Research Journal: 2019-11-06

Zane Billings

2019-11-06

Question

1. Does generating the data using the ODE method get a better fit with noise?
2. Does generating the data using the discretized method get a better fit with noise?
3. Does fitting the data, using data generated with the noisy analytic method, give better results if only the data after the inflection point is used?
4. How do the different generation methods compare to each other?
5. Make sure the entire generation–fitting "pipeline" is working the way I want it to. I.e., ensure that the data fitting functions still work.

Process

- For question 1:
    1. Write a function to generate noisy ODE generated data.
    2. Use the test case I've been using to see if we get a good fit.
- For question 2:
    1. Write a function to generate noisy discretization generated data.
    2. Use the same test case to see if we get a good fit.
- For question 3:
    1. Generate data using the same test case with the noisy analytic method.
    2. Compute models for the data using the standard linear fit function, using only the data before the inflection point, using only the data after the inflection point, using only odd data points, and only even data points.
    3. Compare the SSR for each of these models to determine if fitting after the inflection point is better than fitting all of the data or choosing some other weird subset.
- For question 4:
    1. Generate data using the standard test case via each of the time series generation methods.
    2. Compute the fitted model using the regular linear model fitting function, and see which data regenerates the known parameters the best.
- For question 5, I just need to keep testing and ensure that I can generate data with one of the generation functions, pass this output to the data prepping function, and then pass this output to a model fitting function. Eventually, I want to construct a wrapper function to accept method inputs and do this all in one step.

Results

Setup

First, I sourced in the two scripts containing all of the necessary functions.

```
# Source in function scripts (paths relative to RProject)
source("../Sample_data_generation.R")
```

```
## Loading required package: deSolve
```

```
source("../Least_squares_methods.R")
```

```
## Loading required package: MASS
```

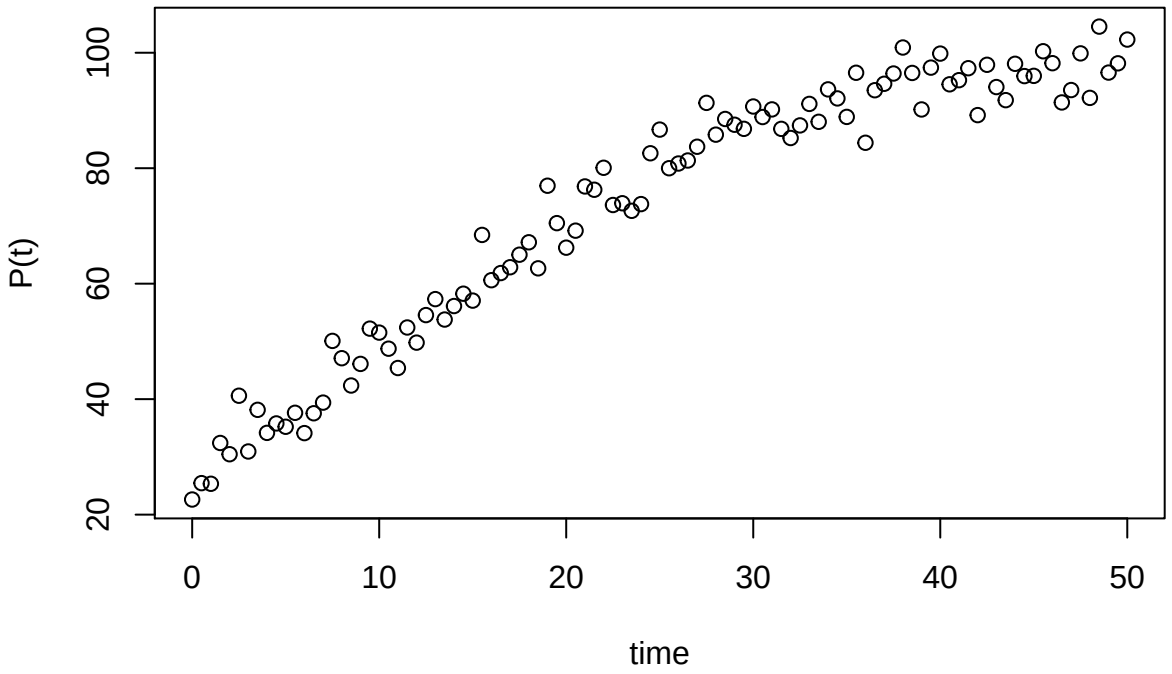
Then, I defined all of the parameters for the test case I wanted to use.

```
# Define test case parameters
P_naught <- 25
K <- 100
r <- 0.1
max_t <- 50
time_step <- 0.5
noise <- 0.04
```

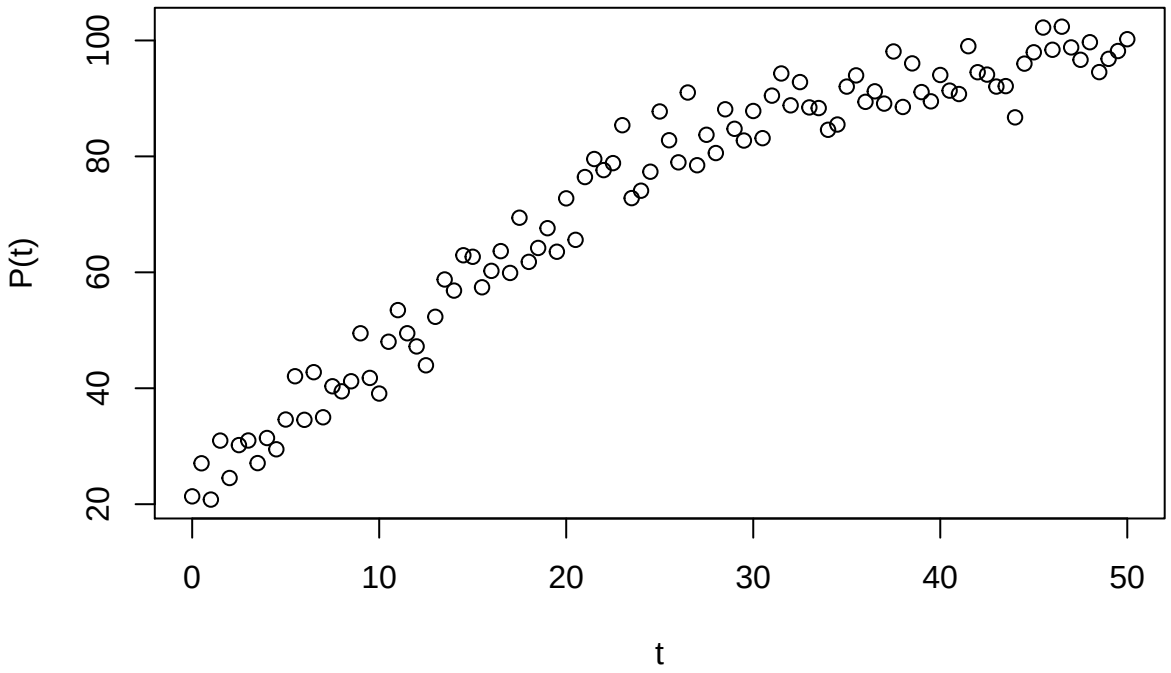
Question 1: Generating data with the noisy ODE solver generator.

First, I generated a sample time series with the test-case parameters, using the noisy ODE solver method. Then, I fit the data using the regular fitting method.

```
sample_ODE_data <-
  generate_noisy_solver_logistic_data(P_naught, K, r, max_t, time_step, noise, TRUE)
```




```
sample_analytic_data <-
  generate_noisy_analytic_logistic_data(P_naught, K, r, max_t, time_step, noise, TRUE)
```




```
prepped_ODE_data <- prep_data(sample_ODE_data)
prepped_analytic_data <- prep_data(sample_analytic_data)
test_ODE_results <- model_logistic_data(prepped_ODE_data)
```

```
## The estimated growth rate is: 0.2391527
## The estimated carrying capacity is: 67.62789
```

```
test_analytic_results <- model_logistic_data(prepped_analytic_data)
```

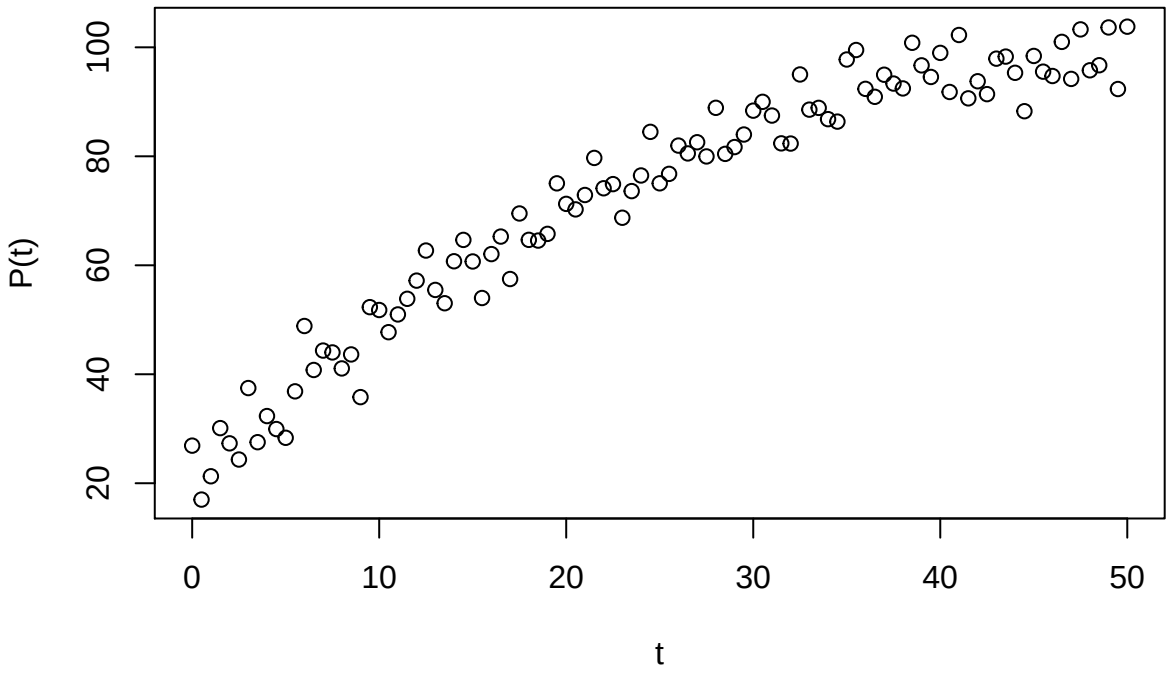
```
## The estimated growth rate is: 0.2480463
## The estimated carrying capacity is: 64.75435
```

So, we see that both estimates for K are about the same, but it appears that, at least for this one case, the logistic solver gets a closer estimate of r. Maybe we should look at this further.

Question 2: Generating data with the noisy Euler generator.

In this case, I generated data using the Euler discretization method with noise. I compared this to the analytic solution from earlier after fitting the Euler generated data.

```
sample_euler_data <-
  generate_noisy_euler_logistic_data(P_naught, K, r, max_t, time_step, noise, TRUE)
```




```
prepped_euler_data <- prep_data(sample_euler_data)
test_euler_results <- model_logistic_data(prepped_euler_data)
```

```
## The estimated growth rate is: 0.2351614
## The estimated carrying capacity is: 64.92487
```

```
test_analytic_results <- model_logistic_data(prepped_analytic_data)
```

```
## The estimated growth rate is: 0.2480463
## The estimated carrying capacity is: 64.75435
```

These two methods appear to be giving us about the same estimation for both parameters. The estimated r is about doubled in both cases, and the carrying capacity is about 2/3 of what we expect in both cases. So, the noisy Euler data probably isn't doing any better.

Question 3: SSR comparisons for noisy analytic data.

We have the prepped noisy analytic data already ready from before, so we can just computed a bunch of results.

```
cat("Regular fit:\n")
```

```
## Regular fit:
```

```
test_reg_results <- model_logistic_data(prepped_analytic_data)
```

```
## The estimated growth rate is: 0.2480463
## The estimated carrying capacity is: 64.75435
```

```
cat("Front fit:\n")
```

```
## Front fit:
```

```
test_front_results <- model_beginning_only(prepped_analytic_data)
```

```
## The estimated growth rate is: 0.8619685
## The estimated carrying capacity is: 35.06418
```

```
cat("Back fit:\n")
```

```
## Back fit:
```

```
test_back_results <- model_end_only(prepped_analytic_data)
```

```
## The estimated growth rate is: 1.894903
## The estimated carrying capacity is: 80.77734
```

```
cat("Even fit:\n")
```

```
## Even fit:
```

```
test_even_results <- model_every_other_even(prepped_analytic_data)
```

```
## The estimated growth rate is: -0.1293255
## The estimated carrying capacity is: 50.76514
```

```
cat("Odd fit:\n")
```

```
## Odd fit:
```

```
test_odd_results <- model_every_other_odd(prepped_analytic_data)
```

```
## The estimated growth rate is: 0.5286181
## The estimated carrying capacity is: 62.88196
```

I didn't get to the rest this week, but the updated code was pushed to GitHub.

Conjectures and Future Questions

- I need to test all of the fitting functions to make sure they are working correctly. I am not convinced based on the answers.
- I still need to work on Q4 and Q5 from this document.
- Explore sensitivities of each of the methods to noise.
- Implement smoothing.
- Refactor everything so it works as intended and has one wrapper function to call the fitting method.