pybats-detection: Quick start guide

André Menezes and Eduardo Gabriel

24 fevereiro, 2022

Contents

1 Step 0: Install `pybats-detection`

Use:

```
>>> git clone git@github.com:Murabei-OpenSource-Codes/pybats-detection.git pybats-detection
>>> cd pybats-detection
>>> python setup.py install
```

The `pybats-detection` provides two main modules, namely: `Intervention` and `Monitoring`. For each class we shall show a quick start guide of how to use them.

2 `Intervention` class

2.1 Step 1: Load the required modules

```
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> from pybats.dglm import dlm
```

```
>>> from matplotlib.pyplot import figure
>>> from pybats_detection.loader import load_cp6
>>> from pybats_detection.intervention import Intervention
```

2.2 Step 2: Load or import your data

Some data from literature are included with the package. The `load_cp6()` function load the time series of monthly total sales of tobacco and related products marketed by a major company in the UK.

```
>>> cp6 = load_cp6()
>>> cp6.head()
```

```
##          time  sales
## 0 1955-01-01    620
## 1 1955-02-01    633
## 2 1955-03-01    652
## 3 1955-04-01    652
## 4 1955-05-01    661
```

2.3 Step 3: Define the model

```
>>> a = np.array([600, 1])
>>> R = np.array([[100, 0], [0, 25]])
>>> mod = dlm(a0=a, R0=R, ntrend=2, deltrend=[0.90, 0.98])
>>> mod.get_coef()
```

```
##              Mean  Standard Deviation
## Intercept     600                10.0
## Local Slope     1                 5.0
```

2.4 Step 4: Specify the interventions

```
>>> list_interventions = [
>>>     {"time_index": 12, "which": ["variance", "noise"],
>>>      "parameters": [{"v_shift": "ignore"},
>>>                     {"h_shift": np.array([0, 0]),
>>>                      "H_shift": np.array([[1000, 25], [25, 25]])}]
>>>      },
>>>     {"time_index": 25, "which": ["noise", "variance"],
>>>      "parameters": [{"h_shift": np.array([80, 0]),
>>>                      "H_shift": np.array([[100, 0], [0, 0]])},
>>>                     {"v_shift": "ignore"}]},
>>>     {"time_index": 37, "which": ["subjective"],
>>>      "parameters": [{"a_star": np.array([970, 0]),
>>>                      "R_star": np.array([[50, 0], [0, 5]])}]}
>>> ]
```

2.5 Step 5: Initialize the Intervention class

```
>>> dlm_intervention = Intervention(mod=mod)
```

2.6 Step 6: Use the method `fit`

```
>>> results = dlm_intervention.fit(
>>>     y=cp6["sales"], interventions=list_interventions)
```

2.7 Step 7: Examine the results

```
>>> results.keys()
```

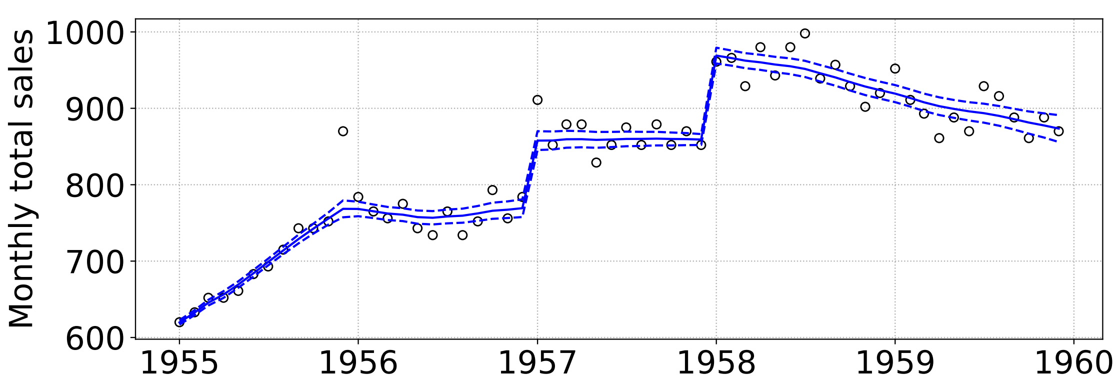
```
## dict_keys(['filter', 'smooth', 'model'])
```

```
>>> dict_smooth = results.get("smooth")
>>> data_posterior = dict_smooth.get("posterior")
>>> data_level = data_posterior[data_posterior["parameter"] == "Intercept"].copy()
>>> data_level.head()
```

```
##    parameter        mean  variance  t    ci_lower    ci_upper
## 0  Intercept  619.752268  2.150703  1  616.819766  622.684771
## 1  Intercept  632.628581  2.620156  2  629.391811  635.865351
## 2  Intercept  645.582048  3.487580  3  641.847738  649.316358
## 3  Intercept  656.292416  4.831934  4  651.896911  660.687920
## 4  Intercept  669.266732  4.427634  5  665.059135  673.474328
```

2.8 Step 8: Plot the results

```
>>> figure(figsize=(12, 4))
>>> plt.plot(cp6["time"], cp6["sales"], "o",
>>>          markersize=6, color="black", fillstyle="none")
>>> plt.plot(cp6["time"], data_level["mean"], color="blue")
>>> plt.plot(cp6["time"], data_level["ci_lower"], color="blue",
>>>          linestyle="dashed")
>>> plt.plot(cp6["time"], data_level["ci_upper"], color="blue",
>>>          linestyle="dashed")
>>> plt.grid(linestyle="dotted")
>>> plt.xlabel("Time")
>>> plt.ylabel("Monthly total sales")
>>> plt.show()
```



3 Monitoring class

3.1 Step 1: Load the required modules

```
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> from pybats.dglm import dlm
>>> from matplotlib.pyplot import figure
>>> from pybats_detection.monitor import Monitoring
>>> from pybats_detection.loader import load_telephone_calls
```

3.2 Step 2: Load or import your data

The `load_telephone_calls()` function load the time series of the average number of calls per day in each month to Cincinnati directory assistance.

```
>>> telephone_calls = load_telephone_calls()
>>> telephone_calls.head()
```

```
##          time  average_daily_calls
## 0 1962-01-01                  350
## 1 1962-02-01                  339
## 2 1962-03-01                  351
## 3 1962-04-01                  364
## 4 1962-05-01                  369
```

3.3 Step 3: Define the model

```
>>> a = np.array([350, 0])
>>> R = np.eye(2)
>>> np.fill_diagonal(R, val=[100])
>>> mod = dlm(a, R, ntrend=2, deltrend=0.95)
>>> mod.get_coef()
```

```
##              Mean  Standard Deviation
## Intercept     350                10.0
## Local Slope     0                10.0
```

3.4 Step 4: Initialize the `Monitoring` class

```
>>> monitor = Monitoring(mod=mod, bilateral=True, prior_length=40)
```

3.5 Step 6: Use the method `fit`

```
>>> results = monitor.fit(y=telephone_calls["average_daily_calls"], h=4,
>>>                       tau=0.135, change_var=[6, 2], type="location",
>>>                       distr="normal", verbose=True)
```

```
## Upper potential outlier detected at time 48 with H=1.1424e-01, L=1.1424e-01 and l=1
## Upper potential outlier detected at time 60 with H=2.6440e-03, L=2.6440e-03 and l=1
## Upper potential outlier detected at time 72 with H=9.3858e-02, L=9.3858e-02 and l=1
## Lower parametric change detected at time 83 with H=1.1669e+02, L=3.8854e-15 and l=7
## Upper parametric change detected at time 97 with H=8.2742e-01, L=1.9329e-04 and l=3
```

```
## Lower potential outlier detected at time 115 with H=5.2280e-02, L=5.2280e-02 and l=1
## Lower parametric change detected at time 138 with H=1.4859e+01, L=1.4859e+01 and l=3
## Lower potential outlier detected at time 140 with H=2.0949e-03, L=2.0949e-03 and l=1
## Lower potential outlier detected at time 141 with H=1.3331e-07, L=1.3331e-07 and l=1
## Lower potential outlier detected at time 142 with H=1.8002e-02, L=1.8002e-02 and l=1
## Lower potential outlier detected at time 146 with H=8.9106e-06, L=8.9106e-06 and l=1
## Lower potential outlier detected at time 147 with H=1.2986e-16, L=1.2986e-16 and l=1
## Lower potential outlier detected at time 148 with H=1.9833e-05, L=1.9833e-05 and l=1
```

3.6 Step 7: Examine the results

```
>>> results.keys()
```

```
## dict_keys(['filter', 'smooth', 'model'])
```

```
>>> dict_filter = results.get("filter")
>>> dict_filter.keys()
```

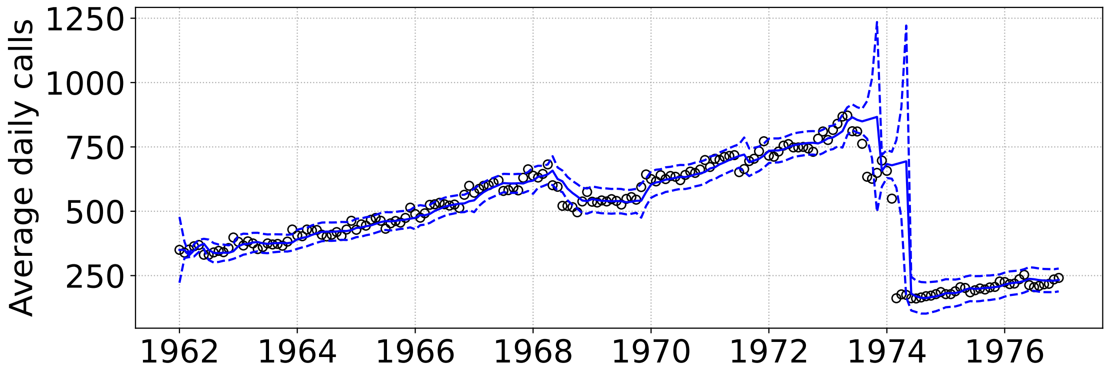
```
## dict_keys(['predictive', 'posterior'])
```

```
>>> data_predictive = dict_filter.get("predictive")
>>> data_predictive.head()
```

```
##    t  prior    y           f  ...  l_upper  what_detected    ci_lower    ci_upper
## 0  1   True  350  350.000000  ...        1        nothing  222.304223  477.695777
## 1  2   True  339  350.000000  ...        1        nothing  318.483933  381.516067
## 2  3   True  351  328.311860  ...        1        nothing  321.629479  334.994242
## 3  4   True  364  348.024290  ...        1        nothing  324.103892  371.944688
## 4  5   True  369  365.042878  ...        1        nothing  341.620552  388.465204
##
## [5 rows x 16 columns]
```

3.7 Step 8: Plot the results

```
>>> figure(figsize=(12, 4))
>>> plt.plot(telephone_calls["time"], telephone_calls["average_daily_calls"], "o",
>>>          markersize=6, color="black", fillstyle="none")
>>> plt.plot(telephone_calls["time"], data_predictive["f"], color="blue")
>>> plt.plot(telephone_calls["time"], data_predictive["ci_lower"], color="blue",
>>>          linestyle="dashed")
>>> plt.plot(telephone_calls["time"], data_predictive["ci_upper"], color="blue",
>>>          linestyle="dashed")
>>> plt.grid(linestyle="dotted")
>>> plt.xlabel("Time")
>>> plt.ylabel("Average daily calls")
>>> plt.show()
```


Average daily calls
1250
1000
750
500
250
1962
1964
1966
1968
1970
1972
1974
1976