`pybats-detection`: A python package for outlier and structural changes detection in time series analysis

André Menezes and Eduardo Gabriel

16 março, 2022

Contents

Smoothing

A brief introduction of the Smoothing class in a simulated example. A time series $\mathbf{Y} = (y_1, \ldots, y_T)$ was generated using the `RandomDLM` class which has the arguments (n, V, W): the number of observations, observational variance and state vector variance. This class has three methods that simulate data using different mechanisms:

- `.level`: dynamic level model;
- `.growth`: dynamic growth model;
- `.level_with_covariates`: dynamic level model where Y is simulated given X, a matrix of fixed covariates.

For now, we stick with `.level`, simulating $n = 100$ observations with both observational and state vector variance equals to one 1, the starting level is set to 100. The simulated data is plotted below.

```
>>> # Generating level data model
>>> np.random.seed(66)
>>> rdlm = RandomDLM(n=100, V=1, W=1)
>>> df_simulated = rdlm.level(
>>>     start_level=100,
>>>     dict_shift={})
>>> y = df_simulated["y"]
```

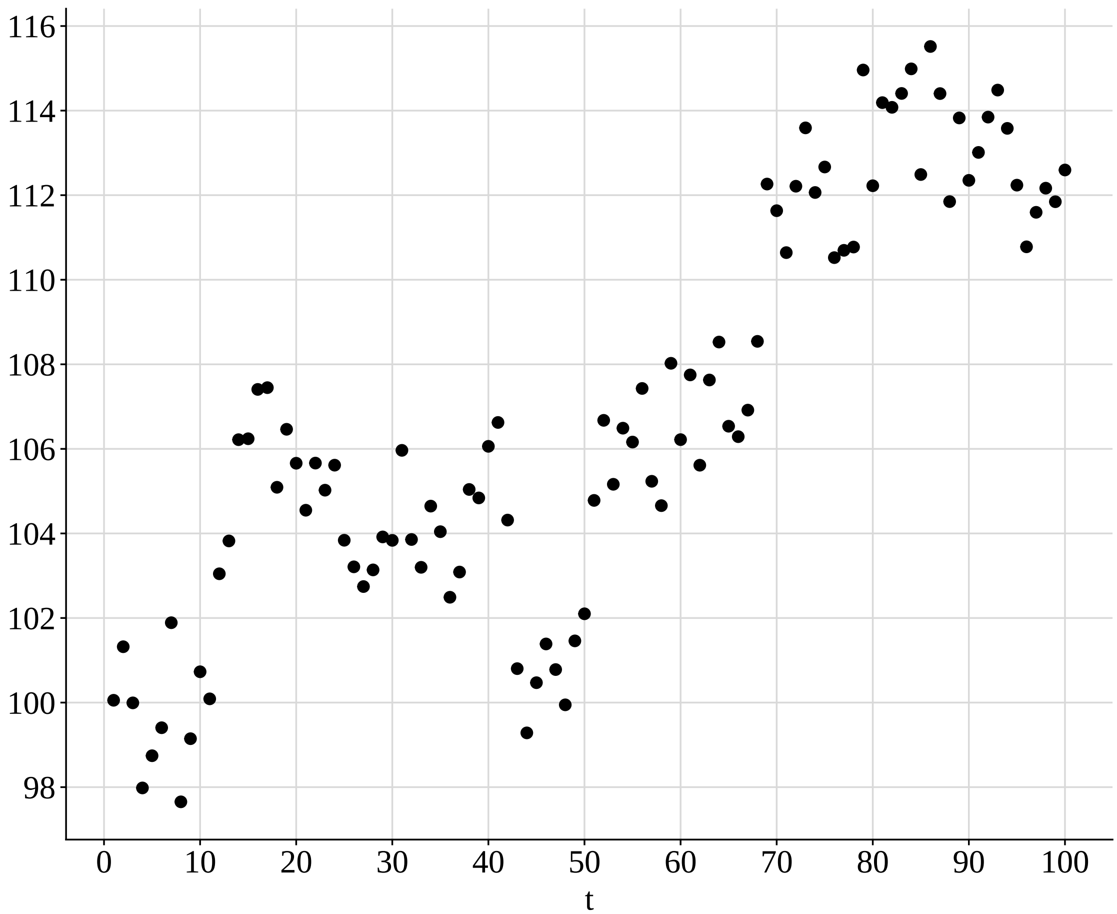

Figure 1: Simulated data

The Smoothing class allows you to perform a retrospective analysis for $\mathbf{Y}$, obtaining the distribution of $(\boldsymbol{\theta}_{T-k}|D_T)$, for $k \geq 1$, the k-step smoothed distribution for the state vector at time T, which is analogous to the k-step ahead forecast distribution $(\boldsymbol{\theta}_{t+k}|D_t)$.

To use Smoothing, first it is necessary to define the model components with prior values, which is done with the `dlm` class available in the `pybats` package. In this case, it was considered a DLM with level and growth. The prior vector and covariances are defined by $\mathbf{a}$ and $\mathbf{R}$. Lastly, the discount factor denoted by `deltrend` is a constant in the interval [0, 1], which is used to coordinate the adaptive capacity of predictions with increasing variance of model components.

```
>>> # Define model components
>>> a = np.array([100, 0])
>>> R = np.eye(2)
>>> np.fill_diagonal(R, val=1)
>>> mod = dlm(a, R, ntrend=2, deltrend=.95)
```

Given this, the method `.fit` will initialize the model and the loop forecast, observe and update begin. The prior and posterior moments $(\mathbf{a}_t, \mathbf{m}_t, \mathbf{C}_t, \mathbf{R}_t)$ will be computed for all t and saved. Subsequently, these moments will be used to obtain the moments for $(\boldsymbol{\theta}_{T-k}|D_T)$, recursively with $k \geq 1$, and denoted by $(\mathbf{a}_T(-k), \mathbf{m}_T(-k), \mathbf{C}_T(-k), \mathbf{R}_T(-k))$.

```
>>> # Fit with monitoring
>>> smooth = Smoothing(mod=mod)
>>> smooth_fit = smooth.fit(y=y)
```

This will return a dictionary with moments for: smoothed and filtered predictive distributions and for the posterior distributions of the model components. Each one can be obtained using the respective key

```
>>> smooth_fit.get('smooth').get('predictive')
```

```
>>> smooth_fit.get('smooth').get('posterior')
```

```
>>> smooth_fit.get('filter').get('predictive')
```

```
>>> smooth_fit.get('filter').get('posterior')
```

Below the results for the predictive and posterior smoothed distributions

smoothed predictive

The results for the smoothed predictive distribution consists of: $f_T(-k), q_T(-k)$ and the bounds for the credibility interval (`ci_lower`, `ci_upper`). Given by

$$f_T(-k) = \mathbf{F}^{'}\mathbf{a}_T(-k), \qquad q_T(-k) = \mathbf{F}^{'}\mathbf{R}_T(-k)\mathbf{F}$$

The credibility interval is is obtained from the corresponding smoothed distributions for the mean response of the series. Since V is considered unknown, then

$$(\mu_T(-k)|D_T) \sim T_{n_T}[f_T(-k), q_T(-k)]$$

For this simulated example, the results for the smoothed predictive distribution for the mean response are

```
>>> smooth_fit.get('smooth').get('predictive').round(2).head()
```

Table 1: Smothed predictive distribution results

qk	t	fk	df	ci_lower	ci_upper
0.31	1	99.97	1	98.85	101.1
0.27	2	100.07	2	99.05	101.1
0.24	3	100.12	3	99.14	101.1
0.23	4	100.20	4	99.24	101.2
0.22	5	100.39	5	99.47	101.3

as for the filtered distribution

```
>>> smooth_fit.get('smooth').get('predictive').round(2).head()
```

Table 2: Filtered predictive distribution results

parameter	mean	variance	t	ci_lower	ci_upper
Intercept	99.97	0.31	1	98.85	101.1
Intercept	100.07	0.27	2	99.05	101.1
Intercept	100.12	0.24	3	99.14	101.1
Intercept	100.20	0.23	4	99.24	101.2
Intercept	100.39	0.22	5	99.47	101.3

Plotting the filtered vs smoothed predictive distributions results is possible to see difference, primarily in the length of the credibility interval.

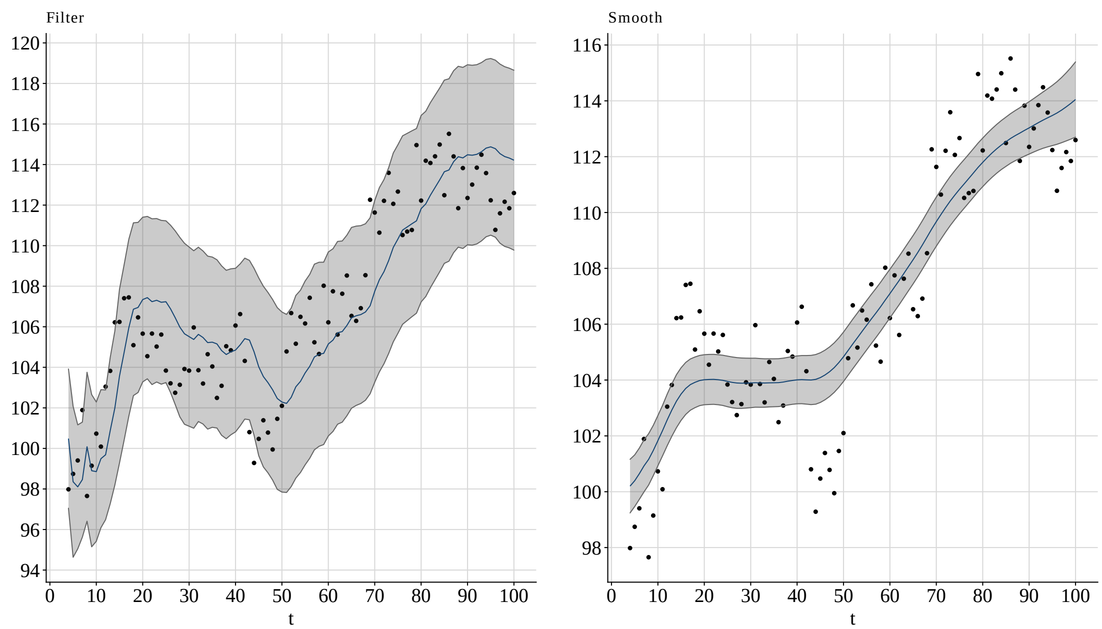

Figure 2: Mean response for Filtered and Smoothed predictive distributions with 95% credibility intervals.

smoothed posterior

The results for the posterior distributions are analogous, where

- parameter: Indicator for the respective state space parameter in $\boldsymbol{\theta}$;
- mean: The smoothed posterior distribution mean for time $t = T - k$ ($\mathbf{m}(-k)$);
- variance: The smoothed posterior distribution variance for time t ($\mathbf{C}(-k)$).
- credibility interval (`ci_lower`, `ci_upper`): The credibility interval obtained from the corresponding smoothed posterior distributions. Since V is considered unknown, then

$$(\boldsymbol{\theta}_{T-k}|D_T) \sim T_{n_T}[\mathbf{a}_T(-k), \mathbf{R}_T(-k)].$$

```
>>> smooth_fit.get('smooth').get('posterior').round(2).head()
```

Table 3: Smothed posterior distribution results

parameter	mean	variance	t	ci_lower	ci_upper
Intercept	99.97	0.31	1	98.85	101.1
Intercept	100.07	0.27	2	99.05	101.1
Intercept	100.12	0.24	3	99.14	101.1
Intercept	100.20	0.23	4	99.24	101.2
Intercept	100.39	0.22	5	99.47	101.3

As before we plot the results for filtered and smoothed distributions, in this case for each state space parameter. As expected, the smoothed posterior distributions show a less erratic behavior with shorter credibility intervals.

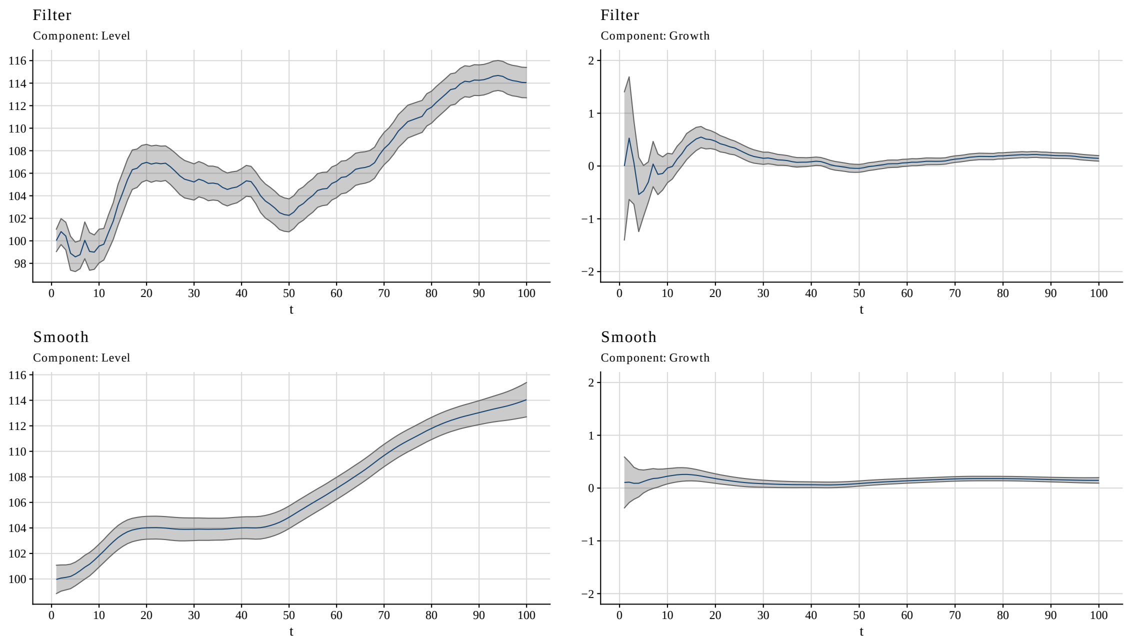

Figure 3: Mean response for Filtered and Smoothed posterior distributions for each model component with 95% credibility intervals.

Aplication: AirPassangers dataset

Below is a practical example with the classic Box & Jenkins airline data, Monthly totals of international airline passengers (1949 to 1960). This data has a clear multiplicative seasonality, using a linear model (with additive seasonality) may be a naive approximation for this data. But, just for the sake of comparison between filtered and smoothing we stick with the linear model.

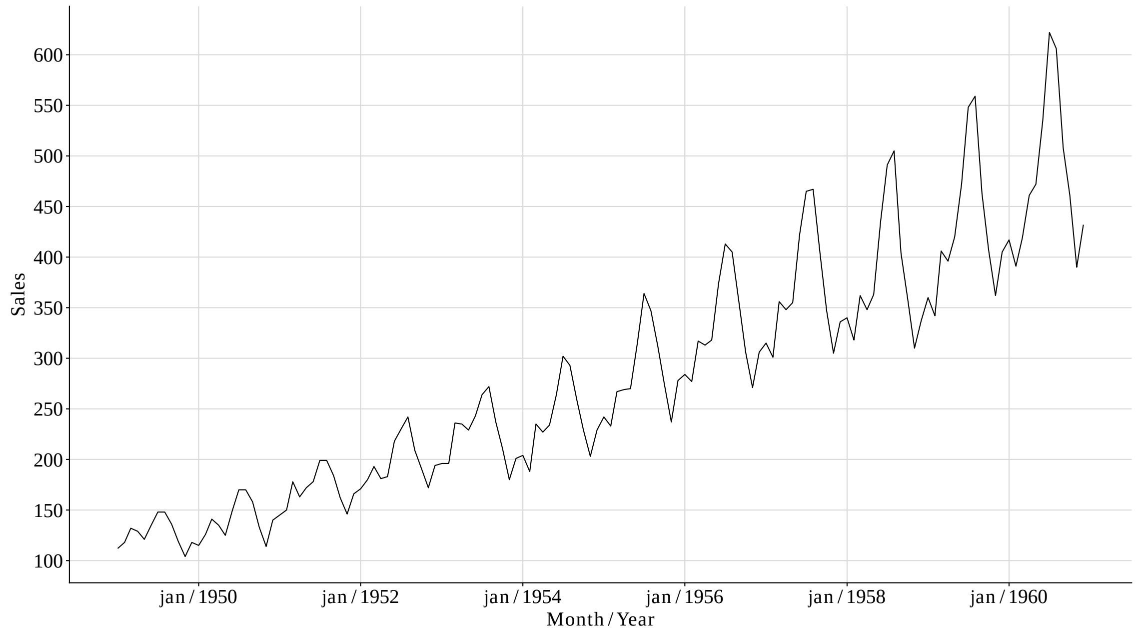

Figure 4: Monthly totals of international airline passengers, 1949 to 1960.

Using a normal DLM with three main components: Trend, Growth and Seasonality. The seasonality is modeled using the Fourier form representation, which depends on the parity of a period p and the number of harmonics components. Formally, the $\mathbf{r}^{th}$ harmonic component is given by

$$S_r(.) = a_r \cos(\alpha r) + b_r \sin(\alpha r), \quad r = 1, \ldots, h, \quad a_r = 2\pi/p, \quad h <= p/2$$

Here it was specified a yearly seasonal effect with period $p = 12$ and the first two harmonics. The discount factor for the level and growth components is set to 0.95, and 0.98 for the seasonal components. The results are plotted below.

```
>>> a = np.array([112, 0, 0, 0, 0, 0])
>>> R = np.eye(6)
>>> np.fill_diagonal(R, val=100)
>>> mod = dlm(a, R, ntrend=2, deltrend=.95, delseas=.98,
>>>           seasPeriods=[12], seasHarmComponents=[[1, 2]])
```

Since the seasonality was modeled using harmonic components, the model has a total of six parameters: level, growth and four for seasonality (a_1, b_1, a_2, b_2). For simplicity, the results for de posterior distributions considered the sum of the harmonic components, whose moments are given by

$$\mu_{seas} = \mathbf{F}'_{seas}\mathbf{a}_T(-k), \qquad \sigma^2_{seas} = \mathbf{F}'_{seas}\mathbf{R}_T(-k)\mathbf{F}_{seas}$$

where $\mathbf{F}'_{seas} = [0, 0, 1, 0, 1, 0]$. The results are illustrated below.

Manual Intervention

CP6

To illustrate the subjective intervention class we use the CP6 data graphed below. This time series runs from January 1955 to December 1959, providing monthly total sales, in monetary terms on a standard scale,

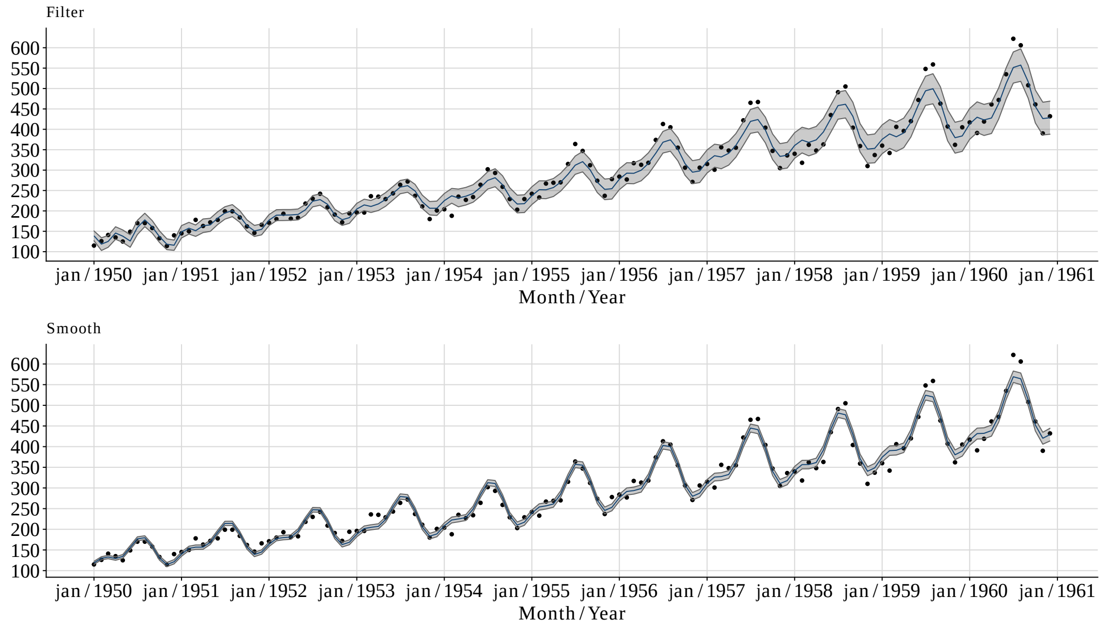

Figure 5: Mean response for Filtered and Smoothed predictive distributions with 95% credibility intervals.

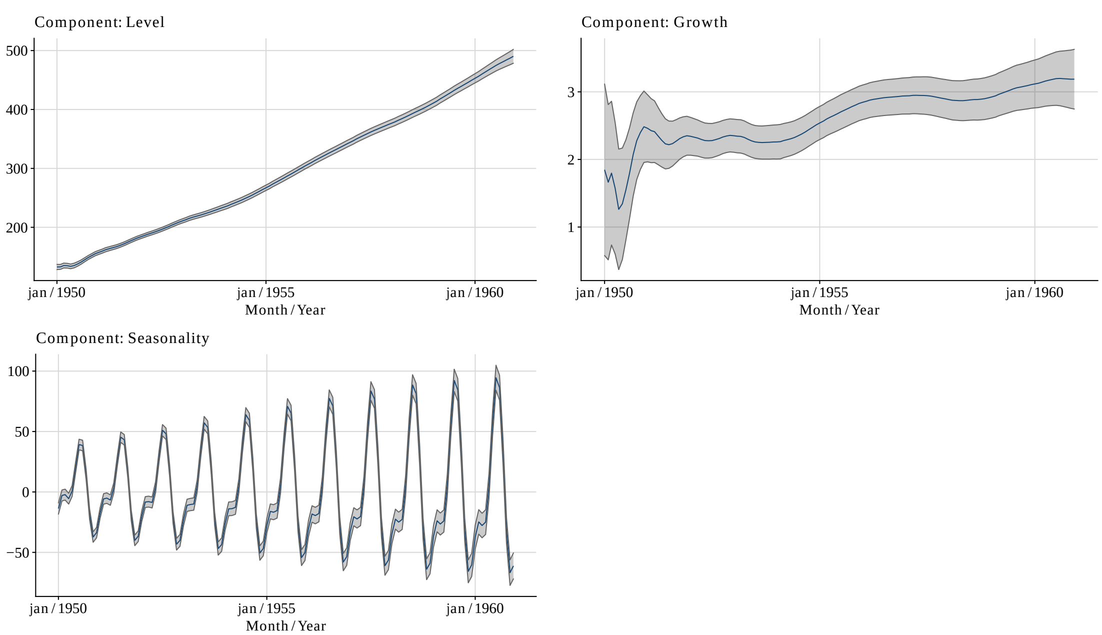

Figure 6: Mean response for Filtered and Smoothed posterior distributions for each model component with 95% credibility intervals.

of a product by a major company in UK. Note that the use of standard time series models may not wield satisfactory results as there a some points that need some attention:

1. During 1955 the market grows fast at a fast but steady rate,
2. A jump in December 1955.
3. The sales flattens off for 1956.
4. There is a major jump in the sales level in early 1957.
5. Another jump in early 1958.
6. Throughout the final two years, there is a steady decline back to late 1957.

```
>>> cp6 = load_cp6()
```

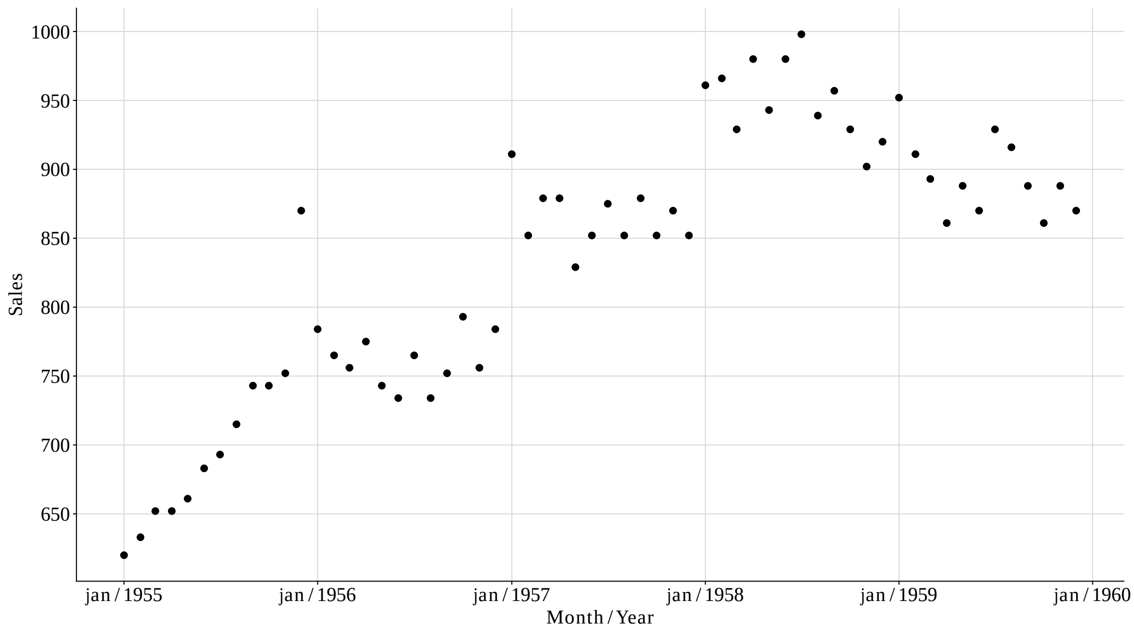

Figure 7: CP6 sales series

Fit Without Intervention

Given this, let's see how a standard dlm performs. The model used is defined below.

```
>>> # Define the growth model
>>> a = np.array([600, 1])
>>> R = np.array([[100, 0], [0, 25]])
>>> mod = dlm(a, R, ntrend=2, deltrend=[0.90, 0.98])
```

```
>>> # Filter and Smooth without intervention
>>> smooth = Smoothing(mod=mod)
>>> out_no_int = smooth.fit(y=cp6["sales"])
>>> dict_filter_no_int = out_no_int.get("filter").get("predictive")
```

Note that until November 1955 the forecast distribution was quite acceptable, the credibility interval was relatively small and the errors was were distributed around zero and inside the interval. But with the jump in December 1955 the level rises dramatically, the biggest problem is not the model's inability to efficiently predict this point, but the influence it has on future predictions. Note that for most of the year 1956 the predicted sales overestimation the real sales, giving a cluster of negative errors $(y_t - f_t)$. In early 1957 another jump was observed, but in this case, it was accompanied by a regime change. And this has great impact in the amplitude of the credibility intervals. In early 1958 another regime change, followed by a change in

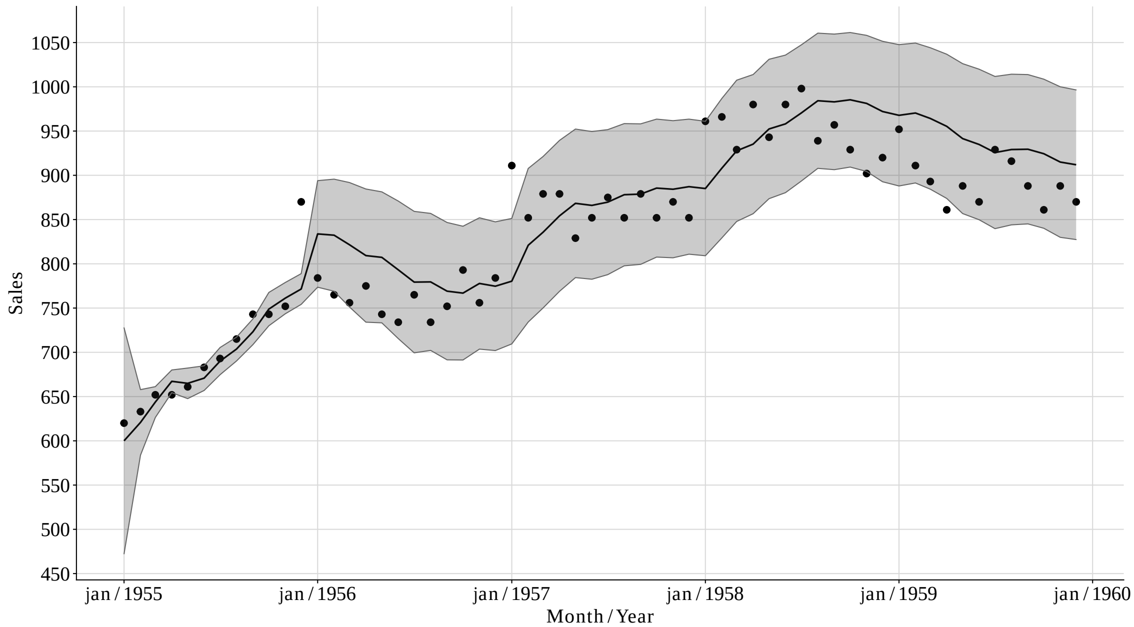

Figure 8: Mean response for Filtered predictive distribution with 95% credibility interval

grow, that is not properly modeled since from August 1958 to January 1960 all errors were negative with the exception of July 1959.

Fit With Intervention

With the intervention class it is possible to consider outside information to define the prior distribution at the time t. This can be done in two ways: noise or subjective. Which must be provided in a list of dictionaries containing the time the intervention will be carried out and the type. Lets start with a empty list

```
>>> intervention_list = []
```

Noise Intervention in Prior Variance

In our example, suppose that a change in growth for the year 1956 was anticipated. An increase in uncertainty about level and growth can be done by the addition of a matrix H_t to R_t at time $t = 12$ given by

$$H_t = \begin{bmatrix} 100 & 25 \\ 25 & 25 \end{bmatrix}$$

Thus, there is an increase (a positive shift) in the prior variance of the components. In our list of interventions we have

```
>>> intervention_list = [{
>>>   "time_index": 12, "which": ["noise"],
>>>   "parameters": [{
>>>     "h_shift": np.array([0, 0]),
>>>     "H_shift": np.array([[100, 25], [25, 25]])}]
>>> }]
```

where

- `time_index`: time of intervention;
- `which`: type of intervention (in this case, a noise intervention);

- `parameters`: the values for the intervention.
    - `h_shit`: Shift in mean (we'll see more about that later).
    - `H_shift`: Shift in variance.

Noise Intervention in Prior Mean and Variance

It is also possible to intervene in the prior mean. Suppose an increase in the market level is expected for the year 1957, we can add a change in level of 80 units and increase the variance by 100 at January ($t = 25$)

$$\mathbf{h}_{25} = \begin{bmatrix} 80 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{H}_{25} = \begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix}$$

now, updating our intervention list

```
>>> intervention_list = [{
>>>   "time_index": 12, "which": ["noise"],
>>>   "parameters": [{
>>>     "h_shift": np.array([0, 0]),
>>>     "H_shift": np.array([[100, 25], [25, 25]])}],
>>>
>>>   "time_index": 25, "which": ["noise"],
>>>   "parameters": [{
>>>     "h_shift": np.array([80, 0]),
>>>     "H_shift": np.array([[100, 0], [0, 0]])}]
>>> }]
```

In January 1958 ($t = 37$) another jump in level is anticipated, this time of about 100 units with a feeling of increased certainly about the new level and also a anticipated uncertainly for the growth. At this time, the prior mean and variance given by

$$\mathbf{a}_{37} = \begin{bmatrix} 864.5 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_{37} = \begin{bmatrix} 91.7 & 9.2 \\ 9.2 & 1.56 \end{bmatrix}$$

are simply altered to

$$\mathbf{a}^*_{37} = \begin{bmatrix} 970 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{R}^*_{37} = \begin{bmatrix} 50 & 0 \\ 0 & 5 \end{bmatrix}$$

Observational Variance Intervention

It is also possible to perform interventions on observational variance. This can be useful for outlier anticipation.

Suppose that at the end of 1955 there will be an announcement of future price increases which will result in forward-buying. So, a intervention at December 1955 ($t = 12$) will allow for an anticipated outlier. In late 1956, there is a a view that the marked change in the new year will begin with a maverick value, as the product that are to be discontinued are sold cheaply.

This interventions can be done by including a variance intervention in our list of interventions for the respective time:

```
>>> 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]])}]}
>>> ]
```

Performing the fit (filter and smoothing) with interventions

Finally, the fit with intervention can be done using the `Intervention` class. In the `.fit` method we will initialize the model and the loop forecast, observe and update, this time with the interventions given in `list_interventions`, begin. This will return a dictionary with the same structure as presented in the smoothing section.

```
>>> manual_interventions = Intervention(mod=mod)
>>> out_int = manual_interventions.fit(
>>>     y=cp6["sales"], interventions=list_interventions)
>>> dict_filter_int = out_int.get("filter").get("predictive")
```

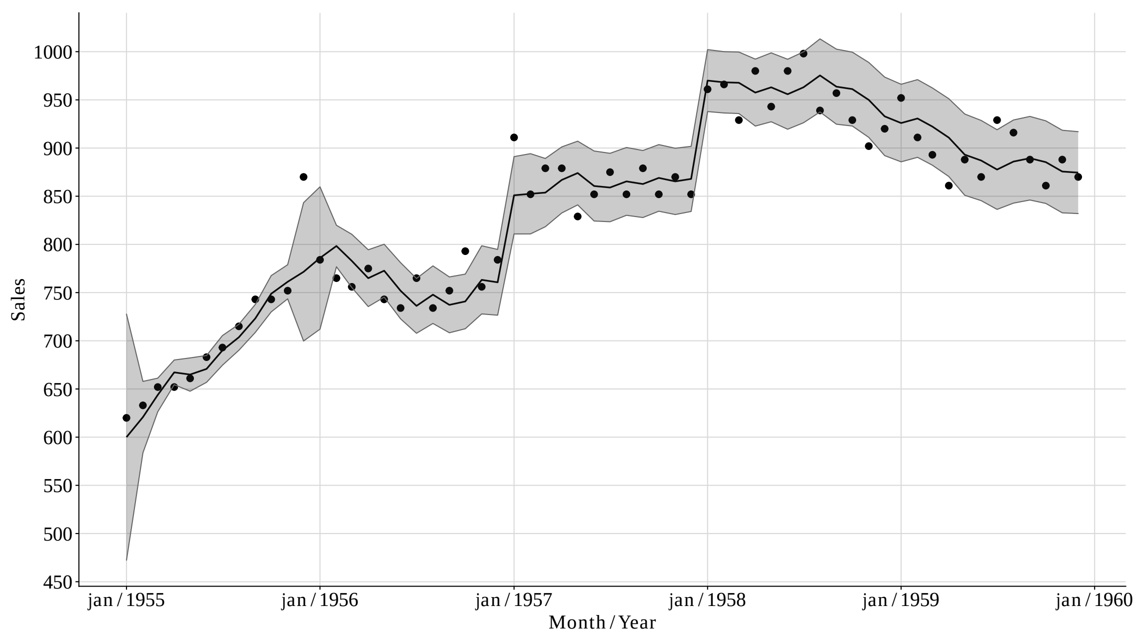

Figure 9: Mean response for filtered predictive distribution with 95% credibility interval and ideal interventions

Automatic Monitoring

The automatic monitoring method sequentially evaluate the forecasting activity to detect breakdowns, based on Bayes factor.

Again with the CP6 data, the automatic monitoring can be done using the `Monitoring` class.

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

```
>>>
>>> # Fit with monitoring
>>> monitor = Monitoring(mod=mod, bilateral=False)
>>> fit_monitor = monitor.fit(y=cp6["sales"], h=4, tau=0.135, change_var=[100])
```

```
## Potential outlier detected at time 12 with H=1.0298e-18, L=1.0298e-18 and l=1
## Potential outlier detected at time 19 with H=6.6255e-02, L=6.6255e-02 and l=1
## Potential outlier detected at time 25 with H=2.5545e-10, L=2.5545e-10 and l=1
## Potential outlier detected at time 37 with H=1.7544e-08, L=1.7544e-08 and l=1
## Potential outlier detected at time 55 with H=8.3786e-03, L=8.3786e-03 and l=1
```

```
>>> forecast_df = fit_monitor.get("filter").get("predictive")
```

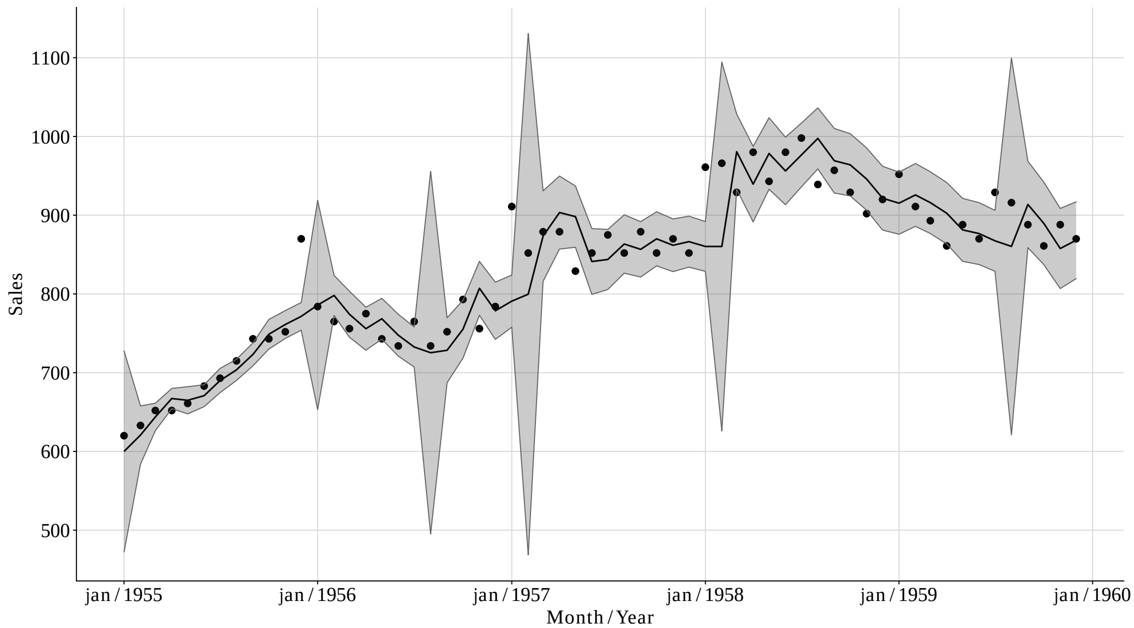

Figure 10: Mean response for Filtered predictive distribution with 95% credibility interval