Replication_code_section4

September 20, 2024

1 Replication code for Section 4, Song et al. (2024)

This file replicates the output in Section 4 of

"PyTimeVar: A Python Package for Trending Time-Varying Time Series Models."

Authors: Mingxuan Song, Bernhard van der Sluis, and Yicong Lin.

Date current version: September 2024

Section 4 illustrates our package to study time-varying herding effects in the Chinese renewable energy market.

[1]:
```python
from PyTimeVar import BoostedHP
from PyTimeVar import LocalLinear
from PyTimeVar import GAS
from PyTimeVar import Kalman
from PyTimeVar.datasets import herding
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

1.1 Replication code for Figure 7 in Section 4.1

[2]:
```python
############### Importing data for replicating the results in Section 4.1
np.random.seed(123)
herd_data = herding.load(start_date='2015-01-05', end_date='2022-04-29')
vY = herd_data[['CSAD_AVG']].values
mX = np.ones_like(vY)
```

[3]:
```python
############### Local linear estimation
# LLr_model = LocalLinear(vY=vY, mX=mX, bw_selection='lmcv_6') ##### this␣
 ↪function will cost around 48 mins.
                                                            ##### We use the␣
 ↪resulted bandwidth directly in the next line
LLr_model = LocalLinear(vY=vY, mX=mX, h=0.14)
LLr_trend = LLr_model.fit()
```

```
Bandwidth specified by user:  0.1400
```

```
Note: this bandwidth will be used for any bootstrap implementation.
```

```
[4]: ############### Boosted HP filter
bHPmodel = BoostedHP(vY=vY, dLambda=1600, iMaxIter=100)
bHPtrend, bHPresiduals = bHPmodel.fit(boost=True, stop="bic",dAlpha=0.05,␣
 ↪verbose=False)
```

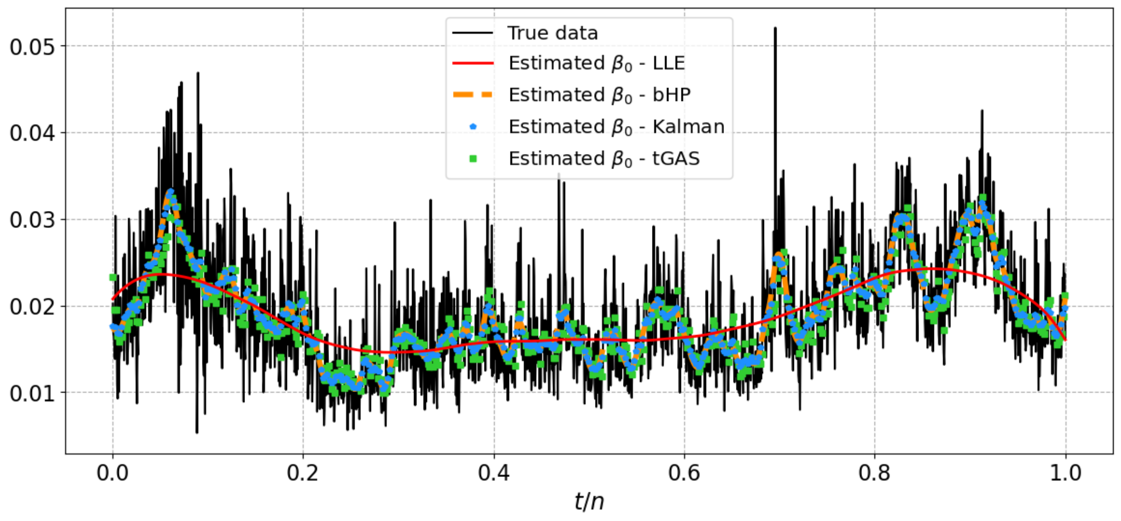
```
[5]: ############### Kalman smoother
kalmanmodel = Kalman(vY=vY, mX=mX)
smooth_trend = kalmanmodel.fit('smoother')
```

```
Optimization success
```

```
[6]: ############### t-GAS model
gasmodel = GAS(vY=vY, mX=mX, method='student')
tGAStrend, tGASparams = gasmodel.fit()
```

```
Time taken: 194.81 seconds
```

```
[7]: ############### Replication code for Figure 7
x_axis_trend = np.arange(1/len(vY),(len(vY)+1)/len(vY),1/len(vY))
plt.figure(figsize=(14, 6))

line2, = plt.plot(x_axis_trend, vY, label='True data',color='black')
line4, = plt.plot(x_axis_trend, tGAStrend, label='Estimated $\\beta_{0}$ -␣
 ↪tGAS',color='limegreen',linestyle='',marker='s',markersize=5,markevery=3)
line5, = plt.plot(x_axis_trend, bHPtrend, label='Estimated $\\beta_{0}$ -␣
 ↪bHP',color='darkorange',linestyle='--',linewidth=4)
line1, = plt.plot(x_axis_trend, smooth_trend, label='Estimated $\\beta_{0}$ -␣
 ↪Kalman',color='dodgerblue',linestyle='',marker='p',markersize=4,markevery=4)
line3, = plt.plot(x_axis_trend, LLr_trend[0], label='Estimated $\\beta_{0}$ -␣
 ↪LLE',color='red', linewidth=2)

# Customize the plot
plt.legend(handles=[line2, line3, line5, line1, line4],fontsize="x-large")
plt.xlabel('$t/n$',fontsize="xx-large")
plt.tick_params(axis='both', labelsize=16)
plt.grid(linestyle='dashed')

plt.show()
```

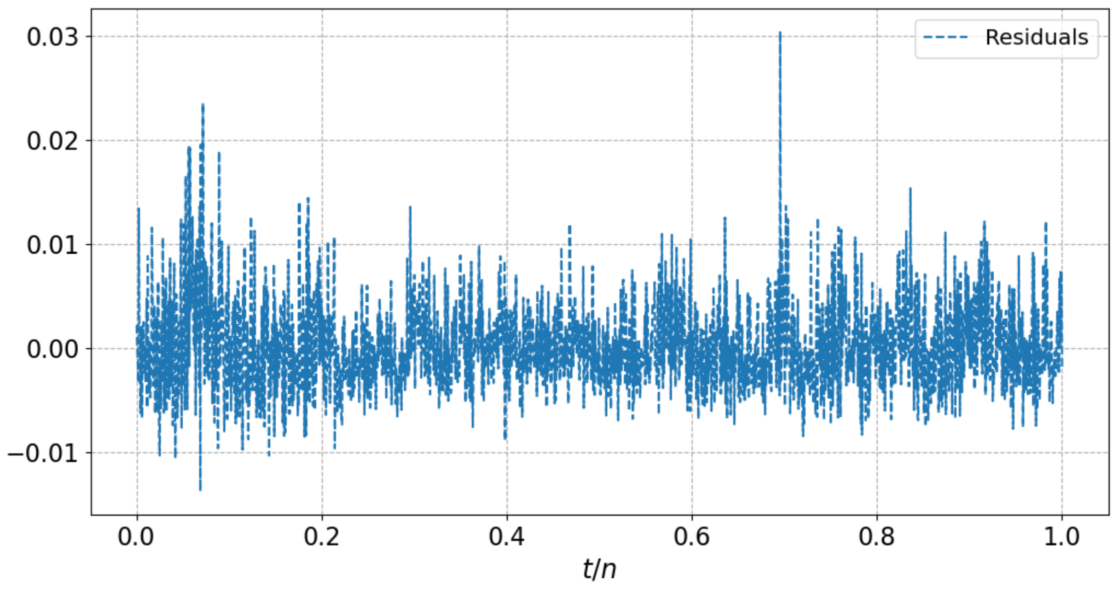


1.2 Replication code for Figure 8 in Section 4.2

[9]:
```
############### Importing data for replicating the results in Section 4.2 and␣
 ↪Appendix A.5
vY,mX = herding.load(data_replication=True)
breakindex_1 = 284
breakindex_5 = 1459
np.random.seed(123)

############### Local linear estimation
LLr_model = LocalLinear(vY=vY, mX=mX, h=0.0975)
LLr_betahat = LLr_model.fit()
```

```
Bandwidth specified by user:  0.0975

Note: this bandwidth will be used for any bootstrap implementation.
```

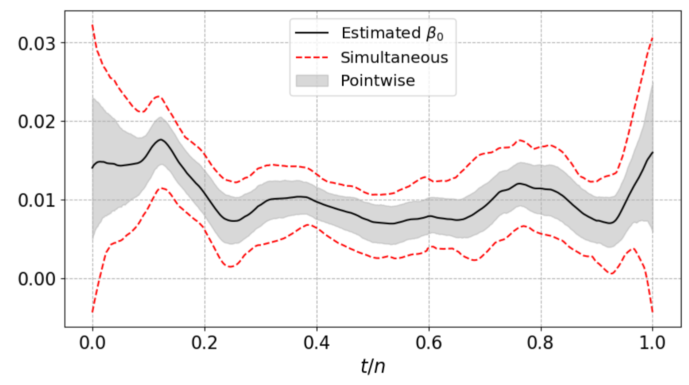
[10]:
```
############### The code below replicates Figure 8 in Section 4.2
############### plot residuals, replication of 8a
residuals = LLr_model.plot_residuals()
```



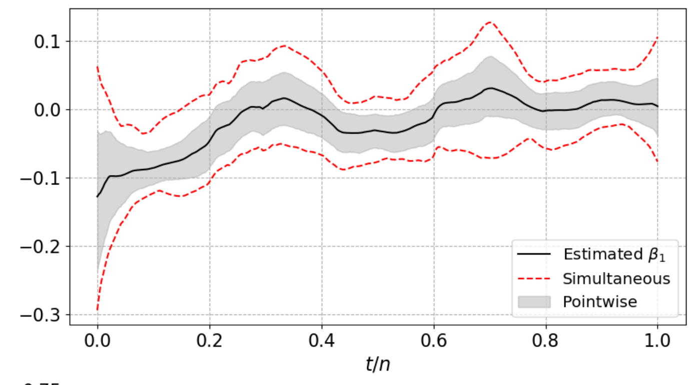
```
[11]: ############### plot LBWB full sample bands, replication of 8b.
      ############### Only the plot of beta_3 is reported in the paper (takes about␣
       ↪27 mins).
      S_LB_full, S_UB_full, P_LB_full, P_UB_full = LLr_model.
       ↪confidence_bands(plots=True)
```

```
Calculating LBWB Bootstrap Samples
```

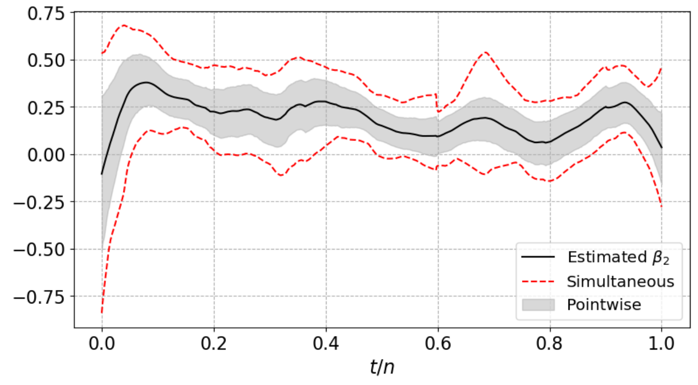
```
100%|      | 1299/1299 [27:20<00:00,  1.26s/it]
```



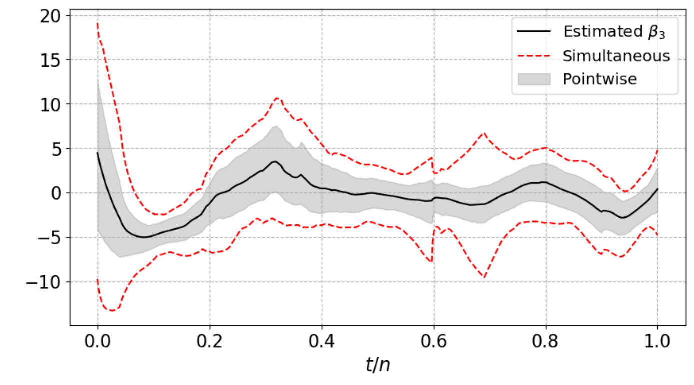
Estimated β_0
Simultaneous
Pointwise
t/n



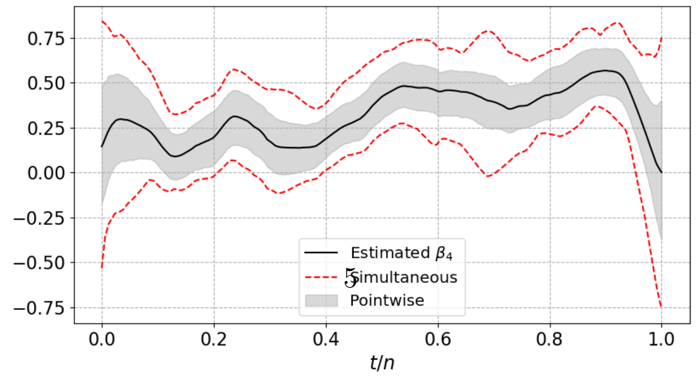
Estimated β_1
Simultaneous
Pointwise
t/n



Estimated β_2
Simultaneous
Pointwise
t/n



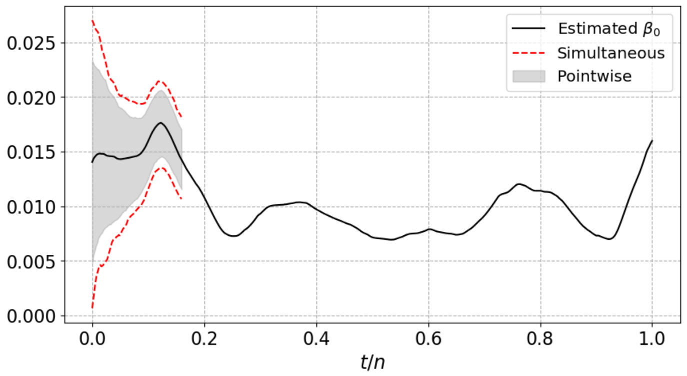
Estimated β_3
Simultaneous
Pointwise
t/n

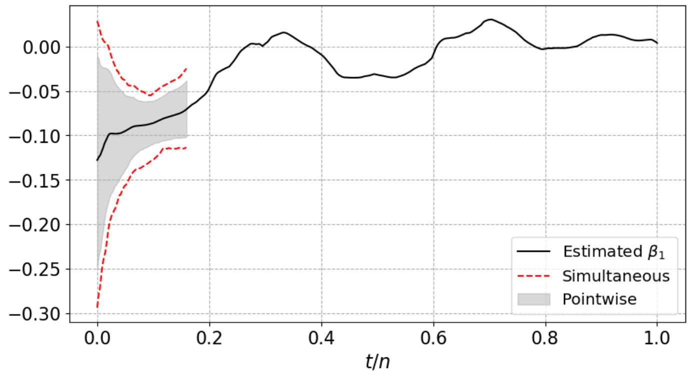


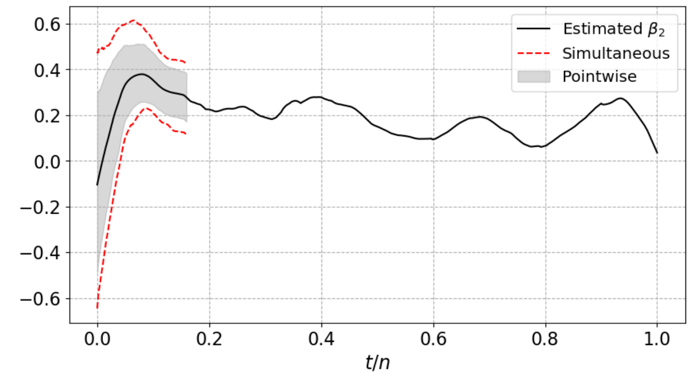
Estimated β_4
Simultaneous
Pointwise
t/n

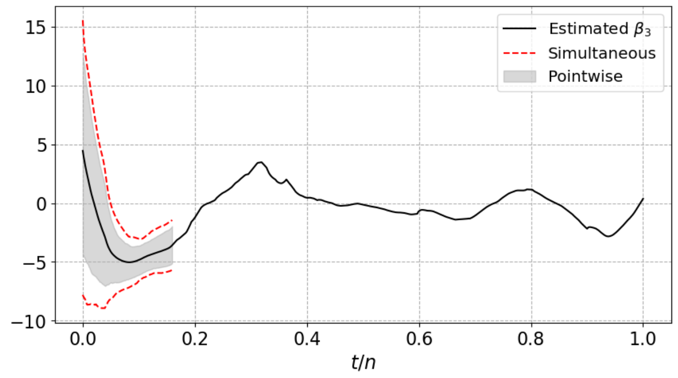
```
[12]: ############### plot LBWB G1 bands, replication of 8c.
      ############### Only the plot of beta_3 is reported in the paper (takes about 8␣
       ↪mins).
      S_LB_G1, S_UB_G1, P_LB_G1, P_UB_G1 = LLr_model.confidence_bands(plots=True,␣
       ↪Gsubs=[(0, breakindex_1)])
```

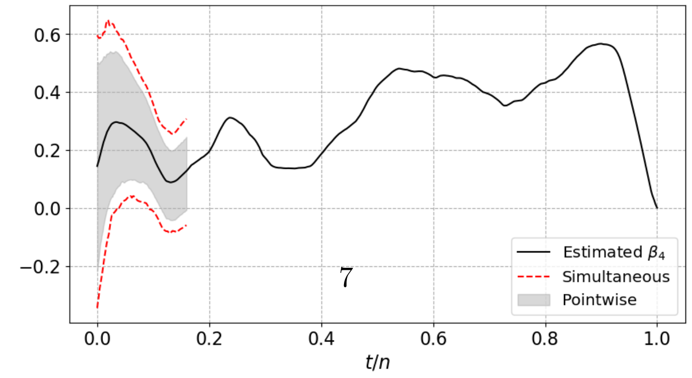
```
Calculating LBWB Bootstrap Samples

100%|      | 1299/1299 [06:35<00:00,  3.28it/s]
```


Estimated β_0
Simultaneous
Pointwise
0.000
0.005
0.010
0.015
0.020
0.025
0.0
0.2
0.4
0.6
0.8
1.0
t/n


Estimated β_1
Simultaneous
Pointwise
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
0.0
0.2
0.4
0.6
0.8
1.0
t/n

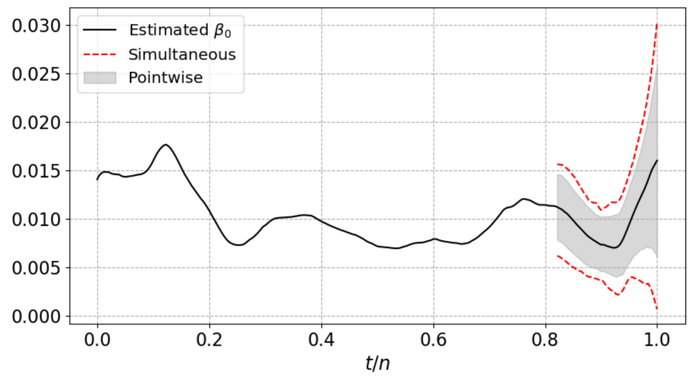

Estimated β_2
Simultaneous
Pointwise
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
t/n

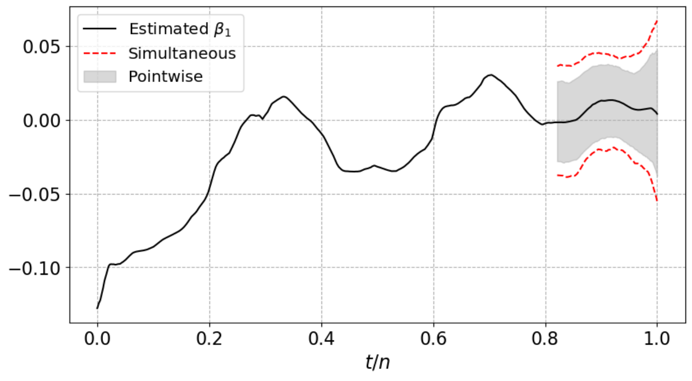

Estimated β_3
Simultaneous
Pointwise
15
10
5
0
−5
−10
0.0
0.2
0.4
0.6
0.8
1.0
t/n

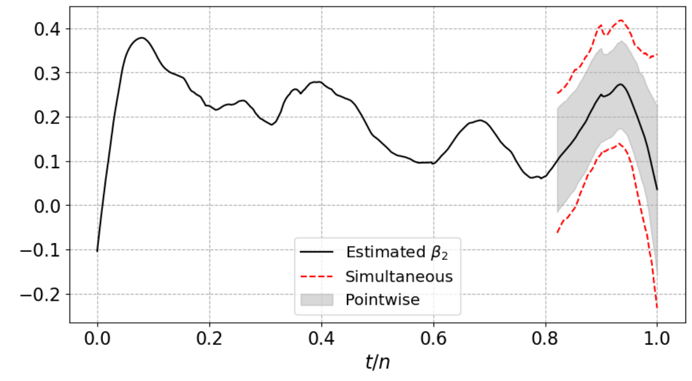

Estimated β_4
Simultaneous
Pointwise
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
t/n

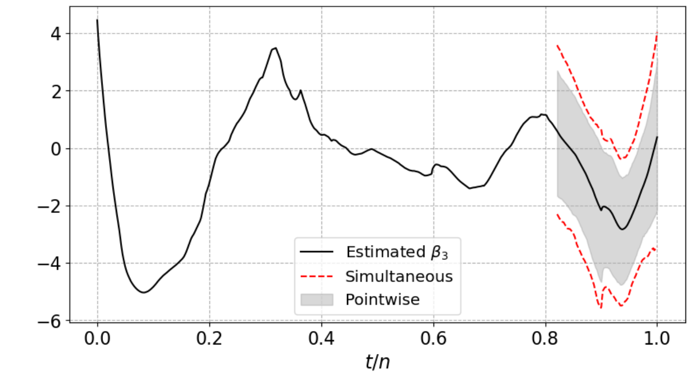
```
[14]: ############### plot LBWB G6 bands, replication of 8d.
      ############### Only the plot of beta_3 is reported in the paper (takes about 9␣
       ↪mins).
      S_LB_G6, S_UB_G6, P_LB_G6, P_UB_G6 = LLr_model.confidence_bands(plots=True,␣
       ↪Gsubs=[(breakindex_5, len(vY))])
```

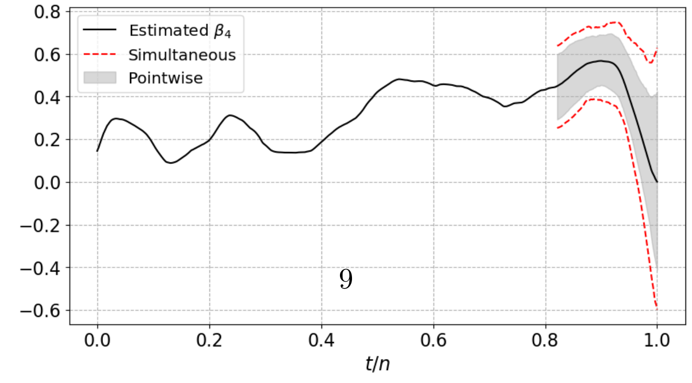
```
Calculating LBWB Bootstrap Samples

100%|      | 1299/1299 [08:56<00:00,  2.42it/s]
```


Estimated β_0
Simultaneous
Pointwise
t/n


Estimated β_1
Simultaneous
Pointwise
t/n


Estimated β_2
Simultaneous
Pointwise
t/n


Estimated β_3
Simultaneous
Pointwise
t/n


Estimated β_4
Simultaneous
Pointwise
t/n

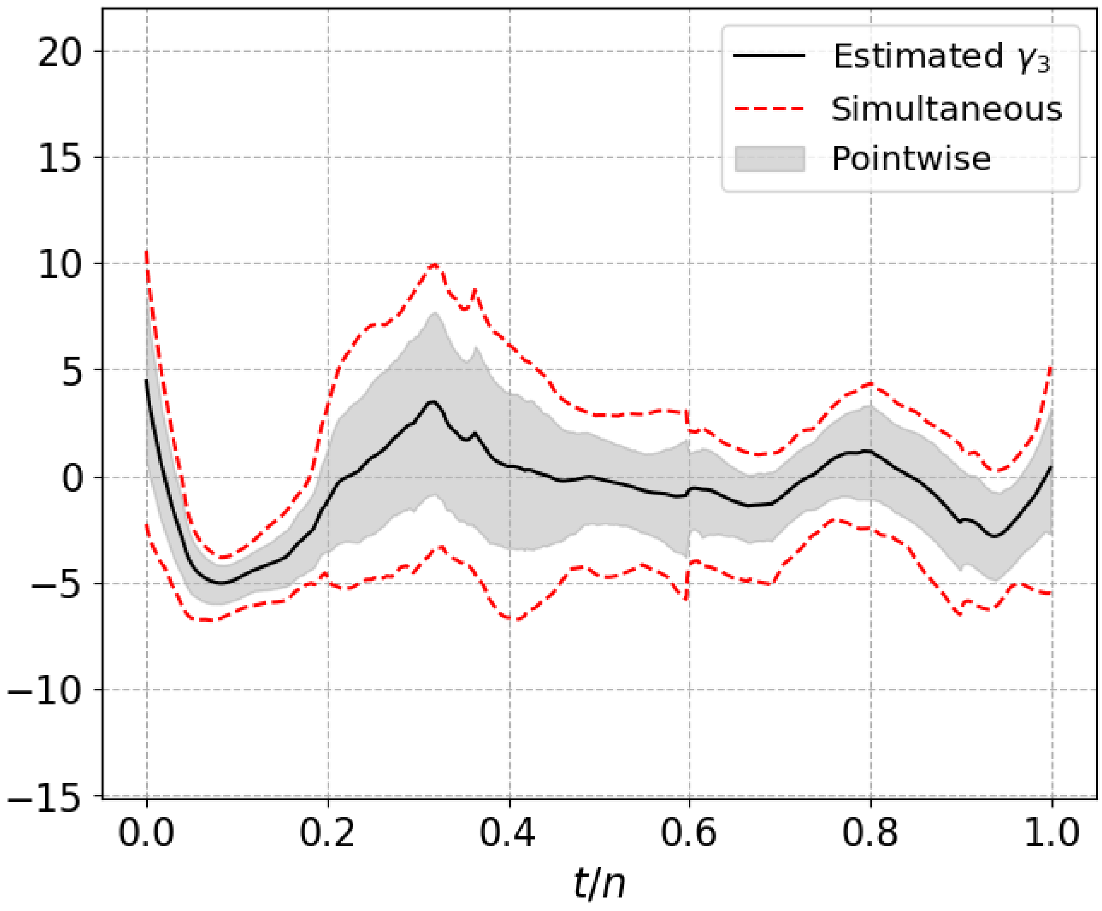
## Replication code for Figure 9 in Appendix A.5

[13]:
```
############### Define the function to unify the format of plots as in Figure 9
def plot_figure9(LLr_betahat, S_LB_full, S_UB_full, P_LB_full, P_UB_full):
    x_axis=np.arange(0,1,1/len(vY))
    plt.figure(figsize=(7.5, 6))
    plt.plot(x_axis, LLr_betahat[3], color='black',  label='Estimated␣
 ↪$\gamma_3$')
    plt.plot(x_axis, S_LB_full[3], color='red', linestyle='dashed')
    plt.plot(x_axis, S_UB_full[3], color='red', linestyle='dashed',␣
 ↪label='Simultaneous')
    plt.fill_between(x_axis, P_LB_full[3], P_UB_full[3], color='grey', alpha=0.
 ↪3, label='Pointwise')
    plt.grid(linestyle='dashed')
    plt.xlabel('$t/n$',fontsize="xx-large")
    plt.ylim(-15.1, 22)
    plt.tick_params(axis='both', labelsize=16)
    plt.legend(fontsize="x-large")
    plt.show()
```

[32]:
```
############### Sieve bootstrap (SB) bands, Figure 9a (takes about 35 mins)
S_LB_full_SB, S_UB_full_SB, P_LB_full_SB, P_UB_full_SB = LLr_model.
 ↪confidence_bands(plots=False, bootstrap_type="SB")
```

```
Calculating SB Bootstrap Samples

100%|      | 1299/1299 [46:43<00:00,  2.16s/it]
```

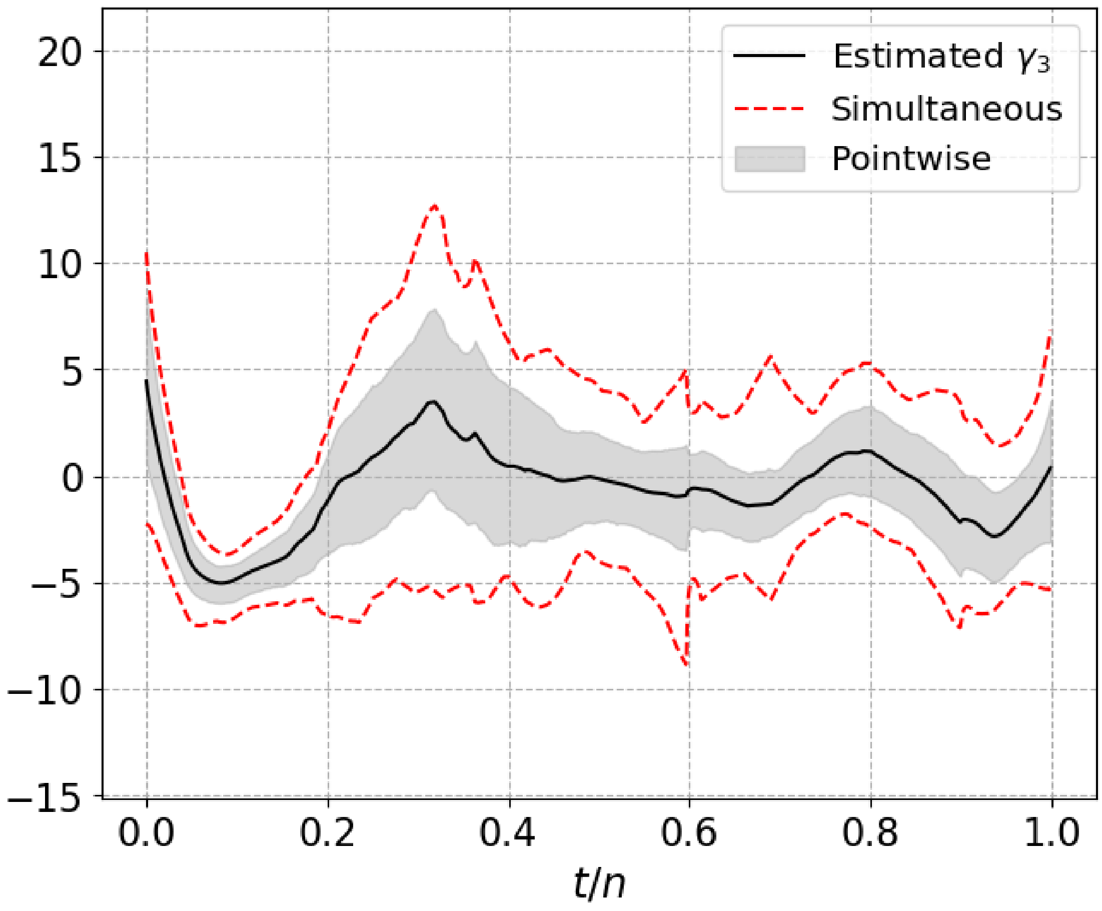
[44]:
```
############### replication code for Figure 9a
plot_figure9(LLr_betahat, S_LB_full_SB, S_UB_full_SB, P_LB_full_SB,␣
 ↪P_UB_full_SB)
```



[38]:
```
############### Sieve wild bootstrap (SWB) bands, Figure 9b (takes about 33␣
↪mins)
S_LB_full_SWB, S_UB_full_SWB, P_LB_full_SWB, P_UB_full_SWB = LLr_model.
↪confidence_bands(plots=False, bootstrap_type="SWB")
```

```
Calculating SWB Bootstrap Samples

100%|      | 1299/1299 [33:09<00:00,  1.53s/it]
```

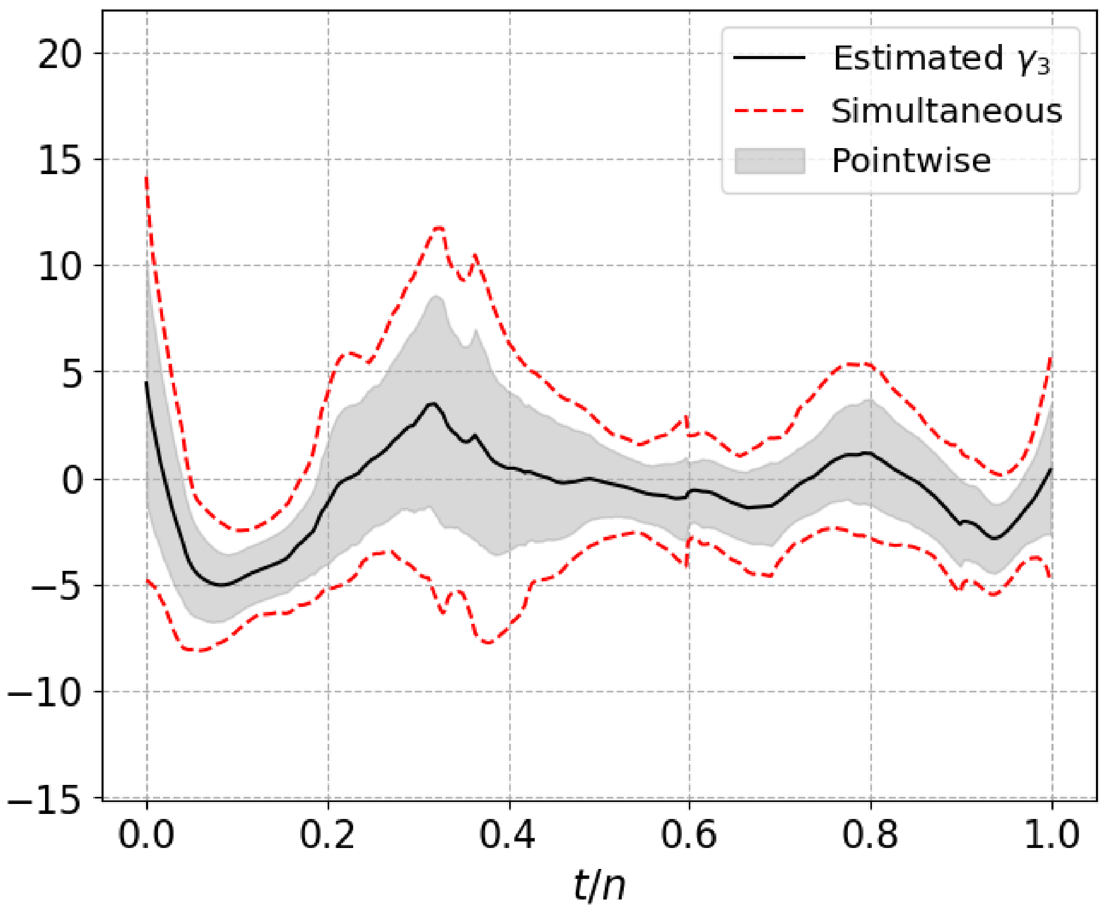
[49]:
```
############### replication code for Figure 9b
plot_figure9(LLr_betahat, S_LB_full_SWB, S_UB_full_SWB, P_LB_full_SWB,␣
↪P_UB_full_SWB)
```



```
[42]: ############### Wild bootstrap (WB) bands, Figure 9c (takes about 28 mins)
      S_LB_full_WB, S_UB_full_WB, P_LB_full_WB, P_UB_full_WB = LLr_model.
       ↪confidence_bands(plots=False, bootstrap_type="WB")
```

```
Calculating WB Bootstrap Samples

100%|          | 1299/1299 [27:57<00:00,  1.29s/it]
```

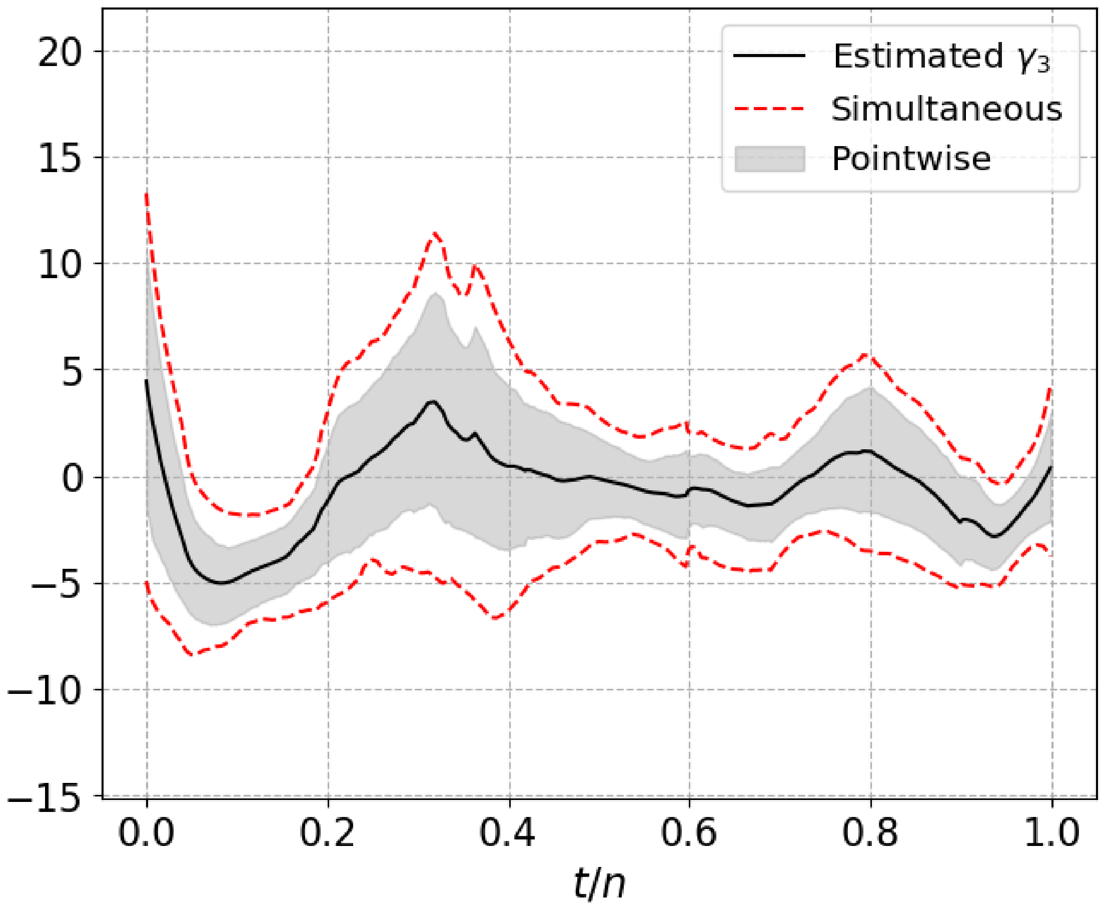
```
[46]: ############### replication code for Figure 9c
      plot_figure9(LLr_betahat, S_LB_full_WB, S_UB_full_WB, P_LB_full_WB,␣
       ↪P_UB_full_WB)
```



```
[47]: ############### Autoregressive wild bootstrap (AWB) bands, Figure 9d (takes␣
       ↪about 27 mins)
      S_LB_full_AWB, S_UB_full_AWB, P_LB_full_AWB, P_UB_full_AWB = LLr_model.
       ↪confidence_bands(plots=False, bootstrap_type="AWB")
```

```
Calculating AWB Bootstrap Samples

100%|      | 1299/1299 [28:23<00:00,  1.31s/it]
```

```
[48]: #### ########### replication code for Figure 9d
      plot_figure9(LLr_betahat, S_LB_full_AWB, S_UB_full_AWB, P_LB_full_AWB,␣
       ↪P_UB_full_AWB)
```


Estimated γ_3
Simultaneous
Pointwise
20
15
10
5
0
−5
−10
−15
0.0
0.2
0.4
0.6
0.8
1.0
t/n