The Journal of Open Source Software

ScenTrees.jl: A Julia Package for Generating Scenario Trees and Scenario Lattices for Multistage Stochastic Programming

Kipngeno Benard Kirui[1], **Alois Pichler**[1], **and Georg Ch. Pflug**[2]

1 University of Technology, Chemnitz, Germany **2** University of Vienna

DOI: 10.21105/joss.01912

Software

- Review
- Repository
- Archive

Editor: Melissa Weber Mendonça

Reviewers:

- @juliohm
- @matbesancon

Submitted: 05 November 2019
Published: 06 February 2020

Summary

Stochastic processes have random and uncertain outcomes. Decisions for stochastic problems involving such processes must be made at the different stages of the problem. Mathematical problems with such uncertain parameters, described by an underlying probability distribution, are sometimes intractable to solve. For algorithmic treatment, such programs should be approximated by simpler ones, in the same manner as functions are represented as vectors on finite grids on digital computers. In most applications, these underlying distributions are approximated by discrete distributions with a finite number of scenarios, or possible realizations, for the random variables. This approximation procedure is often called `scenario generation`.

There is a vast literature describing different methods of scenario generation. Many authors including Høyland & Wallace (2001), Pflug (2001), Kovacevic & Pichler (2015), Pflug & Pichler (2015) and Pflug & Pichler (2016) have addressed different approximation techniques for stochastic processes. But still there is no open-source implementation of the various algorithms in the public domain.

We therefore present `ScenTrees.jl`, a open-source Julia (Bezanson, Edelman, Karpinski, & Shah, 2017) package for generating scenario trees and scenario lattices which can be used, for example, for multistage stochastic optimization problems. It allows users to represent possible sequences of stochastic processes in form of a `scenario tree` in the case of a discrete time stochastic process and a `scenario lattice` for Markovian data processes. In extension, it also provides users with a platform for generating new and additional trajectories in case of a limited data using conditional density estimation. The theory and design of the `ScenTrees.jl` package follows the concept in Pflug & Pichler (2015) directly. It starts with an initial tree, which is a qualified guess by expert opinion, and uses the stochastic approximation procedure to improve the values on the nodes of the tree with samples form the stochastic process to be approximated. The transition probabilities from one node to another are also addressed.

To assess the quality of this approximation, a distance between the initial distribution and its approximation is defined. Typically, we want an approximating tree that has a minimal distance to the original process. We therefore employ the `process distance` (also called `multistage distance`) (Pflug, 2009) to quantify the quality of approximation of the scenario tree. The process distance extends and generalizes the Wasserstein distance to stochastic processes. It was analyzed by Pflug & Pichler (2012) and used by Kovacevic & Pichler (2015) directly to generate scenario trees.

Main features of the package

`ScenTrees.jl` is generally applicable to any type of stochastic process. The key features of the packages are:

1. Generation of scenario trees and scenario lattices from stochastic processes using the stochastic approximation procedure. Here, the branching structure of the scenario tree or the scenario lattice is fixed and then stochastic approximation procedure is used to improve/update the values of the nodes considering all the data available for every iteration. This improvement goes on until the specified number of iterations have been performed and then the process distance is calculated.[1]
2. Generation of scenarios based on limited data using conditional density estimation. In the case of a limited data with an unknown distribution, we employ conditional density estimation to generate new and different samples based on the these available trajectories. The newly generated samples are able to capture the necessary and important characteristics in the original data as well as patterns in the original data. These samples can thus be used in the stochastic approximation procedure to generate scenario trees or scenario lattices.[2]

Implementation details and various examples to demonstrate different methods can be found in the package's documentation.[3]

The following section provides a typical example on how to use `ScenTrees.jl` to generate scenario trees and scenario lattices by combining stochastic approximation procedure and conditional density estimation procedure.

Example: Scenario generation from observed trajectories

Consider the following comma-separated limited data. The data has 1000 trajectories in 5 stages. To use `ScenTrees.jl`, the first step is to load the package as well as the data into Julia as follows.

```
# Load the package
julia> using ScenTrees, CSV
# Load the data from a directory
julia> df = CSV.read("../RData.csv");
# Convert the DataFrame into a Matrix
julia> data = Matrix(df);
```

The following shows an example of a non-Markovian trajectory generated from the data using conditional density estimation by employing the Logistic distribution for the kernels.

```
julia> Example = kernel_scenarios(data, Logistic; Markovian = false)()
[1.5595,0.8150,1.5058,2.6475,4.6137]
```

The generated data has a length equal to the number of columns of the original data. These generated trajectories are the ones we will use to approximate a scenario tree and a scenario lattice in the following sections.

[1] Tutorial4: https://kirui93.github.io/ScenTrees.jl/latest/tutorial/tutorial4/
[2] Tutorial41: https://kirui93.github.io/ScenTrees.jl/latest/tutorial/tutorial41/
[3] Documentation: https://kirui93.github.io/ScenTrees.jl/stable/

Approximation with a scenario tree

We want to approximate the above data using a scenario tree with a branching vector `(1,3,3,3,2)` and `1,000,000` iterations as follows.

```
julia> kernTree = tree_approximation!(Tree([1,3,3,3,2],1),
                  kernel_scenarios(data, Logistic; Markovian = false),
                  100000,2,2);
julia> tree_plot(kernTree)
julia> savefig("rwdataTree.pdf")
```

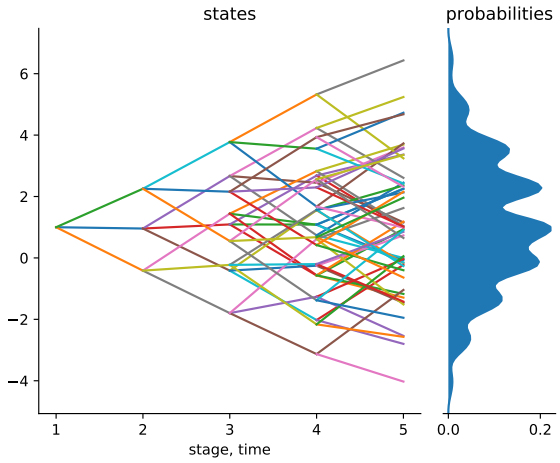

Figure 1: Tree Approximation from Kernel Density Samples

The number of possible trajectories in the scenario tree equals the number of leaves in that scenario tree. In the above scenario tree, there are 54 possible trajectories since the number of leaves is $(1 \times 3 \times 3 \times 3 \times 2) = 54$. The algorithm returns the multistage distance between the above scenario tree and the original stochastic process as $d = 0.23092$.

Approximation with a scenario lattice

Consider a scenario lattice with a branching vector $(1, 3, 4, 5, 6)$. Clearly, this scenario lattice has 5 stages as shown by the number of elements in the branching vector, which is equal to number of columns in the data. We consider $1,000,000$ iterations for the stochastic approximation algorithm and `r=2` parameter for the multistage distance. To generate this scenario lattice, we need Markovian trajectories and therefore we set `Markovian = true` to specify that the trajectories to be generated are Markovian, which is different for scenario trees.

```
julia> rwdataLattice = lattice_approximation([1,3,4,5,6],
                       kernel_scenarios(data, Logistic; Markovian=true),
                      1000000, 2);
julia> plot_lattice(rwdataLattice)
julia> savefig("rwdataLattice.pdf")
```

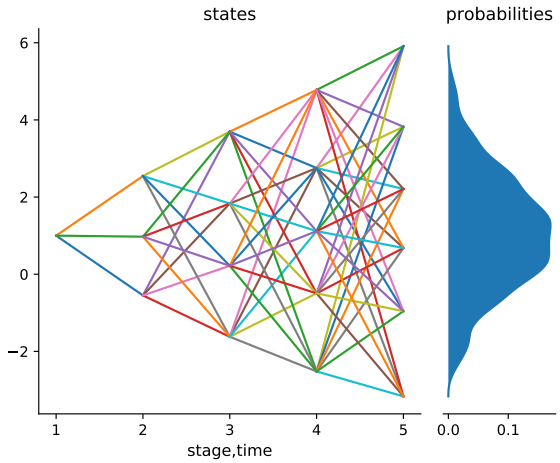

Figure 2: Lattice Approximation from Kernel Density Samples

One should note that the number of nodes in the scenario lattice is equal to the sum of elements in the branching structure, i.e., $1 + 3 + 4 + 5 + 6 = 19$ nodes. The scenario tree considered above has 94 nodes and 54 possible scenarios while the scenario lattice above has 19 nodes and $(1 \times 2 \times 3 \times 4 \times 5) = 360$ scenarios. This shows generally with fewer number of nodes can have many possible trajectories than a scenario tree with more number of nodes. The algorithm returns the multistage distance between the scenario lattice and the original process as $d = 1.1718$.

Acknowledgments

This work was supported by the German Academic Exchange Service (DAAD).

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. doi:10.1137/141000671

Høyland, K., & Wallace, S. W. (2001). Generating scenario trees for multistage decision problems. *Management Science*, *47*(2), 295–307. doi:10.1287/mnsc.47.2.295.9834

Kovacevic, R. M., & Pichler, A. (2015). Tree approximation for discrete time stochastic processes: A process distance approach. *Annals of Operations Research*, *235*(1), 395–421. doi:10.1007/s10479-015-1994-2

Pflug, G. C. (2001). Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, *89*(2), 251–271. doi:10.1007/s101070000202

Pflug, G. C., & Pichler, A. (2012). A distance for multistage stochastic optimization models. *SIAM Journal on Optimization*, *22*(1), 1–23. doi:10.1137/110825054

Pflug, G. C., & Pichler, A. (2015). Dynamic generation of scenario trees. *Computational Optimization and Applications*, *62*(3), 641–668. doi:10.1007/s10589-015-9758-0

Pflug, G. C., & Pichler, A. (2016). From empirical observations to tree models for stochastic optimization: Convergence properties. *SIAM Journal on Optimization*, *26*(3), 1715–1740. doi:10.1137/15M1043376

Pflug, G. Ch. (2009). Version-independence and nested distributions in multistage stochastic optimization. *SIAM Journal on Optimization*, *20*(3), 1406–1420. doi:10.1137/080718401