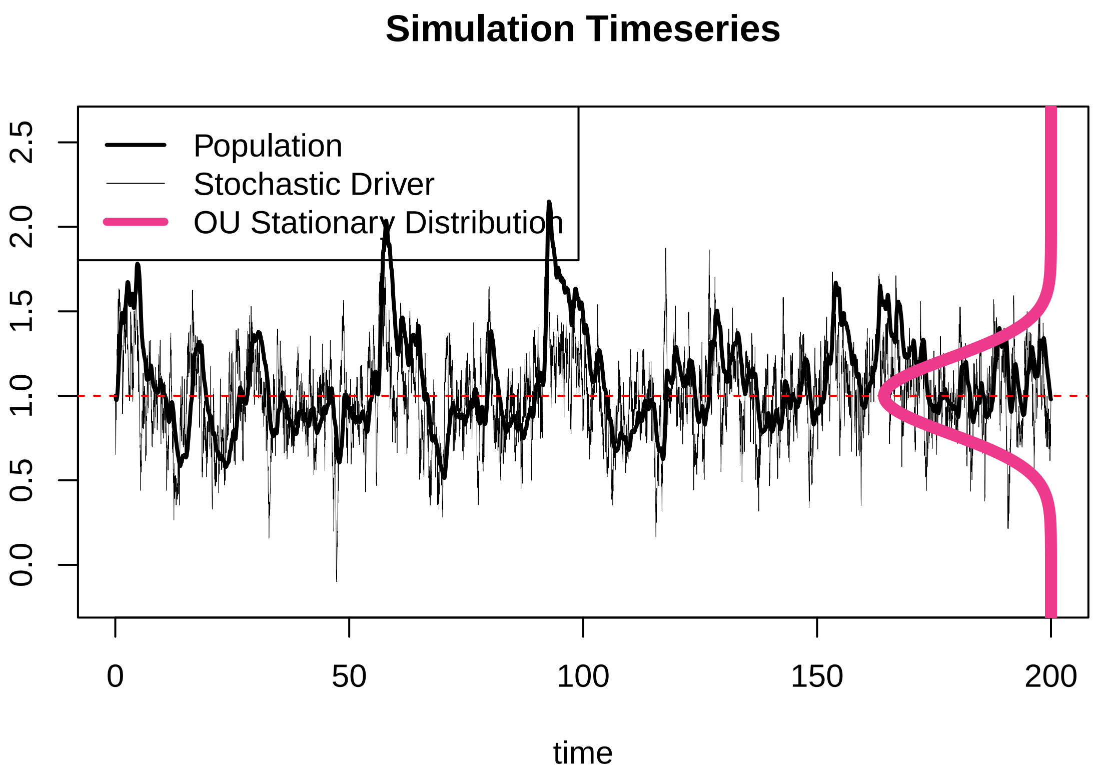


```
plot(t, ACF.22, type='l')
abline(v=0, col="red")
```

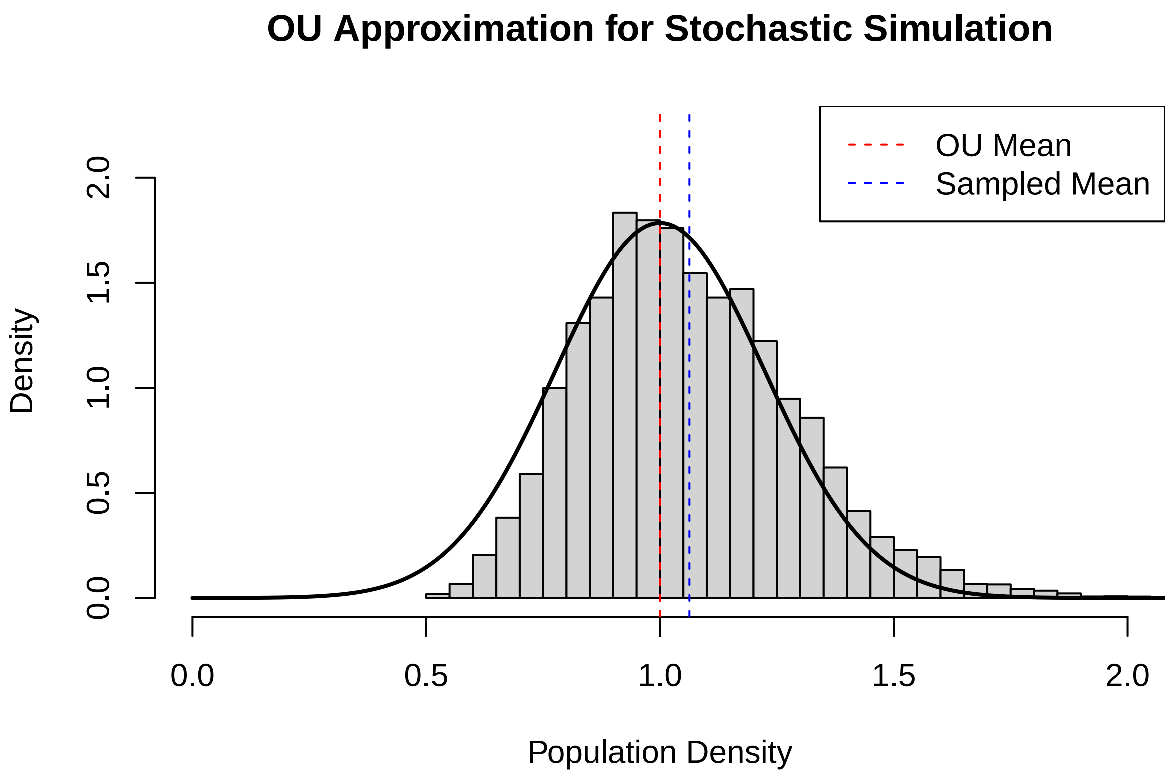


```r
# Stochastic Orbit
timesteps <- 100000
dt <- 0.02

parameters <- list(B=B, L=L, m=m, v=v, d=d, x=xeq)
results <- array(NA, dim=c(timesteps, 2))
results[1,] <- c(xeq, 0)

for(t in 2:timesteps) {
  x = results[t-1,1]
  theta = results[t-1,2]
  parameters <- list(B=B, L=L*exp(theta), m=m, v=v, d=d, x=x)
  dx = model(parameters)*dt
  dtheta = -a*theta*dt + b*sqrt(dt)*rnorm(1)
  results[t,] <- c(x+dx, theta+dtheta)
}
```

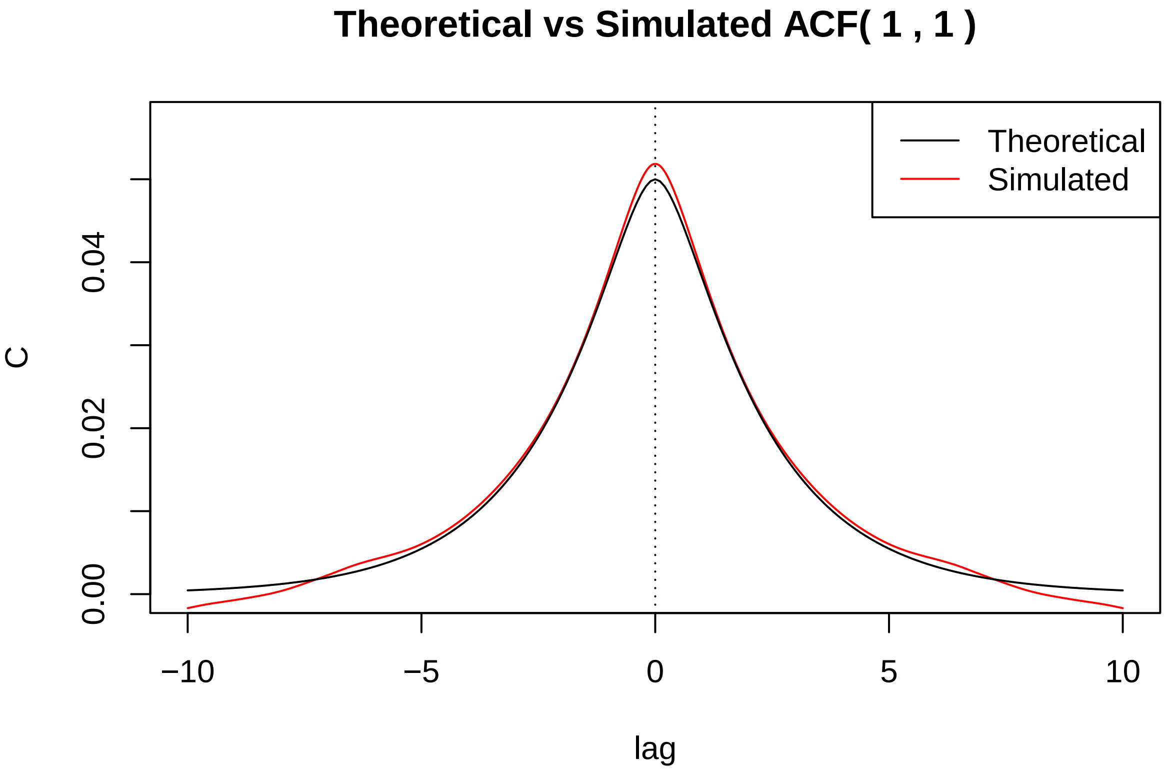
```r
plot.length <- 10000
plot((1:plot.length)*dt, results[1:plot.length,2]+xeq, type='l', lwd=0.2, ylim = c(-0.2,2.6*xeq),
     main="Simulation Timeseries", xlab="time", ylab="")
lines((1:plot.length)*dt, results[1:plot.length,1], type='l', lwd=2)
abline(h=xeq, col='red', lty=2)

x <- seq(-0.5, 2.7*xeq, 0.01)
y <- plot.length*dt-20*dnorm(x, mean=xeq, sd=sqrt(K[1,1]))
lines(y, x, type='l', lwd=6, col="violetred2")

legend("topleft", legend=c("Population", "Stochastic Driver", "OU Stationary Distribution"),
       lty=1, lwd = c(2, 0.5, 4), col=c("black", "black", "violetred2"))
```

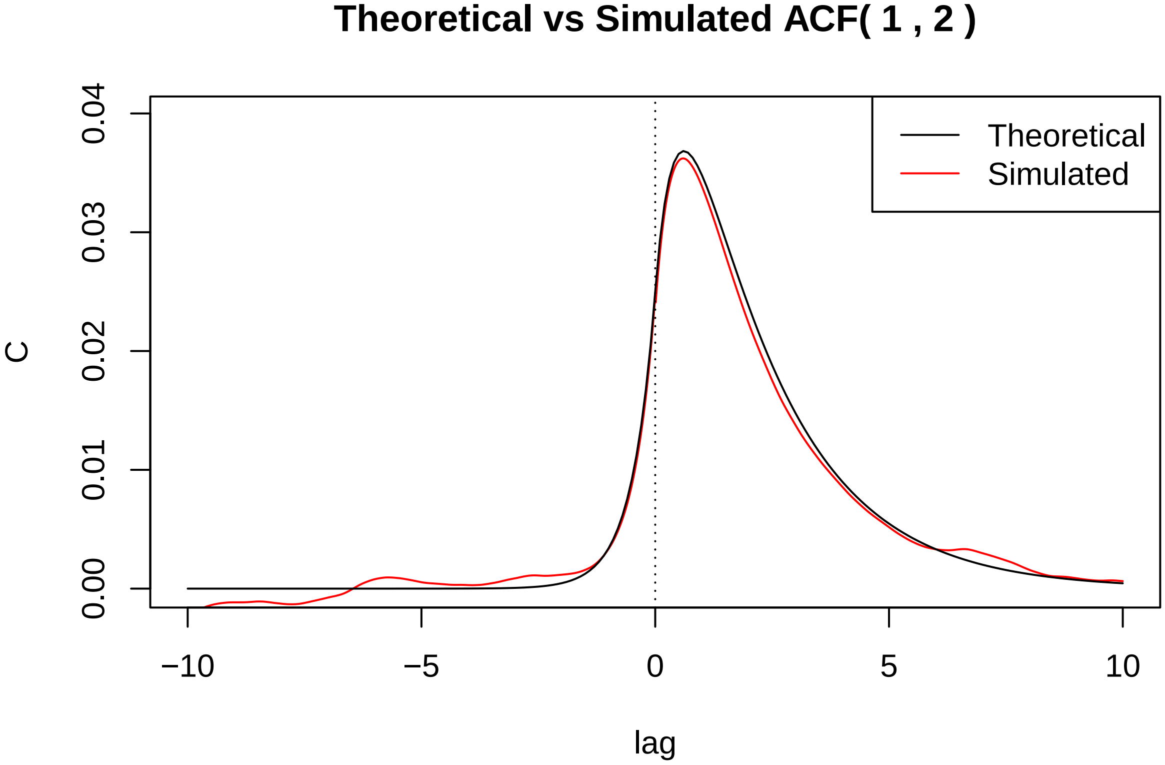


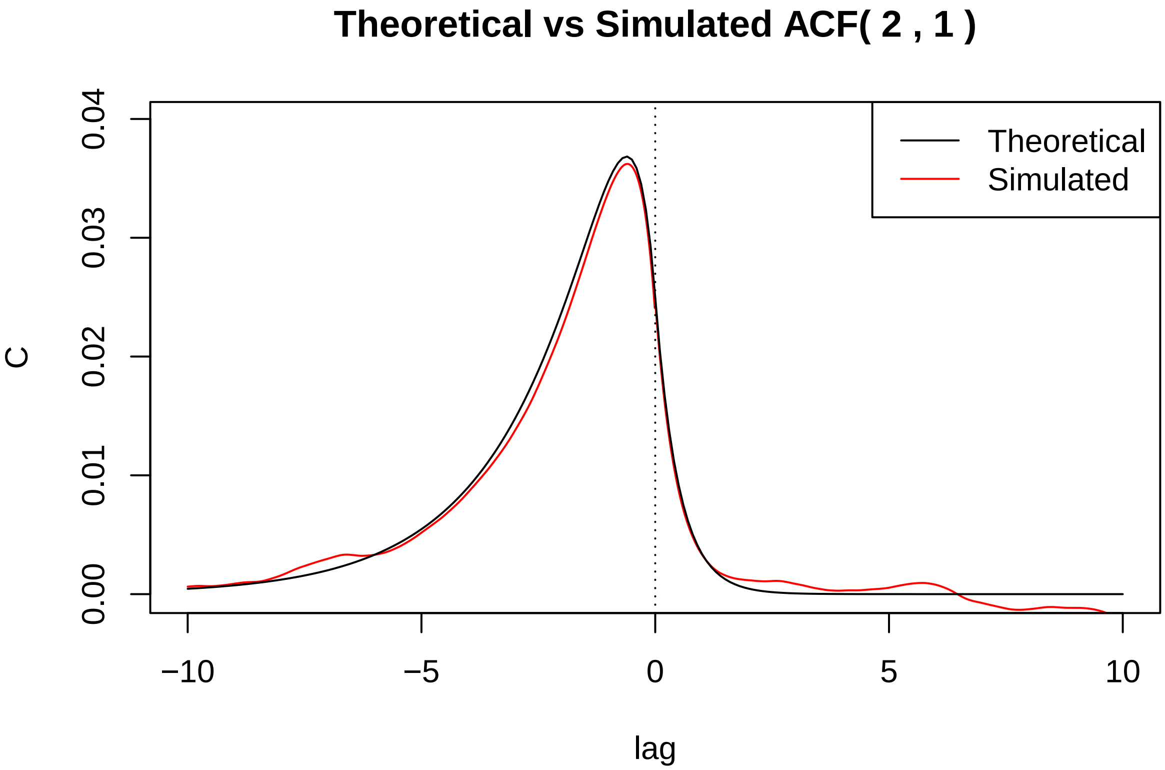
```
plot.range <- seq(0, 2.5*xeq, 0.01)
hist(results[,1], probability = TRUE, xlim = c(0,2*xeq), ylim=c(0, 2.25), breaks = 30,
     main="OU Approximation for Stochastic Simulation",
     xlab="Population Density")
lines(plot.range, dnorm(plot.range, mean=xeq, sd=sqrt(K[1,1])), lwd=2)
abline(v=xeq, lty=2, col="red")
abline(v=mean(results[,1]), lty=2, col="blue")
legend("topright", legend = c("OU Mean", "Sampled Mean"), lty=2, col=c("red", "blue"))
```

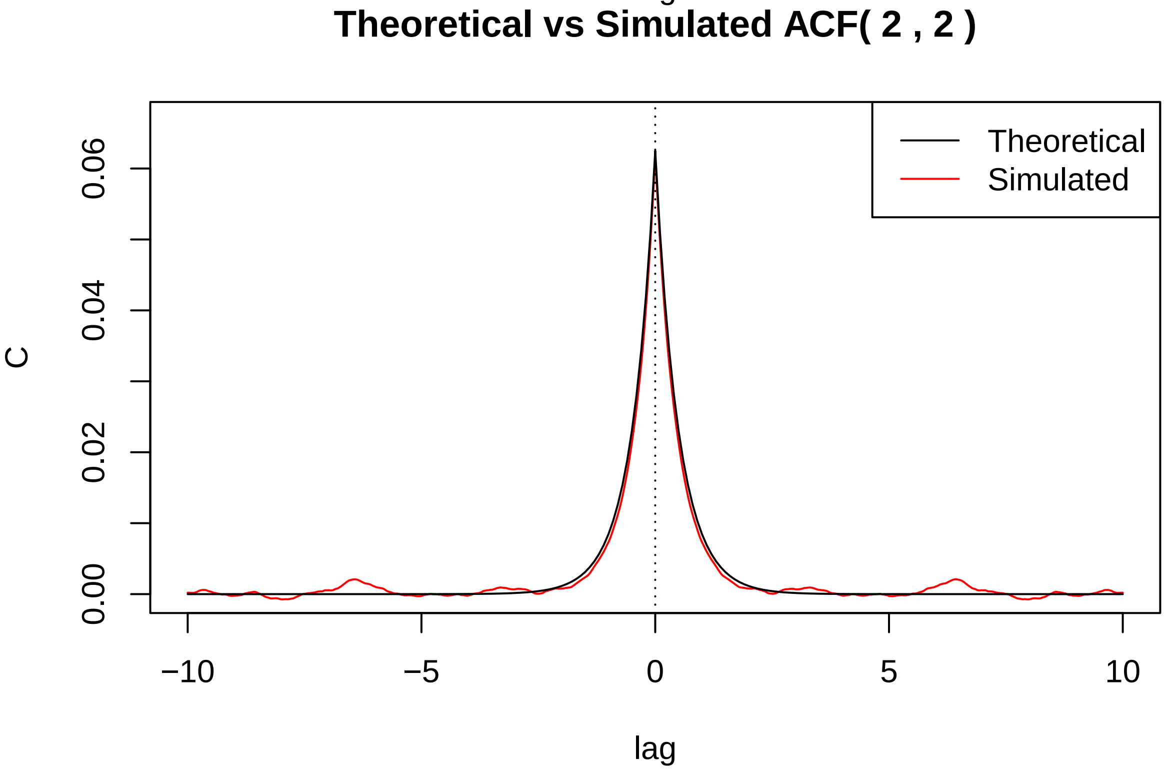


```
acf.empirical.pos <- acf(results, type="covariance",
                         lag.max=max(t.pos)/dt, plot = FALSE)$acf
acf.empirical.neg <- acf(results[nrow(results):1,], type="covariance",
                         lag.max=abs(min(t.neg))/dt, plot = FALSE)$acf

acf.11.empirical <- c(rev(acf.empirical.neg[,1,1]), acf.empirical.pos[,1,1])
acf.12.empirical <- c(rev(acf.empirical.neg[,1,2]), acf.empirical.pos[,1,2])
acf.21.empirical <- c(rev(acf.empirical.neg[,2,1]), acf.empirical.pos[,2,1])
acf.22.empirical <- c(rev(acf.empirical.neg[,2,2]), acf.empirical.pos[,2,2])

acf.data.empirical <- cbind(acf.11.empirical, acf.12.empirical, acf.21.empirical, acf.22.empirical)
acf.data.theoretical <- cbind(ACF.11, ACF.12, ACF.21, ACF.22)

plot.range <- seq(min(t.neg), max(t.pos), length.out=length(acf.11.empirical))

ind.i <- c(1,1,2,2)
ind.j <- c(1,2,1,2)
for(i in 1:ncol(acf.data.empirical)) {
  plot(plot.range, acf.data.empirical[,i], type='l',
       ylim=c(0, 1.1*max(acf.data.empirical[,i])), col="red",
       main = paste("Theoretical vs Simulated ACF(",ind.i[i],",",ind.j[i],")" ), xlab="lag", ylab="C")
  lines(c(t.neg, t.pos), acf.data.theoretical[,i], type='l')
  abline(v=0, lty=3)
  legend("topright", legend=c("Theoretical", "Simulated"), lty=1, col=c("black", "red"))
}
```


Theoretical vs Simulated ACF(1 , 1)
Theoretical
Simulated
C
0.00
0.02
0.04
−10
−5
0
5
10
lag


Theoretical vs Simulated ACF(1 , 2)
Theoretical
Simulated
C
0.00
0.01
0.02
0.03
0.04
−10
−5
0
5
10
lag





DEMOGRAPHIC FLUCTUATIONS IN SMALL SYSTEMS

Let m denote the number of juveniles and n denote the number of adults. The quasi-equilibrium resulting from the fast dynamics of juveniles is then:

$$\hat{m} = \frac{\lambda n}{\mu + \frac{\nu n}{\Omega}}$$

where Ω is the system size. The resulting birth and death rates of adults is:

$$B_n = \beta \hat{m} = \frac{\beta \lambda n}{\mu + \frac{\nu n}{\Omega}}$$

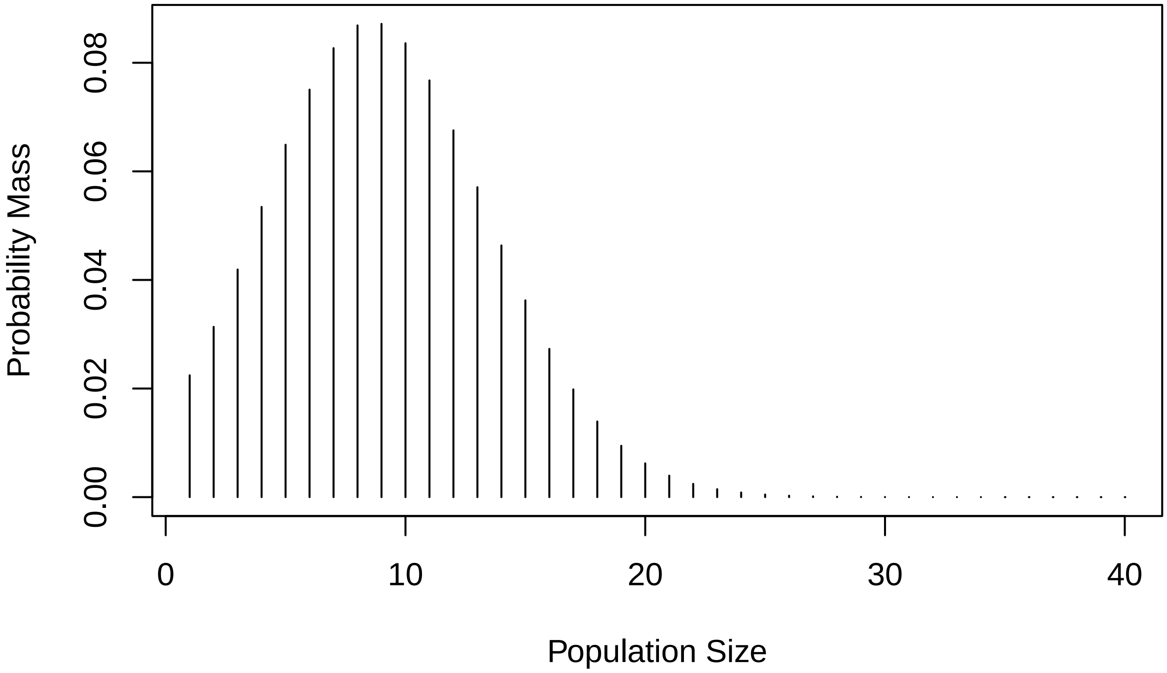
$$D_n = \delta n$$

```
# Birth rate
Birth <- function(n, parameters) {
  with(parameters, {
    B*L*n/(m + v*n/omega)
  })
}

# Death rate
Death <- function(n, parameters) {
  with(parameters, {
    d*n
  })
}
```

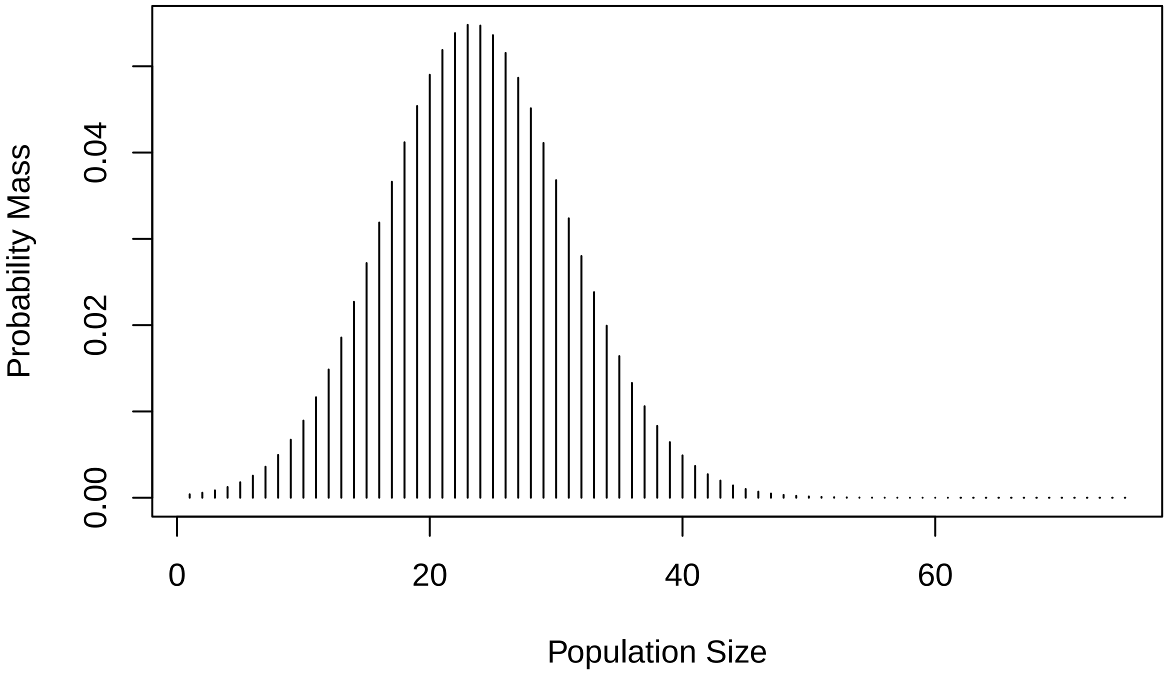
```
stationary.dist <- function(parameters, max.pop) {
  with(parameters, {
    birth.rates <- Birth(1:max.pop, parameters)
    death.rates <- Death(1:max.pop, parameters)

    A <- diag(-(birth.rates+death.rates))
    A.sub <- diag(birth.rates[1:(max.pop-1)])
    A.sup <- diag(death.rates[(2:max.pop)])
    A[2:max.pop, 1:(max.pop-1)] <- A[2:max.pop, 1:(max.pop-1)] + A.sub
    A[1:(max.pop-1), 2:max.pop] <- A[1:(max.pop-1), 2:max.pop] + A.sup

    B.D.eigens <- eigen(A)
    stationary.dist <- abs(B.D.eigens$vectors[,max.pop])
    stationary.dist <- stationary.dist / sum(stationary.dist)

    return(stationary.dist)
  })
}
```

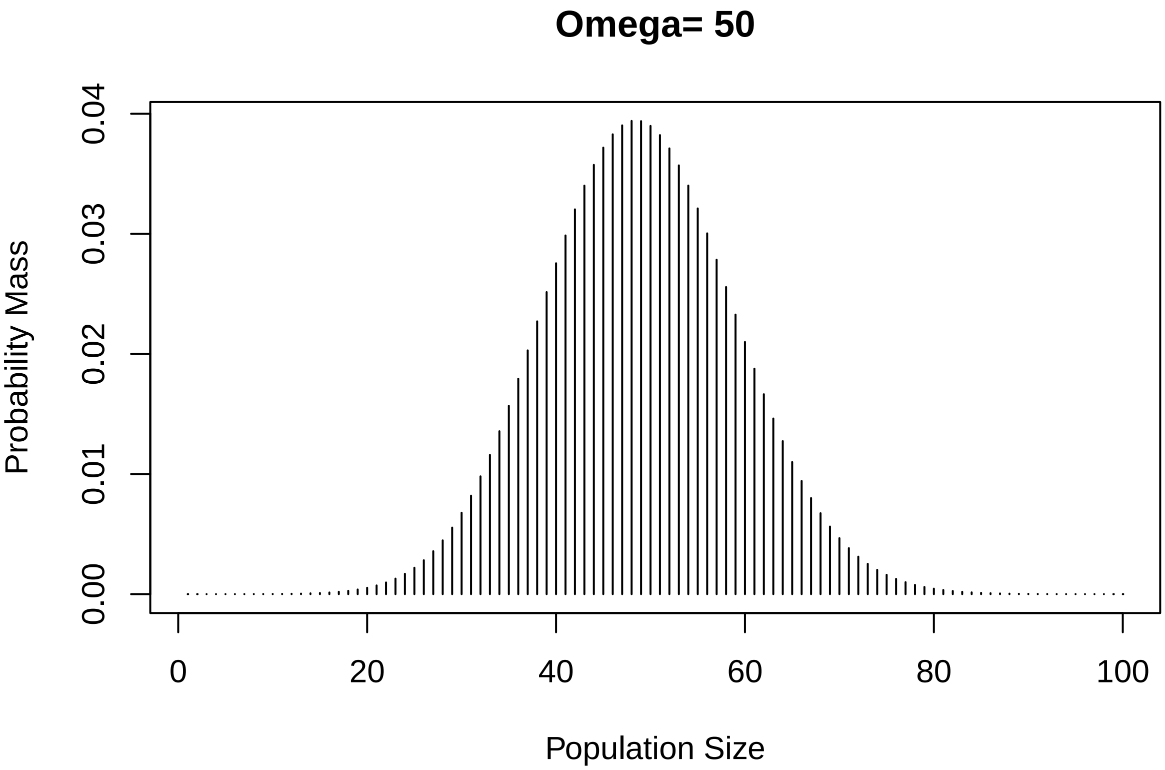
```
max.pops <- c(40, 75, 100, 200)
omegas <- c(10, 25, 50, 100)

for(i in 1:4) {
  B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omegas[i])
  plot(stationary.dist(B.D.parameters, max.pops[i]), type='h',
       main=paste("Omega=", omegas[i]), xlab="Population Size", ylab="Probability Mass")
}
```

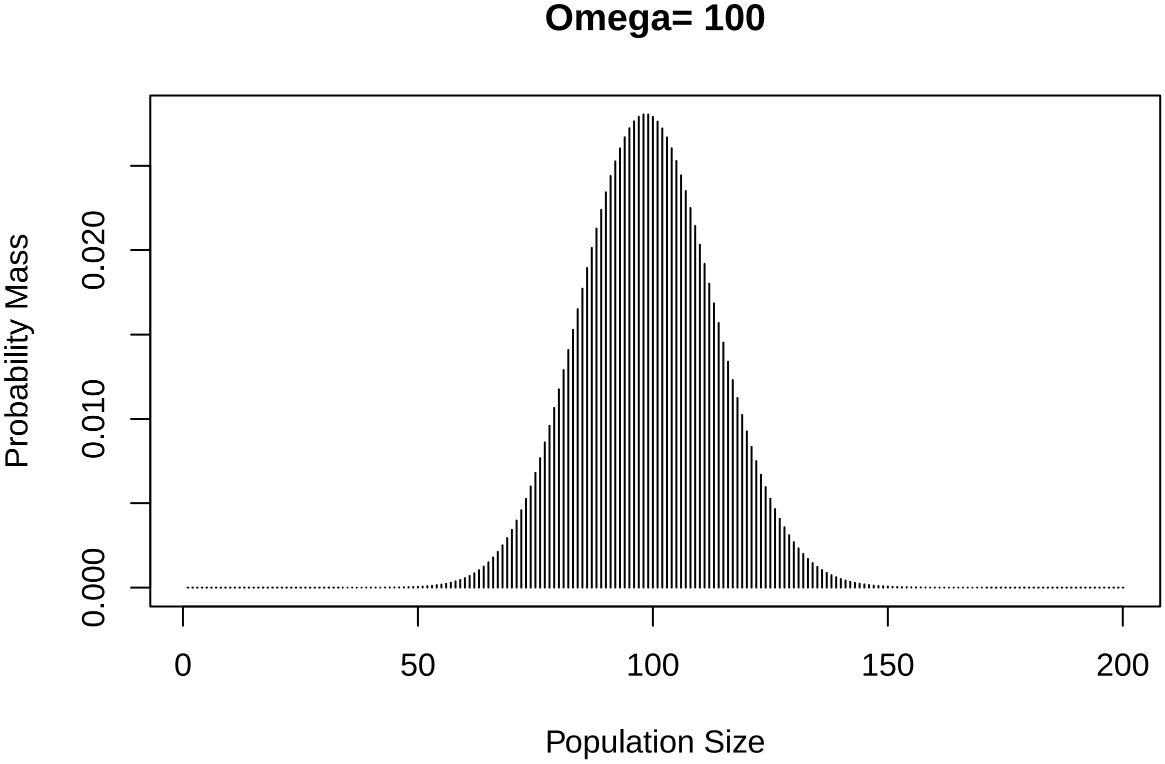
Omega= 10



Omega= 25

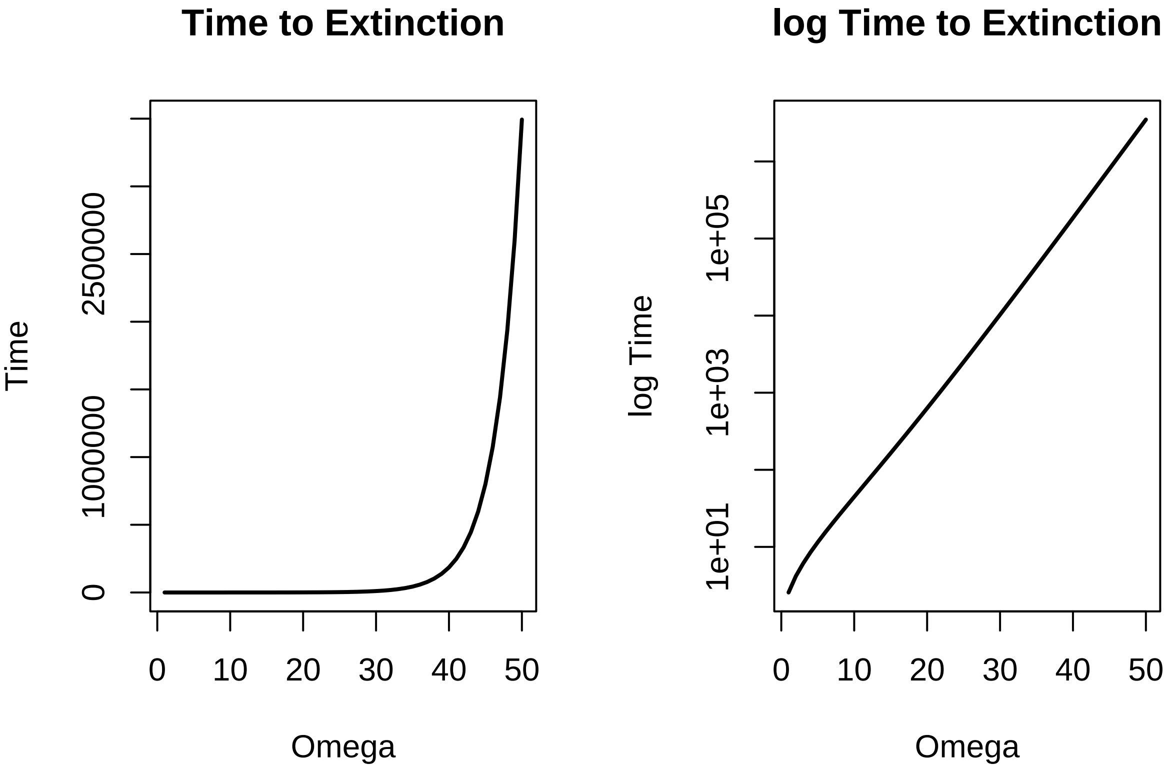






```r
# Time to Extinction
omegas <- 1:50
max.pop <- 100
ext.times <- c()

for(i in omegas) {
  B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=i)
  s.dist <- stationary.dist(B.D.parameters, max.pop)
  ext.times <- c(ext.times, 1/(s.dist[1]*Death(1, B.D.parameters)))
}
```

```
par(mfrow=c(1,2))
plot(omegas, ext.times, type='l', lwd=2,
     main="Time to Extinction", xlab="Omega", ylab="Time")
plot(omegas, ext.times, type='l', log='y', lwd=2, cex=0.5,
     main="log Time to Extinction", xlab="Omega", ylab="log Time")
```



FOKKER-PLANCK APPROXIMATION FOR SEMI-LARGE SYSTEMS

First we convert the population counts to densities such that $x = \frac{n}{\Omega}$. Moreover, we equate the per capita birth and death rates of the two models by $\frac{B_n}{n} = \frac{b(x)}{x}$ and $\frac{D_n}{n} = \frac{d(x)}{x}$. The resulting birth and death rates of adults are:

$$b(x) = \frac{\beta\lambda x}{\mu + \nu x}$$
$$d(x) = \delta x$$

which are the same birth and death rates we had in the original formulation. The Fokker-Planck equation is thus:

$$\partial_t p(x,t) = -\partial_x (b(x) - d(x))p(x,t) + \frac{\varepsilon}{2}\partial_x^2 (b(x) + d(x))p(x,t)$$

where $\varepsilon = \frac{1}{\Omega}$.

The stationary distribution of the Fokker-Planck equation is the solution to:

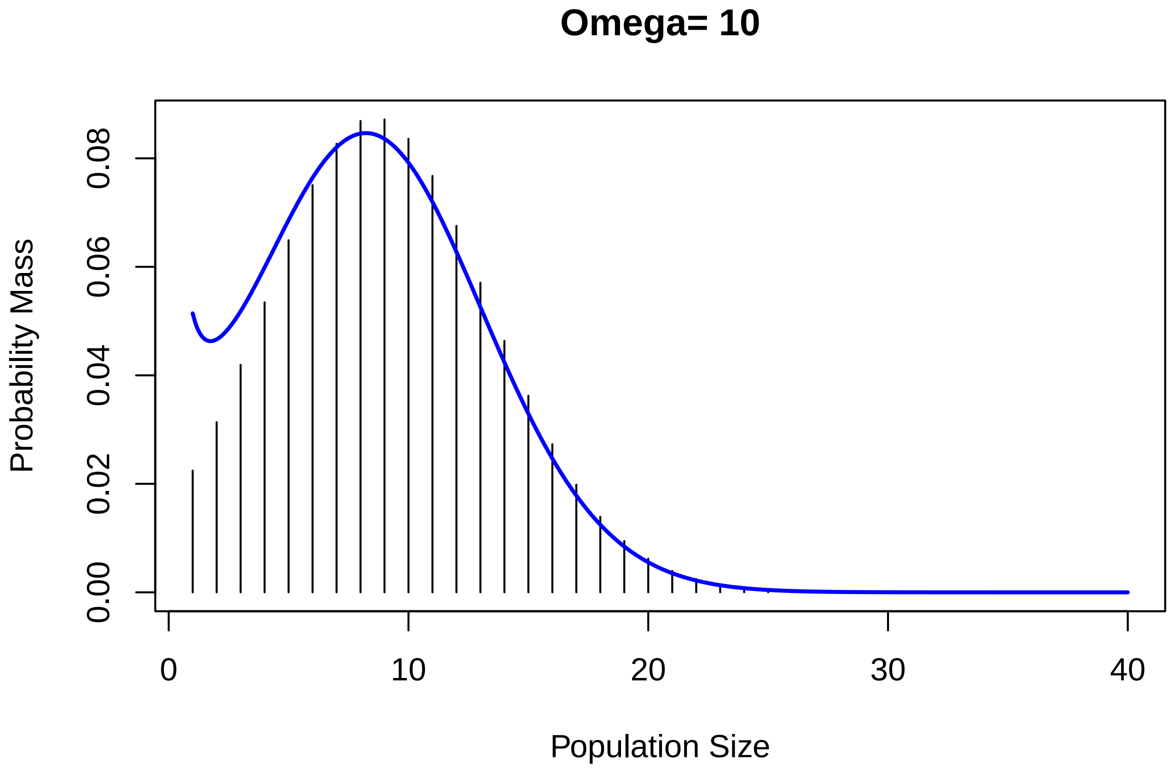
$$\hat{p}(x) = \frac{C}{b(x)+d(x)} \exp\left[\frac{2}{\varepsilon}\int_{\varepsilon}^{x} \frac{b(\xi)-d(\xi)}{b(\xi)+d(\xi)} d\xi\right]$$

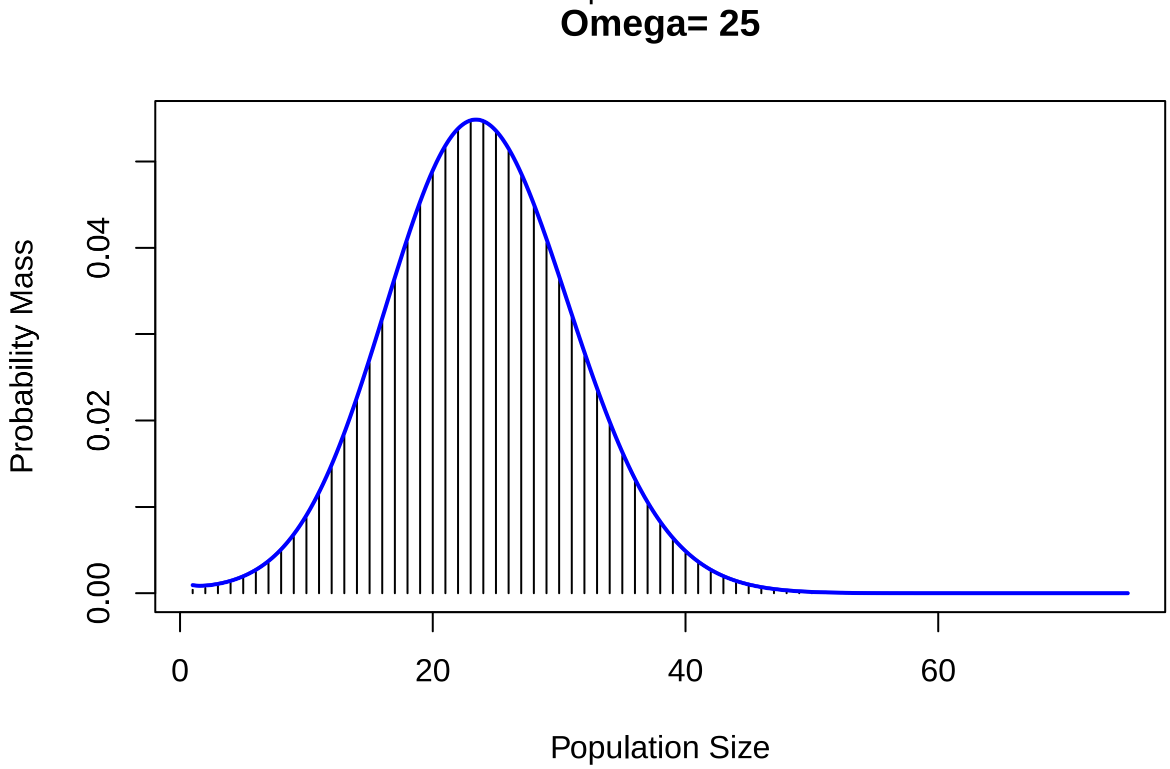
where C is a normalizing constant.

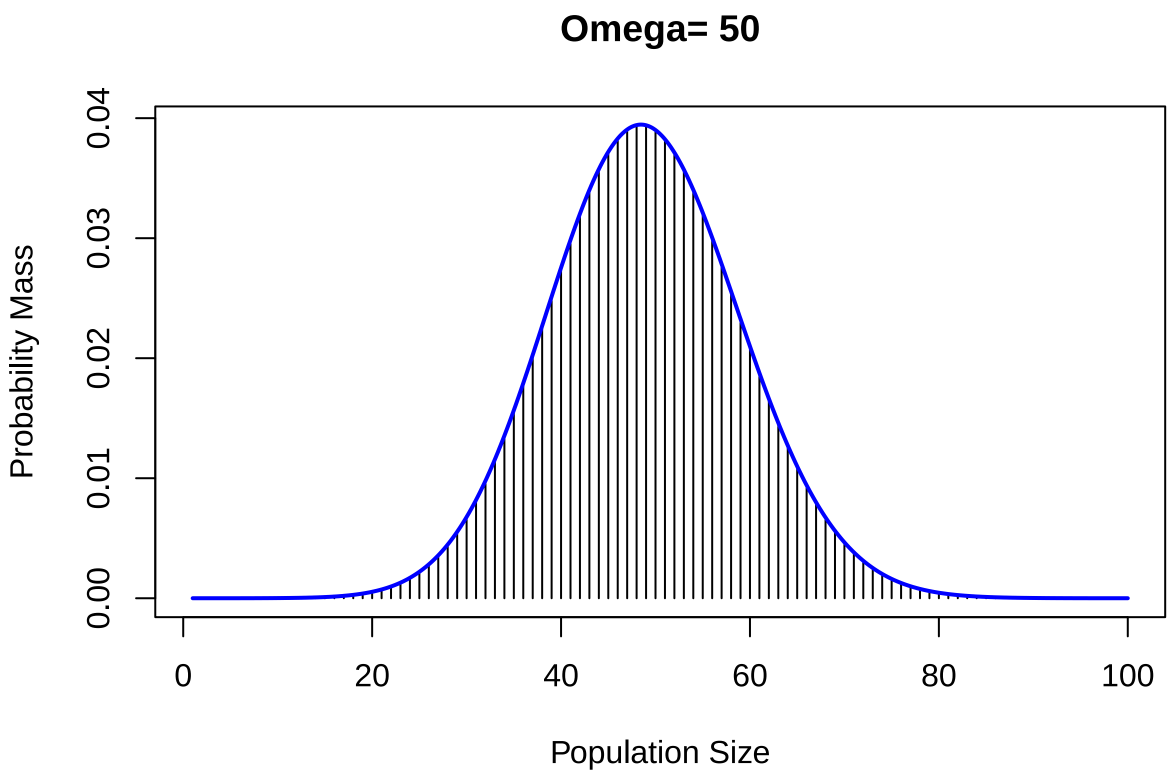
```r
# Birth rate
birth <- function(x, parameters) {
  with(parameters, {
    B*L*x/(m+v*x)
  })
}
# Birth rate
death <- function(x, parameters) {
  with(parameters, {
    d*x
  })
}
```

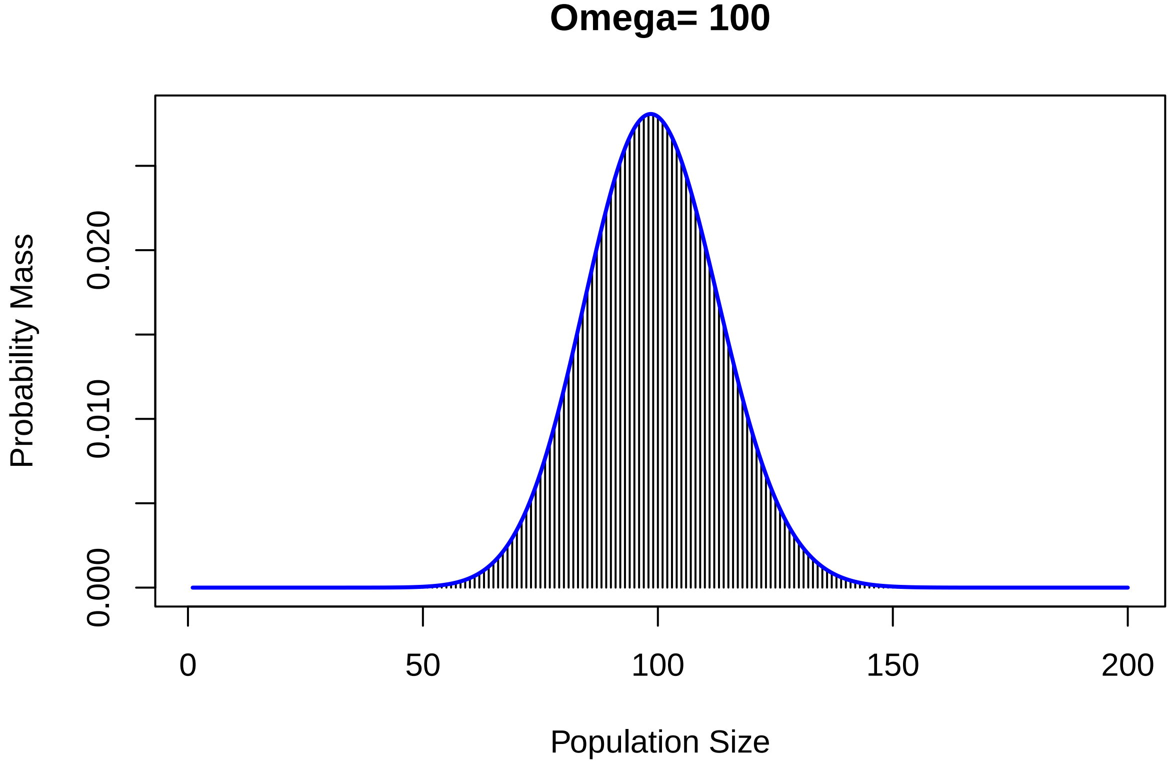
```r
omega <- 50
B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omega)

b.rate <- birth(xs, B.D.parameters)
d.rate <- death(xs, B.D.parameters)
integrand.sym <- as_expr((b.rate-d.rate)/(b.rate+d.rate))

integrand.fun <- function(xs) {
    eval(integrand.sym)
}
F.P.stationary <- function(parameters, x) {
  with(parameters, {
    b.rate <- birth(x, parameters)
    d.rate <- death(x, parameters)

    integral <- integrate(Vectorize(integrand.fun), lower=1/omega, upper=x)$value
    p.x <- 1/(b.rate+d.rate)*exp(2*omega*integral)

    return(p.x)
  })
}
```

```r
max.pops <- c(40, 75, 100, 200)
omegas <- c(10, 25, 50, 100)

for(i in 1:4) {
  B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omegas[i])

  inc <- 0.1
  FP <- 1/omegas[i]*sapply(seq(1,max.pops[i], inc), function(x){
    F.P.stationary(B.D.parameters, 1/omegas[i]*x)
    })
  FP <- 1/inc*FP/sum(FP)

  plot(stationary.dist(B.D.parameters, max.pops[i]), type='h',
       main=paste("Omega=", omegas[i]),
        xlab="Population Size", ylab="Probability Mass")
  lines(seq(1,max.pops[i], inc), FP, lwd=2, col="blue")
}
```







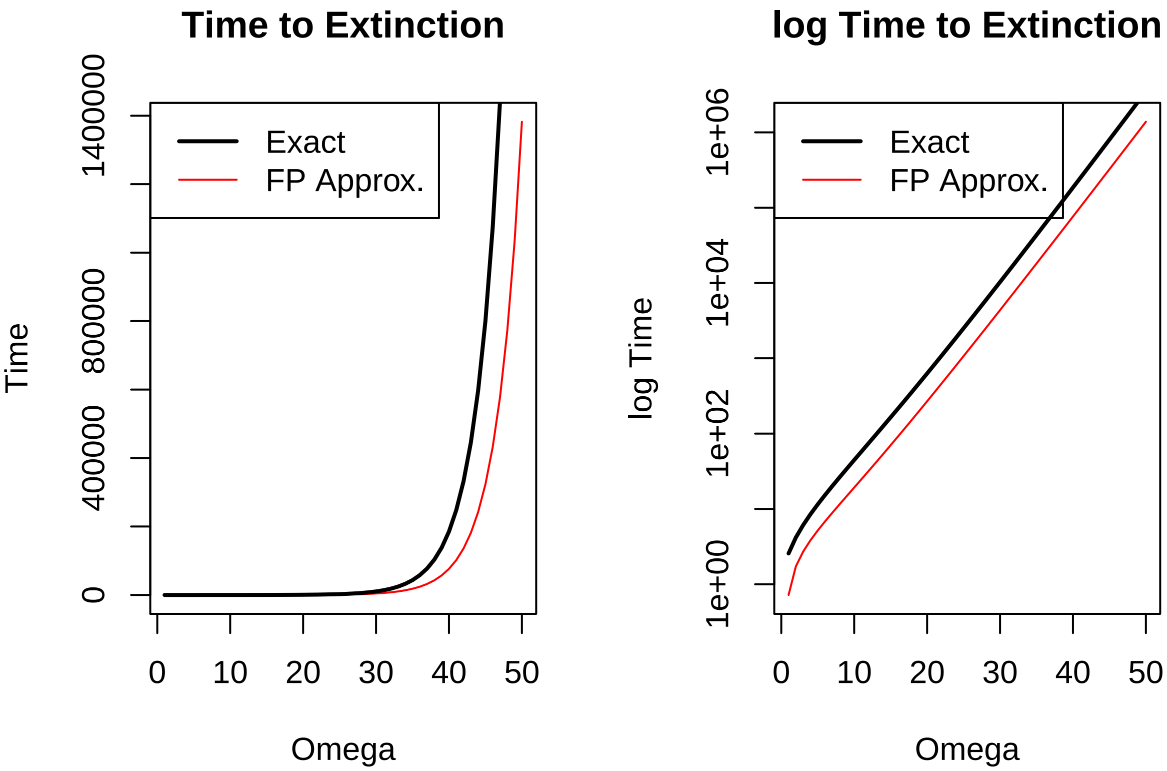


EXTINCTION TIME WITH FOKKER-PLANCK APPROXIMATION

```
# Time to Extinction
omegas <- 1:50
ext.times.FP <- c()
```

```r
for(om in omegas) {
  B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=om)

  inc <- 0.01
  range <- seq(1/om, 2, inc)
  FP <- sapply(range, function(x){F.P.stationary(B.D.parameters, x)})
  FP <- FP/sum(FP*inc)
  p.eps <- FP[1]

  ext.times.FP <- c(ext.times.FP, 1/(p.eps*death(1/om, B.D.parameters)))
}
```

```r
par(mfrow=c(1,2))
plot(omegas, ext.times.FP, type='l', col="red",
     main="Time to Extinction", xlab="Omega", ylab="Time")
lines(omegas, ext.times, type='l', lwd=2)
legend("topleft", legend=c("Exact", "FP Approx."),
       lwd=c(2,1), col=c("black", "red"))

plot(omegas, ext.times.FP, type='l', log='y', col="red",
     main="log Time to Extinction", xlab="Omega", ylab="log Time")
lines(omegas, ext.times, type='l', lwd=2)
legend("topleft", legend=c("Exact", "FP Approx."),
       lwd=c(2,1), col=c("black", "red"))
```



ORNSTEIN-UHLENBECK APPROXIMATION TO SEMI-LARGE SYSTEMS

The SDE corresponding to the above diffusion equation is:

$$dx = (b(x) - d(x))xdt + \sqrt{\varepsilon(b(x) + d(x))}dW$$

We proceed with the OU approximation by linearizing this around $\hat{x}$:

$$d(x-\bar{x}) = (b'(\bar{x}) - d'(\bar{x}))(x-\bar{x})dt + \sqrt{\varepsilon(b(\bar{x}) + d(\bar{x}))}dW$$

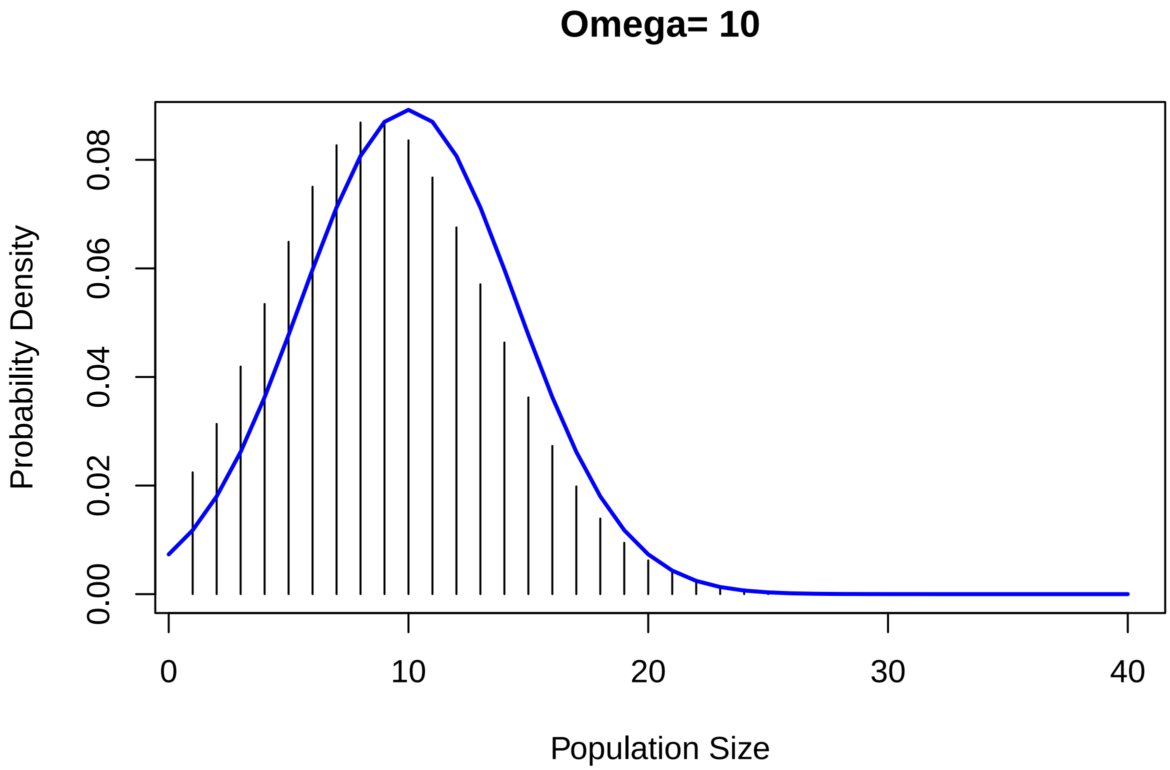
```
b.deriv <- D(as_expr(birth(xs, B.D.parameters)), 'xs')
b.deriv <- as_expr(subs(eval(b.deriv), 'xs', xeq))
d.deriv <- D(as_expr(death(xs, B.D.parameters)), 'xs')

b.eq <- birth(xeq, B.D.parameters)
d.eq <- death(xeq, B.D.parameters)
```

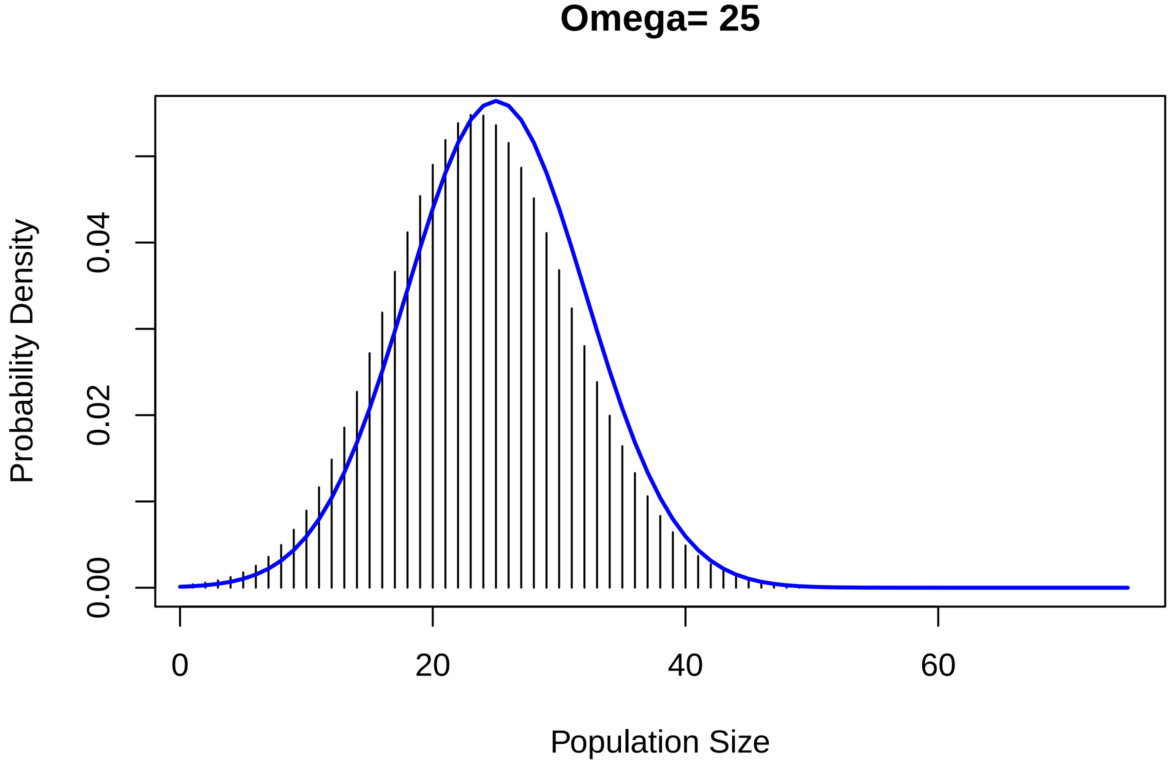
The OU approximation gives us the stationary distribution:

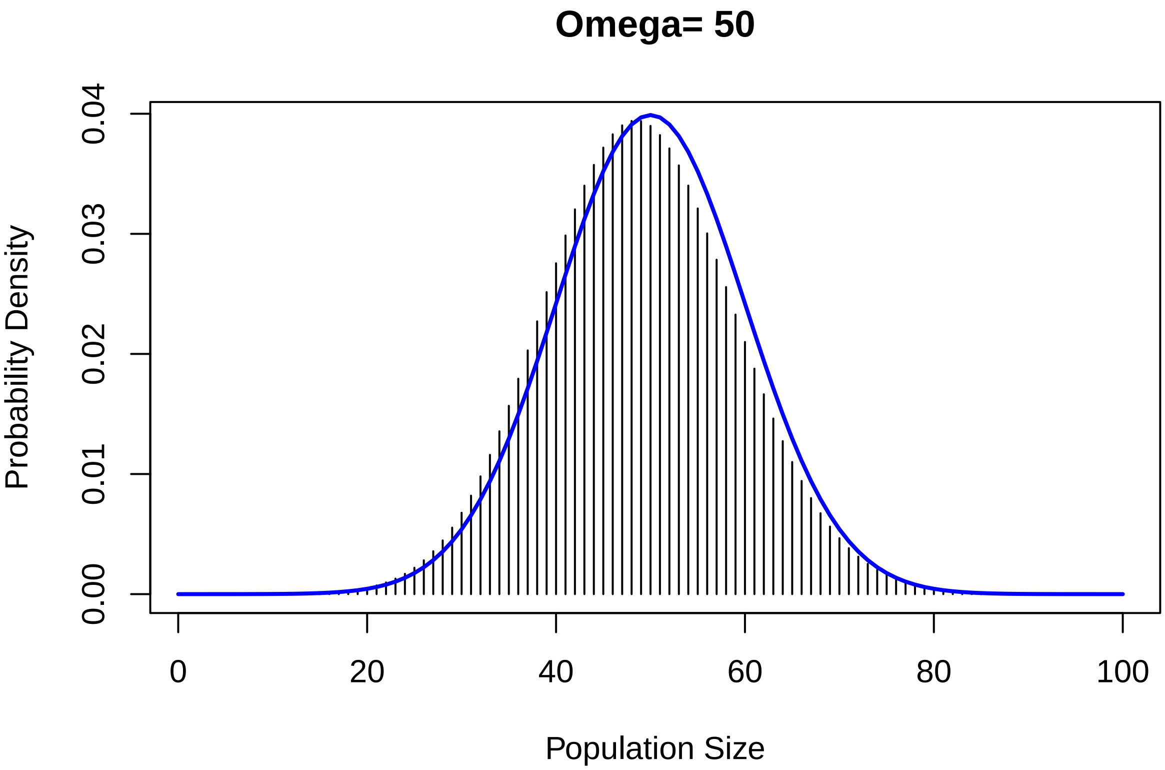
$$x \sim N(\bar{x}, \frac{b^2}{2a})$$

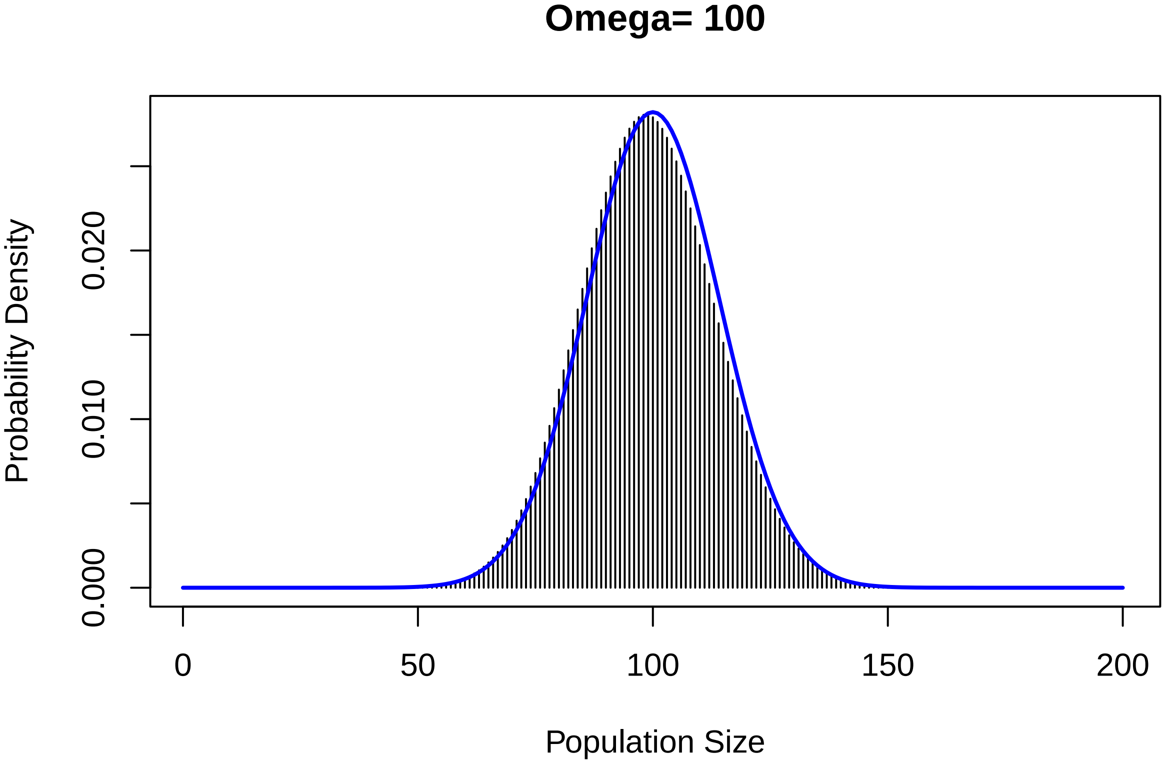
where $a = d'(\bar{x}) - b'(\bar{x})$ and $b = \varepsilon(b(\bar{x}) + d(\bar{x}))$.

```
max.pops <- c(40, 75, 100, 200)
omegas <- c(10, 25, 50, 100)

for(i in 1:4) {
  a.OU <- d.deriv - b.deriv
  b.OU <- sqrt(1/omegas[i]*(b.eq+d.eq))

  B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omegas[i])

  plot.range <- 0:max.pops[i]
  OU <- 1/omegas[i]*dnorm(plot.range*1/omegas[i], mean=xeq, sd = b.OU/sqrt(2*a.OU))

  plot(stationary.dist(B.D.parameters, max.pops[i]), type='h',
       main=paste("Omega=", omegas[i]),
       xlab="Population Size", ylab="Probability Density")
  lines(plot.range, OU, type='l', col="blue", lwd=2)
}
```


Omega= 10
Probability Density
0.00
0.02
0.04
0.06
0.08
0
10
20
30
40
Population Size


Omega= 25
Probability Density
0.00
0.02
0.04
0
20
40
60
Population Size

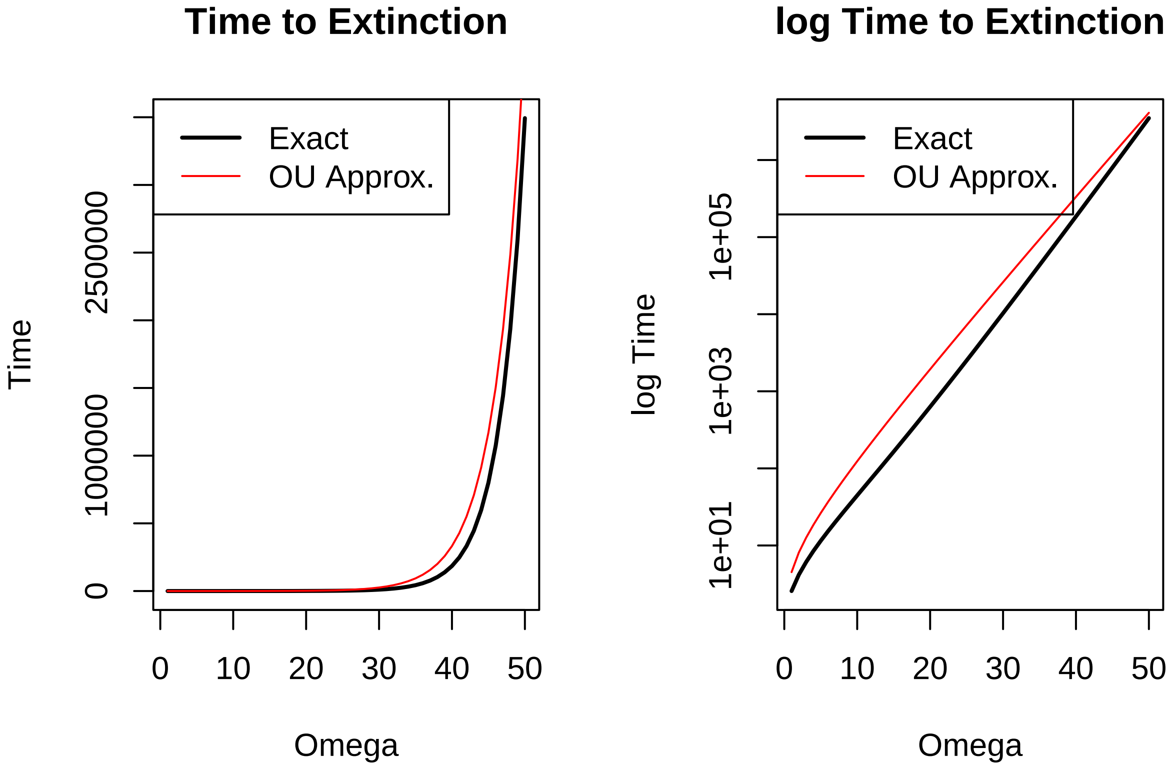




TIME TO EXTINCTION WITH OU APPROXIMATION

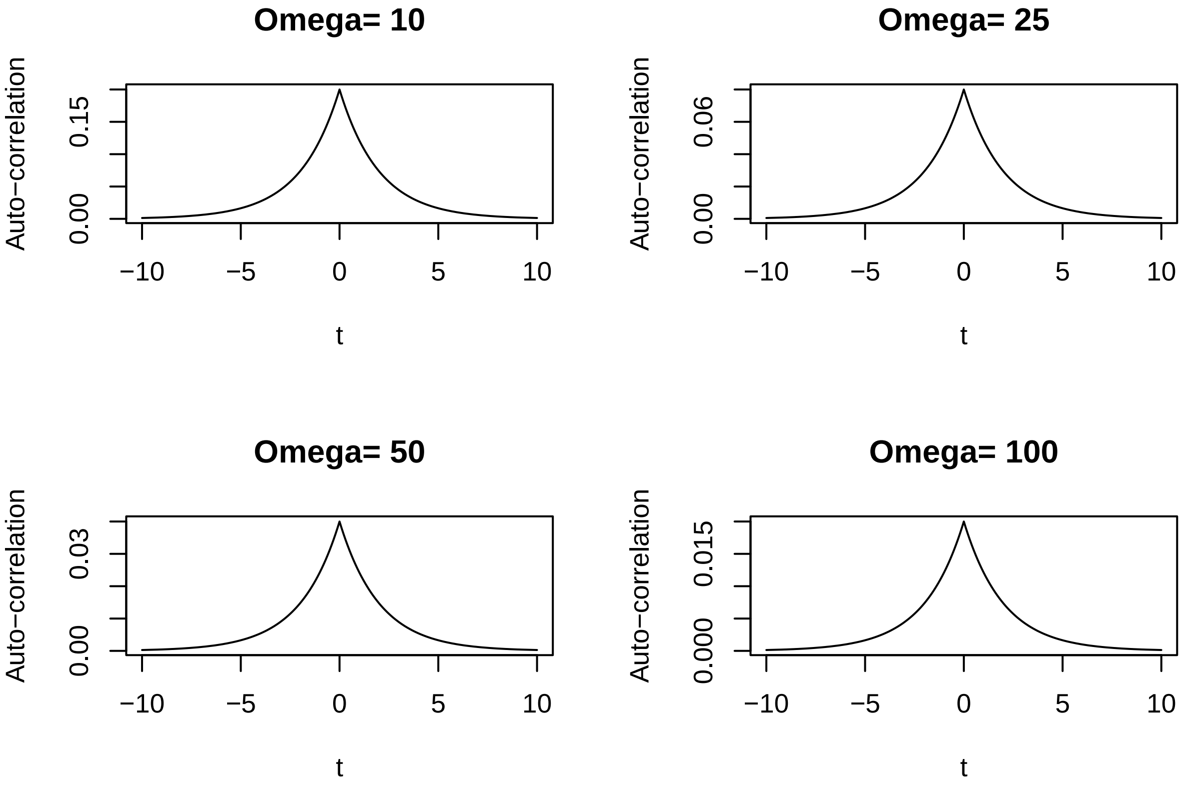
```
omegas <- 1:50
ext.times.OU <- c()
for(om in omegas) {
  b.OU <- sqrt(1/om*(b.eq+d.eq))
  p.eps <- dnorm(1/omega, mean=xeq, sd=b.OU/sqrt(2*a.OU))
  ext.times.OU <- c(ext.times.OU, 1/(p.eps*death(1/om, B.D.parameters)))
}
```

```
par(mfrow=c(1,2))
plot(omegas, ext.times, type='l', lwd=2,
     main="Time to Extinction", xlab="Omega", ylab="Time")
lines(omegas, ext.times.OU, type='l', col="red")
legend("topleft", legend=c("Exact", "OU Approx."),
       lwd=c(2,1), col=c("black", "red"))

plot(omegas, ext.times, type='l', lwd=2, log='y',
     main="log Time to Extinction", xlab="Omega", ylab="log Time")
lines(omegas, ext.times.OU, type='l', col="red")
legend("topleft", legend=c("Exact", "OU Approx."),
       lwd=c(2,1), col=c("black", "red"))
```



AUTO-CORRELATION OF ORNSTEIN-UHLENBECK APPROXIMATION

```
omegas <- c(10, 25, 50, 100)
t.range <- seq(-10, 10, 0.1)

par(mfrow=c(2,2))
for(om in omegas) {
  b.OU <- sqrt(1/om*(b.eq+d.eq))
  s2 <- b.OU^2/(2*a.OU)
  AC <- s2*exp(-abs(t.range*a.OU))

  plot(t.range, AC, type='l',
       main=paste("Omega=", om), xlab="t", ylab="Auto-correlation")
}
```



MEAN TIME TO EXTINCTION CONDITIONAL ON POPULATION DENSITY

The mean time to extinction conditional on population density can be calculated from the Kolomogrov backward equation:

$$\partial_t R(x,t) = -\partial_x (b(x) - d(x)) R(x,t) - \frac{\varepsilon}{2}(b(x) + d(x)) \partial_x^2 R(x,t)$$

where $R(x,t)$ is the cumulative probability density of extinction time t, given the current population density x. Upon integration, this yields the 2nd order Boundary Value Problem:

$$\frac{\varepsilon}{2}(b(x) + d(x)) \frac{d^2 E[T(x)]}{dx^2} + (b(x) - d(x)) \frac{dE[T(x)]}{dx} = -1$$

subject to boundary conditions:

$$E[T(0)] = 0$$

$$\left. \frac{dE[T(x)]}{dx} \right|_{x = x_{max}} = 0$$

The 2nd order non-autonomous ODE needs to be written as a pair of first order non-autonomous ODEs before solving the BVP:

$$\frac{dE[T(x)]}{dx} = s(x)$$

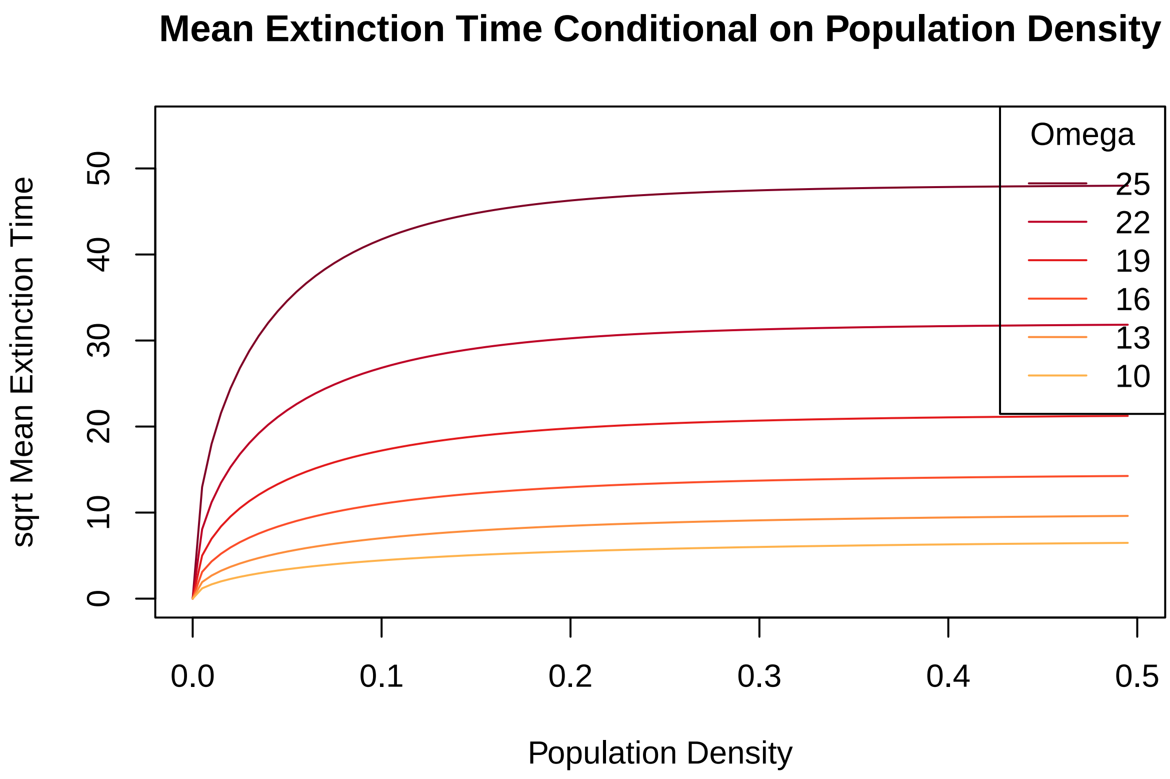
$$\frac{ds(x)}{dx} = -\frac{2}{\varepsilon(b(x)+d(x))}[(b(x)-d(x))s(x)+1]$$

```
library(bvpSolve)
```

```
## Loading required package: deSolve

##
## Attaching package: 'bvpSolve'

## The following object is masked from 'package:stats':
##
##     approx
```

```
library(RColorBrewer)

f2 <- function(x, y, parms) {
  with(parms, {
    drift <- birth(x, BVP.parameters)-death(x, BVP.parameters)
    volatility <- birth(x, BVP.parameters)+death(x, BVP.parameters)

    dy <- y[2]
    dy2 <- -(drift*y[2]+1)*2*omega/volatility

    return(list(c(dy, dy2)))
  })
}
```

```
x <- seq(1e-9, 5, 0.005)

omegas <- rev(seq(10, 25, 3))
BVP.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omegas[1])

colors <- rev(brewer.pal(9, "YlOrRd"))

sol <- bvpcol(yini = c(0, NA), yend = c(NA, 0), bspline = TRUE,
              x = x, func = f2, parms=BVP.parameters)
plot(x[1:length(x)/10], sqrt(sol[1:length(x)/10,2]),
     ylim=c(0, 55), type='l', col=colors[1],
     main="Mean Extinction Time Conditional on Population Density",
     xlab="Population Density", ylab="sqrt Mean Extinction Time")

for(i in 2:length(omegas)) {
  BVP.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omegas[i])
  sol <- bvpcol(yini = c(0, NA), yend = c(NA, 0), bspline = TRUE,
                x = x, func = f2, parms=BVP.parameters)
  lines(x[1:length(x)/10], sqrt(sol[1:length(x)/10,2]), col=colors[i])
}
legend("topright", legend = omegas, col = colors, lty = 1, title="Omega")
```

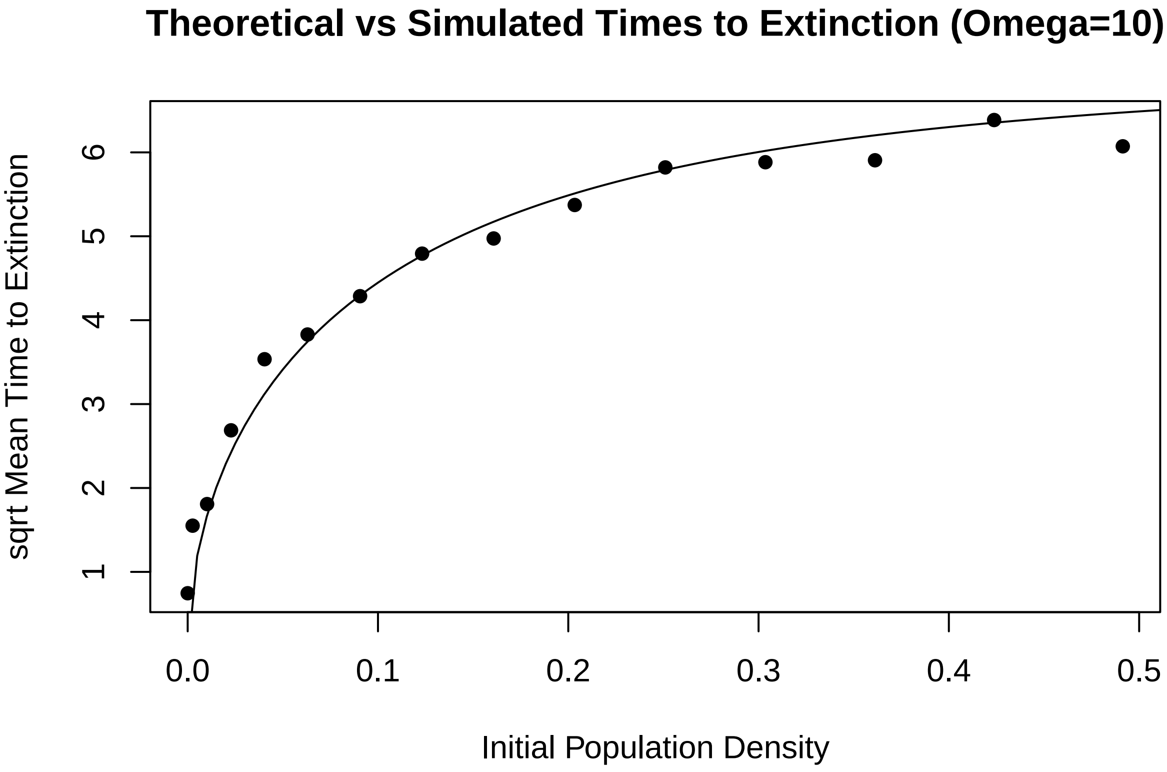


As expected, time to extinction increases as the system size increases, and does to at an accelerating rate. Moreover, note that the plot shows the square root transformed time to extinction, so the actual rate of acceleration is much faster.

```
# Stochastic Orbit
inits <- seq(0.001, 0.75, 0.05)^2
nreps <- length(inits)*250
timesteps <- 10000
dt <- 0.03
omega <- 10

B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, x=xeq, omega=omega)
results <- array(NA, dim=c(timesteps, nreps))

results[1, ] <- rep(inits, nreps/length(inits))

for(t in 2:timesteps) {
  x = results[t-1, ]
  B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, x=x, omega=omega)
  b.rate <- birth(x, B.D.parameters)
  d.rate <- death(x, B.D.parameters)
  dx = (b.rate-d.rate)*dt + sqrt(1/omega*(b.rate+d.rate)*dt)*rnorm(nreps)
  results[t,] <- x+dx
}
```

```
pts <- 1:length(inits)
ext.data <- c()

n.extinct <- rowSums(is.na(results)[,seq(pts[1], nreps, length(inits))])
new.extinctions <- n.extinct[2:length(n.extinct)] - n.extinct[1:(length(n.extinct)-1)]
```

```
ext.data <- c(ext.data, sum(new.extinctions*dt*(c(2:timesteps)))/(nreps/length(inits)))

for(i in 2:length(inits)) {
  n.extinct <- rowSums(is.na(results)[,seq(pts[i], nreps, length(inits))])
  new.extinctions <- n.extinct[2:length(n.extinct)] - n.extinct[1:(length(n.extinct)-1)]
  ext.data <- c(ext.data, sum(new.extinctions*dt*(c(2:timesteps)))/(nreps/length(inits)))
}

x <- seq(1e-5, 5, 0.005)
BVP.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omega)
sol <- bvpcol(yini = c(0, NA), yend = c(NA, 0), bspline = TRUE,
                x = x, func = f2, parms=BVP.parameters)

plot(inits, sqrt(ext.data), pch=16,
     main="Theoretical vs Simulated Times to Extinction (Omega=10)",
     xlab="Initial Population Density", ylab="sqrt Mean Time to Extinction")
lines(x[1:length(x)/2], sqrt(sol[1:length(x)/2,2]))
```

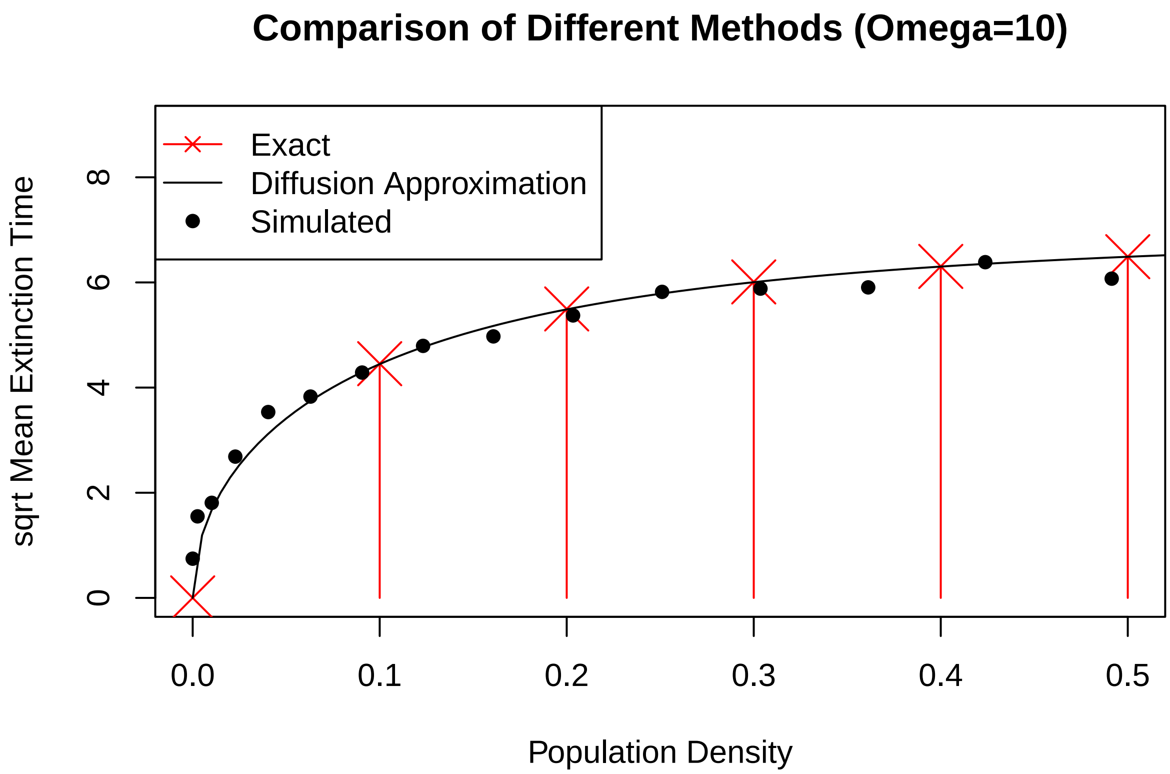


The SDE and diffusion approximations are quite similar to each other. But how do they compare to the exact time to extinction for the birth death process?

```
B.D.parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omega)
n.max <- 50

births <- Birth(1:n.max, B.D.parameters)
deaths <- Death(1:n.max, B.D.parameters)
p.birth <- births/(births+deaths)
p.death <- 1 - p.birth

T.M <- diag(1, n.max+1)
```

```
T.M[3:nrow(T.M), 2:(ncol(T.M)-1)] <- T.M[3:nrow(T.M), 2:(ncol(T.M)-1)] - diag(p.death[2:n.max])
T.M[2:(nrow(T.M)-1), 3:ncol(T.M)] <- T.M[2:(nrow(T.M)-1), 3:ncol(T.M)] - diag(p.birth[1:(n.max-1)])
# set boundary condition: T_(n.max) = T_(n.max-1)
T.M[nrow(T.M), (ncol(T.M)-1):ncol(T.M)] <- c(-1, 1)

# solution vector
T.v <- c(0, 1/(births+deaths))
# for boundary condition
T.v[length(T.v)] <- 0

ext.times.BD <- solve(T.M, T.v)
```

```
plot.max <- round(0.5*omega)

plot((0:plot.max)/omega, sqrt(ext.times.BD[1:(plot.max+1)]), type='h', col="red", ylim=c(0, 9),
     main="Comparison of Different Methods (Omega=10)",
     xlab="Population Density", ylab="sqrt Mean Extinction Time")
points((0:plot.max)/omega, sqrt(ext.times.BD[1:(plot.max+1)]), col="red", pch=4, cex=3)
points(inits, sqrt(ext.data), pch=16)
lines(x, sqrt(sol[,2]))
legend("topleft", legend=c("Exact", "Diffusion Approximation", "Simulated"),
       lty=c(1,1, NA), pch=c(4, NA, 16), col=c("red", "black", "black"))
```



The SDE and diffusion approximations are surprisingly close to the exact solution despite the small system size. Extinction time increases rapidly at small densities, but then saturates due to the boundary condition at the maximum population size (which assumes that large populations rapidly decay to the carrying capacity, so the time to extinction not too different from a population already at carrying capacity).

MULTI-TYPE BIRTH-DEATH PROCESS

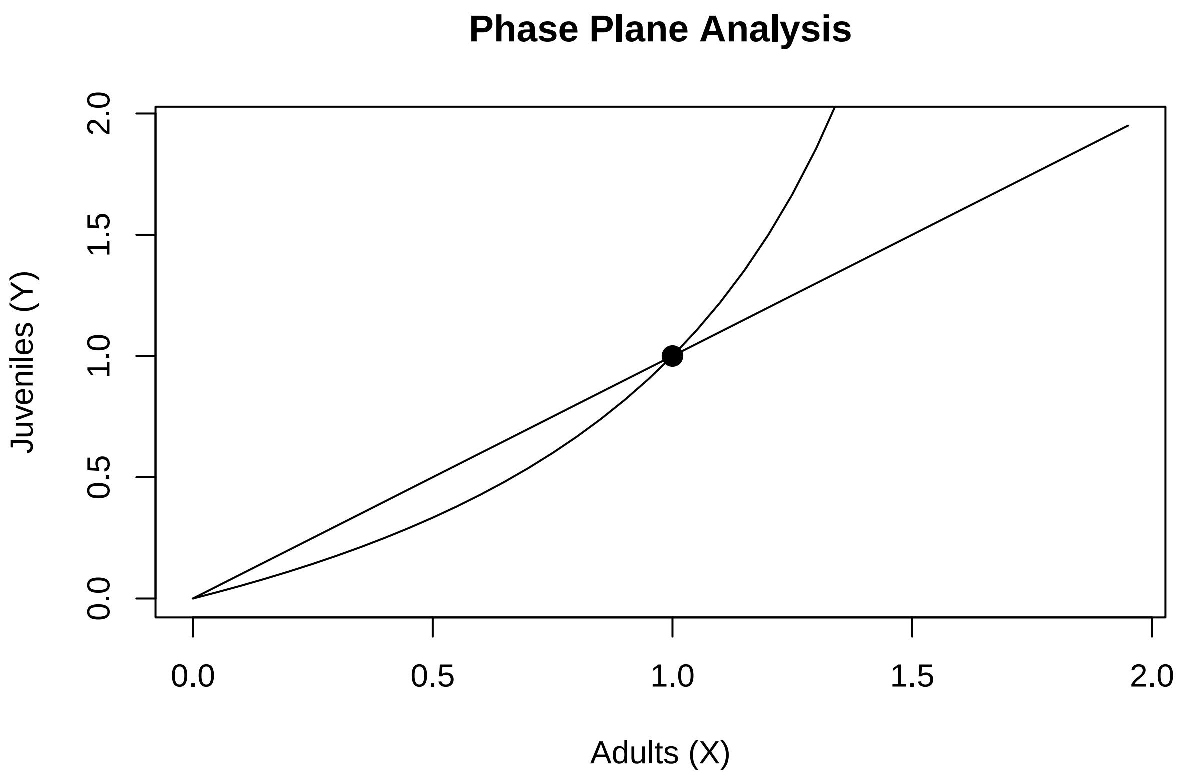
We started the analysis with a time-scale separation, assuming the birth and death rates of juveniles were very high compared to other rates. Now we relax this assumption and take a look at the full 2-dimensional system. The model equations are:

$$\frac{dx}{dt} = \beta y - \delta x$$

$$\frac{dy}{dt} = \lambda x - \nu xy - \mu y$$

```r
model <- function(parameters) {
  with(parameters, {
    dx = B*y - d*x
    dy = L*x-v*x*y-m*y
    return(list(dx=dx, dy=dy))
  })
}

def_sym('ys')
parameters.xy <- list(B=B, L=L, m=m, v=v, d=d, x=xs, y=ys)

y.0 <- equilibrium(model(parameters.xy)$dx, ys, parameters.xy)[[1]]$ys
x.0 <- equilibrium(model(parameters.xy)$dy, xs, parameters.xy)[[1]]$xs

eq <- solve_sys(cbind(xs, ys), cbind(x.0, y.0), list(xs, ys))
xeq <- as_expr(eq[[2]]$xs)
yeq <- as_expr(eq[[2]]$ys)

x.0 <- as_expr(subs(x.0, 'ys', expression(y)))
y.0 <- as_expr(subs(y.0, 'xs', expression(x)))
```

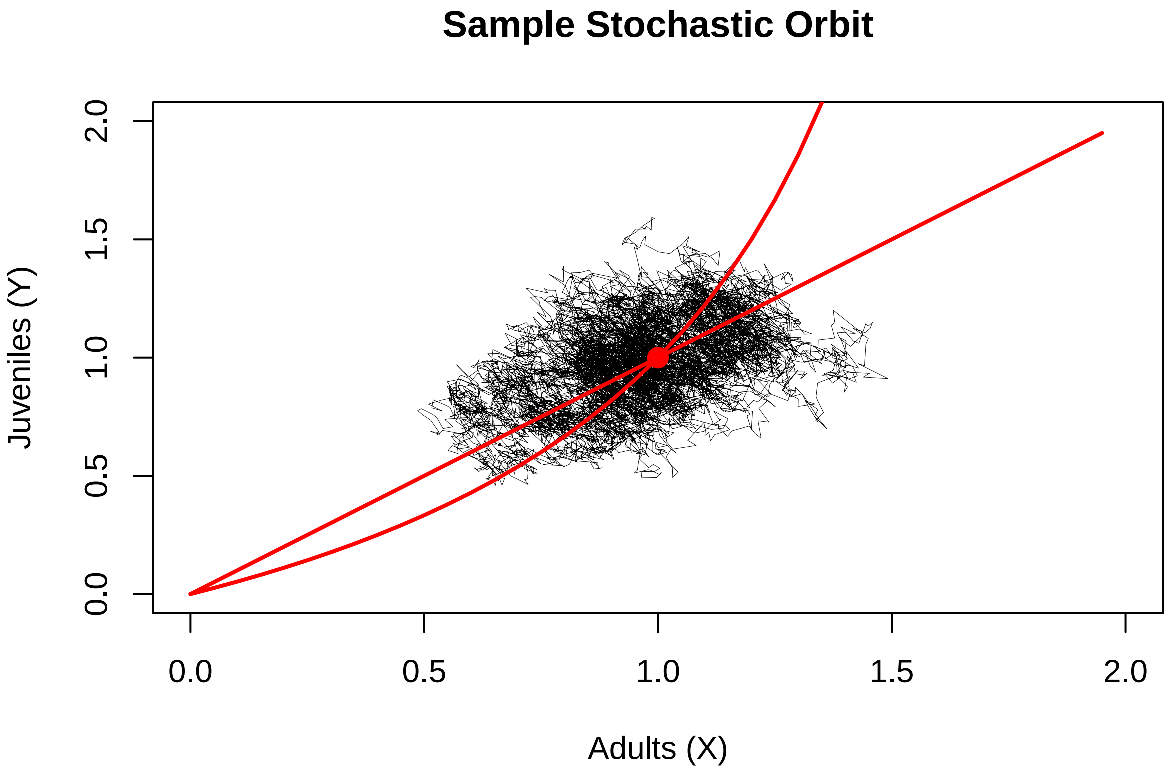
```r
x <- seq(0, 1.99, 0.05)
y <- seq(0, 1.99, 0.05)

plot(x, eval(y.0), type='l',
     main="Phase Plane Analysis", xlab="Adults (X)", ylab="Juveniles (Y)")
lines(y, eval(x.0), type='l')
points(xeq, yeq, pch=16, cex=1.5)
```



```r
K <- function(parameters) {
  with(parameters, {
    S.xx <- B*y + d*x
    S.yy <- (B+m+v*x)*y + L*x
    S.xy <- -B*y
    S.yx <- S.xy
    Covs <- 1/omega*matrix(c(S.xx, S.yx, S.xy, S.yy), 2, 2)
    return(Covs)
  })
}
```

```r
dt=0.01
timesteps <- seq(0, 5000, dt)
omega=50

results <- array(NA, dim = c(length(timesteps), 2))
results[1,] <- c(1, 1)
for(t in 2:length(timesteps)) {
  x <- results[t-1,1]
  y <- results[t-1,2]
  parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omega, x=x, y=y)

  inc <- model(parameters)
  E.dxy <- c(inc$dx, inc$dy)

  L.K <- t( chol(K(parameters)+diag(1e-6, 2)) )

  results[t,] <- results[t-1,] + E.dxy*dt + sqrt(dt)*L.K%*%rnorm(2)
}
```

```
x <- seq(0, 1.99, 0.05)
y <- seq(0, 1.99, 0.05)

plot(results[1:10000,1], results[1:10000,2], lwd=0.2, type='l',
     xlim=c(0, 2), ylim=c(0, 2),
     main="Sample Stochastic Orbit", xlab="Adults (X)", ylab="Juveniles (Y)")
lines(x, eval(y.0), type='l', col="red", lwd=2)
lines(y, eval(x.0), col="red", lwd=2)
points(xeq, yeq, pch=16, cex=1.5, col="red")
```



```
library(pracma)
```

```
##
## Attaching package: 'pracma'

## The following object is masked from 'package:deSolve':
##
##     rk4

## The following objects are masked from 'package:caracas':
##
##     eye, hessian, inv, jacobian, nullspace, ones, pinv, rref, taylor,
##     zeros
```

```
library(viridis)
```

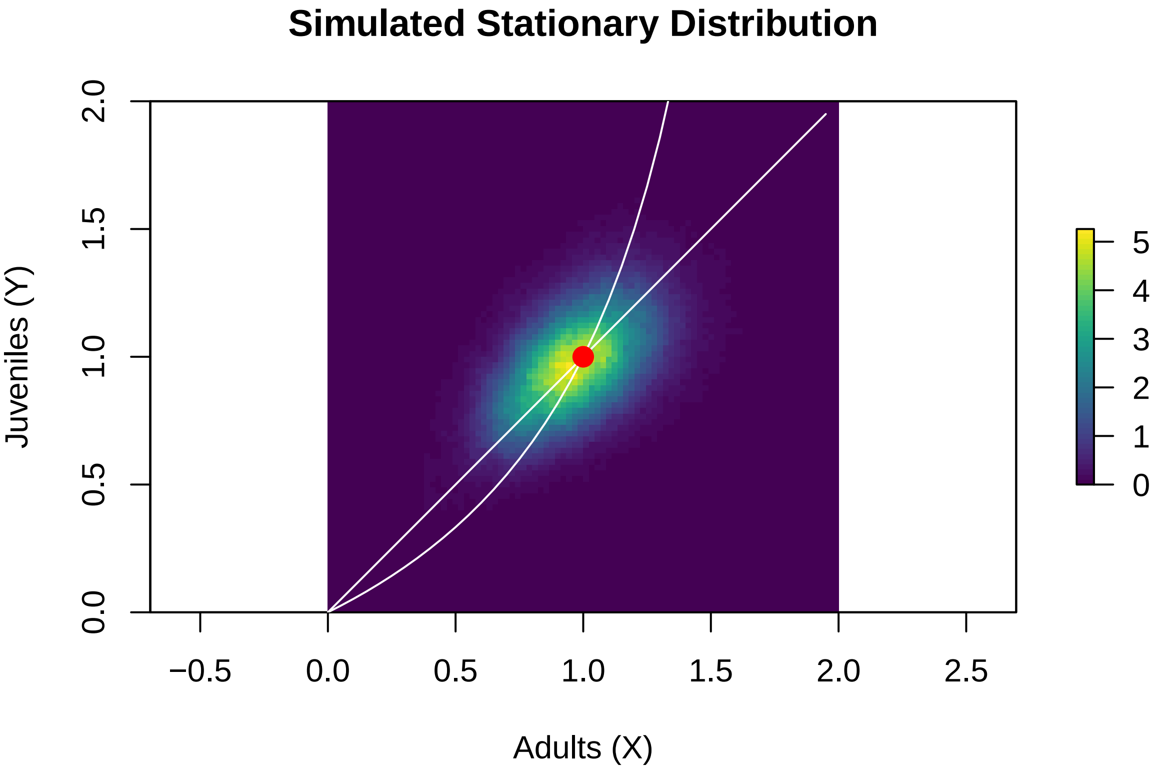
```
## Loading required package: viridisLite
```

```
library(graphics)
library(raster)
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'raster'

## The following object is masked from 'package:caracas':
##
##     subs
```

```
bins <- 90
e <- extent(0, 2, 0, 2)
r <- raster(e, nrow=bins, ncol=bins)
z <- rasterize(results, r, rep(1, nrow(results))/nrow(results)/(2/bins)^2, fun=sum, background=0)

plot(z, col=viridis_pal()(50), xlim=c(0, 2), ylim=c(0, 2),
     main="Simulated Stationary Distribution", xlab="Adults (X)", ylab="Juveniles (Y)")
lines(x, eval(y.0), col="white")
lines(y, eval(x.0), col="white")
points(xeq, yeq, pch=16, col="red", cex=1.5)
```

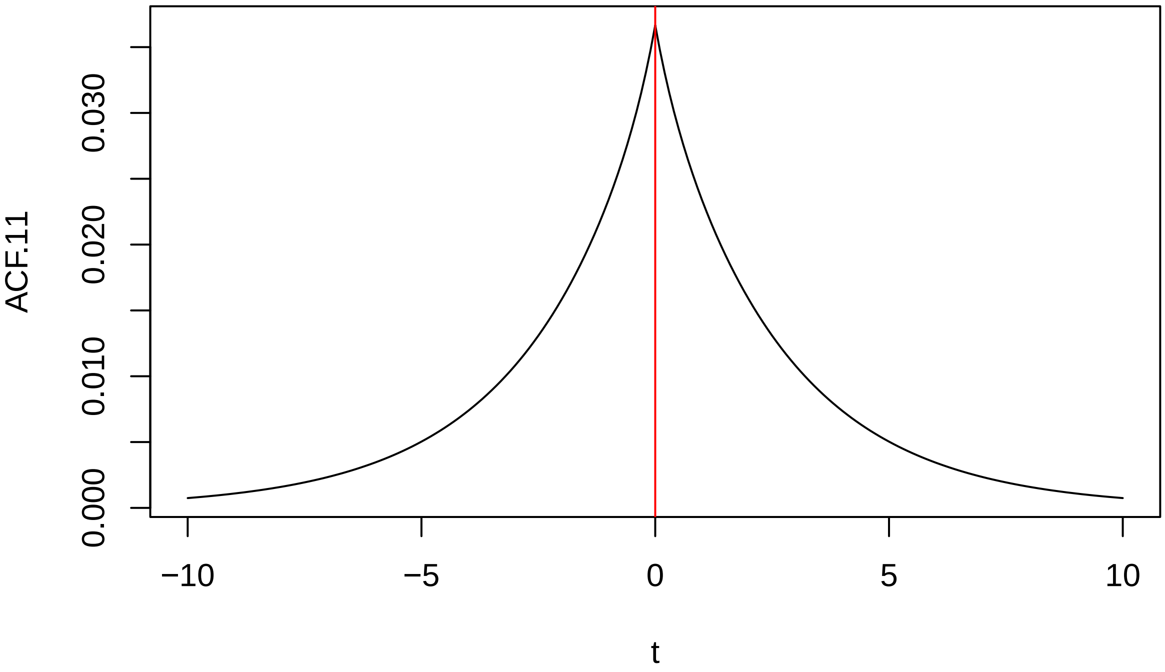


ORNSTEIN-UHLENBECK APPROXIMATION TO THE MULTI-TYPE BIRTH DEATH PROCESS

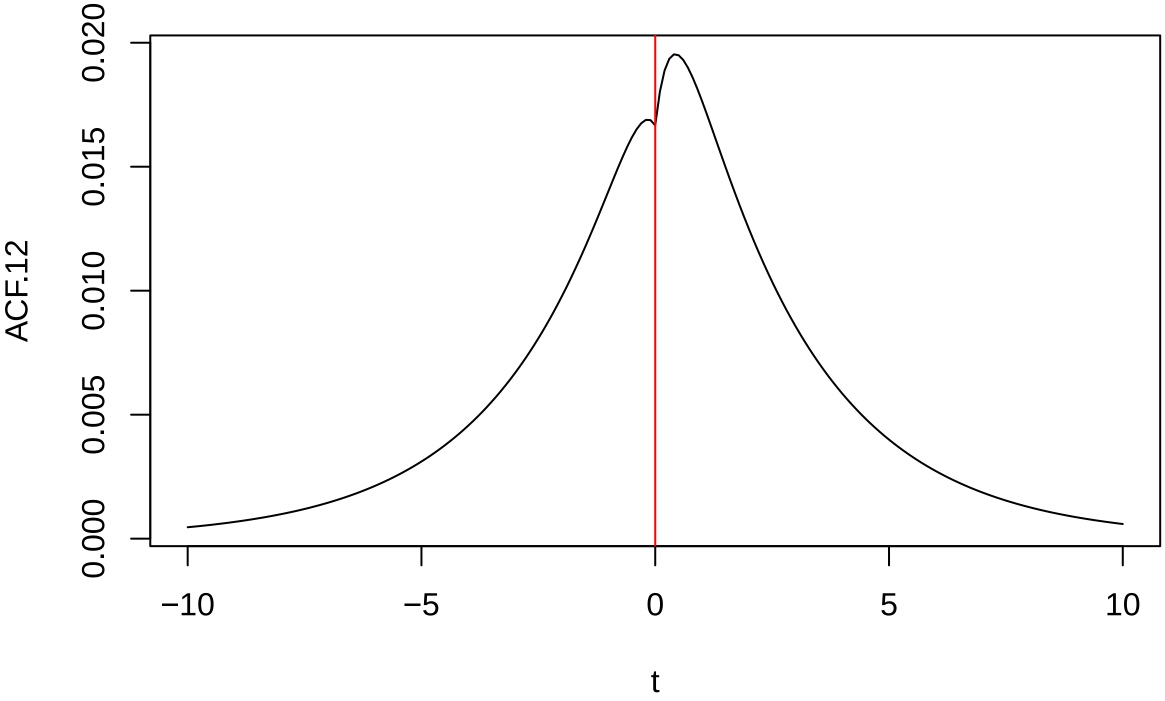
```
parameters <- list(B=B, L=L, m=m, v=v, d=d, omega=omega, x=xeq, y=yeq)

d.xy <- model(parameters.xy)
dx <- d.xy$dx
dy <- d.xy$dy

J <- caracas::jacobian(c(dx, dy), c(xs, ys))
A.OU <- as_expr(caracas::subs(J, c('xs', 'ys'), c(xeq, yeq)))

B.OU <- t( chol(K(parameters)) )
```

```r
# Covariance Matrix
def_sym("s11", "s12", "s21", "s22")
Cov <- t(matrix(c("s11", "s12", "s21", "s22"), 2, 2))
Cov <- as_sym(Cov)
cov.eq <- as_sym(A.OU)%*%Cov + Cov%*%as_sym(t(A.OU)) + as_sym(B.OU)%*%as_sym(t(B.OU))
cov.elems <- solve_sys(cbind(cov.eq[1,1], cov.eq[1,2], cov.eq[2,1], cov.eq[2,2]),
                       list(s11, s12, s21, s22))[[1]]

s11 <- as_expr(cov.elems$s11)
s12 <- as_expr(cov.elems$s12)
s21 <- as_expr(cov.elems$s21)
s22 <- as_expr(cov.elems$s22)
Cov <- t(matrix(c(s11, s12, s21, s22), 2, 2))
```

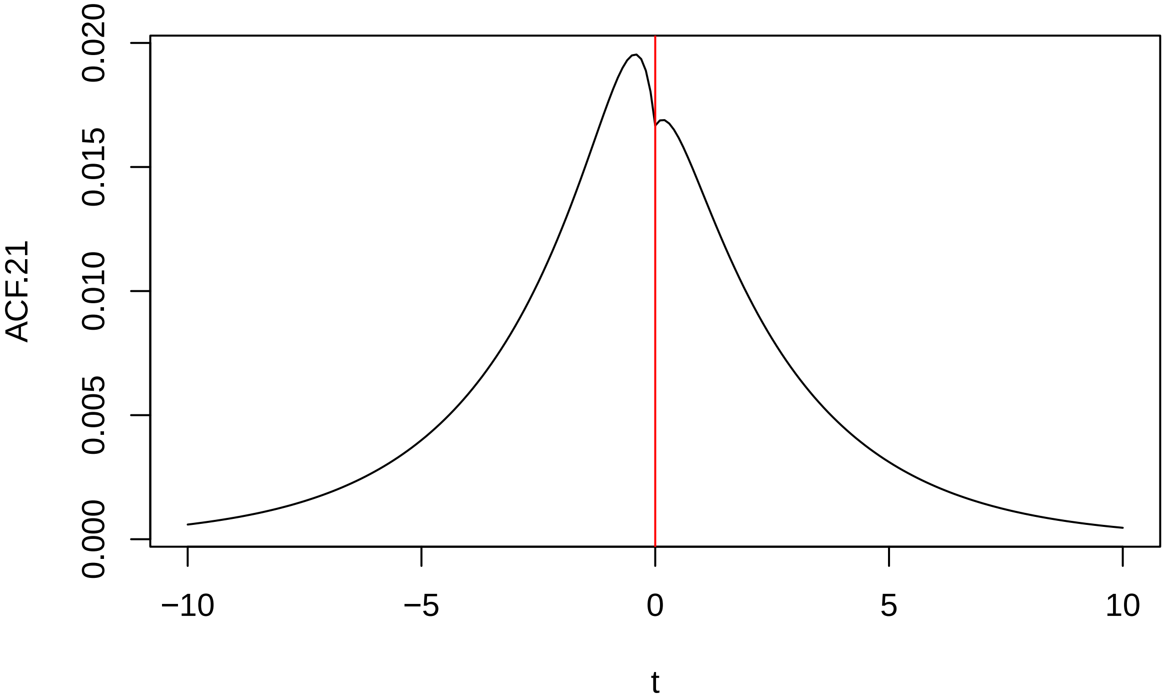
```r
# Auto-correlation Functions
eigens <- eigen(A.OU)
U <- eigens$vectors
Lambda <- eigens$values

t.pos <- seq(0, 10, 0.1)
t.neg <- -rev(t.pos)
t <- c(t.neg, t.pos)

ACF.11.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%Cov)[1,1]})
ACF.11.neg <- sapply(t.neg, function(tau) {(Cov%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[1,1]})
ACF.11 <- c(ACF.11.neg, ACF.11.pos)

ACF.12.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%Cov)[1,2]})
ACF.12.neg <- sapply(t.neg, function(tau) {(Cov%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[1,2]})
ACF.12 <- c(ACF.12.neg, ACF.12.pos)

ACF.21.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%Cov)[2,1]})
ACF.21.neg <- sapply(t.neg, function(tau) {(Cov%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[2,1]})
ACF.21 <- c(ACF.21.neg, ACF.21.pos)

ACF.22.pos <- sapply(t.pos, function(tau) {(U%*%diag(exp(Lambda*tau))%*%solve(U)%*%Cov)[2,2]})
ACF.22.neg <- sapply(t.neg, function(tau) {(Cov%*%t(U%*%diag(exp(-Lambda*tau))%*%solve(U)))[2,2]})
ACF.22 <- c(ACF.22.neg, ACF.22.pos)

plot(t, ACF.11, type='l')
abline(v=0, col="red")
```



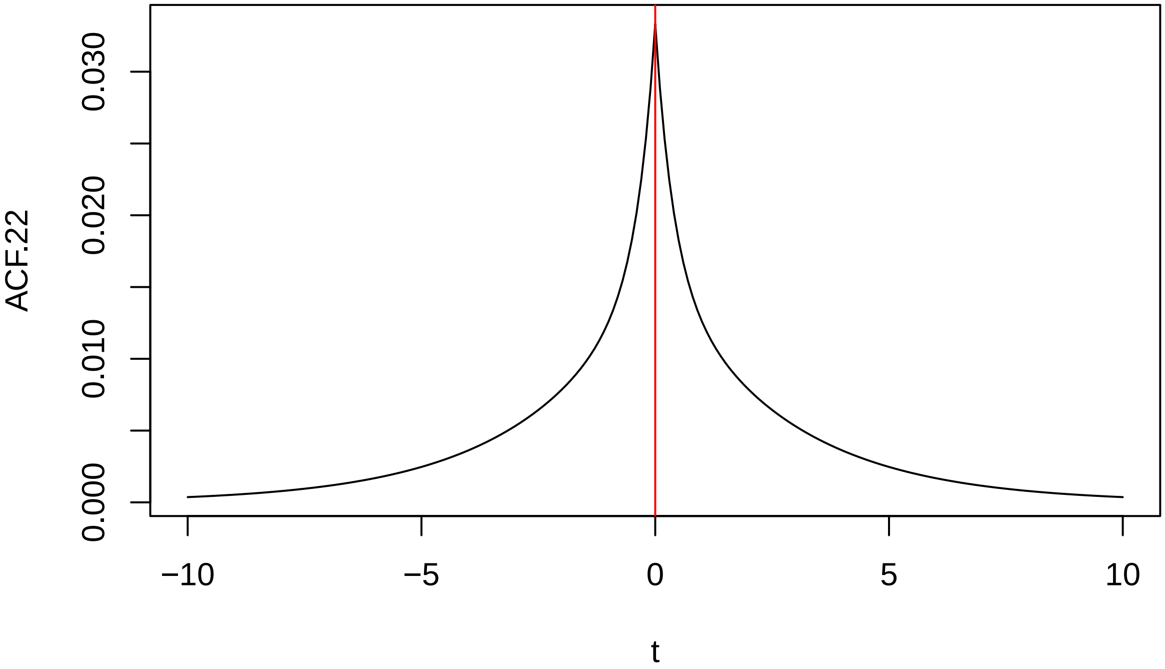
```
plot(t, ACF.12, type='l')
abline(v=0, col="red")
```



```
plot(t, ACF.21, type='l')
abline(v=0, col="red")
```

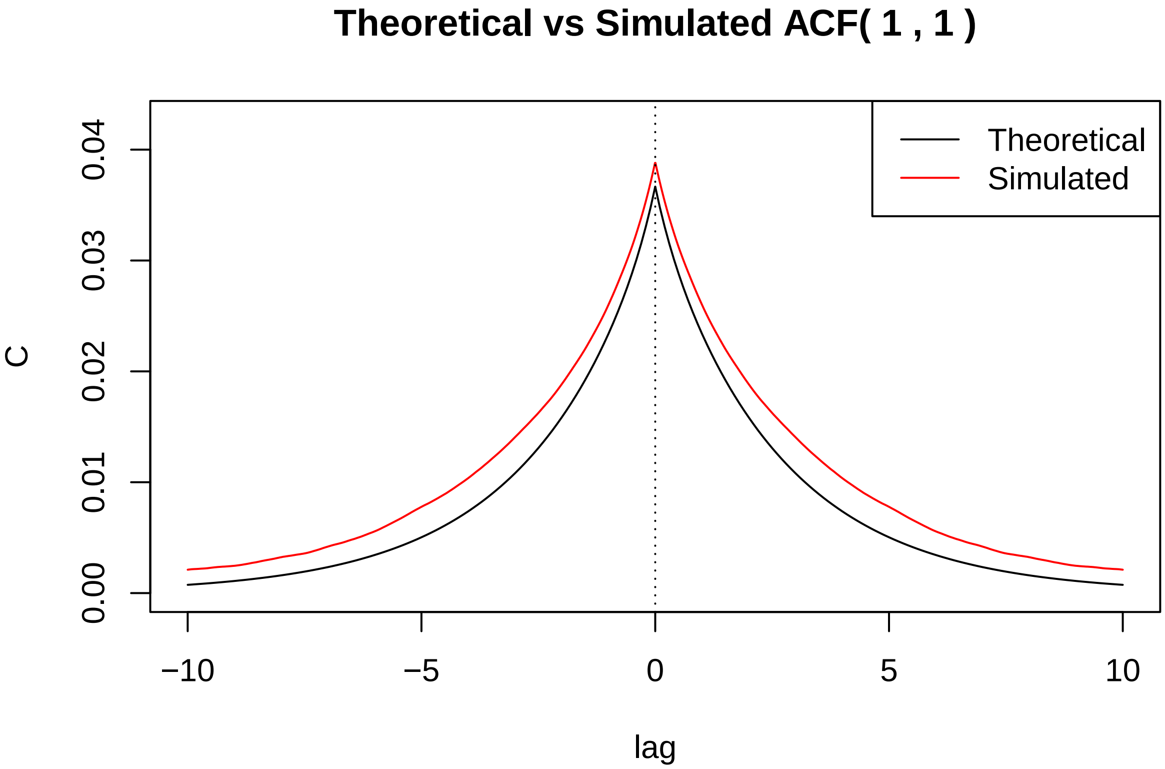


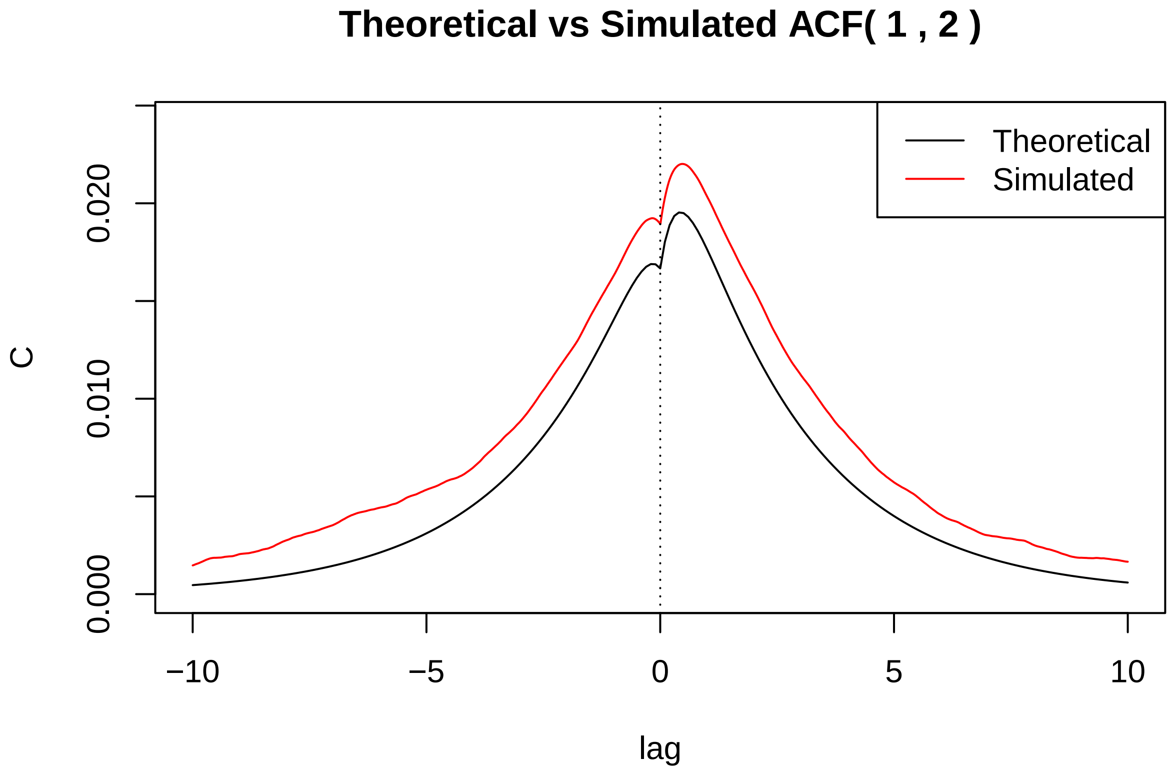
```
plot(t, ACF.22, type='l')
abline(v=0, col="red")
```

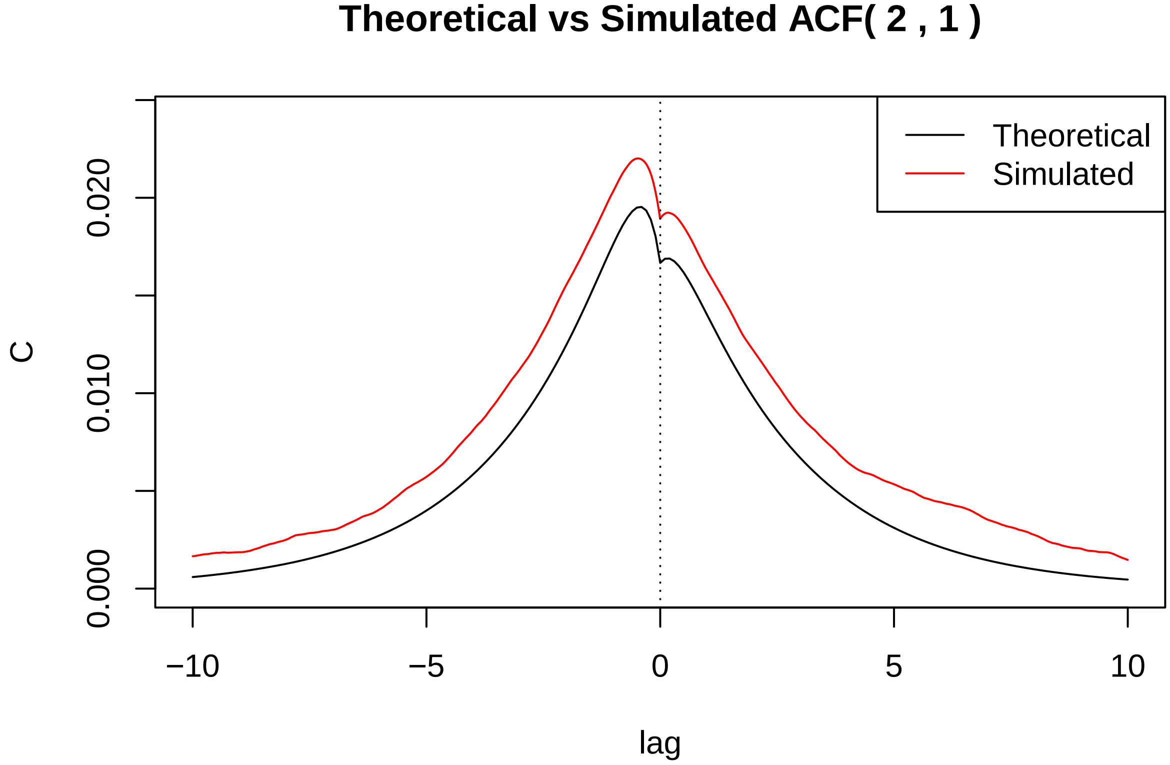


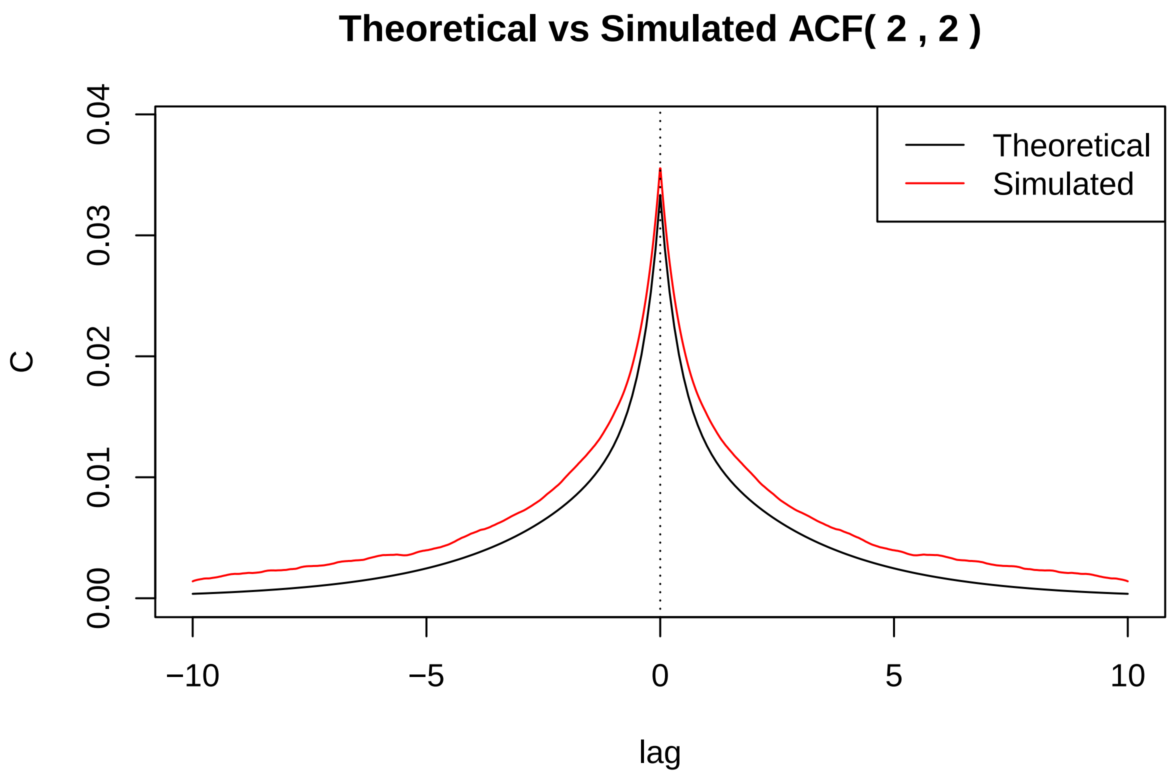
We can see that the auto-covariance between adult and juvenile densities is bimodal, with one peak occuring in the future and another in the past. This is because adult densities predict future juvenile densities through births and interference, while juvenile densities predict future adult densities through maturation. The asymmetry resulting from the different heights of the peaks indicate that juvenile densities are a better predictor of future adult densities than adults are of future juvenile densities. This is likely due to the fact that while adults have a positive effect on juveniles through birth events, this is somewhat countered by the negative effect of interference/cannibalism. Juveniles, on the other hand, directly mature into adults and have no negative effect on adult densities to counter the positive effect.

```r
acf.empirical.pos <- acf(results, type="covariance",
                         lag.max=max(t.pos)/dt, plot = FALSE)$acf
acf.empirical.neg <- acf(results[nrow(results):1,], type="covariance",
                         lag.max=abs(min(t.neg))/dt, plot = FALSE)$acf

acf.11.empirical <- c(rev(acf.empirical.neg[,1,1]), acf.empirical.pos[,1,1])
acf.12.empirical <- c(rev(acf.empirical.neg[,1,2]), acf.empirical.pos[,1,2])
acf.21.empirical <- c(rev(acf.empirical.neg[,2,1]), acf.empirical.pos[,2,1])
acf.22.empirical <- c(rev(acf.empirical.neg[,2,2]), acf.empirical.pos[,2,2])

acf.data.empirical <- cbind(acf.11.empirical, acf.12.empirical, acf.21.empirical, acf.22.empirical)
acf.data.theoretical <- cbind(ACF.11, ACF.12, ACF.21, ACF.22)

plot.range <- seq(min(t.neg), max(t.pos), length.out=length(acf.11.empirical))

ind.i <- c(1,1,2,2)
ind.j <- c(1,2,1,2)
for(i in 1:ncol(acf.data.empirical)) {
  plot(plot.range, acf.data.empirical[,i], type='l',
       ylim=c(0, 1.1*max(acf.data.empirical[,i])), col="red",
       main = paste("Theoretical vs Simulated ACF(",ind.i[i],",",ind.j[i],")" ), xlab="lag", ylab="C")
  lines(c(t.neg, t.pos), acf.data.theoretical[,i], type='l')
  abline(v=0, lty=3)
  legend("topright", legend=c("Theoretical", "Simulated"), lty=1, col=c("black", "red"))
}
```

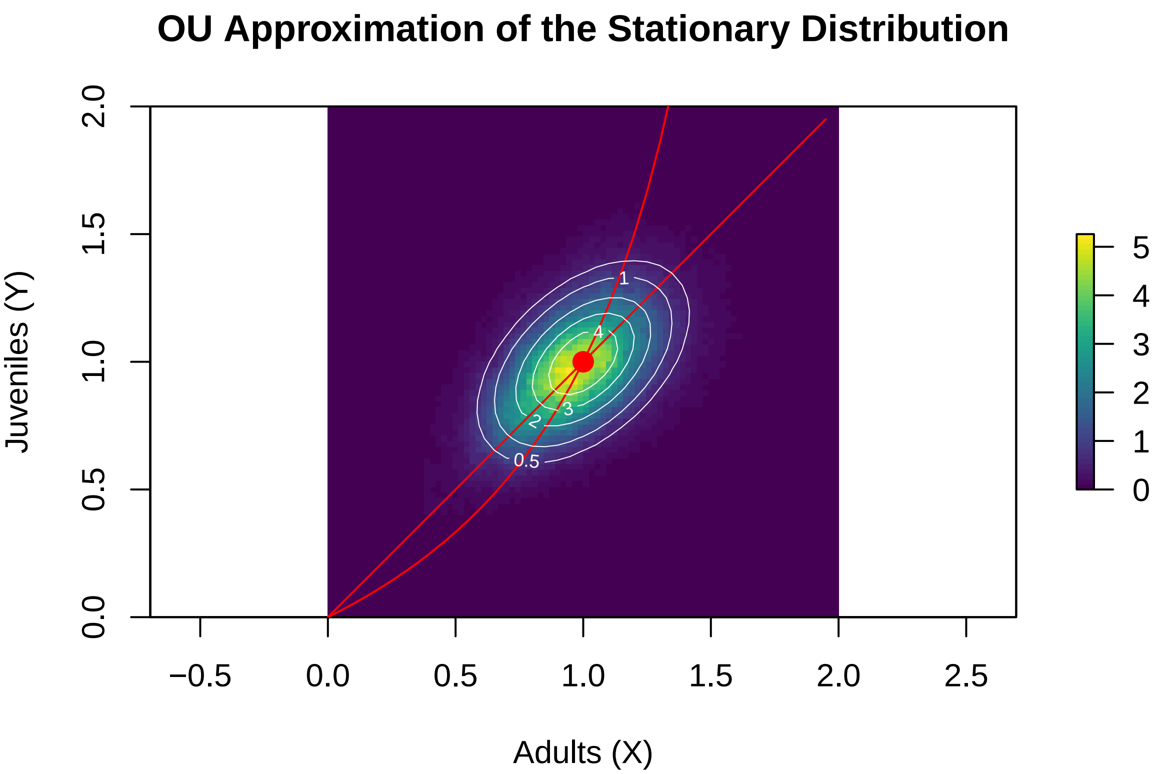



Theoretical vs Simulated ACF(1 , 2)
Theoretical
Simulated
C
0.000
0.010
0.020
−10
−5
0
5
10
lag

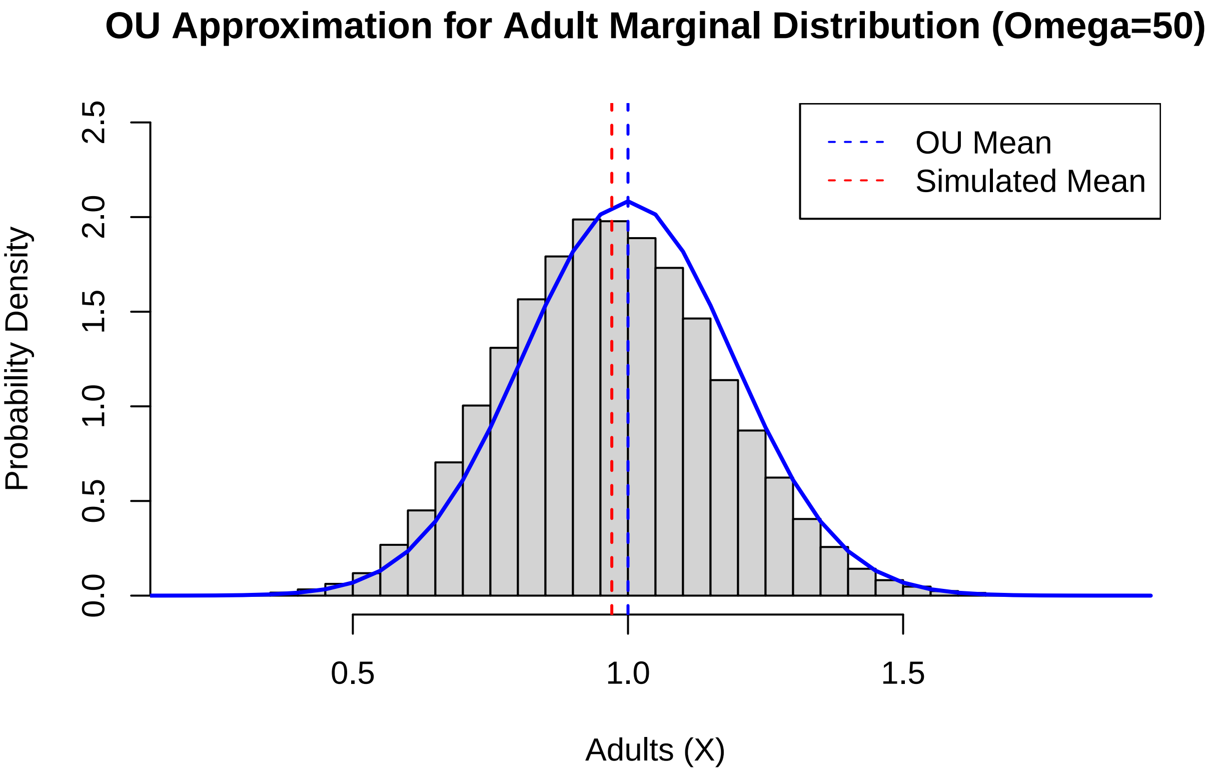

Theoretical vs Simulated ACF(2 , 1)
Theoretical
Simulated
C
0.000
0.010
0.020
−10
−5
0
5
10
lag



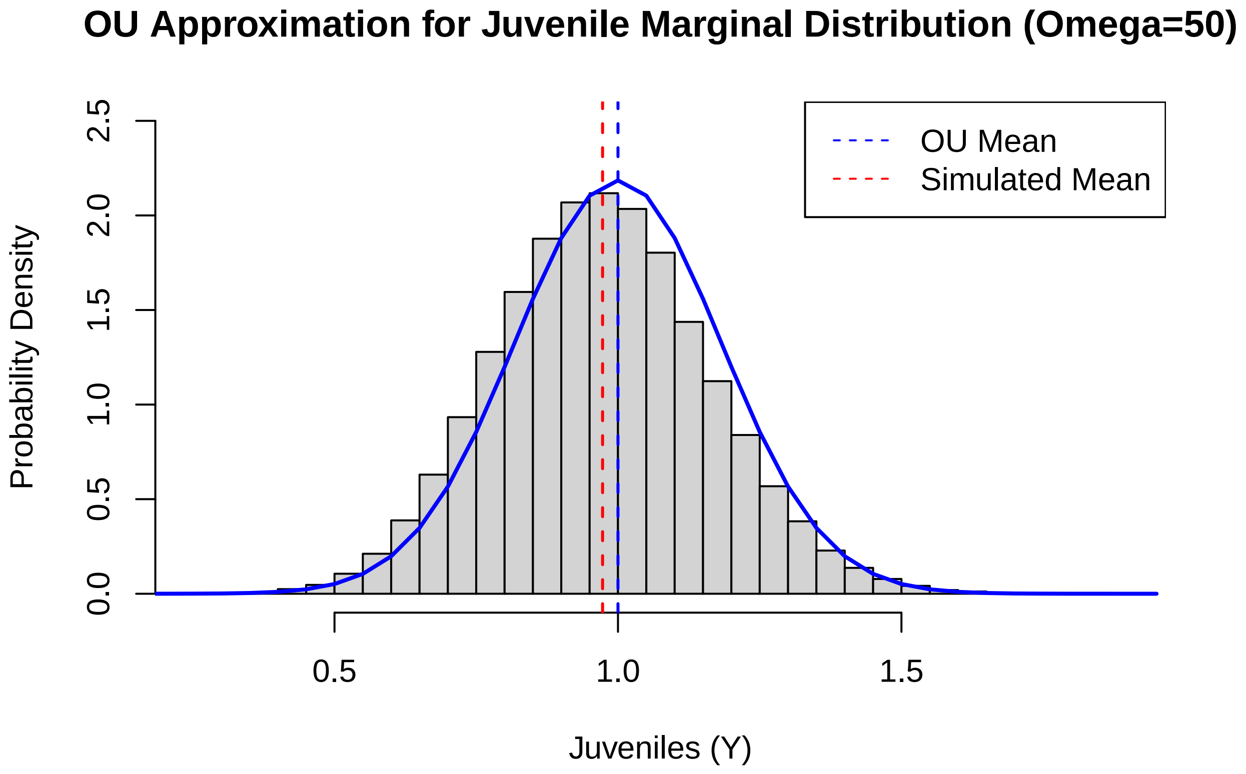
```
library(mnormt)

#create bivariate normal distribution
mu    <- c(xeq, yeq)
f     <- function(x, y) dmnorm(cbind(x, y), mu, Cov)
dens  <- outer(x, y, f)

#create contour plot
plot(z, col=viridis_pal()(50), xlim=c(0, 2), ylim=c(0, 2),
     main="OU Approximation of the Stationary Distribution", xlab="Adults (X)", ylab="Juveniles (Y)")
lines(x, eval(y.0), col="red")
lines(y, eval(x.0), col="red")
points(xeq, yeq, pch=16, col="red", cex=1.5)
contour(x, y, dens, add=TRUE, col="white", levels=c(4, 3, 2, 1, 0.5), lwd=0.5)
```



```
hist(results[,1], prob=T, breaks=30, ylim=c(0, 2.5),
     main="OU Approximation for Adult Marginal Distribution (Omega=50)",
     xlab="Adults (X)", ylab="Probability Density")
lines(x, dnorm(x, mean=xeq, sd=sqrt(Cov[1,1])), type='l', lwd=2, col="blue")
abline(v=xeq, lty=2, col="blue", lwd=1.5)
abline(v=mean(results[,1]), lty=2, col="red", lwd=1.5)
legend("topright", legend=c("OU Mean", "Simulated Mean"), col=c("blue", "red"), lty=2)
```



```
hist(results[,2], prob=T, breaks=30, ylim=c(0, 2.5),
     main="OU Approximation for Juvenile Marginal Distribution (Omega=50)",
     xlab="Juveniles (Y)", ylab="Probability Density")
lines(x, dnorm(y, mean=yeq, sd=sqrt(Cov[2,2])), type='l', lwd=2, col="blue")
abline(v=yeq, lty=2, col="blue", lwd=1.5)
abline(v=mean(results[,2]), lty=2, col="red", lwd=1.5)
legend("topright", legend=c("OU Mean", "Simulated Mean"), col=c("blue", "red"), lty=2)
```


OU Approximation for Juvenile Marginal Distribution (Omega=50)
OU Mean
Simulated Mean
Probability Density
0.0
0.5
1.0
1.5
2.0
2.5
0.5
1.0
1.5
Juveniles (Y)