How to implement Euler method in R

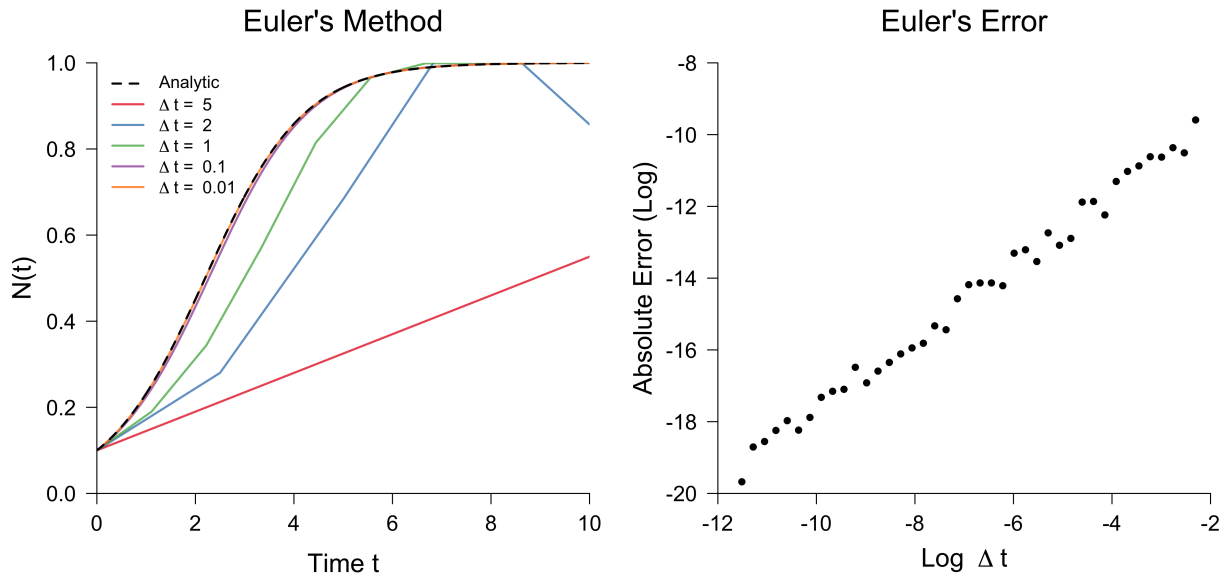
Asked 3 years, 2 months ago Modified today Viewed 578 times Part of R Language Collective

2

I am trying to implement this Euler Method procedure but I am unable to get the required graphs.

```
solve_logistic <- function(N0, r = 1, delta_t = 0.01, times = 1000) {
  N <- rep(N0, times)
  dN <- function(N) r * N * (1 - N)

  for (i in seq(2, times)) {
    # Euler
    N[i] <- N[i-1] + delta_t * dN(N[i-1])

    # Improved Euler
    # k <- N[i-1] + delta_t * dN(N[i-1])
    # N[i] <- N[i-1] + 1 /2 * delta_t * (dN(N[i-1]) + dN(k))

    # Runge-Kutta 4th order
    # k1 <- dN(N[i-1]) * delta_t
    # k2 <- dN(N[i-1] + k1/2) * delta_t
    # k3 <- dN(N[i-1] + k2/2) * delta_t
    # k4 <- dN(N[i-1] + k3) * delta_t
    #
    # N[i] <- N[i-1] + 1/6 * (k1 + 2*k2 + 2*k3 + k4)
  }

  N
}
```

This is the graph I want to make:



And you can also view the original source which I am following for this graph

r Edit tags

Share Edit Follow Close Flag

edited 9 hours ago
MrFlick
194k 17 275 294
Recognized by R Language Collective

asked May 16, 2020 at 9:44
Ijaz Ali
23 3

The answer below was useful ? – Rémi Coulaud Jul 22, 2020 at 9:48

2 Answers

Sorted by:
Reset to default

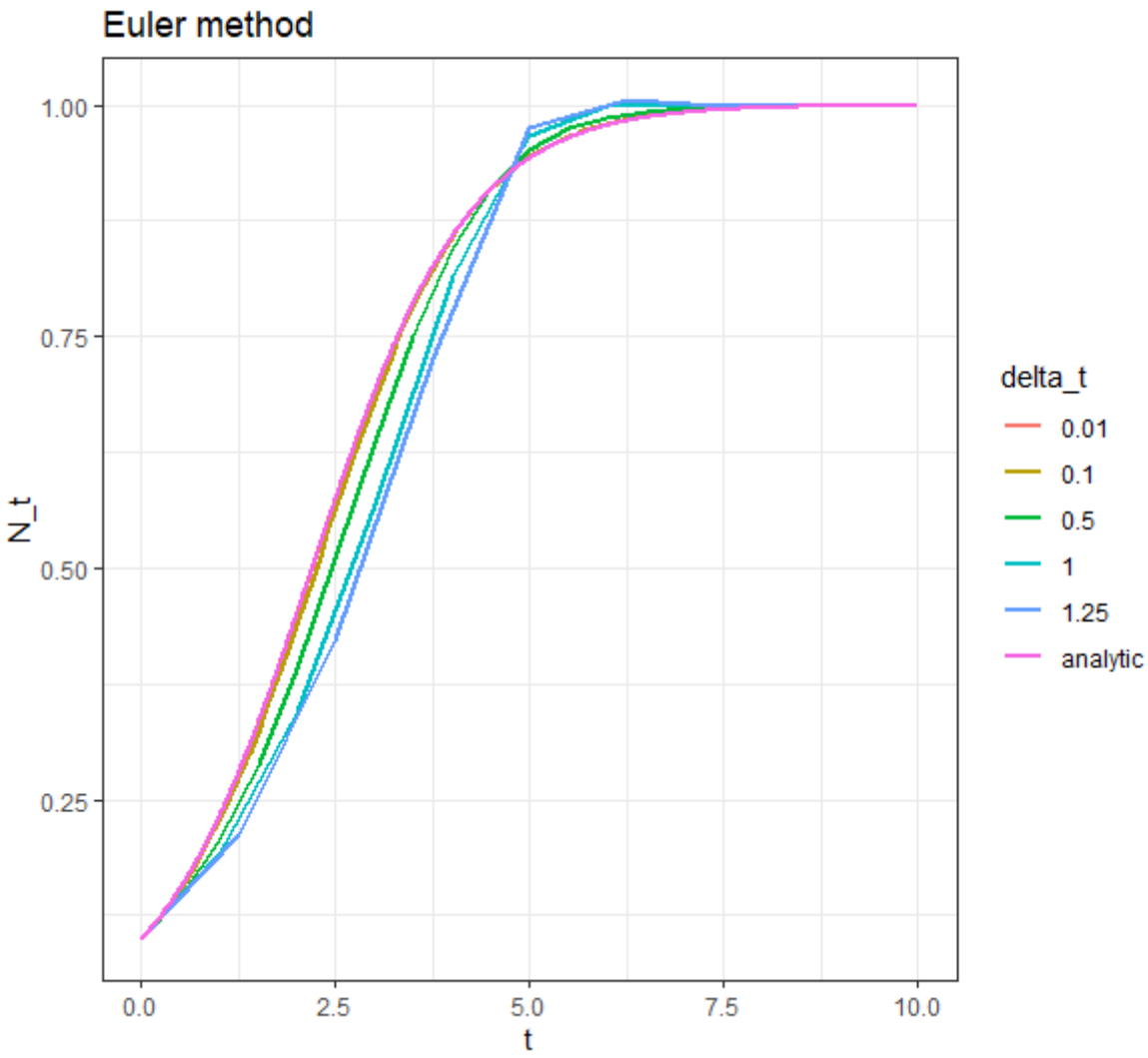
Date modified (newest first)

1

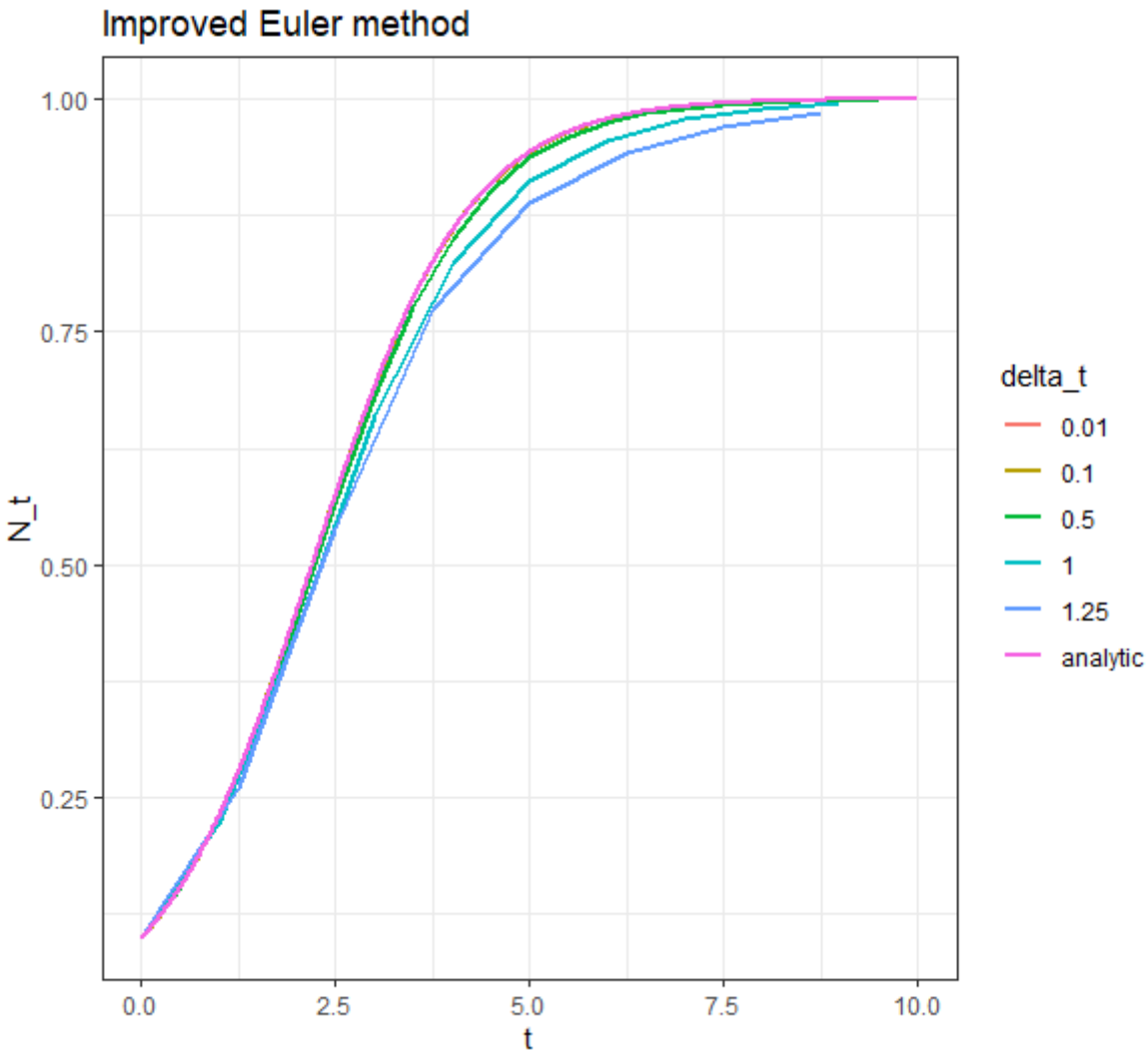
Making the code more generic and modular (separating out the numerical methods), we can plot the solution to the epidemic model initial value problem (differential equation):

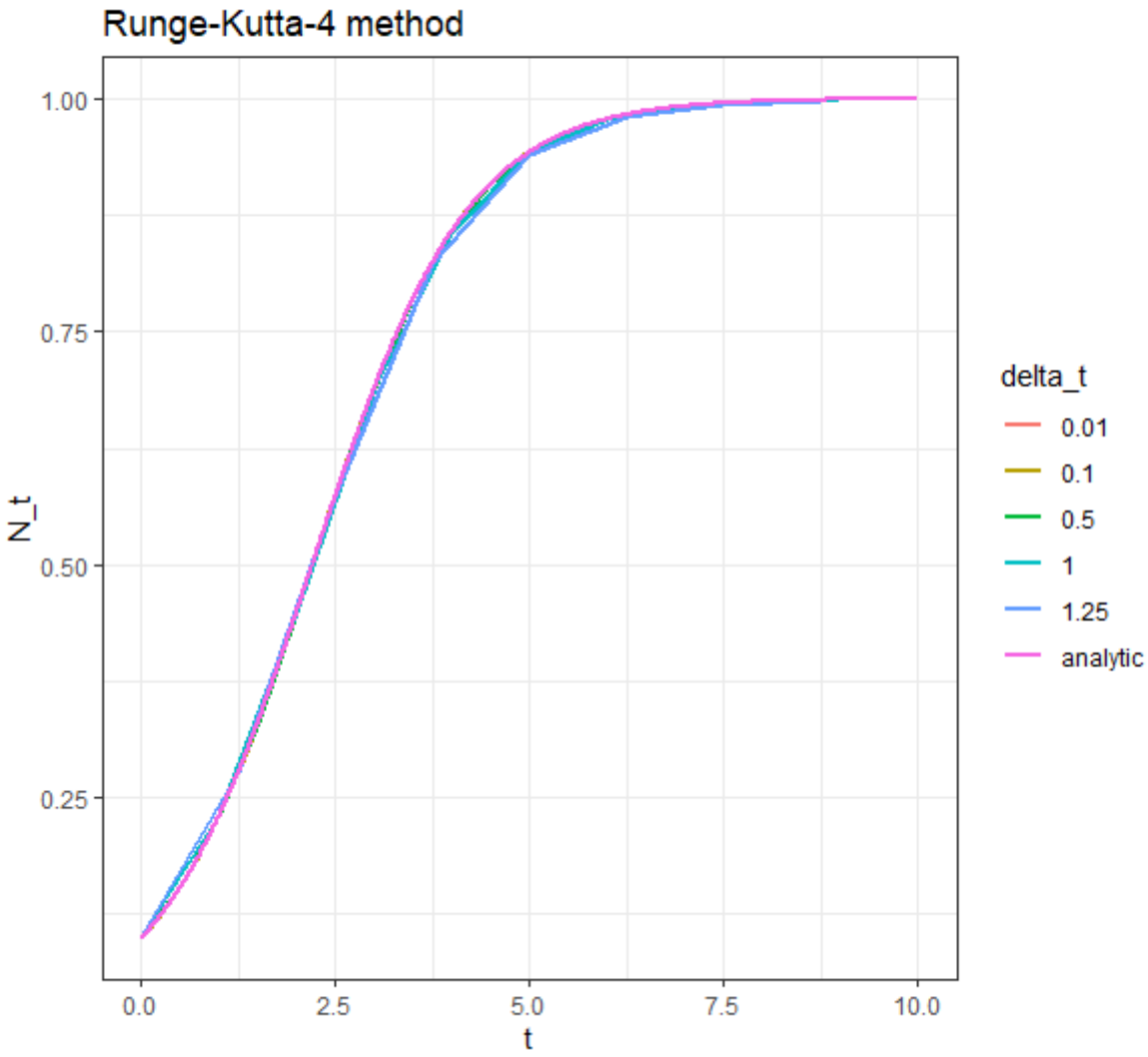
```
Euler_update <- function(xn, delta_t, f) {
  return (xn + delta_t * f(xn))
}

Improved_Euler_update <- function(xn, delta_t, f) {
  k1 <- f(xn) * delta_t
  k2 <- f(xn + k1) * delta_t
  return (xn + 1/2 *  (k1 + k2))
}

Runge_Kutta_4_update <- function(xn, delta_t, f) {
  k1 <- f(xn) * delta_t
  k2 <- f(xn + k1/2) * delta_t
  k3 <- f(xn + k2/2) * delta_t
  k4 <- f(xn + k3) * delta_t
  return (xn + 1/6 * (k1 + 2*k2 + 2*k3 + k4))
}

logistic_function <- function(x, r=1) {
  return (r * x * (1 - x))
}

analytic_solution <- function(x0, ts, r=1) {
  return (1 / (1 + (1-x0)/x0*exp(-r*ts)))
}

solve_logistic <- function(N0, r = 1, delta_t = 0.01, times = 1000,
update_fn=Euler_update) {
  t <- rep(0, times)
  N <- rep(N0, times)

  for (i in seq(2, times)) {
    # Euler
    t[i] <- t[i-1] + delta_t
    N[i] <- update_fn(N[i-1], delta_t, logistic_function)
  }

  return (list(t, N))
}

solution_df <- NULL
```

```
N0 <- 0.1
for (delta_t in c(0.01,0.1,0.5,1,1.25)) {
  sol <- solve_logistic(N0, delta_t=delta_t, times=10/delta_t) #, update_fn =
Runge_Kutta_4_update)
  solution_df <- rbind(solution_df, data.frame(delta_t=delta_t, t=sol[[1]],
N_t=sol[[2]]))
}
solution_df$delta_t <- as.factor(solution_df$delta_t)
ts <- seq(0,10,0.01)
solution_df <- rbind(solution_df, data.frame(delta_t='analytic', t=ts,
N_t=analytic_solution(N0, ts)))

library(dplyr)
library(ggplot2)
solution_df %>% ggplot(aes(t, N_t, col=delta_t)) + geom_line(lwd=1) + theme_bw() +
ggtitle('Euler method')
```



Just by changing the function name for the corresponding algorithm, we can obtain the solution with improved Euler and Runge-Kutta order-4 methods, improving the accuracy (in terms of deviance from the analytic solution) much more and faster convergence,, as shown in the figures below.


Improved Euler method
1.00
0.75
0.50
0.25
N_t
0.0
2.5
5.0
7.5
10.0
t
delta_t
0.01
0.1
0.5
1
1.25
analytic



Share Edit Delete Flag

edited 6 hours ago

answered 10 hours ago

Sandipan Dey
21.3k 2 49 62

1

Your interest for epedimiological model is a good thing.

To obtain a similar graph as you show, you need to code first the analytical solution of `N(t)` which is given on the reference web site.

```
logistic <- function(N0, r, t){
    return(1 / (1 + ((1-N0)/N0) * exp(- r * t)))
}
```

Moreover you should be careful with absisse informations.

```
r <- 1
t <- 1:1000
N0 <- 0.03
delta_t <- 0.01
plot(t * delta_t, logistic(N0 = N0, r = r, t = t * delta_t), type = "l",
     ylim = c(0, 1),
     ylab = "N(t)",
     xlab = "times")
```

```
lines(t * delta_t, solve_logistic(N0 = N0, times = max(t)),
      col = "red", lty = 2)
```

It gives you part of the graphic, now you are able to compute error of the method and test with another delta.

The Euler method is a numerical method for EDO resolution based on Taylor expansion like gradient descent algorithm .

```
solve_logistic <- function(N0, r = 1, delta_t = 0.01, times = 1000) {
  N <- rep(N0, times)
  dN <- function(N) r * N * (1 - N)

  for (i in seq(2, times)) {
    # Euler (you follow the deepest slope with a small step delta)
    N[i] <- N[i-1] + delta_t * dN(N[i-1])
  }
  N
}
```

Share Edit Follow Flag

edited Jun 22, 2020 at 8:59

answered May 16, 2020 at 14:16

Rémi Coulaud
1,684 1 8 19