Physics 215 (1st Semester AY 2022-2023)

Richelle Jade L. Tuquero

Session 0.1: Julia Try-out

Generate plots of the result of a logistic map

$$x_{t+1} = r * x_t * (1 - x_t)$$

over different r-values starting from a random initial seed value x_0. You can use the `rand()` function for this one.

Submit the Jupyter notebook as your work containing the codes, plots and analysis (using the Markdown cells).

1. Investigate how the behavior changes based on the known properties of a Logistic Map.
2. Generate relevant plots as you see fit for analyzing your program.
3. Try using functions to make your code neat and reusable.
4. Submit the Jupyter notebook with the following filename: SURNAME-FIRST_NAMES-Session-0_1-Julia-Try-Out.ipynb replacing [SURNAME] and [FIRST_NAMES] appropriately.

In [1]:
```
using Pkg
Pkg.activate(".")
```

```
  Activating project at `~/Desktop/Physics 215/Submission/Session 0.1`
```

In [2]:
```
# Adding packages
Pkg.add("Plots")
Pkg.update()

# Importing packages
using Plots
```

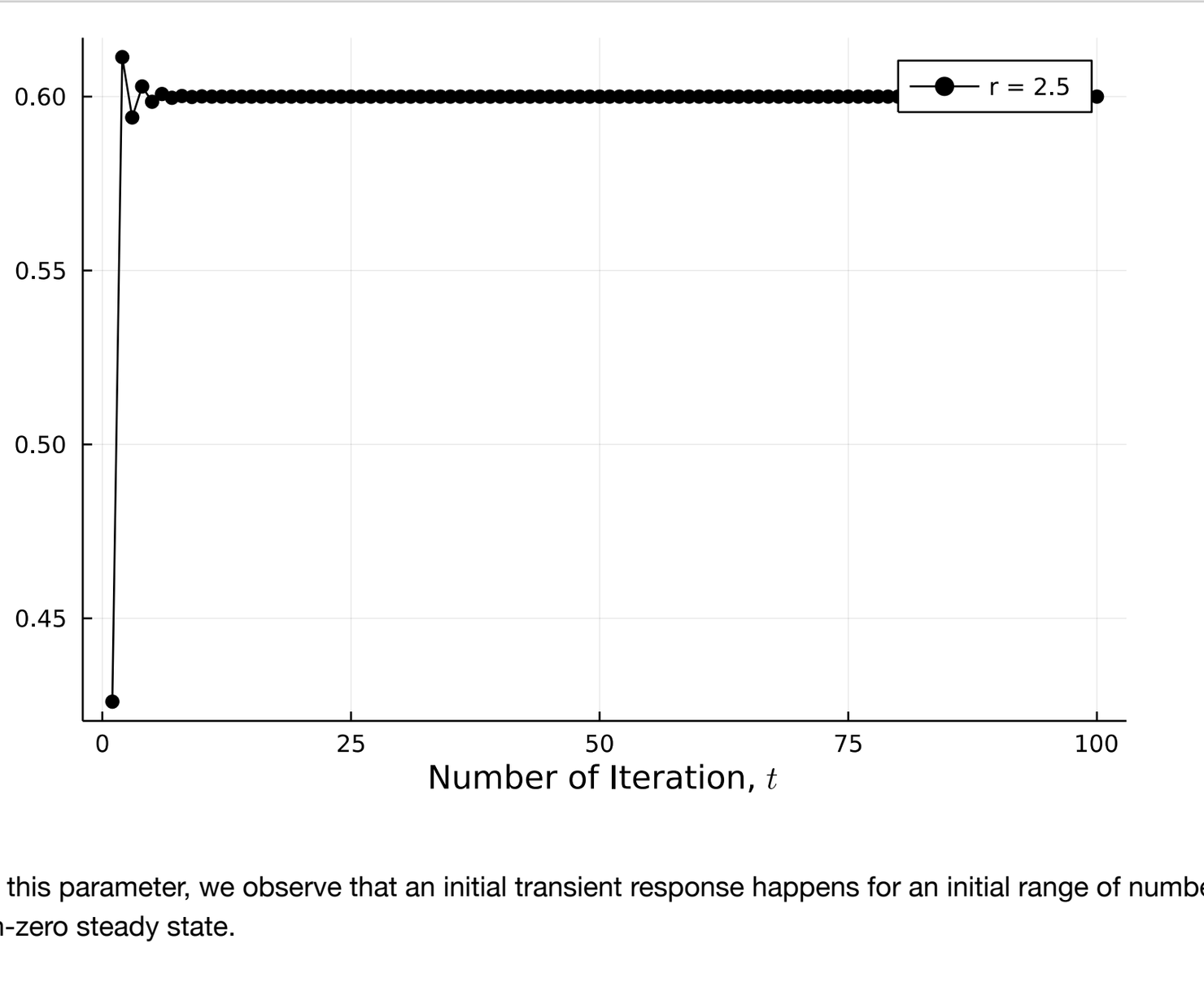
```
    Updating registry at `~/.julia/registries/General.toml`
   Resolving package versions...
  No Changes to `~/Desktop/Physics 215/Submission/Session 0.1/Project.toml`
  No Changes to `~/Desktop/Physics 215/Submission/Session 0.1/Manifest.toml`
    Updating registry at `~/.julia/registries/General.toml`
  No Changes to `~/Desktop/Physics 215/Submission/Session 0.1/Project.toml`
  No Changes to `~/Desktop/Physics 215/Submission/Session 0.1/Manifest.toml`
```

Initial Testing for a single value of the parameter r

We first try to investigate the x-values depending on a value of an arbitrary value of r. Let $r = 2.5$ with a random initial seed x_0 using `rand()` for total $N = 100$ iteration.

In [3]:
```
N = 100      # Number of iteration
r = 2.5      # Parameter
X = zeros(N) # For the recorded value of x after every iteration.

x = rand()   # Initial value of x
x_old = x
X[1] = x

for n in range(2,N)
    x_new = r*x_old*(1-x_old) # Equation of the logistic map
    X[n] = x_new
    x_old = x_new             # Update the value of x_old or x_t
end

# Plotting of the x-values for each iterartion
plot(X, markershape =:circle, markercolor =:black,
    linecolor =:black, label ="r = "*string(r),
    xlabel = "Number of Iteration, \$t\$", ylabel= "\$x_{t}\$")
```

Out[3]:



For this parameter, we observe that an initial transient response happens for an initial range of number of iteration. Afterwards, it then reach equilibrium or a non-zero steady state.

Implementing a function

Before we begin solving for the result of the logistic map for varying r, we first check if the function we want to create is already present. We want to refer to our function as `logistic` which will solve the equation $x_{t+1} = r * x_t * (1 - x_t)$.

In [4]:
```
? logistic
```

```
search:

Couldn't find logistic
Perhaps you meant Registry
```

Out[4]: No documentation found.

Binding `logistic` does not exist.

Since there is no function called `logistic`, we may now define our function for solving the logistic map. To record the x-values for total N iterations, we will be storing its values in a one-dimensional array or vector using `zeros(N)`. Since the logistic map equation updates itself based on its previous value, we will use two variables namely; `x_old` and `x_new` for the old and new values of x.

In [5]:
```
"""
    logistic( r, x, N )
Computes and returns the result of the logistic map x_{t+1} = r * x_t * (1 - x_t) for N iteration.
- Input: `r` for the parameter.
         `x0` for the initial value of x.
         `N` for the total number of iteration.
- Output: `X_t::Vector`
"""

function logistic( r, x0, N )
    X_t = zeros(N)                  # For the recorded value of x after every iteration.
    x_old = x0
    X_t[1] = x0
    for n in range(2,N)
        x_new = r*x_old*(1-x_old) # Equation of the logistic map
        X_t[n] = x_new            # Saving the value for index n in the vector.
        x_old = x_new             # Updating value of x_old or x_t in the equation.
    end
    return X_t                      # Returns the x-value of each iteration.
end
```

Out[5]: `logistic (generic function with 1 method)`

In [6]:
```
? logistic
```

```
search: logistic
```

Out[6]: No documentation found.

`logistic` is a `Function`.

```
# 1 method for generic function "logistic":
[1] logistic(r, x0, N) in Main at In[5]:10
```

Since we will plot the values of x for different r, we create a function `plotLogistic( r, x, N )` to plot the one-dimensional array generated by the function `logistic( r, x, N )`. But before that, we first check if there is a function called `plotLogistic( r, x, N )`.

In [7]:
```
? plotLogistic
```

```
search:

Couldn't find plotLogistic
Perhaps you meant logistic
```

Out[7]: No documentation found.

Binding `plotLogistic` does not exist.

Since there is no defined `plotLogistic` function, we can use it for our self-defined function.

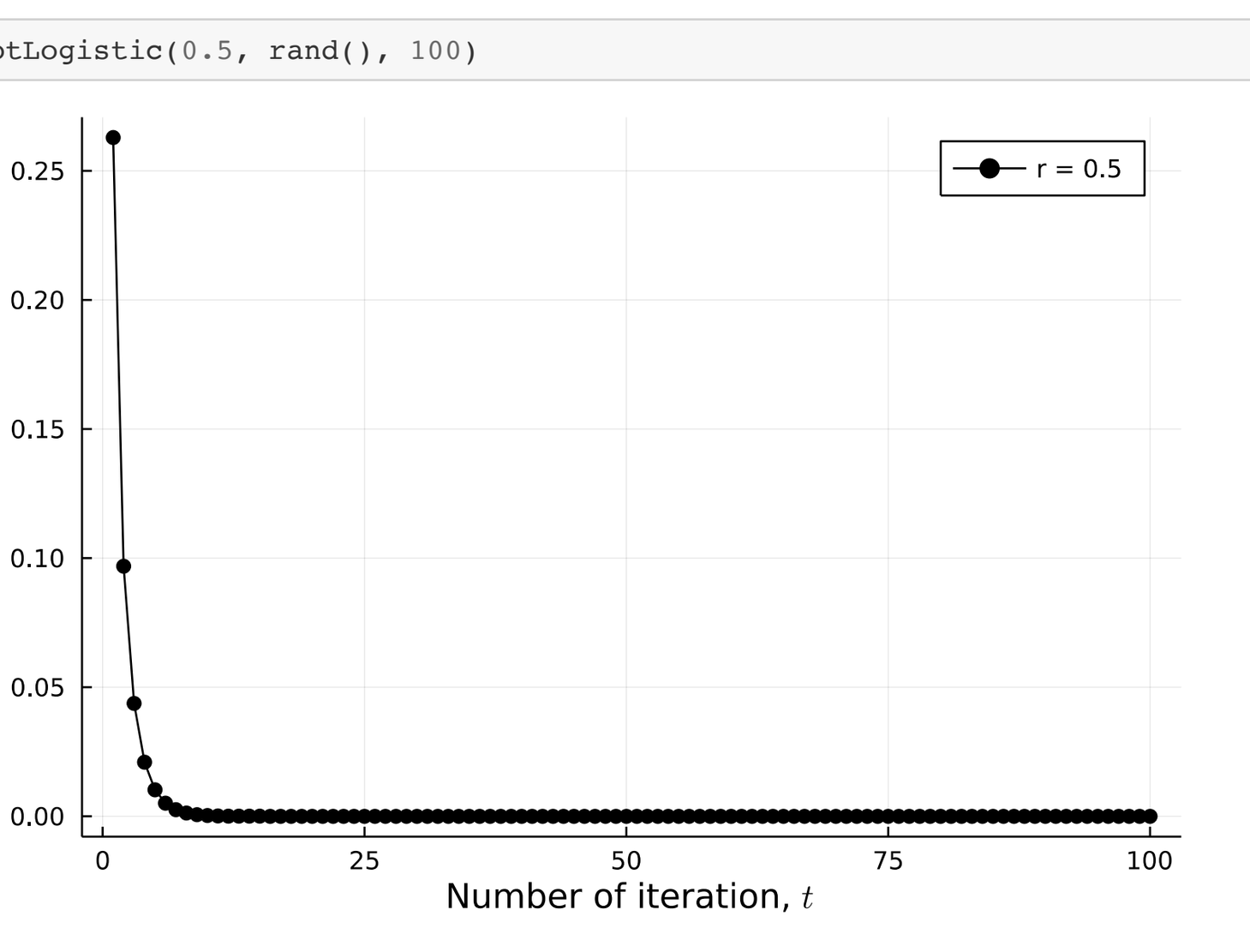
In [8]:
```
"""
    plotLogistic( r, x, N )
This plots the x-values of the logistic map for N iteration with black circles connected by a black line.
"""

function plotLogistic( r, x0, N )
    xt = logistic( r, x0, N )
    return plot(xt, markershape =:circle, markercolor =:black,
                linecolor =:black, label ="r = "*string(r),
                xlabel = "Number of iteration, \$t\$", ylabel= "\$x_{t}\$")
end
```

Out[8]: `plotLogistic (generic function with 1 method)`

We now have our needed functions, we will now investigate the evolution of x_t for different r-values. The r-values that we will use are $r = 0.5, 1.0, 2.0, 2.5, 3.0, 3.5, 4.0$ for 100 iterations. We will observe its behavior using the `plotLogistic` function.

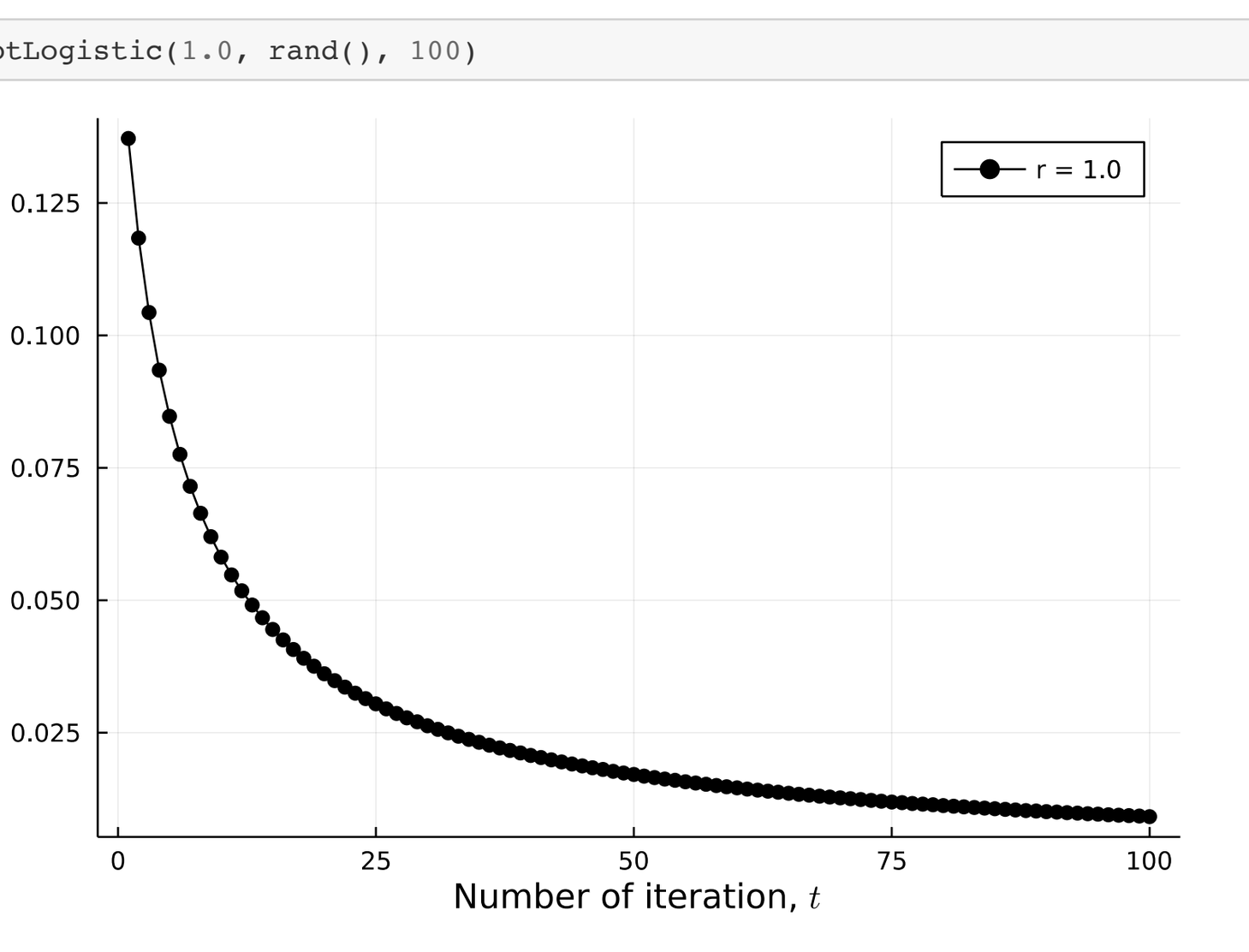
In [9]:
```
plotLogistic(0.5, rand(), 100)
```

Out[9]:



In [10]:
```
plotLogistic(1.0, rand(), 100)
```

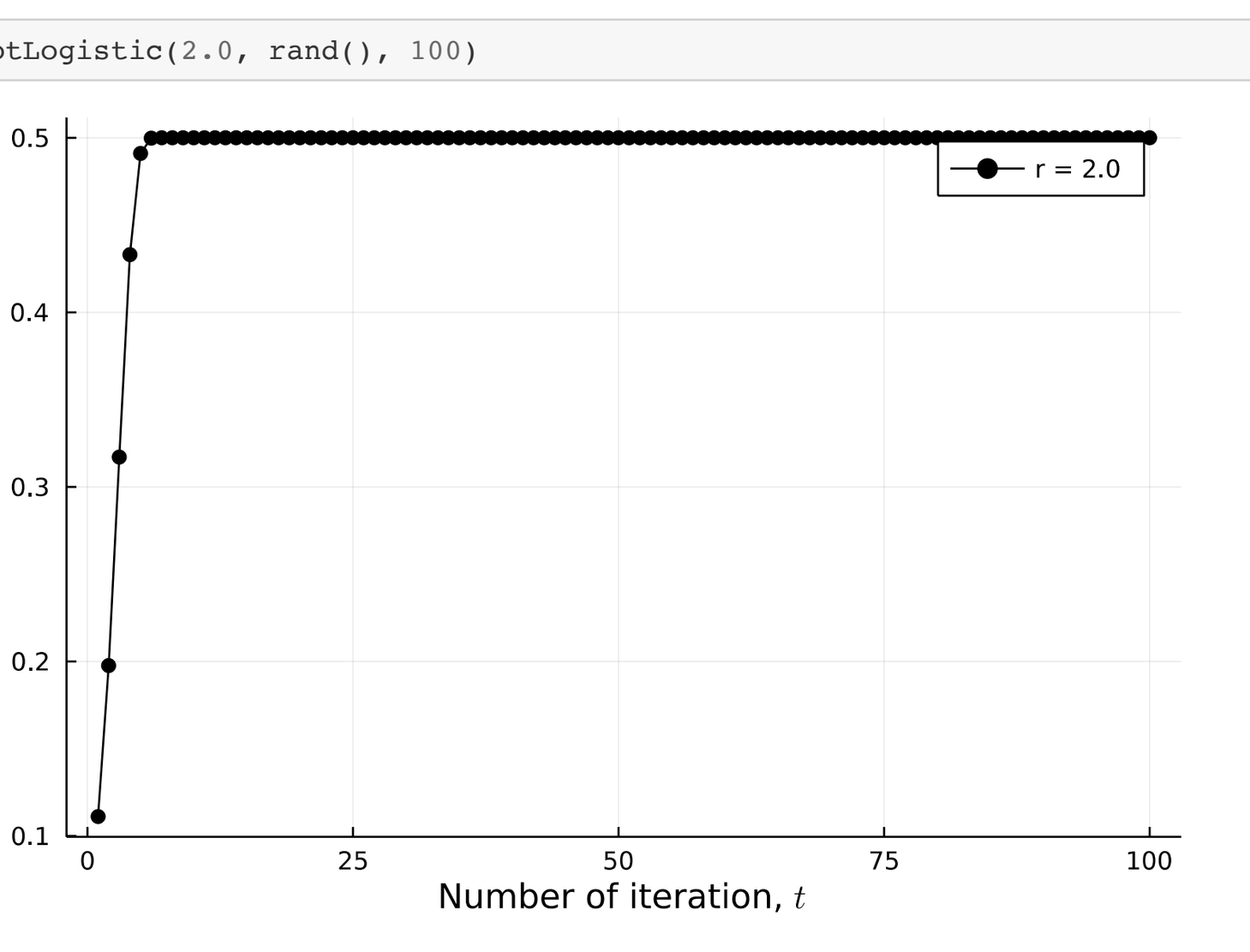
Out[10]:



For the $r = 0.5$ and $r = 1.0$, we observe that x_t decreases until it reaches and becomes 0.

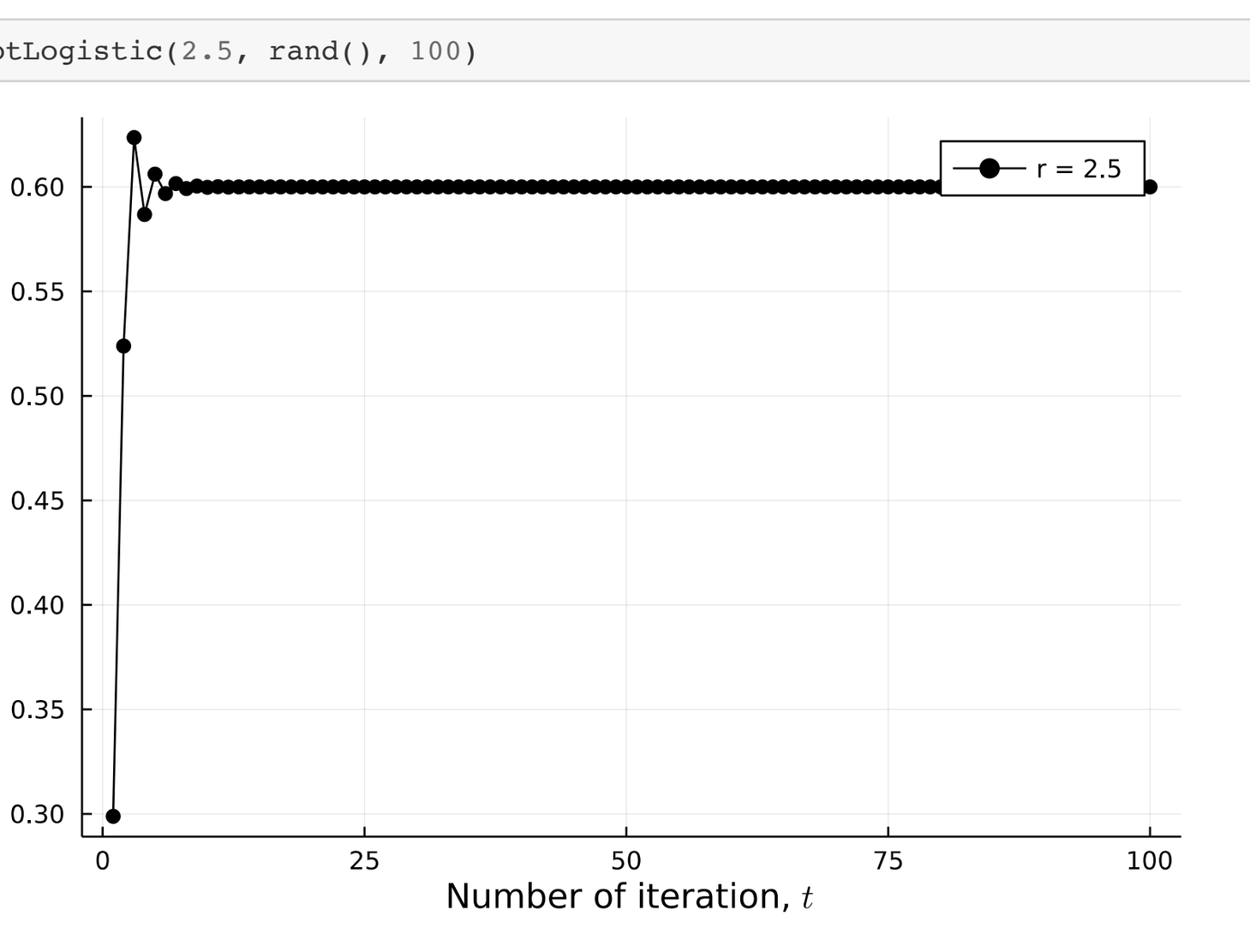
In [11]:
```
plotLogistic(2.0, rand(), 100)
```

Out[11]:



In [12]:
```
plotLogistic(2.5, rand(), 100)
```

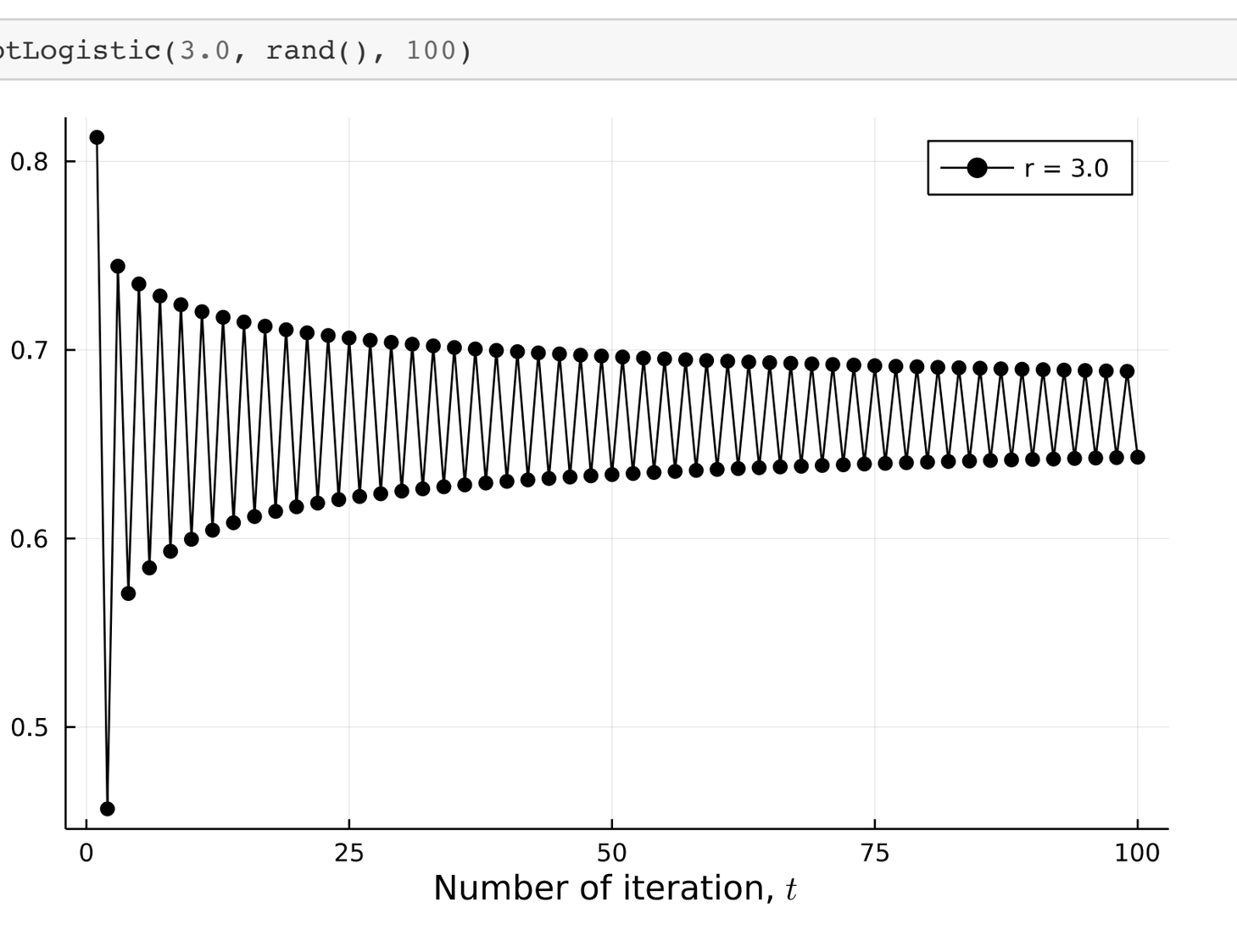
Out[12]:



For $r = 2.0$ and $r = 2.5$, we observe that x_t reaches a non-zero steady state. Moreover, the value of the steady state or equilibrium increases as r increases.

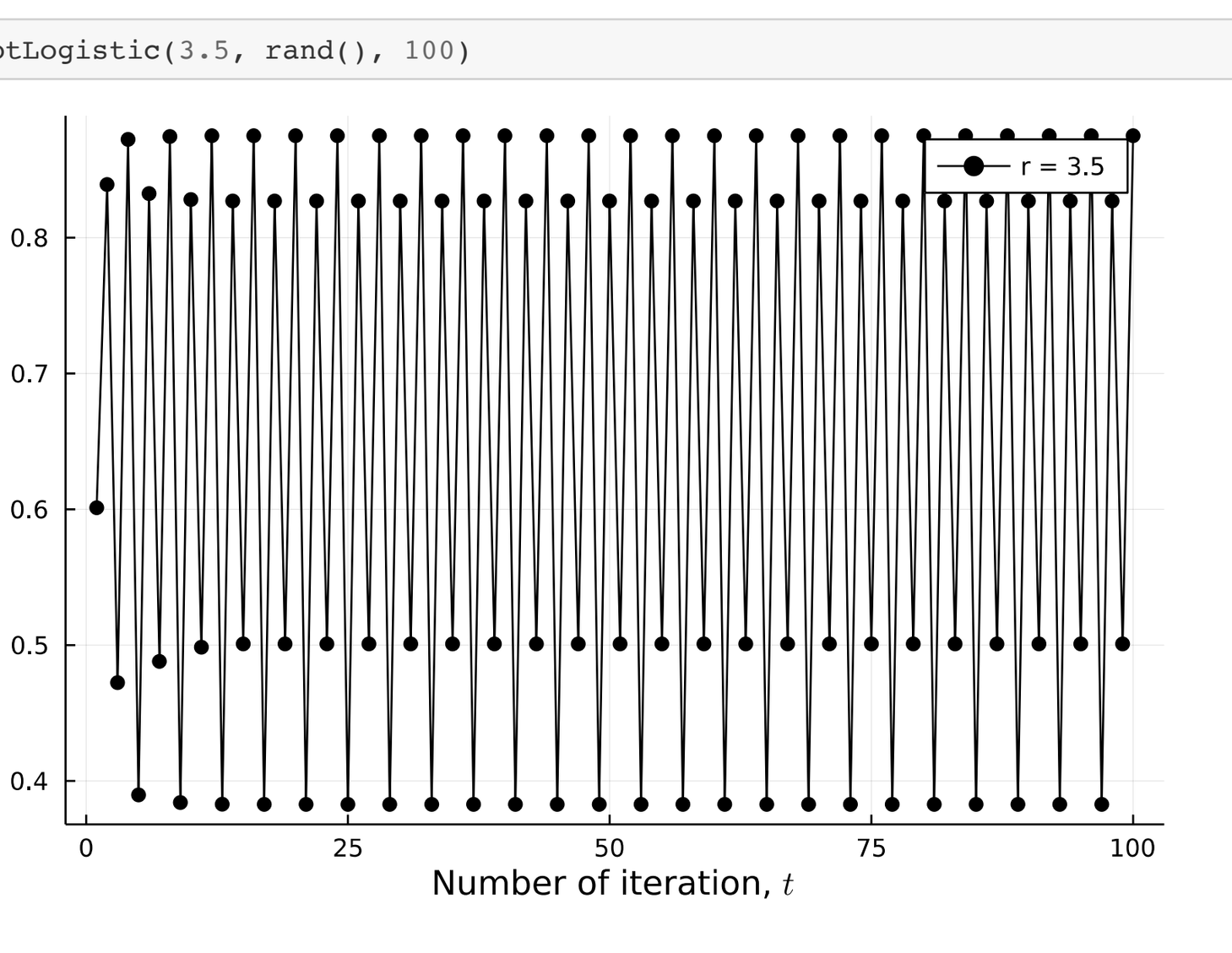
In [13]:
```
plotLogistic(3.0, rand(), 100)
```

Out[13]:



In [14]:
```
plotLogistic(3.5, rand(), 100)
```

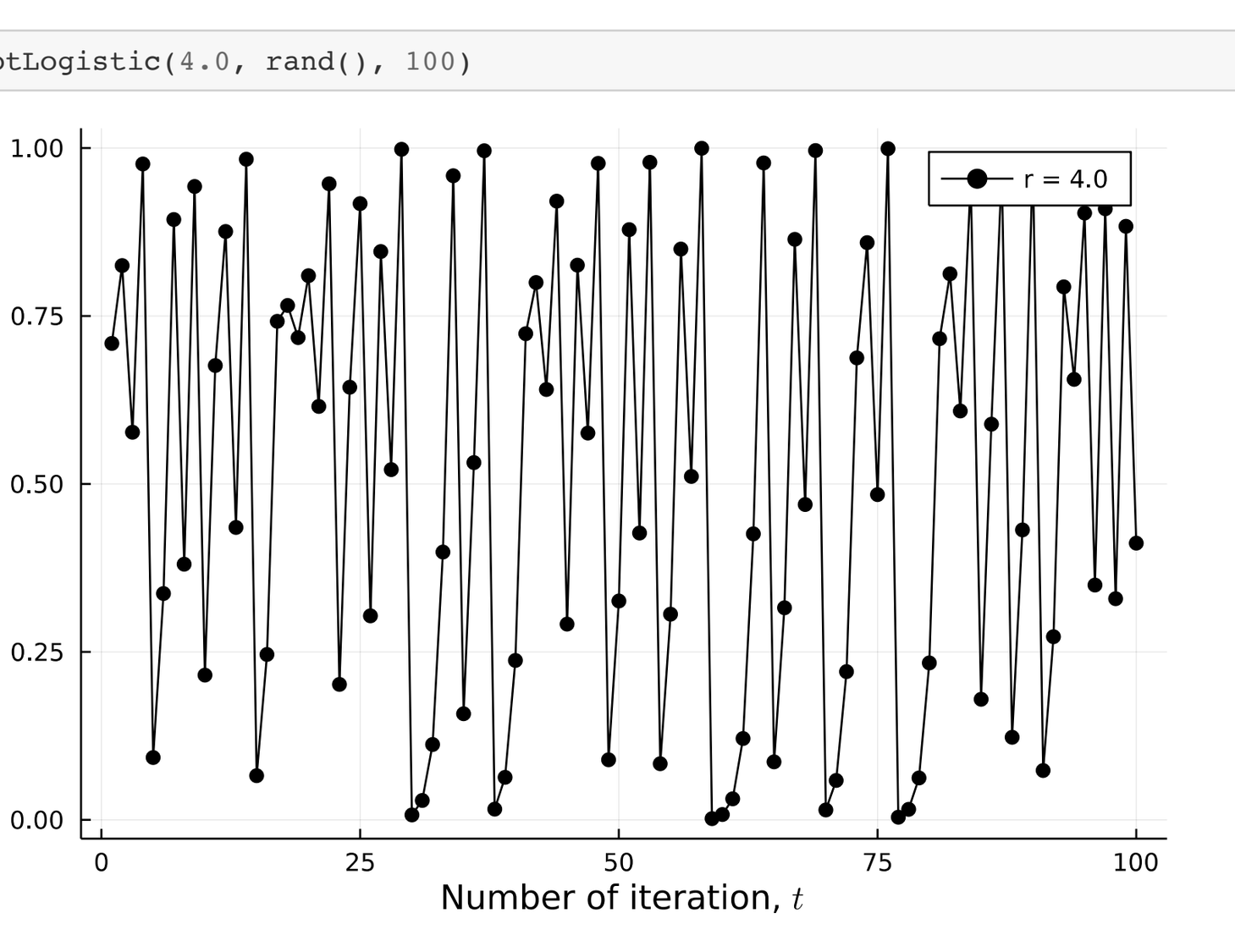
Out[14]:



Then for $r = 3.0$ and $r = 3.5$, we notice that a periodic cycle behavior is present instead of a non-zero steady state.

In [15]:
```
plotLogistic(4.0, rand(), 100)
```

Out[15]:



Lastly, testing for $r = 4.0$ shows that there is no periodic cycle behavior nor a steaady state for this parameter. Instead, we observe a chaotic behavior with random (pseudo-random) behavior. For a more specific discussion on how the behavior of the x-value changes with the parameter r, we also include the bifurcation diagram in the last subsection.

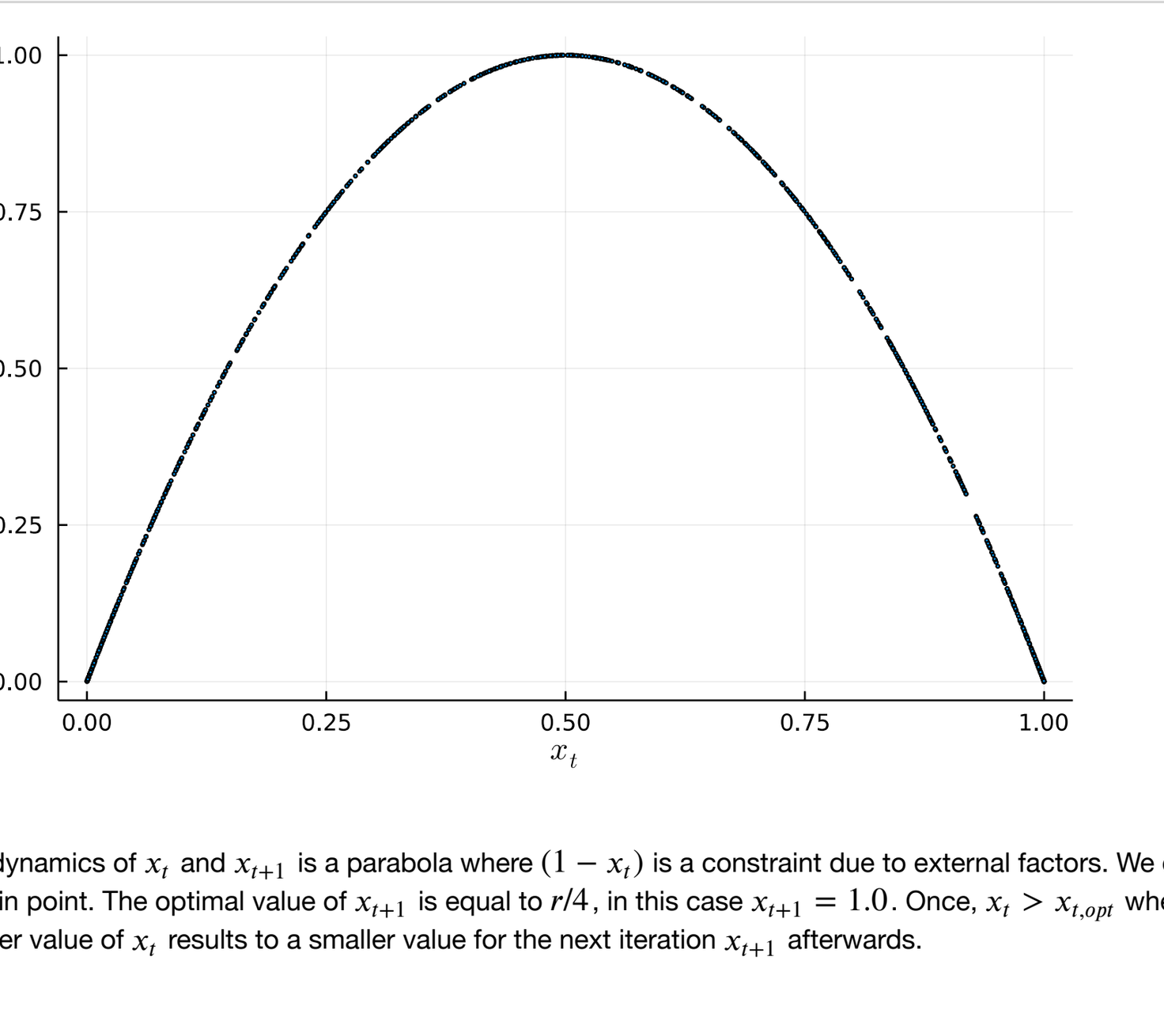
Relationship of x_t and x_{t+1}

Using the function `logistic`, we will now use the function to observe the behavior of x_{t+1} depending on the value of x_t. For this part, we will use $r = 4$.

In [16]:
```
N = 1_000        # Number of iteration
X = zeros(N-1)   # For the values of x_t
Y = zeros(N-1)   # For the values of x_{t+1}

r = 4            # Value of r in the logistic map
x = rand()       # Initial value of x or x_0

X_t = logistic(r, x, N)

# Records the values of x_t and x_{t+1} for plotting
X[1:end] = X_t[1:end-1]
Y[1:end] = X_t[2:end];
```

In [17]:
```
plot(scatter(X, Y, mode = "markers", markersize = 1),
    xlabel = "\$x_t\$", ylabel= "\$x_{t+1}\$", legend = false)
```

Out[17]:



The dynamics of x_t and x_{t+1} is a parabola where $(1 - x_t)$ is a constraint due to external factors. We observe that as x_t increases, x_{t+1} also increases up to a certain point. The optimal value of x_{t+1} is equal to $r/4$, in this case $x_{t+1} = 1.0$. Once, $x_t > x_{t,opt}$ where $x_{t,opt}$ is when $x_{t+1} = r/4$, x_{t+1} decreases. Hence, a greater value of x_t results to a smaller value for the next iteration x_{t+1} afterwards.

Bifurcation diagram

To get a better idea of how x changes depending on the r-values, we simulate the logistic map for different r-values ranging from 0 to 4 by only saving the last value of x. This will be defined in the function `bifurcation` after checking that there is no preexisting function with the same name.

In [18]:
```
? bifurcation
```

```
search:

Couldn't find bifurcation
Perhaps you meant @cfunction
```

Out[18]: No documentation found.

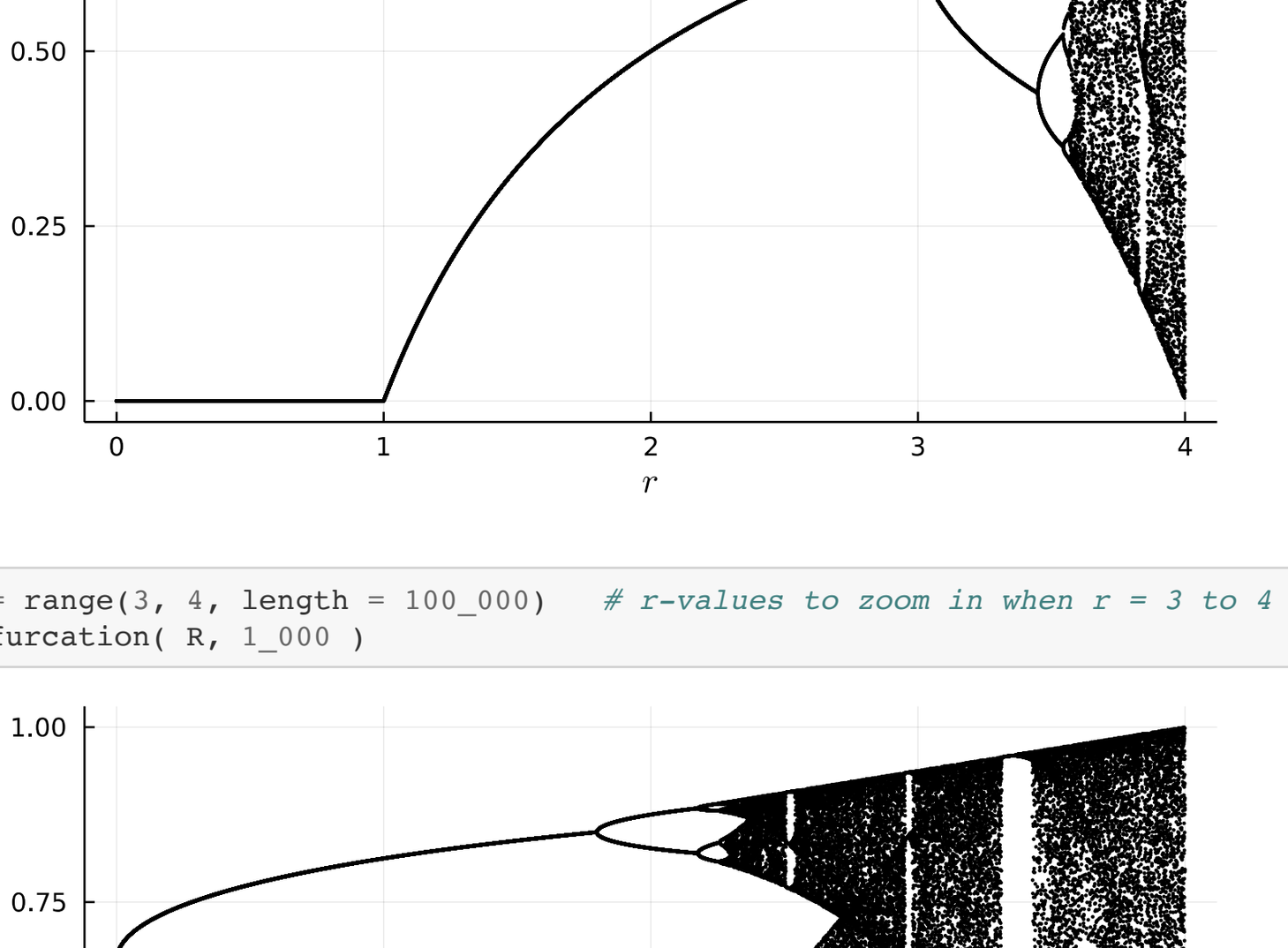
Binding `bifurcation` does not exist.

In [19]:
```
"""
    bifurcation( R, N )
Computes and returns the Nth result of the logistic map x_{t+1} = r * x_t * (1 - x_t)
for N iteration for a set value of R.
- Input: `R` for the range of values of parameter.
         `N` for the total number of iteration.
- Output: `X_r::Vector`
"""

function bifurcation( R, N )
    X_r = zeros(length(R))
    for i in range(1,length(R)) # For each value of r
        r = R[i]
        x = rand()
        x = logistic( r,x, N )
        X_r[i] = x[end]        # Records the final value of x for each r
    end

    plt = plot(scatter(R, X_r, mode = "markers", markersize = 0.5),
        xlabel = "\$r\$", ylabel= "\$x\$", legend = false)
    return plt
end
```

Out[19]: `bifurcation (generic function with 1 method)`

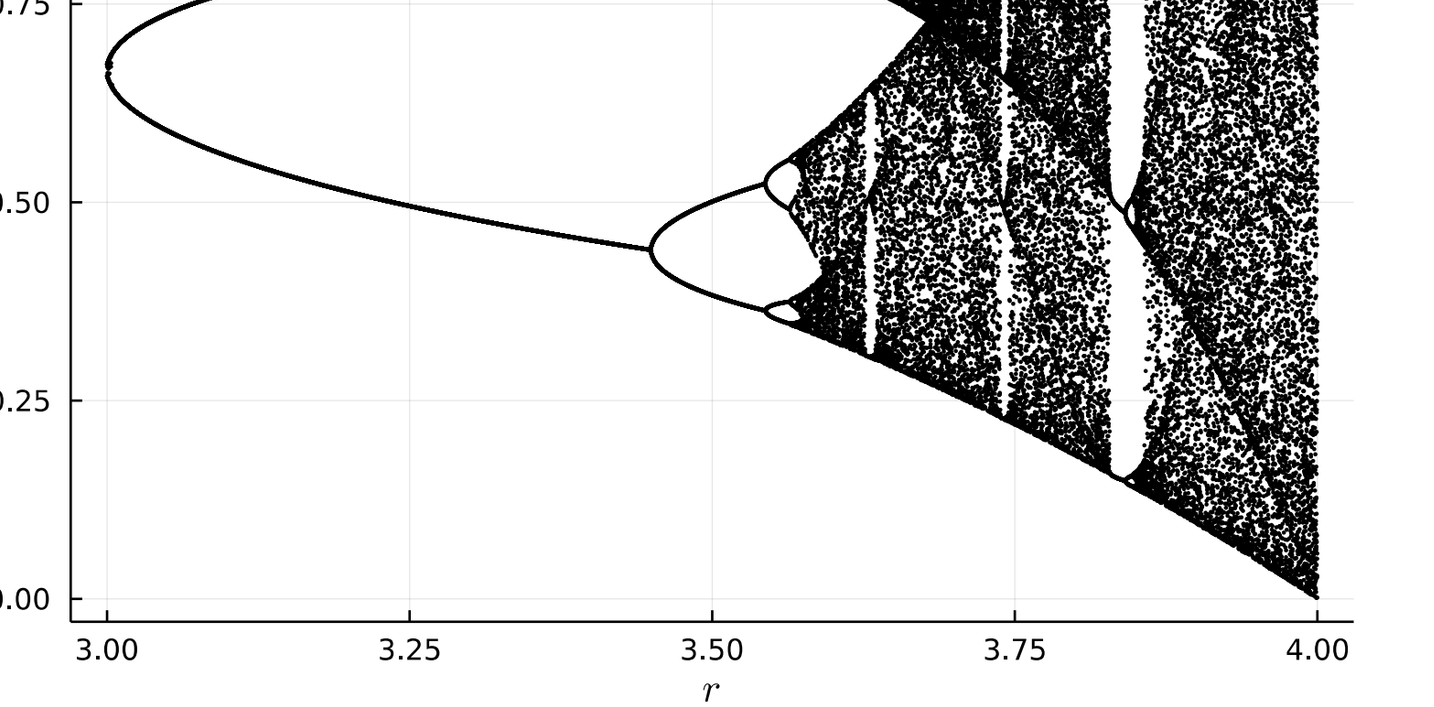
In [20]:
```
R = range(0, 4, length = 100_000)   # r-values
bifurcation( R, 1_000 )
```

Out[20]:



In [21]:
```
R = range(3, 4, length = 100_000)   # r-values to zoom in when r = 3 to 4
bifurcation( R, 1_000 )
```

Out[21]:



We observe that the equilibrium value for $0 \leq r \leq 1$, the equilibrium point ultimately results to 0 regardless of a nonzero value for r. Additionally, when $1 < r < 3$, the x-value increases as r increases. Then, the evolution of x-value for $r \geq 3$, there is a periodic cycle which results to periodic doubling bifurcations. Lastly, we observe a chaotic behavior when r is larger than ≈ 3.6. Note that in the interval for the r-values with chaotic behavior, we also observe a window with periodic cycle behavior then afterwards, it will return back to chaos.