differential-equations

Following the tutorial from:

Differential Equations in R Part 1: Representing Basic Dynamics

https://www.youtube.com/watch?v=1iNXQypailI

Differential Equations in R Part 2: Solving Lotka-Volterra Predation Equations

https://www.youtube.com/watch?v=lJqiasw7OPs

Note: I fixed a few mistakes with the functions where it was using more global scope than I wanted.

Install the library deSolve from Packages -> Install

Use the library deSolve and lattice

```
library(deSolve)
library(lattice)
```

Solving the continous equation

$$\frac{dN}{dt} = rN$$

Create a function

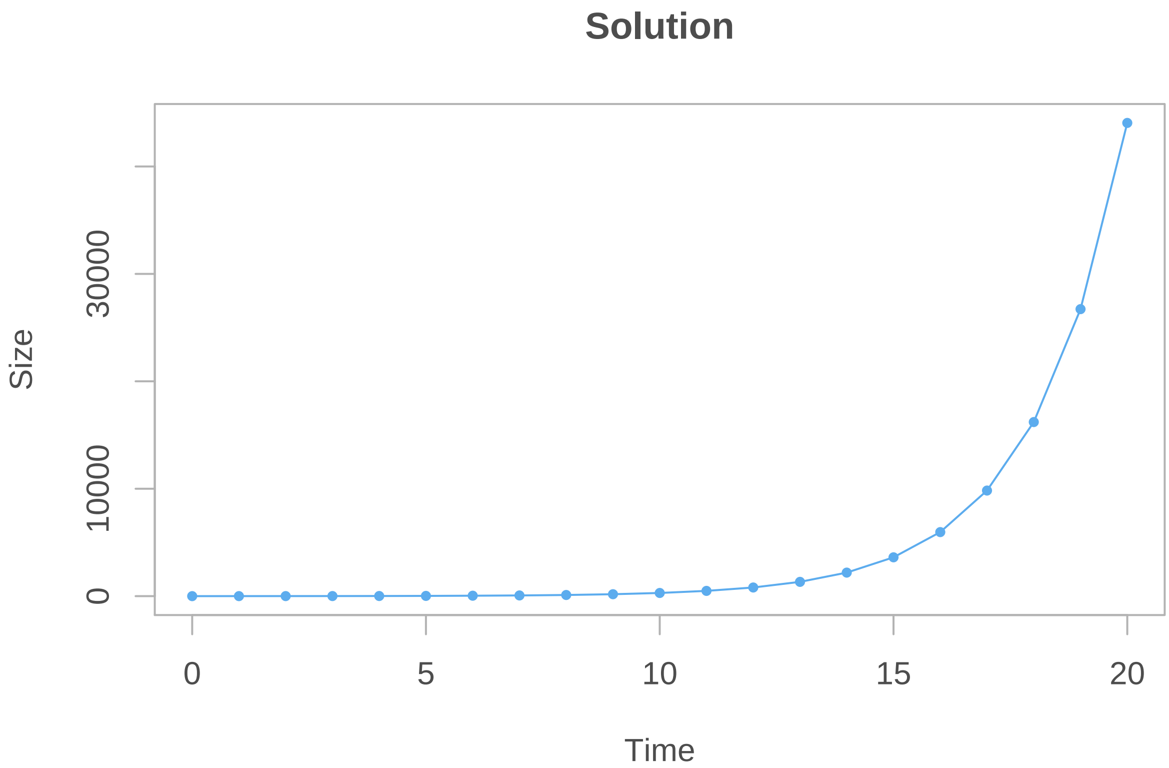
```
cgrowth <- function(times, y, parms) {
  r <- parms[1]
  N <- y[1]
  dN.dt <- r * N
  return(list(dN.dt))
}

p  <- c(p=0.5)
y0 <- c(N=2)
t  <- 0:20

sol <- ode(y = y0, times = t, func = cgrowth, parms = p)
sol
```

```
##    time           N
## 1     0    2.000000
## 2     1    3.297445
## 3     2    5.436572
## 4     3    8.963395
## 5     4   14.778153
## 6     5   24.365058
## 7     6   40.171205
## 8     7   66.231149
## 9     8  109.196755
## 10    9  180.035094
## 11   10  296.827805
## 12   11  489.386525
## 13   12  806.862352
## 14   13 1330.291724
## 15   14 2193.281112
## 16   15 3616.110779
```

```
## 17   16  5961.961552
## 18   17  9829.617263
## 19   18 16206.305348
## 20   19 26719.691878
## 21   20 44053.345008
```

Plotting it would be

```
plot(sol, type='o', xlab="Time", ylab="Size", main="Solution",
     pch=16, cex=0.7, fg="grey70", col="steelblue2", col.axis="grey30",
     col.lab="grey30", col.main="grey30")
```



Solving a continous time logistic equation numerically

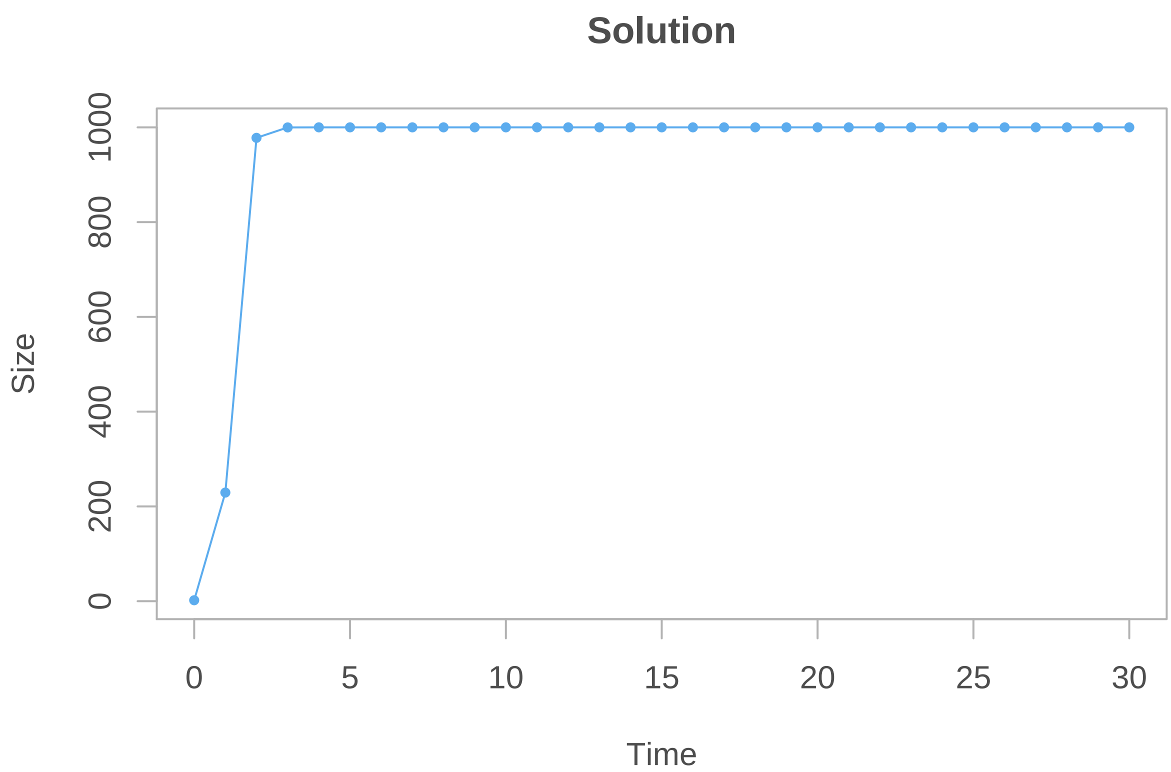
$$\frac{dN}{dt} = rN(1 - \frac{N}{K})$$

```
clogistic <- function(times, y, parms) {
  r <- parms[1]
  K <- parms[2]
  N <- y[1]
  dN.dt <- r * N * (1 - (N / K))
  return(list(dN.dt))
}

p  <- c(r=5, K=1000)
y0 <- c(N=2)
t  <- 0:30

sol2 <- ode(y = y0, times = t, func = clogistic, parms = p)
sol2
```

```
##    time         N
## 1     0    2.0000
## 2     1  229.2406
## 3     2  977.8473
## 4     3  999.8474
## 5     4  999.9990
## 6     5 1000.0000
## 7     6 1000.0000
## 8     7 1000.0000
## 9     8 1000.0000
## 10    9 1000.0000
## 11   10 1000.0000
## 12   11 1000.0000
## 13   12 1000.0000
## 14   13 1000.0000
## 15   14 1000.0000
## 16   15 1000.0000
## 17   16 1000.0000
## 18   17 1000.0000
## 19   18 1000.0000
## 20   19 1000.0000
## 21   20 1000.0000
## 22   21 1000.0000
## 23   22 1000.0000
## 24   23 1000.0000
## 25   24 1000.0000
## 26   25 1000.0000
## 27   26 1000.0000
## 28   27 1000.0000
## 29   28 1000.0000
## 30   29 1000.0000
## 31   30 1000.0000
```

Plotting it would be

```r
plot(sol2, type='o', xlab="Time", ylab="Size", main="Solution",
     pch=16, cex=0.7, fg="grey70", col="steelblue2", col.axis="grey30",
     col.lab="grey30", col.main="grey30")
```

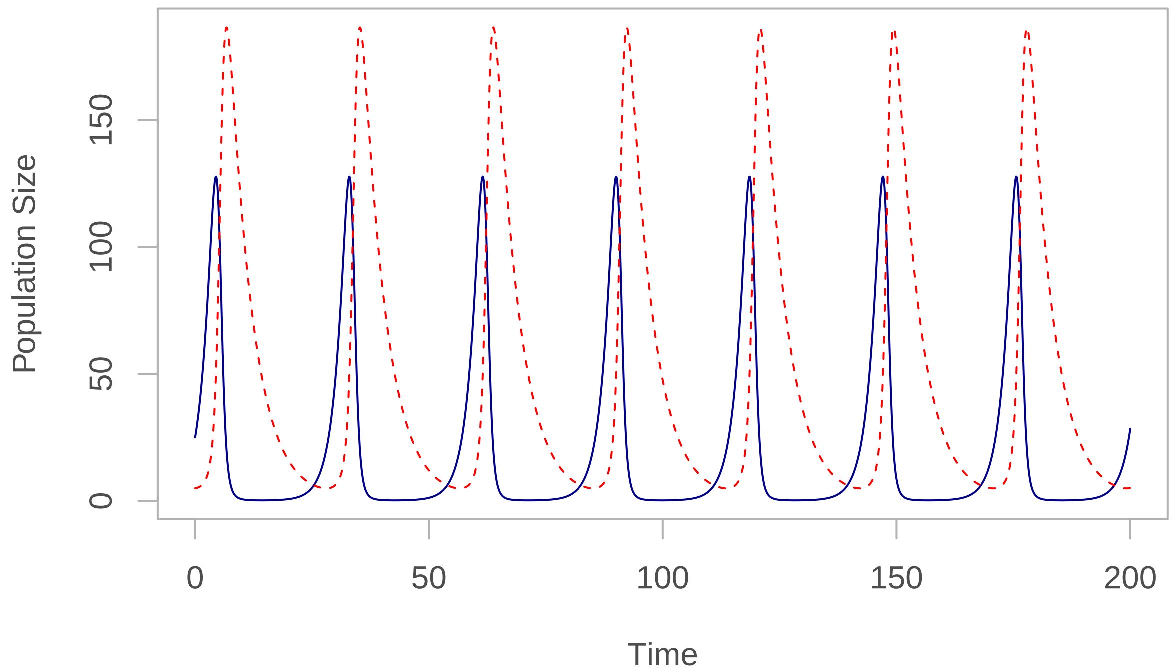


Lotka-Volterra Predation Equations (Coupled Differential Equations)

$$\frac{dN}{dt} = rN - aPN$$
$$\frac{dP}{dt} = -bP + fPN$$

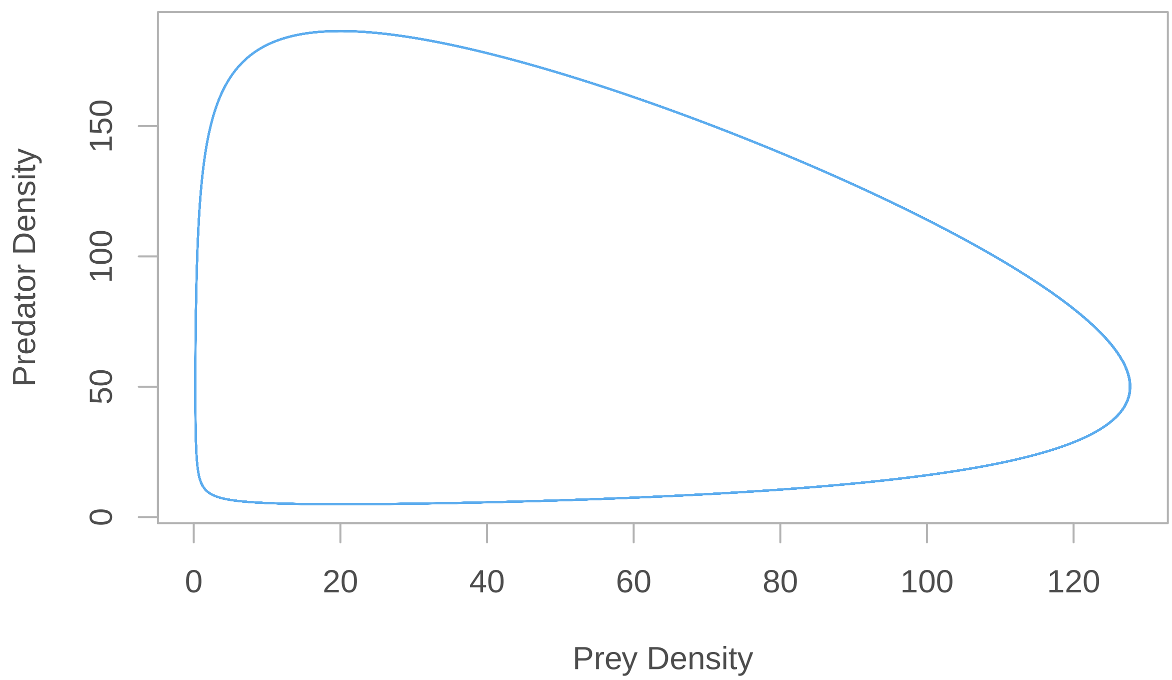
I use the `with` and `as.list` as short hands to not have to unbox/deconstruct variables

```
predpreyLV <- function(t, y, p) {
 with(c(as.list(y), as.list(p)), {
   dNdt <- r * N - a * P * N
   dPdt <- -b * P + f * P * N
   return(list(c(dNdt, dPdt)))
 })
}

r <- 0.50
a <- 0.01
f <- 0.01
b <- 0.20

p      <- c(r=r, a=a, b=b, f=f)
y0     <- c(N=25, P=5)
times  <- seq(0, 200, 0.1)
LV.out <- ode(y = y0, times, predpreyLV, p)

matplot(LV.out[,1], (LV.out[,2:3]), type="l", xlab="Time",
        ylab="Population Size",fg="grey70", col=c("darkblue", "red"),
        col.axis="grey30", col.lab="grey30", col.main="grey30")
```



In phase space it would be

```
plot(LV.out[,2], LV.out[,3], type="l", xlab="Prey Density",
     ylab="Predator Density", fg="grey70", col="steelblue2",
     col.axis="grey30", col.lab="grey30", col.main="grey30")
```



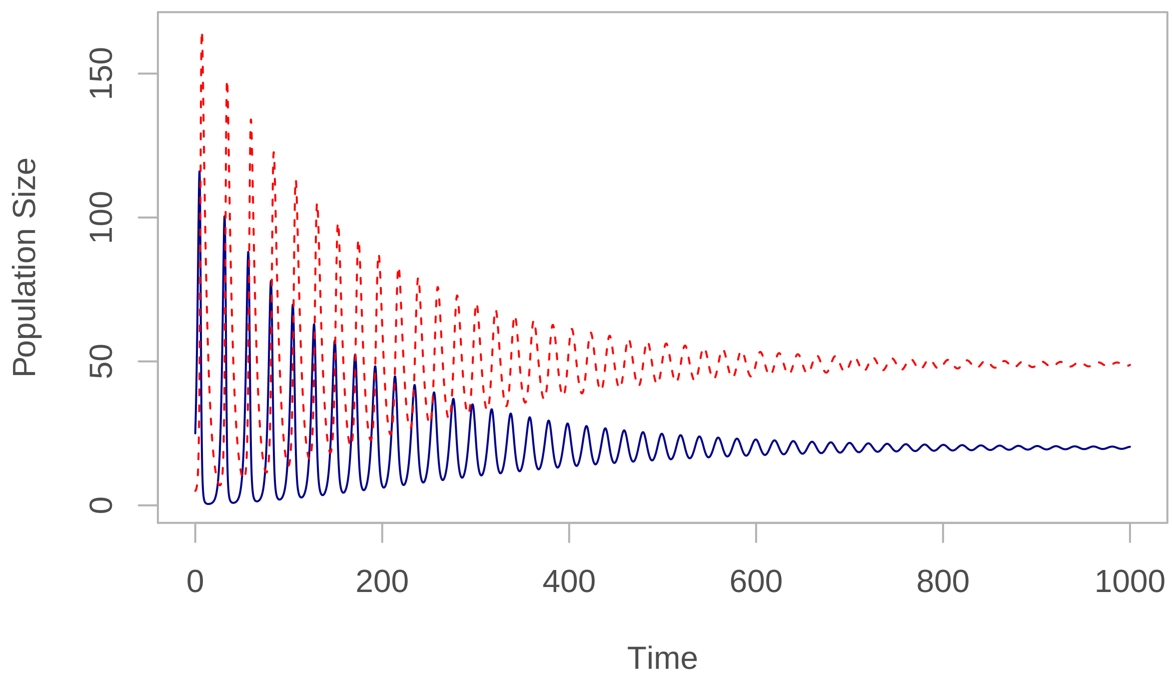
Making it more realistic by adding a carrying capacity to prey density

$$\frac{dN}{dt} = rN[1 - (\frac{N}{K})] - aPN$$

$$\frac{dP}{dt} = -bP + fPN$$

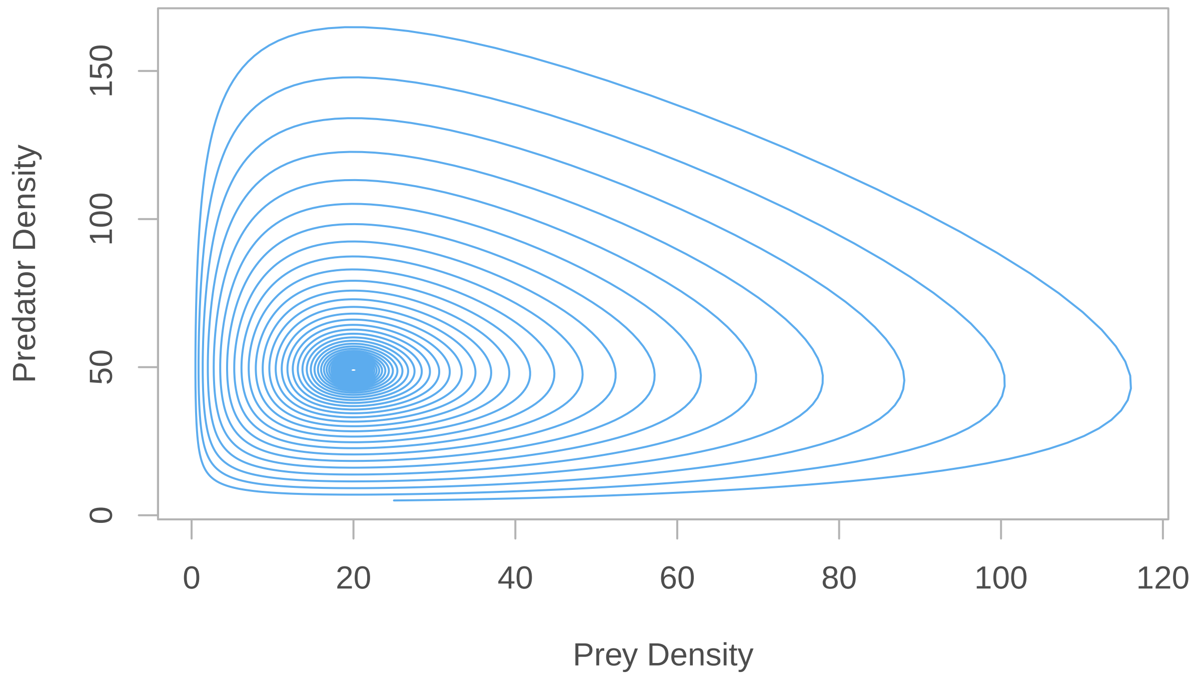
Numerically in R that would be

```
predpreyCarryingLV <- function(t, y, p) {
 with(c(as.list(y), as.list(p)), {
   dNdt <- r * N * (1 - (N / K)) - a * P * N
   dPdt <- -b * P + f * P * N
   return(list(c(dNdt, dPdt)))
 })
}

r <- 0.50
a <- 0.01
f <- 0.01
b <- 0.20
K <- 1000

p      <- c(r=r, a=a, b=b, f=f, K=K)
y0     <- c(N=25, P=5)
times  <- seq(0, 1000, 0.1)
LV.out <- ode(y = y0, times, predpreyCarryingLV, p)

matplot(LV.out[,1], (LV.out[,2:3]), type="l", xlab="Time",
        ylab="Population Size",fg="grey70", col=c("darkblue", "red"),
        col.axis="grey30", col.lab="grey30", col.main="grey30")
```



As a phase diagram it would be

```
plot(LV.out[,2], LV.out[,3], type="l", xlab="Prey Density",
     ylab="Predator Density", fg="grey70", col="steelblue2",
     col.axis="grey30", col.lab="grey30", col.main="grey30")
```


Predator Density
0
50
100
150
0
20
40
60
80
100
120
Prey Density