Solow Growth Model

Mattia Albertini

2024-09-29

Interactive Shiny App

In this script, we create an interactive app that allows you to play around with the Solow Model. Do not alter the following code, just run it and play around. Note: to be able to start the app, you need to have a connection to internet. Have fun!

The Solow Model

The model we will represent is,

$$Y = AK^{\alpha}H^{1-\alpha}$$

in per capita terms,

$$y = Ak^{\alpha} \qquad where \qquad y = \frac{Y}{H}; \quad k = \frac{K}{H}$$

The steady state condition k^* solves,

$$k^* = (1-\delta)k^* + sAk^{\alpha} \longrightarrow k^* = \left(\frac{sA}{\delta}\right)^{\frac{1}{1-\alpha}}; \quad y^* = A\left(\frac{sA}{\delta}\right)^{\frac{\alpha}{1-\alpha}}$$

where δ is the depreciation rate of capital, s is the savings rate, A is the total factor productivity, and α is the capital share.

Define the UI

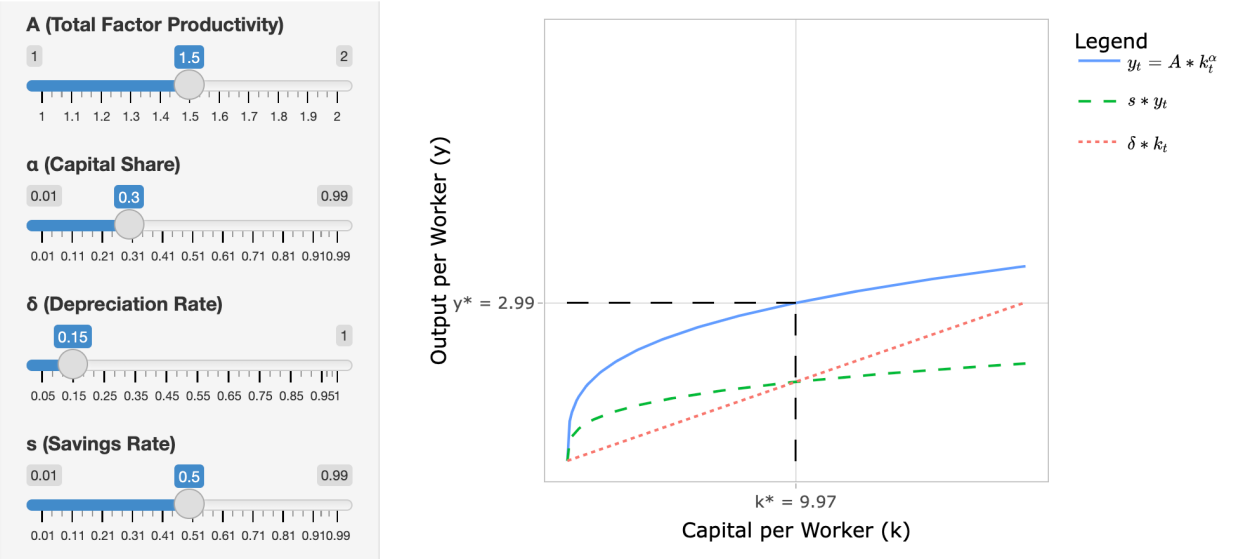
```
# Define UI
ui <- fluidPage(
  titlePanel("Solow Model: Cobb-Douglas Production Function (Per Capita)"),
  withMathJax(),
  sidebarLayout(
    sidebarPanel(
      sliderInput("A", "A (Total Factor Productivity)", min = 1, max = 2, value = 1.5, step = 0.1),
      sliderInput("alpha", "α (Capital Share)", min = 0.01, max = 0.99, value = 0.3, step = 0.01),
      sliderInput("delta", "δ (Depreciation Rate)", min = 0.05, max = 1, value = 0.15, step = 0.05),
      sliderInput("s", "s (Savings Rate)", min = 0.01, max = 0.99, value = 0.5, step = 0.01)
    ),

    mainPanel(
      plotlyOutput("productionPlot")
    )
  )
)
```

Define the Server

```
# Define server logic
server <- function(input, output) {

  output$productionPlot <- renderPlotly({
    # Define capital per worker (k), starting from 0
    k <- seq(0, 100, by = 0.1)  # Array of values for capital per worker, starting from 0

    # Cobb-Douglas Production Function in per capita terms
    y <- input$A * k^input$alpha   # Output per worker (y = A * k^alpha)

    # Savings Function (per worker)
    y_savings <- input$s * y       # Savings = s * y

    # Depreciation (per worker)
    y_depreciation <- input$delta * k  # Depreciation = delta * k

    # Find the intersection point (k* where s * A * k^alpha = delta * k)
    k_star <- (input$s * input$A / input$delta)^(1 / (1 - input$alpha))
    y_star <- input$A * k_star^input$alpha  # y at k_star (on production function)

    # Create a data frame for plotting
    plot_data <- data.frame(k = k, Production = y, Savings = y_savings, Depreciation = y_depreciation)

    # Compute a reasonable y-axis limit based on max A (assume A = 2 as the highest value for example)
    max_A <- 2
    max_y <- max_A * max(k)^input$alpha  # Maximum output per worker for highest A

    # Plotting
    gg <- ggplot(plot_data, aes(x = k)) +
  geom_line(aes(y = Production, color = "$$y_t = A * k_t^{\\alpha}$$"), size = 0.5) +
  geom_line(aes(y = Savings, color = "$$s * y_t$$"), size = 0.5, linetype = "dashed") +
  geom_line(aes(y = Depreciation, color = "$$\\delta * k_t$$"), size = 0.5, linetype = "dotted") +

  # Add vertical line at k_star (same as horizontal)
  geom_segment(aes(x = k_star, y = 0, xend = k_star, yend = y_star),
               linetype = "longdash", color = "black", size = 0.25) +

  # Add horizontal line from the intersection to the y-axis (thinner)
  geom_segment(aes(x = 0, y = y_star, xend = k_star, yend = y_star),
               linetype = "longdash", color = "black", size = 0.25) +

  # Customize x-axis: show only k_star as the tick
  scale_x_continuous(limits = c(0, 20), breaks = k_star, labels = paste0("k* = ", round(k_star, 2))) +

  # Customize y-axis: set fixed limits to prevent auto-scaling
  scale_y_continuous(limits = c(0, max_y), breaks = y_star, labels = paste0("y* = ", round(y_star, 2))) +

  labs(title = "",
       x = "Capital per Worker (k)",
       y = "Output per Worker (y)",
       color = "Legend") +

  theme_light() +

  # Move the label annotations outside the plot window
  theme(axis.text.x = element_text(margin = margin(t = 10)),   # Adjust x-axis label position
        axis.text.y = element_text(margin = margin(r = 10)))   # Adjust y-axis label position

# Render plotly
ggplotly(gg)
  })
}
```

Run the APP

```
shinyApp(ui = ui, server = server)
```



- The total factor productivity A is a scaling factor that shifts the function up and down. The intuition is that for the same amount of capital, the production becomes more efficient and therefore more output can be produced.
- The savings rate s determines the saving function. Note that as it increase, more capital can be devote to production (but not consumption!) and hence the output increases
- The depreciation rate δ determines the amount of capital that depreciates over time. Note that the higher it is, the more capital depreciates and hence we are left with less capital to devote to production. This implies a reduction in the output.
- The capital share α determines the sensitivity (or elasticity) of output to changes in capital. It determines the shape of the production function. Higher capital shares imply a higher responsiveness of output to capital, and therefore a higher capital share determines a higher output.