Chaos in Population Difference equations

Produced by following and tweaking the steps from:

R LABS Difference Equations in R Part 1 Representing Basic Population Dynamics

https://youtu.be/hWHtRPywNnQ

R LABS Difference Equations in R Part 2: Deterministic Chaos and Bifurcation Diagrams

https://youtu.be/KYyS74rb0Hk

We can create a Exponential Growth differenece function using the equation

$$N_{t+1} = \lambda N_t$$

Where N_{t+1} is the new population size, N_t is the old population size, and λ is the per capita birth and death rate.

The resulting R function to solve this numerically would be

```
PopGrowth <- function(lambda, N0, generations) {
  N <- c(N0, numeric(generations - 1))
  for (t in 1:(generations - 1)) {
    N[t + 1] <- lambda * N[t]
  }
  return(N)
}
```

An example function usage is

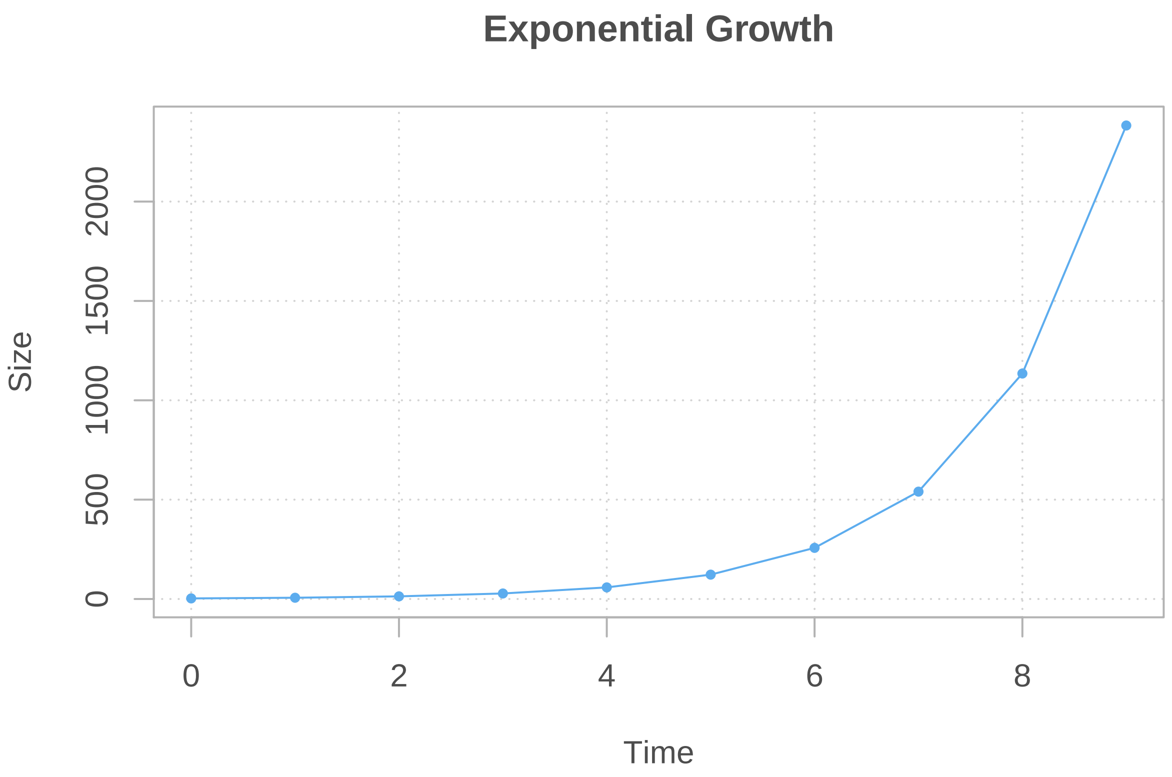
```
lambda      <-  2.1
N0          <-  3.0
generations <- 10.0

Output <- PopGrowth(lambda, N0, generations)
Output
```

```
##  [1]    3.0000    6.3000   13.2300   27.7830   58.3443  122.5230  257.2984
##  [8]  540.3266 1134.6858 2382.8401
```

Plotting the Output would be

```
plot(0:(generations - 1), Output, type='o', xlab="Time", ylab="Size",
     main="Exponential Growth", pch=16, cex=0.7, fg="grey70", col="steelblue2",
     col.axis="grey30", col.lab="grey30", col.main="grey30", panel.first=grid())
```



The explicit solution is obtained by looking at multiple iterations of the difference equation

$$N_1 = \lambda N_0 N_2 = \lambda N_1 = \lambda(\lambda N_0) = \lambda^2 N_0 N_3 = \lambda N_2 = \lambda(\lambda^2 N_0) = \lambda^2 N_0 \ldots \therefore N_t = \lambda^t N_0$$

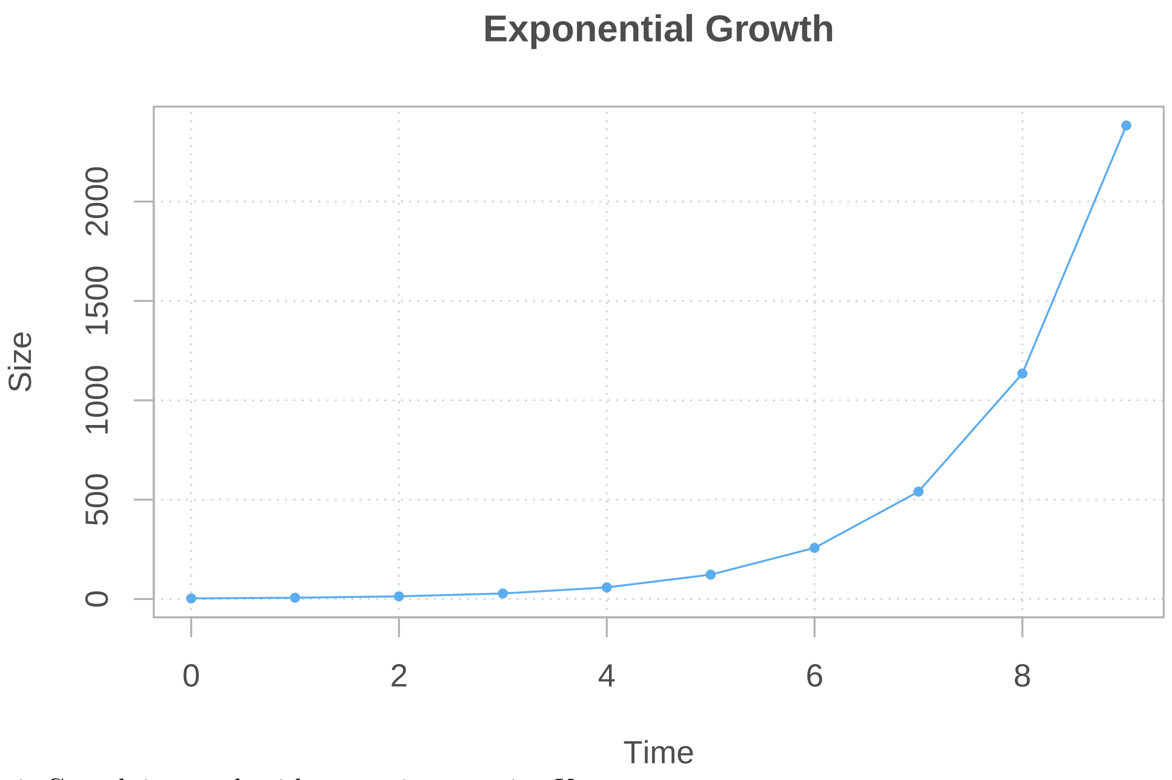
The R code becomes

```
lambda <- 2.1
t      <- 0:9
N0     <- 3
Nt     <- (lambda^t) * N0
Nt
```

```
##  [1]    3.0000    6.3000   13.2300   27.7830   58.3443  122.5230  257.2984
##  [8]  540.3266 1134.6858 2382.8401
```

The plot looks the same

```
plot(t, Nt, type='o', xlab="Time", ylab="Size", main="Exponential Growth",
     pch=16, cex=0.7, fg="grey70", col="steelblue2", col.axis="grey30",
     col.lab="grey30", col.main="grey30", panel.first=grid())
```



Logistic Growth is growth with a carrying capacity K

$$N_{t+1} = N_t + rN_t(1 - [N_t/K])$$

Solving numerically using a function would be

```
dlogistic <- function(K, r, N0, generations) {
  N <- c(N0, numeric(generations -1))
  for (t in 1:(generations-1)) {
    N[t + 1] <- N[t] + r * N[t] * (1 - (N[t]/K))
  }
  return(N)
}
```

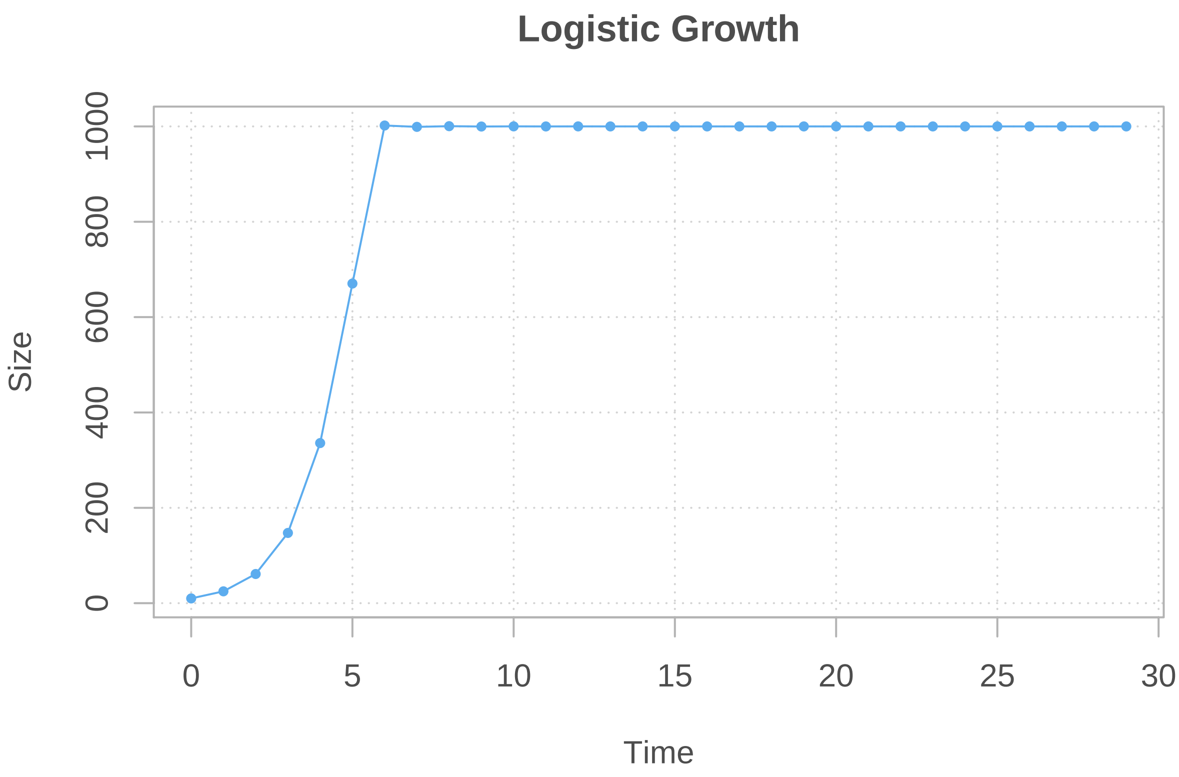
An example function usage is

```
K           <- 1000.0
r           <-    1.5
N0          <-   10.0
generations <-   30.0
Output <- dlogistic(K, r, N0, generations)
Output
```

```
##  [1]   10.00000   24.85000   61.19872  147.37887  335.86637  670.45660
##  [7] 1001.87342  999.05803 1000.46966  999.76484 1000.11750  999.94123
## [13] 1000.02938  999.98531 1000.00735  999.99633 1000.00184  999.99908
## [19] 1000.00046  999.99977 1000.00011  999.99994 1000.00003  999.99999
## [25] 1000.00001 1000.00000 1000.00000 1000.00000 1000.00000 1000.00000
```

Plotting the Output would be

```
plot(0:(generations - 1), Output, type='o', xlab="Time", ylab="Size", main="Logistic Growth",
     pch=16, cex=0.7, fg="grey70", col="steelblue2", col.axis="grey30", col.lab="grey30",
     col.main="grey30", panel.first=grid())
```



The canocial form of the discrete-time logistic is

$$X_{t+1} = rX_t(1 - X_t)$$

Where X_{t+1} is the ratio of existing population to maximum possible population, X_t is the old ratio of existing population to maximum possible population, and r is the (per capita birth rate) - (per capita death rate)

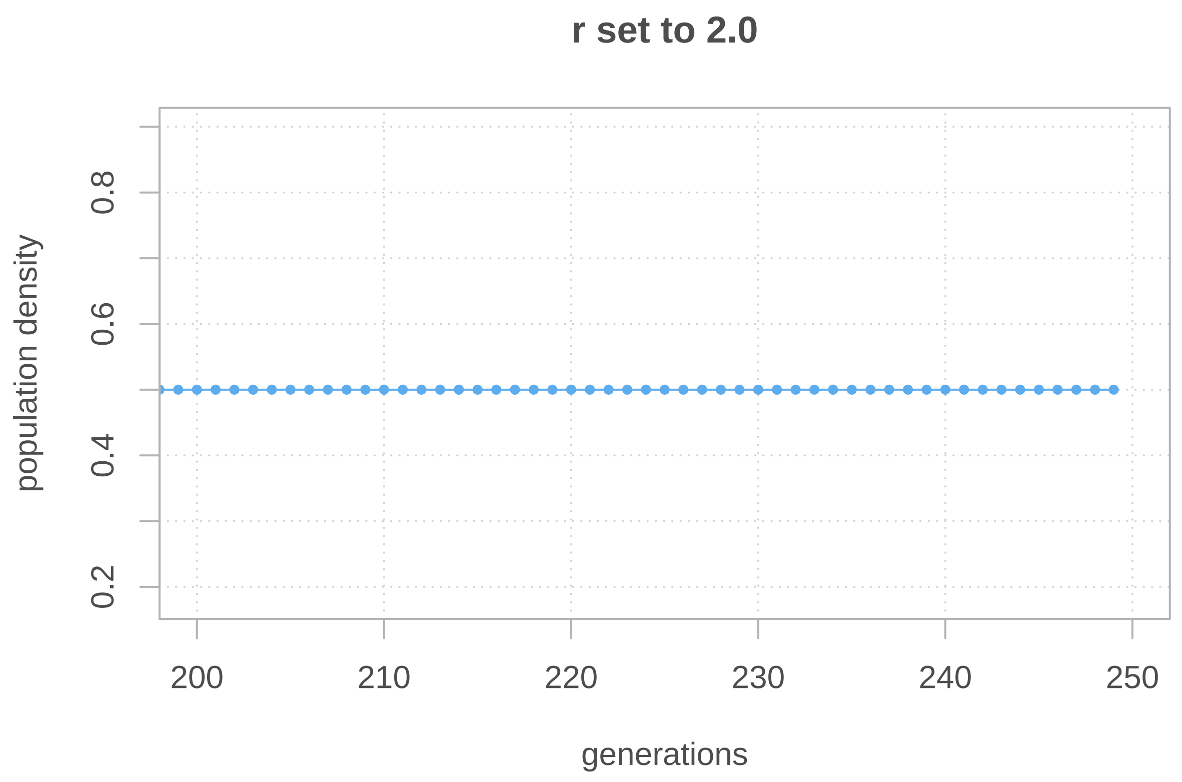
Written as a function in R it is

```
canlogistic <- function(r, X0, generations) {
  X <- c(X0, numeric(generations - 1))
  for (t in 1:(generations - 1)) {
    X[t + 1] <- r * X[t] * (1 - X[t])
  }
  return(X)
}
```

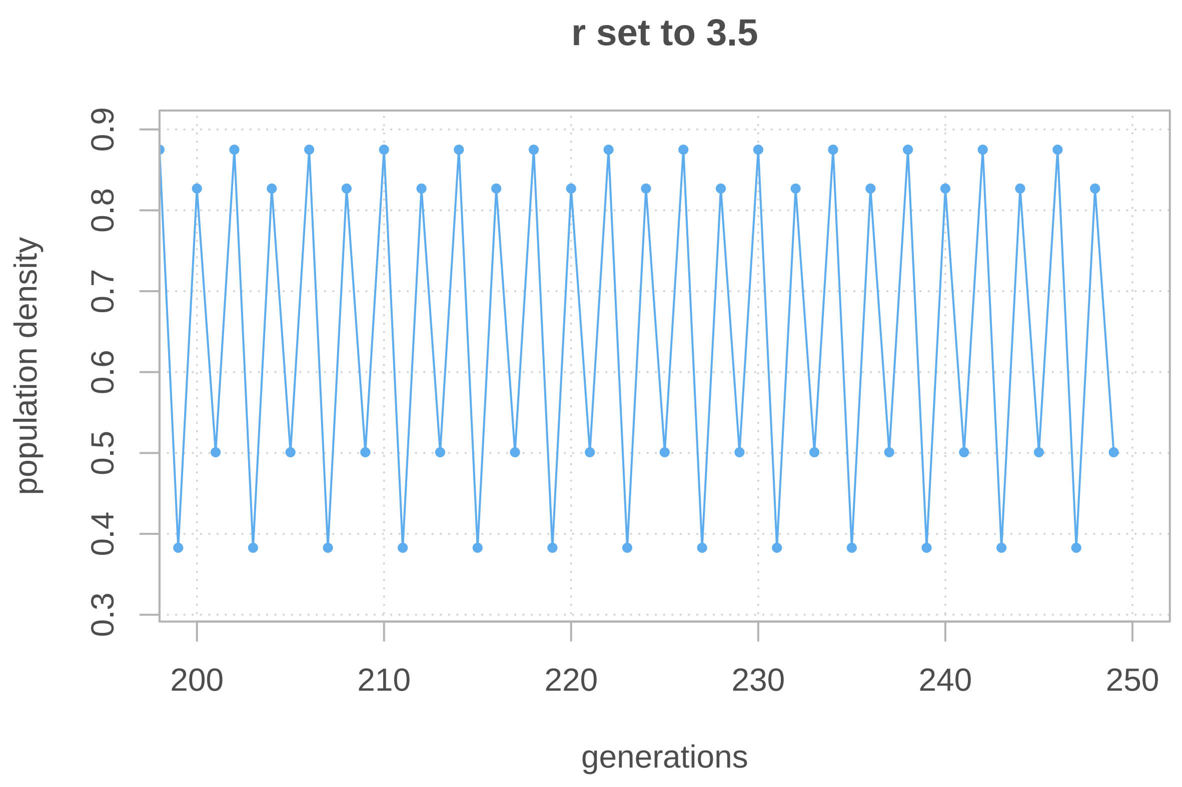
With three different r values, we can see it with a 1 point cycle, a 2 point cycles, and then finally a chaotic regime

```
generations <- 250
Xt.1 <- canlogistic(r = 2.0, X0 = 0.9, generations)
Xt.2 <- canlogistic(r = 3.5, X0 = 0.9, generations)
Xt.3 <- canlogistic(r = 3.9, X0 = 0.9, generations)
```

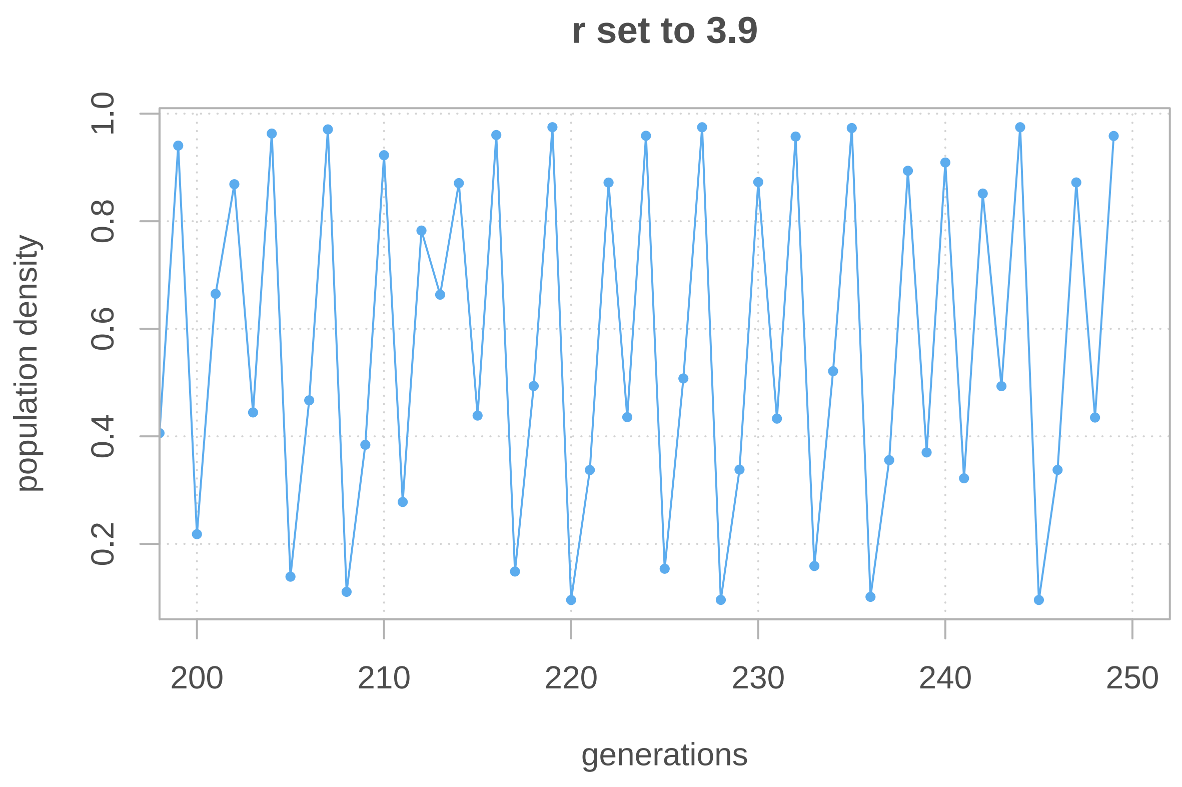
```
plot(0:(generations - 1), Xt.1, type='o', xlim=c(200,250), xlab="generations",
     ylab="population density", main="r set to 2.0", pch=16, cex=0.7, fg="grey70",
     col="steelblue2", col.axis="grey30", col.lab="grey30", col.main="grey30",
     panel.first=grid())
```



```
plot(
  0:(generations - 1), Xt.2, type='o', xlim=c(200,250), xlab="generations",
  ylab="population density", main="r set to 3.5", pch=16, cex=0.7, fg="grey70",
  col="steelblue2", col.axis="grey30", col.lab="grey30", col.main="grey30",
  panel.first=grid())
```



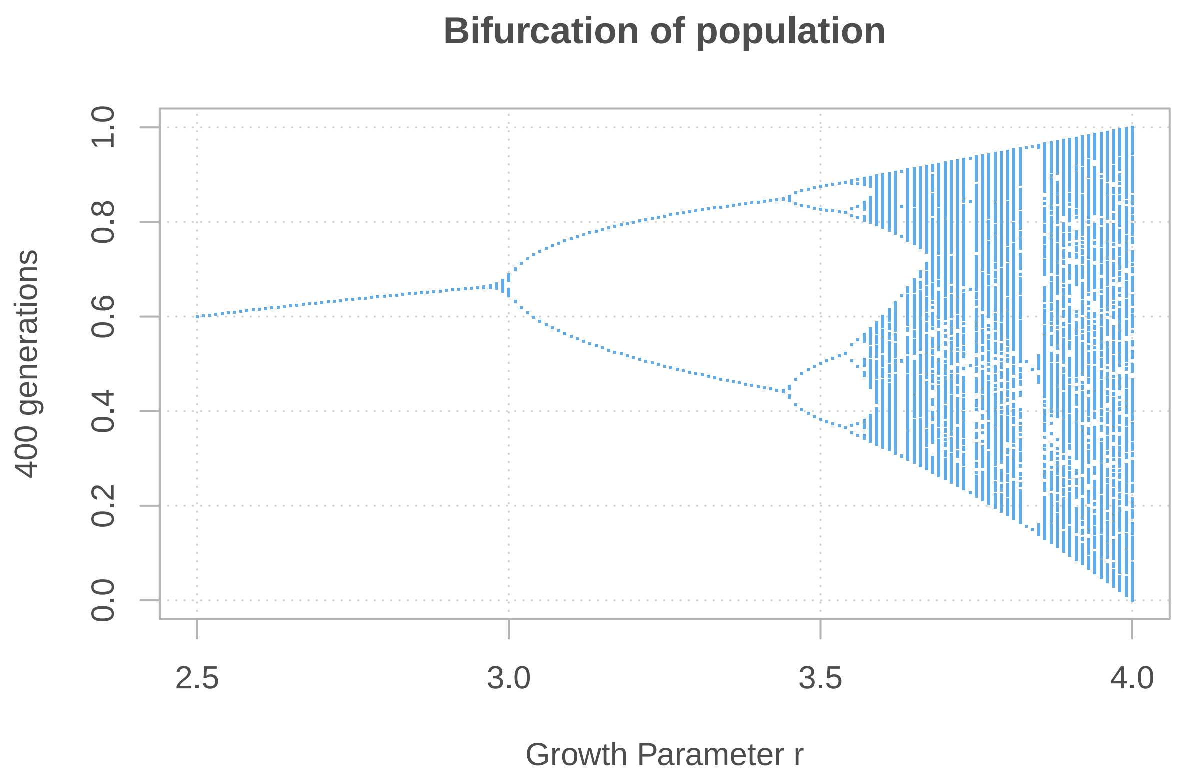
```r
plot(
  0:(generations - 1), Xt.3, type = 'o', xlim=c(200,250), xlab="generations",
  ylab="population density", main="r set to 3.9", pch=16, cex=0.7, fg="grey70",
  col="steelblue2", col.axis="grey30", col.lab="grey30", col.main="grey30",
  panel.first=grid())
```



```
minr        <- 2.5
maxr        <- 4.0
inc         <- 0.01
rd          <- seq(minr, maxr, inc)
ttransients <- 100
trange      <- 400

plot(c(minr, maxr), c(0,1), type="n", pch=".", xlab="Growth Parameter r",
  ylab="400 generations", main="Bifurcation of population", fg="grey70",
  col="steelblue2", col.axis="grey30", col.lab="grey30", col.main="grey30",
  panel.first=grid())

for(r in rd) {
  x <- 0.1
  for(i in 1:ttransients) {
    x <- r * x * (1 - x)
  }
  for(i in 1:trange) {
    x <- r * x * (1 - x)
    points(r, x, pch =".", col="steelblue2")
  }
}
```


Bifurcation of population
400 generations
Growth Parameter r