FINC-672 – Workshop in Finance: Empirical Research

Data Visualization

Prof. Matt Fleckenstein
University of Delaware

mflecken@udel.edu

Goals

- ☐ Data Visualization in Julia using `Plots.jl`

Plotting in Julia

- We are going to use `Plots.jl`[1] for visualizing data.
- To use this package, we first need to install it. Recall that we do this with the Julia package manager as follows.

```
using Pkg;
Pkg.add("Plots");
using Plots;
```

[1] https://github.com/JuliaPlots/Plots.jl

Plotting in Julia

- After installing `Plots.jl`, we need to select a *plotting backed*. Basically, we just need to tell `Plots.jl` which program to use to make plots.[2]
- We are going to use Julia `Plot.jl`'s `GR` backend.

```
Pkg.add("GR");
gr();
Plots.GRBackend();
```

[2] For more information on plotting backends, see `https://docs.juliaplots.org/latest/backends/`.

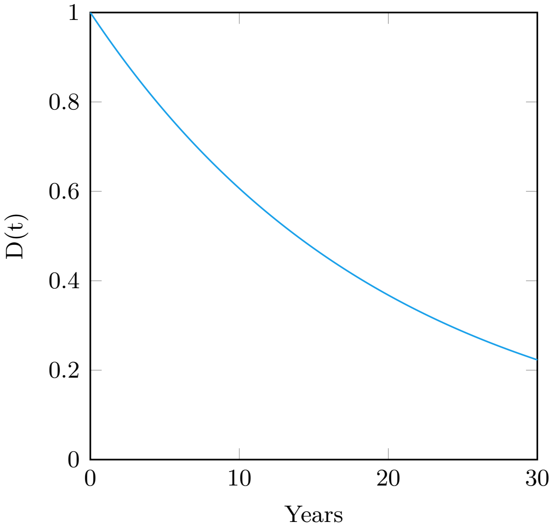
Plotting in Julia

- Now we are ready to make a first plot. Let's start with a simple *line plot*.
- As an example, let's plot the continously-compounded discount-factor curve, assuming that the applicable interest rate is 5% (per annum, continously compounded).
- Recall from your prior finance classes that the continuously compounded discount factor for time t is $D(t) = \exp(-rt)$, where r is the continuously compounded interest rate.
- Let's plot $D(t)$ using $r = 0.05$ for $t = 1...30$ years in 3-month increments.

Plotting in Julia

```
r=0.05;
t=0.25:0.25:30.0;
Dt = exp.(-r.*t);
plot(t, Dt,
     linecolor="blue", linewidth=3,
     title="Continuously Compounded Discount Factors",
     xlabel="Years",
     ylabel="D(t)")
```

Plotting in Julia



Plotting in Julia

- Let's take this step by step.
- First, we simply set $r = 0.05$ and initialize the time t vector which runs from $t = 0.25$ (i.e. 3-month from now) to $t = 30$ years (i.e. 30 years from now). Note that we use the colon `:` to create a `range` that starts at 0.25 and increases by 0.25 each step, i.e. we get 0.25, 0.50, 0.75, 1.00, ..., 30.
- Second, we calculate the discount factors using the dot operator `.` which just means that we apply $exp()$ to each $r * t$ value.
- Finally, we can plot t on the horizontal axis and $D(t)$ on the vertical axis, simply by calling `plot(t,Dt)`.
- We use `linecolor="blue"` to set the color, and we use `linewidth=3` to set the line thickness.
- By using `xlabel` and `ylabel` we add labels on the x-Axis and the y-Axis.

Plotting in Julia

- Next, let's plot multiple graphs in one plot. To illustrate, we will simulate multiple paths of daily prices of a stock over a period of one month from March 1, 2022 to March 31, 2022.
- Recall from prior finance classes, that we often model stock prices as **random walks**. We will let Julia generate random numbers for us, which we will use to calculate the paths of the random walk.
- To generate random numbers from a *standard* normal distribution, we use `Random.jl` package by calling `using Random`
- First, we set a *seed* which basically initializes the random number generator.

```
using Random
Random.seed!(2021)
```

Plotting in Julia

- Next, let's create our vector of days which is going to be our horizontal axis. We need 31 days.

```
day_axis = collect(1:1:31)
```

Plotting in Julia

- Next, let's set the initial stock price (`S_0`) to 100.
- We generate random draws from a standard normal distribution by calling the `randn()` function, which takes as input parameter how many random numbers we want to generate.
- In our example, we need to simulate stock prices for 31 days (`days=31`).
- These random draws are the daily changes in the stock price.
- Finally, to get the path of the stock price over the month, we need to take the cumulative changes of the stock price and add those to the initial price.
- We achieve this by calling `cumsum` which simply returns the cumulative sum of an input vector.
- Let's simulate three paths.

Plotting in Julia

```
S_0 = 100.0;
days = 31;
stock_path_1 = S_0 + cumsum( randn(days) );
stock_path_2 = S_0 + cumsum( randn(days) );
stock_path_3 = S_0 + cumsum( randn(days) );
```

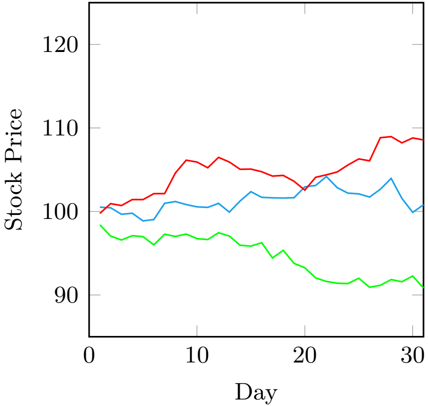
Plotting in Julia

- Finally, we are all set to plot the stock paths.
- Note that we label the individual paths using the `label` statement and use the `legend` statement to place the plot legend in the bottom left corner.
- `xrotation=45` just means that we rotate the labels for the days by 45 degrees.

```
plot(day_axis, [stock_path_1 stock_path_2 stock_path_3],
    xlim=(0,31), ylim=(85.0,125.0)
    xlabel="Day", ylabel="Stock Price",
    label=["Path 1" "Path 2" "Path 3" "Path 4" "Path 5"],
    legend=:bottomleft, xrotation=45, title="Simulated Stock Price Paths")
```

Plotting in Julia

- The final plot will look similar to the one below.[3]



[3] Note that your plot will look slightly different. This is because you are most likely drawing different random numbers.

Wrap-Up

- ☑ Data Visualization in Julia using `Plots.jl`