FractionalDTimeSeries

Calculate the fractional derivative of a time series

Definition ⓘ

```
In[]:= FractionalDTimeSeries::tol =
         "Argument `1` generates more weights than total data points in
           the time series. Please, increase it.";

In[]:= FractionalDTimeSeries[ts_TemporalData, d_?NonNegative, t : _?Positive : 0.0001 ] := Module[
         {timeSeries = ts, diffOrder = d, tolerance = t,
          fdSeries, idxDiff, check, k, timeSeriesValues, totalWeights, weights},
         weights = Reverse[NestWhileList[k = 0;
             (-# * ((diffOrder - k + 1) / k)) &, 1, (k++; Abs[#] > tolerance) &,
             1, ∞, -1]];
         totalWeights = Length[weights];
         timeSeriesValues = QuantityMagnitude@timeSeries["Values"];
         check = (Length[timeSeriesValues] ≥ totalWeights);
         (idxDiff = Range[totalWeights, Length[timeSeriesValues]];
           fdSeries = (weights.timeSeriesValues⟦#⟦1⟧ ;; #⟦2⟧⟧) & /@
             Transpose[{Range[Length[idxDiff]], idxDiff}];
           TimeSeries[fdSeries, {timeSeries["Dates"]⟦totalWeights ;;⟧},
            TemporalRegularity → True]) /; If[check, True,
           ResourceFunction["ResourceFunctionMessage"][FractionalDTimeSeries::tol, t];
           False]]
```

Documentation

Usage ⓘ

FractionalDTimeSeries[*tseries*, *d*]

applies a fractional derivative of order *d* to the time series *tseries*.

FractionalDTimeSeries[*tseries*, *d*, *tol*]

uses tolerance value *tol* for memory preservation.

Details & Options ⓘ

- FractionalDTimeSeries[*tseries*, *d*, *tol*] represents a fractional difference operator of order *d* for the time series *tseries*, given a tolerance *tol*, as formulated by Marcos Lopez de Prado (2018).

- The order d should be a positive real.
- By default, the tolerance is set to 0.0001.

Examples

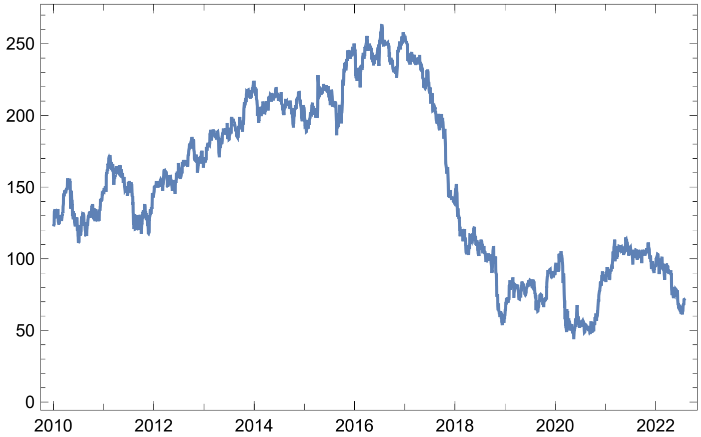
Basic Examples

Get a time series of prices and visualize it:

```
In[∘]:= ts = FinancialData["GE", "Close", "Jan. 1, 2010"]

Out[∘]= TimeSeries[ + Time: 04 Jan 2010 to 27 Jul 2022
                      Data points: 3163 ]
```

```
In[∘]:= DateListPlot@ts
```

Out[∘]=

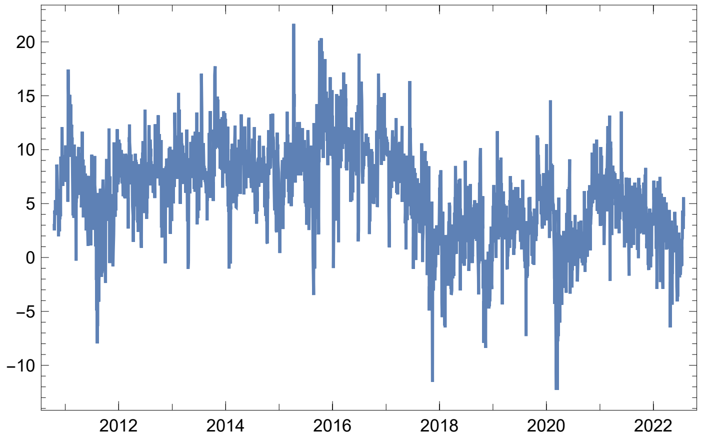


Compute the half difference of the time series and visualize it:

```
In[∘]:= fdseries = FractionalDTimeSeries[ts, 0.5]

Out[∘]= TimeSeries[ + Time: 18 Oct 2010 to 27 Jul 2022
                      Data points: 2964 ]
```

```
In[◦]:= DateListPlot@fdseries
```

Out[◦]=



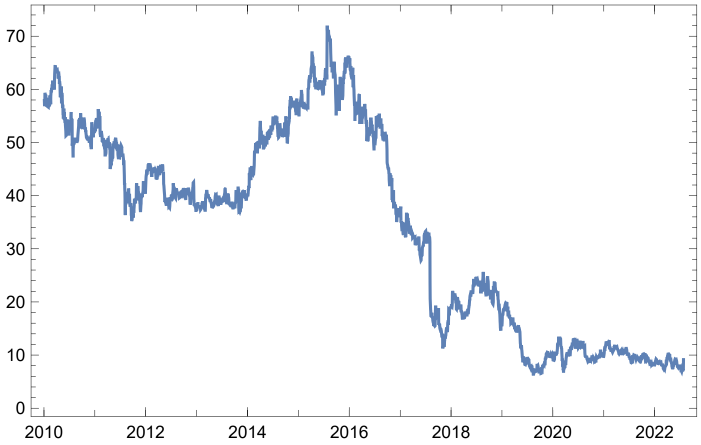
Scope

Get a time series of prices:

```
In[◦]:= ts = FinancialData["TEVA", "Close", "Jan. 1, 2010"]

Out[◦]= TimeSeries[ ⊞ Time: 04 Jan 2010 to 27 Jul 2022
                      Data points: 3162 ]
```

```
In[◦]:= DateListPlot@ts
```

Out[◦]=

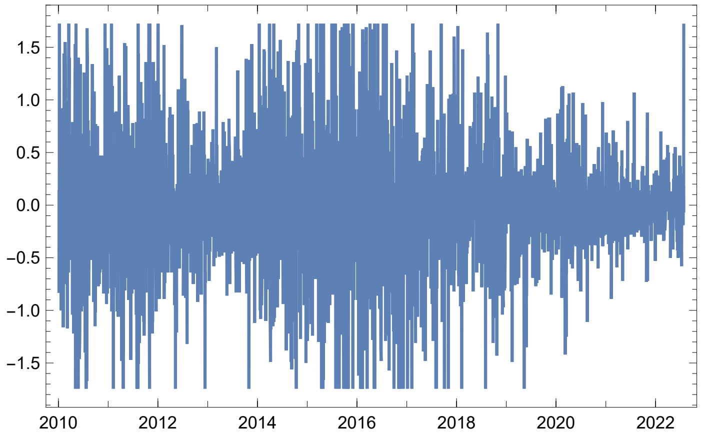


Computing the FractionalDTimeSeries of order 1 leads to the usual first difference:

```
In[◦]:= firstDiff = FractionalDTimeSeries[ts, 1]

Out[◦]= TimeSeries[ ⊞ Time: 05 Jan 2010 to 27 Jul 2022
                      Data points: 3161 ]
```

```
In[∘]:= DateListPlot@firstDiff
```

Out[∘]=

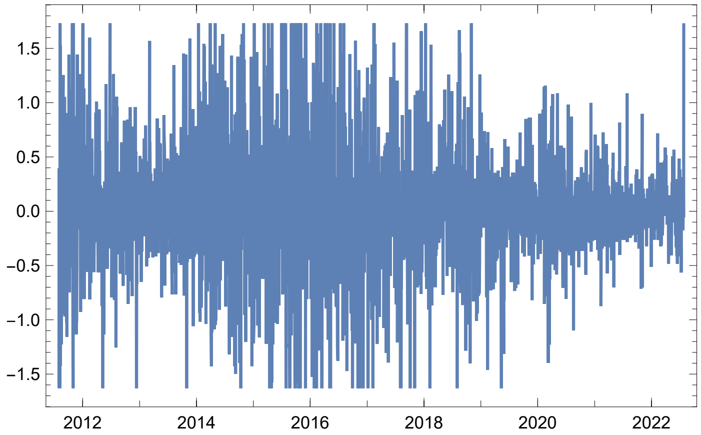


However, applying the half-difference twice on the original time series yields an equivalent first difference, but with fewer data points and on a slightly different scale:

```
In[∘]:= doublefdseries = FractionalDTimeSeries[FractionalDTimeSeries[ts, 0.5], 0.5]

Out[∘]= TimeSeries[ ⊞  Time: 02 Aug 2011 to 27 Jul 2022
                       Data points: 2764 ]
```

```
In[∘]:= DateListPlot@doublefdseries
```

Out[∘]=



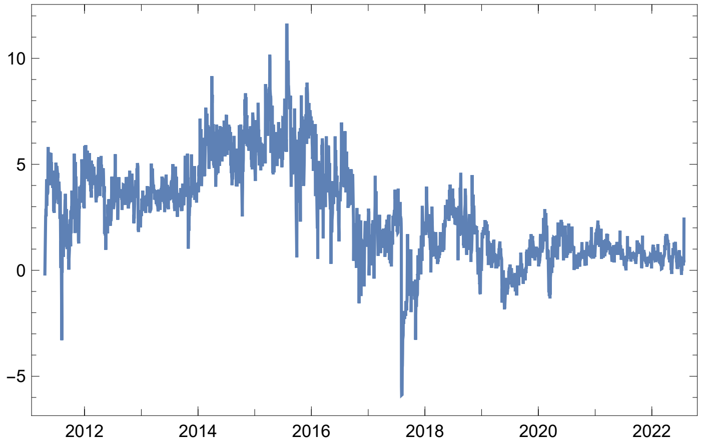
Compute the FractionalDTimeSeries of order 0.3531 (with default $tol = 0.0001$) to the original time series:

```
In[∘]:= ts = FinancialData["TEVA", "Close", "Jan. 1, 2010"]

Out[∘]= TimeSeries[ ⊞  Time: 04 Jan 2010 to 27 Jul 2022
                       Data points: 3162 ]
```

```
In[∘]:= fdseries = FractionalDTimeSeries[ts, 0.3531]

Out[∘]= TimeSeries[ ⊞  Time: 21 Apr 2011 to 27 Jul 2022
                       Data points: 2834 ]
```

```
In[∘]:= DateListPlot@fdseries
```

Out[∘]=

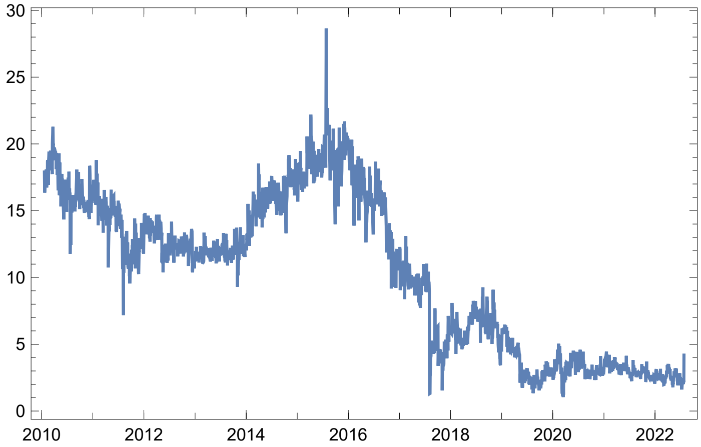


In the same time series, a FractionalDTimeSeries of order 0.3531 and *tol* = 0.01 will generate a differentiated time series with more data points:

```
In[∘]:= fdseries = FractionalDTimeSeries[ts, 0.3531, 0.01]

Out[∘]= TimeSeries[ ⊞  Time: 20 Jan 2010 to 27 Jul 2022
                       Data points: 3151 ]
```

```
In[∘]:= DateListPlot@fdseries
```

Out[∘]=



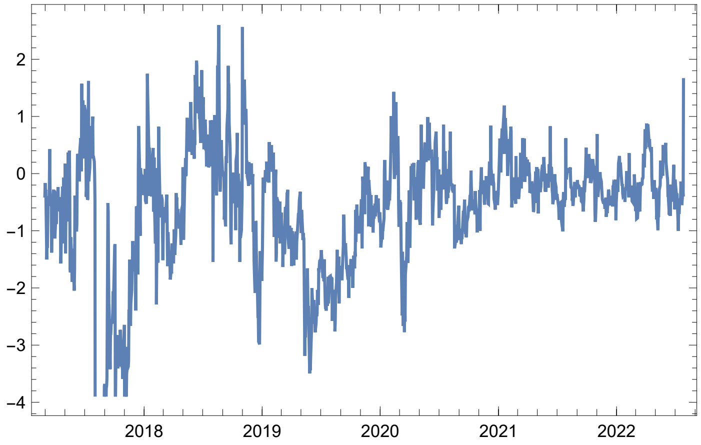
In contrast, a FractionalDTimeSeries of the same order 0.3531, but *tol* = 0.00001, will generate a differentiated time series with *fewer* data points:

In[◦]:=
```
fdseries = FractionalDTimeSeries[ts, 0.3531, 0.00001]
```

Out[◦]=
```
TimeSeries[ ⊞  Time: 28 Feb 2017 to 27 Jul 2022
               Data points: 1362 ]
```

In[◦]:=
```
DateListPlot@fdseries
```

Out[◦]=



Options

Applications

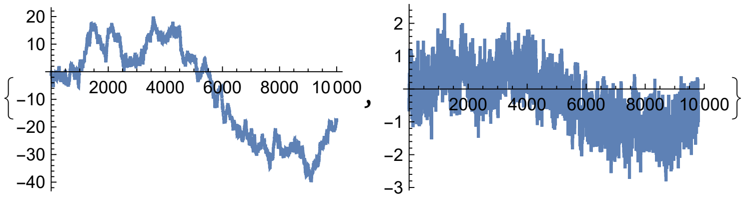
Simulate an ARIMAProcess with linear trend, and apply a fractional differentiation of order 0.5 to it:

In[◦]:=
```
BlockRandom[SeedRandom[170];
 sample = RandomFunction[ARIMAProcess[{-0.1}, 1, {0.2}, 0.1], {1, 1*^4}]]
```

Out[◦]=
```
TemporalData[ ⊞  Time: 1 to 10000
                 Data points: 10000     Paths: 1 ]
```

In[◦]:=
```
{ListLinePlot[sample], ListLinePlot[FractionalDTimeSeries[sample, 0.5]["Values"]]}
```

Out[◦]=



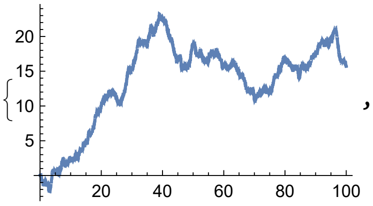
Simulate an FractionalBrownianMotionProcess with Hurst index 0.65, and apply a fractional differentiation of order 0.45 to it:

```
In[∘]:= BlockRandom[SeedRandom[150];
         sample = RandomFunction[FractionalBrownianMotionProcess[0.65], {0, 100, 0.01}]]
```

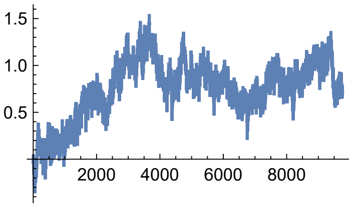
```
Out[∘]= TemporalData[ Time: 0 to 100
                      Data points: 10001    Paths: 1 ]
```

```
In[∘]:= {ListLinePlot[sample], ListLinePlot[FractionalDTimeSeries[sample, 0.45]["Values"]]}
```

Out[∘]= {



,



}

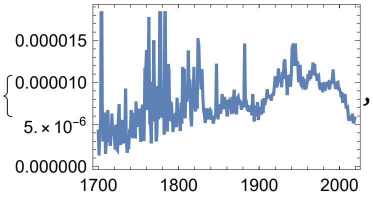
Get a time series of the word "Peru" using WordFrequencyData, and apply a fractional differentiation of order 0.35 and *tol* = 0.01 to it:

```
In[∘]:= sample = WordFrequencyData["Peru", "TimeSeries"]
```

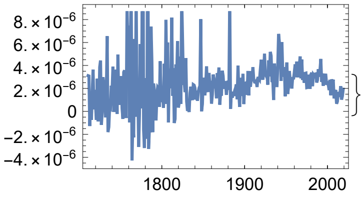
```
Out[∘]= TimeSeries[ Time: 01 Jan 1700 to 01 Jan 2019
                    Data points: 320 ]
```

```
In[∘]:= {DateListPlot[sample], DateListPlot[FractionalDTimeSeries[sample, 0.35, 0.01]]}
```

Out[∘]= {



,



}

Properties and Relations

Possible Issues

FractionalDTimeSeries requires the order to be a positive real number. Negative orders (integration) are currently not implemented:

```
In[∘]:= FractionalDTimeSeries[FinancialData["F", "Close", "Jan. 1, 2000"], -0.5]
```

```
Out[∘]= FractionalDTimeSeries[TimeSeries[ Time: 03 Jan 2000 to 27 Jul 2022
                                           Data points: 5678 ], -0.5]
```

FractionalDTimeSeries requires the tolerance to be positive:

```
In[∘]:= FractionalDTimeSeries[FinancialData["F", "Close", "Jan. 1, 2000"], 0.5, -0.0001]

Out[∘]= FractionalDTimeSeries[TimeSeries[ ⊞  Time: 03 Jan 2000 to 27 Jul 2022
                                                Data points: 5678            ], 0.5, -0.0001]
```

If the tolerance is too small, no transformation will be performed:

```
In[∘]:= FractionalDTimeSeries[FinancialData["F", "Close", "Jan. 1, 2000"], 0.5, 0.0000001]
```

••• ResourceFunction: FractionalDTimeSeries::tol: Argument 1. × 10^{-7} generates more weights than total data points in the time series. Please, increase it.

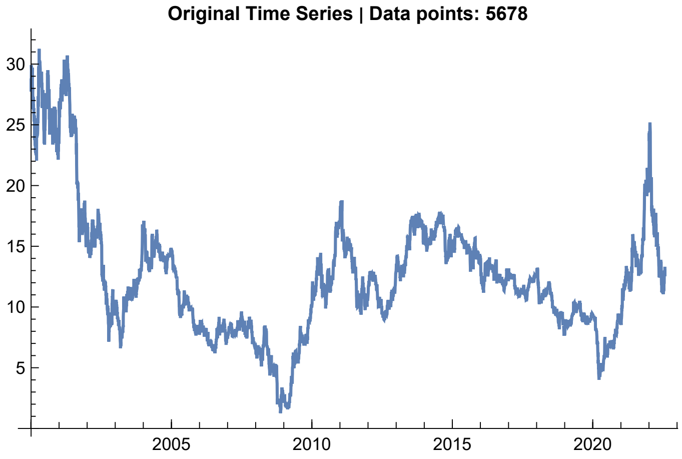
```
Out[∘]= FractionalDTimeSeries[TimeSeries[ ⊞  Time: 03 Jan 2000 to 27 Jul 2022
                                                Data points: 5678            ], 0.5, 1. × 10^-7]
```

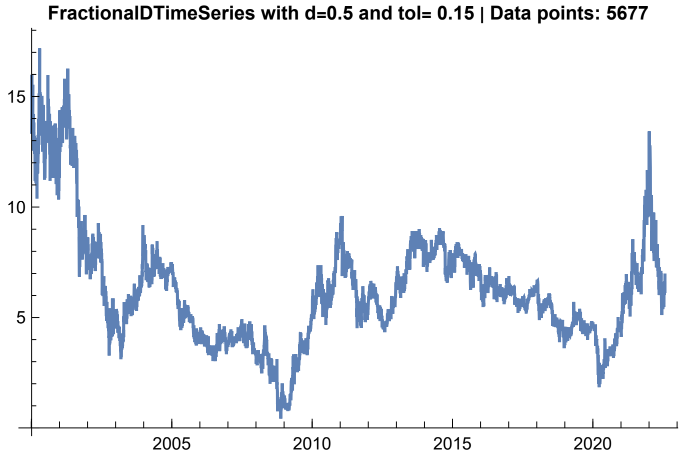
If the tolerance is set too high, the transformed time series will be essentially the same as the original one, but in a different scale:

```
In[∘]:= ts = FinancialData["F", "Close", "Jan. 1, 2000"];
        fdshightol = FractionalDTimeSeries[ts, 0.5, 0.15]

Out[∘]= TimeSeries[ ⊞  Time: 04 Jan 2000 to 27 Jul 2022
                          Data points: 5677            ]
```

In[ ]:=
```
Row[
 {ListLinePlot[ts, ImageSize → 350, PlotLabel →
    Style["Original Time Series | Data points: " <> ToString@(Length@(ts["Values"])),
     Darker@Black, 10, Bold]], ListLinePlot[fdshightol, ImageSize → 350,
   PlotLabel → Style["FractionalDTimeSeries with d=0.5 and tol= 0.15 | Data points: " <>
      ToString@(Length@(fdshightol["Values"])),
     Darker@Black, 10, Bold]]}]
```

Out[ ]=





Neat Examples

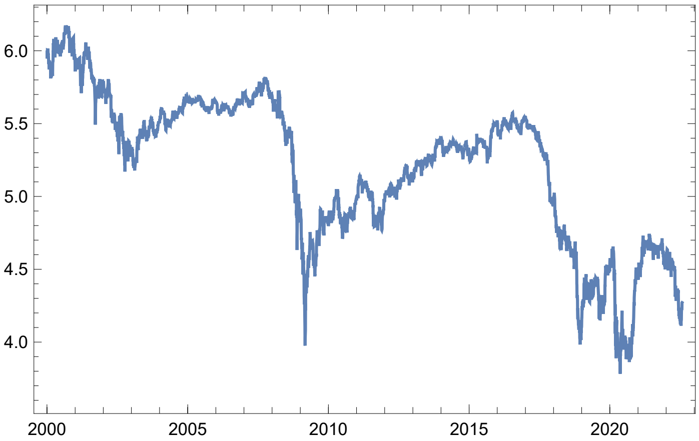
A logarithmically-transformed time series:

In[ ]:=
```
ts = Log@QuantityMagnitude@FinancialData["GE", "Close", "Jan. 1, 2000"]
```

Out[ ]=
```
TimeSeries[ Time: 03 Jan 2000 to 27 Jul 2022
            Data points: 5679 ]
```

```
In[∘]:= DateListPlot@ts
```

Out[∘]=



Find the relationship between the statistical value of a UnitRootTest and the CorrelationFunction of a transformed time series, by defining several values for d^* between 0 and 1, and using them as the order of differentiation:

```
In[∘]:= baseDs = Range[0, 1, 0.005];
        setFSeries = Table[{d, FractionalDTimeSeries[ts, d]}, {d, baseDs}];
```

Apply a UnitRootTest (red axis) on each differentiated time series to see up from which level of d is possible to reject H_0. Also, compute its CorrelationFunction at lag 1 (blue axis) to compare:

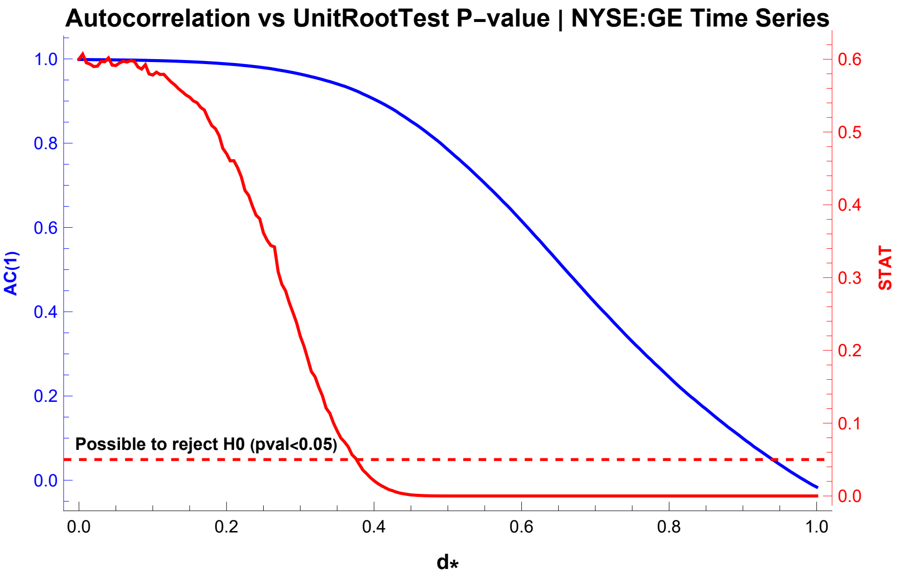
```
In[∘]:= statACFDataPlot = {#⟦1⟧, CorrelationFunction[#⟦2⟧, 1], UnitRootTest@#⟦2⟧} & /@ setFSeries;
```

Plot the results:

```
In[∘]:= corrPlot = ListLinePlot[
          {#[[1]], #[[2]]} & /@ statACFDataPlot, ImagePadding → {{50, 50}, {45, 2}},
          PlotStyle → Blue, Frame → {True, True, False, False},
          FrameStyle → {Automatic, Blue, Automatic, Automatic},
          FrameTicks → {None, All, None, None},
          PlotLabel → Style[
            "Autocorrelation vs UnitRootTest P-value | NYSE:GE Time Series", Black, 13, Bold],
          FrameLabel → {{Style["AC(1)", Bold], None}, {Style["d*", 11, Bold], None}},
          ImageSize → 500];
        statPlot = ListLinePlot[
          {#[[1]], #[[3]]} & /@ statACFDataPlot, ImagePadding → {{50, 50}, {45, 2}},
          PlotStyle → Red,
          Frame → {False, False, False, True},
          FrameTicks → {{None, All}, {None, None}},
          FrameStyle → {Automatic, Automatic, Automatic, Red},
          FrameLabel → {{None, Style["STAT", Bold]}, {None, None}},
          PlotLabel → Style[" "],
          ImageSize → 500,
          GridLines → {None, {0.05}},
          GridLinesStyle → Directive[AbsoluteThickness[3 / 2], Red, Dashed],
          Epilog → {Text[Style["Possible to reject H0 (pval<0.05)", 9, Bold],
             Scaled[{0.015, 0.125}], {-1, 0}]}
         ];
```

```
In[∘]:= Overlay[{corrPlot, statPlot}]
```

Out[∘]=



Source & Additional Information

Contributed By ⓘ

Frank Salvador Ygnacio Rosas

Keywords ⓘ

- Fractional calculus
- Fractional differentiation
- Time series

Categories

☐ Cloud & Deployment
☐ Data Manipulation & Analysis
☐ External Interfaces & Connections
☐ Geographic Data & Computation
☐ Graphs & Networks
☐ Images
☐ Knowledge Representation & Natural Language
☐ Notebook Documents & Presentation
☐ Repository Tools
☐ Social, Cultural & Linguistic Data
☐ Strings & Text
☐ System Operation & Setup
☐ User Interface Construction
☐ Wolfram Physics Project
☐ Core Language & Structure
☐ Engineering Data & Computation
☑ Financial Data & Computation
☐ Geometry
☑ Higher Mathematical Computation
☐ Just For Fun
☐ Machine Learning
☐ Programming Utilities
☐ Scientific and Medical Data & Computation
☐ Sound & Video
☑ Symbolic & Numeric Computation
☑ Time–Related Computation
☐ Visualization & Graphics

Related Symbols ⓘ

- FractionalD
- ItoProcess
- TimeSeries

Related Resource Objects

- Resource Name (resources from any Wolfram repository)

Source/Reference Citation

- Lopez de Prado, M. (2018). *Advances in Financial Machine Learning*. Wiley, New Jersey, 75-88.
- Walasek, R. and Gajda, J. (2021). "Fractional differentiation and its use in machine learning". *International Journal of Advances in Engineering Sciences and Applied Mathematics*, Vol.13, No. 3, pp. 270-277. DOI: 10.1007/s12572-021-00299-5

Links

- Wolfram Community - "Fractional Differentiation and fBm: a study of long-memory processes"

Tests

```
In[ ]:= MyFunction[x, y]
Out[ ]= x y
```

Compatibility

Wolfram Language Version

13.0+

Operating System

Required Features

Environments

Cloud Support

Author Notes

Additional information about limitations, issues, etc.

Submission Notes

Additional information for the reviewer.