Journal: Is lambda related to r?

Zane Billings

31 March, 2020

Is lambda related to the value of r in the original population?

While I previously found that the amount of smoothing needed to recover the value of r from dimensionless data was much higher than I had expected, now I want to find out how r, the population intrinsic growth rate, relates to λ^*, the optimal smoothing value. I have three main hypothesis:

1. The value of λ^* is constant for all r,
2. The value of λ^* is proportional to r, or
3. The value of λ^* is chaotic with respect to the value of r.

Right now, I am leaning towards hypothesis 2: I think there will be a linear relationship between the value of r and the value of λ^*.

Eventually I will wrap this up into a function, but right now I just want to do the experiment, so I will be copy-pasting most of my code from the previous experiment and wrapping it inside another loop. First I will source the necessary scripts and set a random seed, as well as setup variables I need.

Note here that I added the functions from the previous test to the `Helpers.R` script, so I didn't have to redefine them here.

```r
source(here::here("Scripts","Dimensionless_exploration.R"))
```

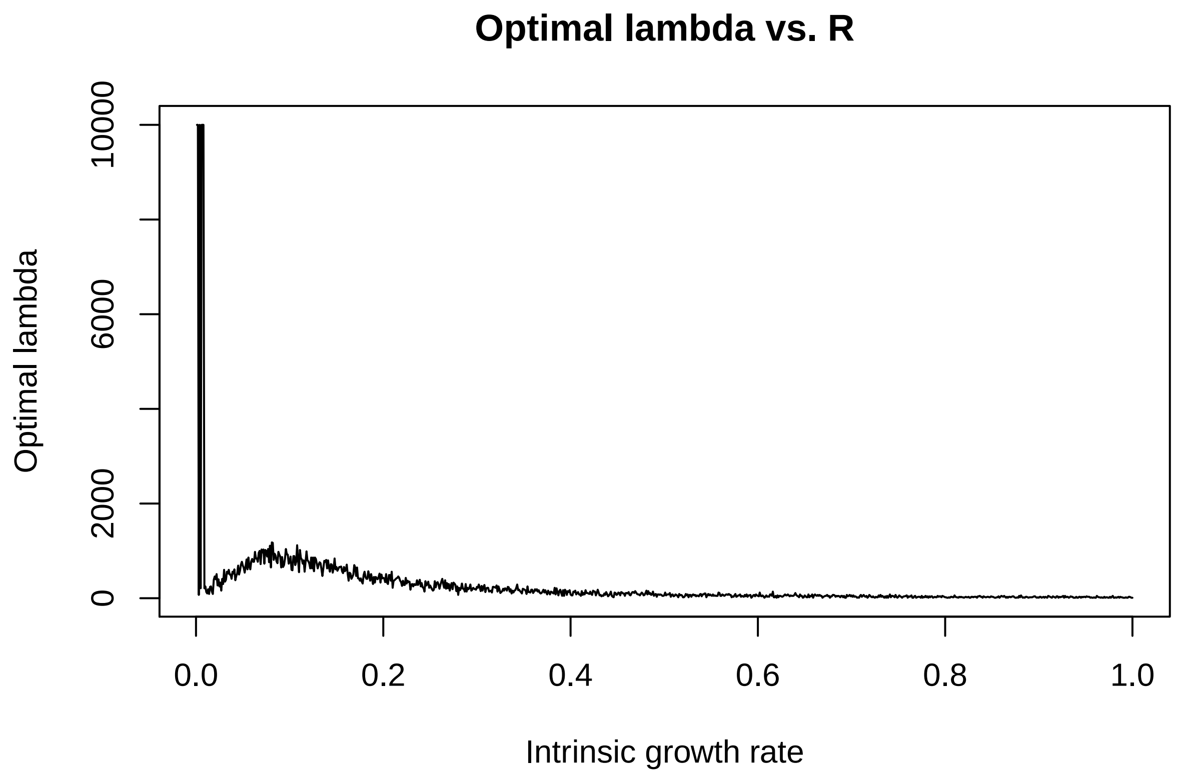
```
## Loading required package: MASS
```

```r
source(here::here("Scripts", "Helpers.R"))
source(here::here("Scripts", "Least_squares_methods.R"))

rs <- seq(0.001, 1, 0.001) # Values of r to test
lambdas <- seq(10, 10000, 10) # Values of lambda for each r
k <- 3 # k-fold cross validation constant
lstar <- numeric(length(rs))
```

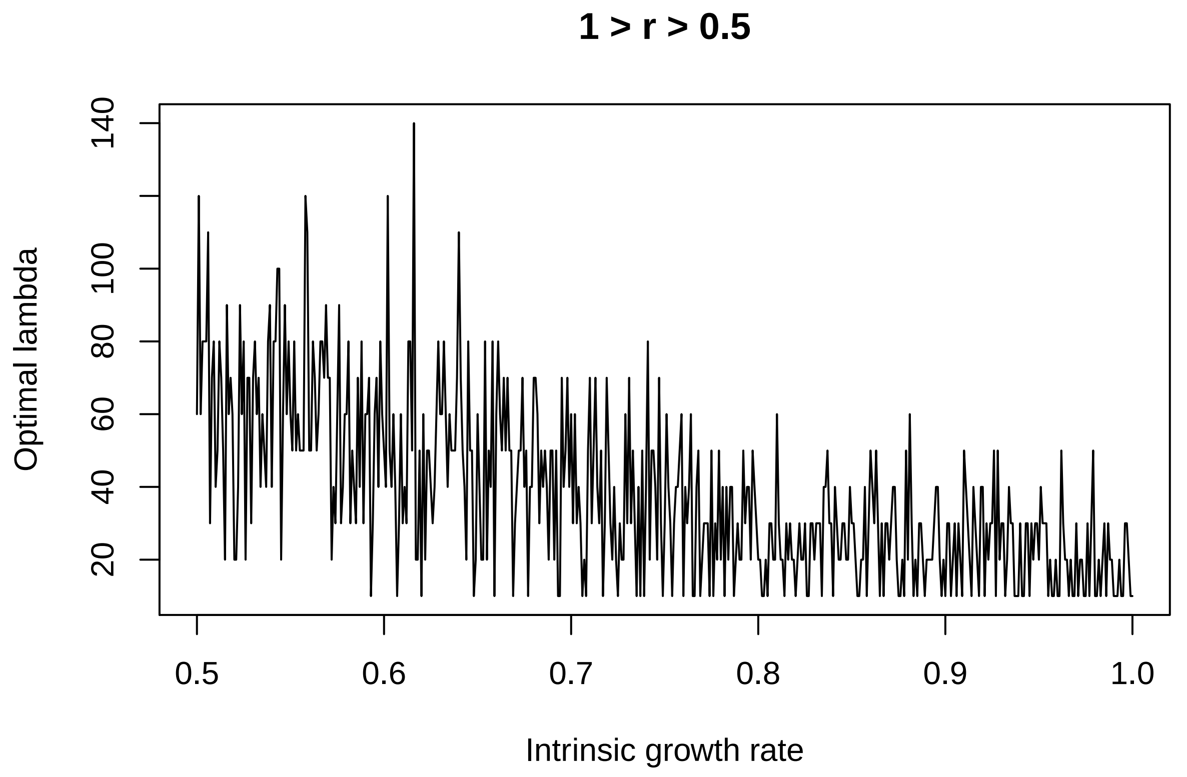
```r
for (r in rs) {
  # Generate and partition data
  df <- generate_dimensionless_logistic_data(.1, r, 50, 0.1)
  parts <- random_partition(df, k)
  combos <- combn(1:k, k-1, simplify = F)

  # Empty vector to contain cross-validation error values
  cves <- numeric(length(lambdas))

  # Do all of this stuff for each lambda using the current data.
  for (a in 1:length(lambdas)) {
    l <- lambdas[[a]]
    errors <- numeric(length(parts))
    # Need to do this k number of times (# of parts)
    for (i in 1:k) {
      # Grab the first combo of indices to use
      to_use <- combos[[i]]
      # Make a blank dataframe
      train <- data.frame()
      # This part has to be done k-1 times to build training data.
      for (j in to_use) {
        # Get the next data frame.
        new_dat <- parts[[j]]
        # Add this data frame to the bottom of the rest.
        train <- rbind(train, new_dat)
      }

      # Now train has k-1 of the parts in it.
      # Set test to be the part that wasn't selected.
      not_used <- (1:k)[!(1:k %in% to_use)]
      test <- parts[[not_used]]
      # Prep and model the data, specify time step manually.
      prepped <- prep_data(train, 0.1)
      mod <- model_logistic_data_dimensionless_smoothing(prepped, l)
      # Next need to calculate predicted values using the estimated growth rate from the model.
      #P = (P0e^(rt))/(1+P0(1-e^(rt)))
      t <- test$t
      pred <- (0.1*exp(mod*t))/(1+0.1*(exp(mod*t)-1))
      errors[i] <- calculate_SSE(test$P,pred)
    }

    cves[a] <- mean(errors)
  }
  lstar[match(r,rs)] <- lambdas[which(cves == min(cves))]
}
```

Now we need to collect the values of r and λ^* together and we can visualize the results.

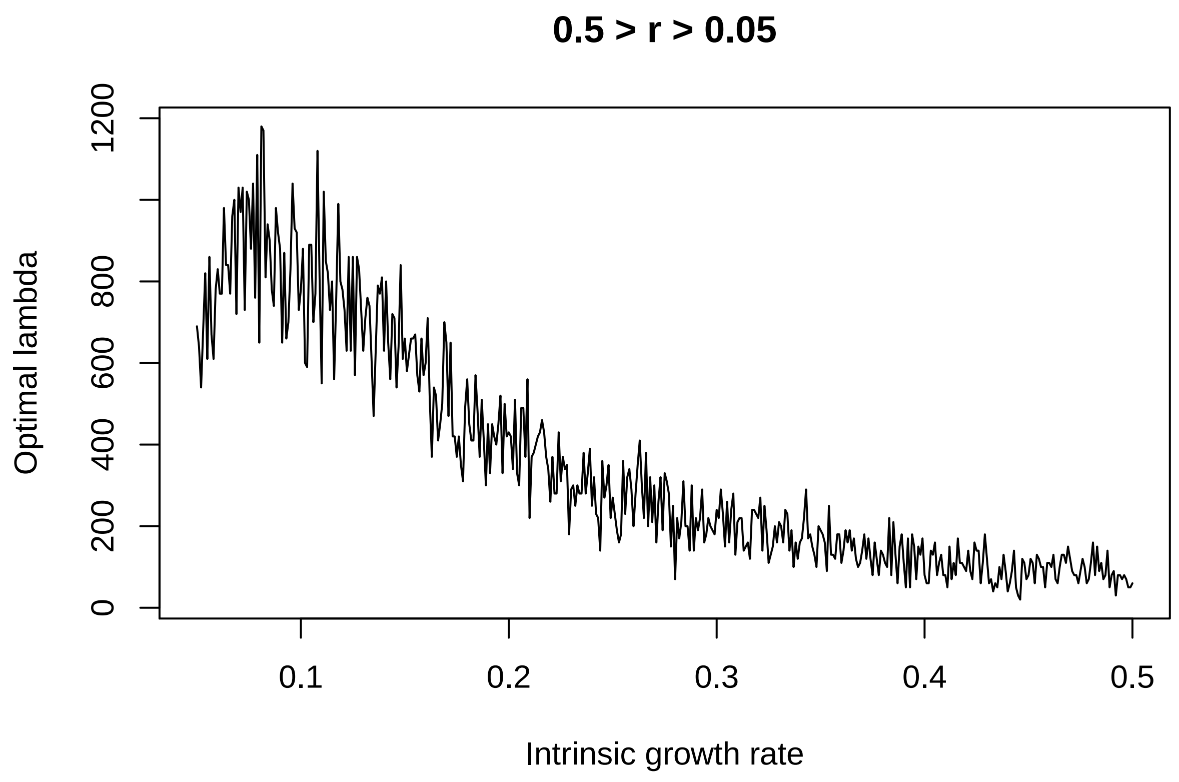
```r
res <- data.frame("r" = rs,
                  "l" = lstar)
with(res, plot(r,l, type = "l",
               xlab = "Intrinsic growth rate",
               ylab = "Optimal lambda",
               main = "Optimal lambda vs. R"))
```



```r
with(res[res$r >= 0.5 & res$r <= 1, ], plot(r,l, type = "l",
               xlab = "Intrinsic growth rate",
               ylab = "Optimal lambda",
               main = "1 > r > 0.5"))
```

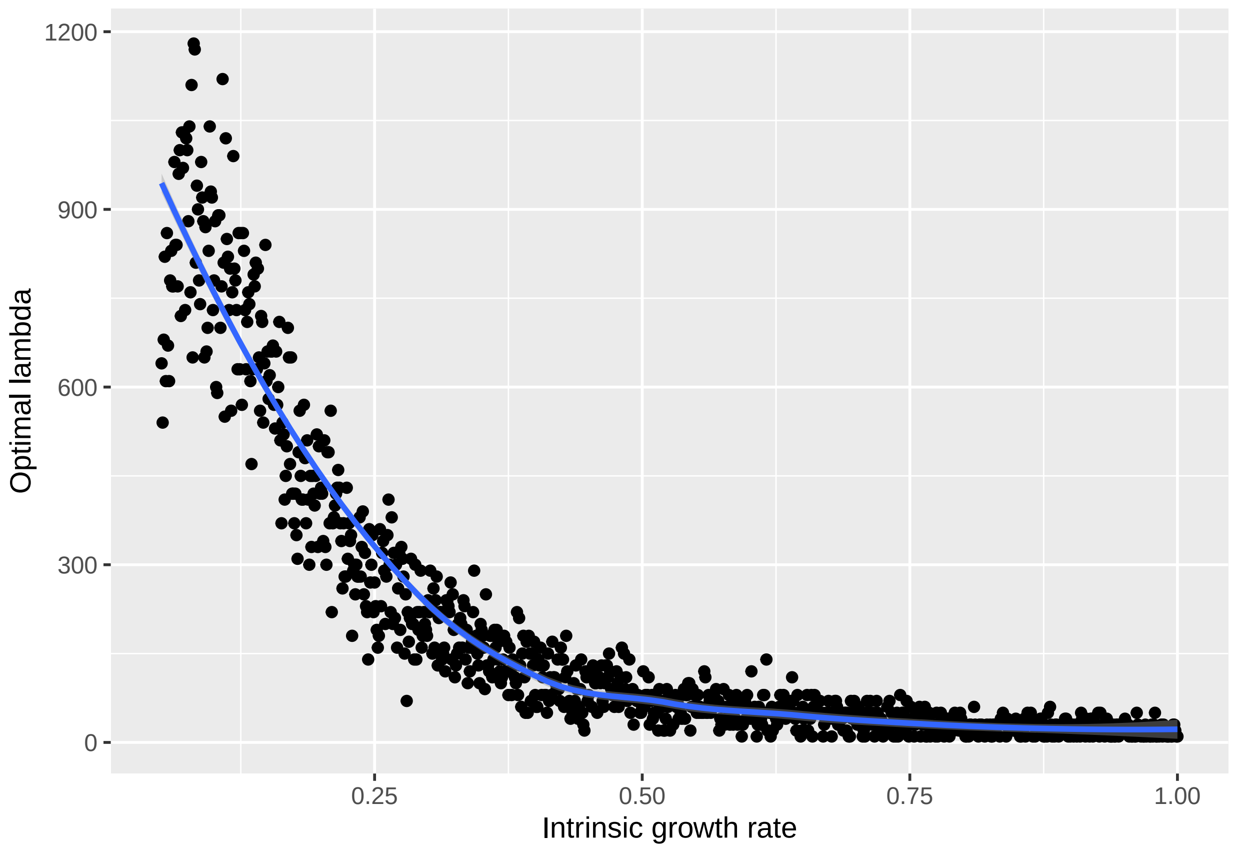


```r
with(res[res$r <= 0.5 & res$r >= 0.05, ], plot(r,l, type = "l",
               xlab = "Intrinsic growth rate",
               ylab = "Optimal lambda",
               main = "0.5 > r > 0.05"))
```

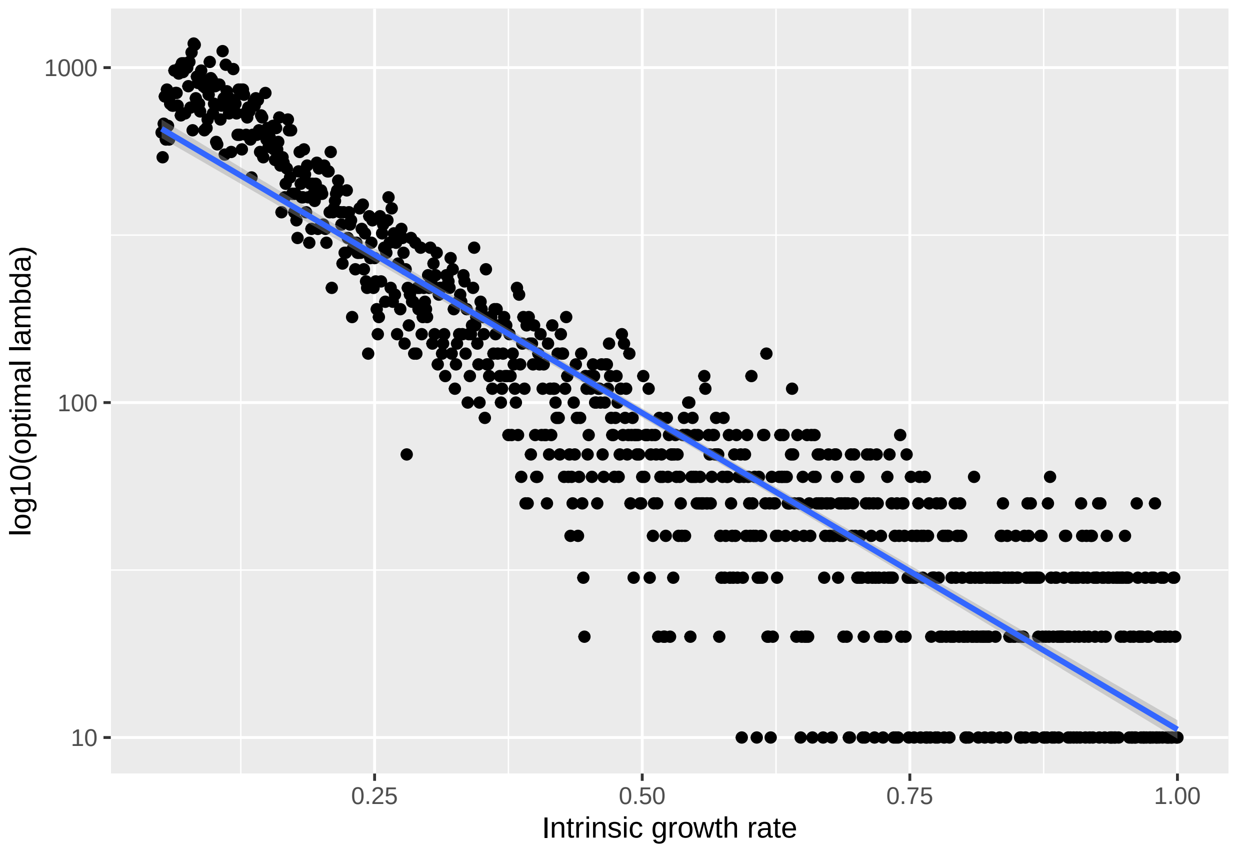


THis curve (starting at around r = 0.05 to exclude the major outliers) can probably be modeled well by an expontential decay curve. Here, I have just modeled it with a LOESS curve but an exponential fit might be worth examining.

```r
library(tidyverse)
res %>%
  filter(res$r > 0.05) %>%
  ggplot(aes(x = r, y = l)) +
    geom_point() +
    labs(x = "Intrinsic growth rate", y = "Optimal lambda") +
    geom_smooth(method = "loess")
```



```
res %>%
  filter(res$r > 0.05) %>%
  ggplot(aes(x = r, y = l)) +
    geom_point() +
    scale_y_log10() +
    labs(x = "Intrinsic growth rate", y = "log10(optimal lambda)") +
    geom_smooth(method = "lm")
```


1000
100
10
log10(optimal lambda)
0.25
0.50
0.75
1.00
Intrinsic growth rate