PyCipio: Bayesian Time-series Prediction

Mikkel Werling (201706722)
Emil Rønn (2017...)
Victor Møller Poulsen (2017...)

1 Introduction

1.1 Time Series Forecasting

1.2 Decomposition of a Signal

$$y(t) = g(t) + s(t) + \varepsilon$$

$$y(t) = g(t) \cdot s(t) \cdot \varepsilon$$

$$g(t) = \alpha + \beta \cdot x$$

$$s(t) = \sum_{n=1}^{N} \left(a_n cos(\frac{2\pi nt}{P}) + b_n sin(\frac{2\pi nt}{P}) \right)$$

$$F(t) = \left[cos(\frac{2\pi 1t}{7}), \ldots, sin(\frac{2\pi 8t}{7}) \right]$$

$$s(t) = F(t) \cdot \omega$$

$$y(t) = alpha + beta \cdot x + s_1(t) + s_2(t)$$

1.3 Bayesian Framework

2 Implementation and Architecture

2.1 Object Oriented Programming

2.2 Primary Functions

```
PyCipio.__init__(data, time, values, index = None, split = 0.7):
```

Description:

Initializing the class. Assumes that data is a pandas DataFrame object.

Arguments:

data (pd.DataFrame):

Dataframe containing a column containing time indices and a column containing values.

Additionally, data can also contain a column which specifies groups in the data, but is not required.

time (str):

Column name in data, which specifies time indices.

values (str):

Column name in data, which specifies the values of y.

index (str, optional):

Column name in data, which specifies a grouping variable. If this variable is given,

the analysis will be carried out independently for each grouping. Defaults to None.

split (float, optional):

Float indicating the proportion of data used for training. Defaults to 0.7.

Example:

```
Pc = PyCipio(data = data,
        time = "x",
        values = "y",
```

```
                    index = "group",
                    split = 0.8)
```

```
PyCipio.fit(p1, p2, p1_mode, p2_mode, divisor = 20, deviation = 0.2):
```

Description:

Fits the model and plots the prior predictive distribution.

Arguments:

p1 (tuple):

Tuple of integers, where the first value is the value of p and the second value is the number of components. First value can be specified as a float, while the second value must be an integer.

p2 (tuple):

Tuple of integers, where the first value is the value of p and the second value is the number of components. First value can be specified as a float, while the second value must be an integer.

p1_mode (str):

String indicating whether the seasonal component should be multiplicative or additive. If anything else than "multiplicative" is specified, the mode defaults to additive.

p2_mode (str):

String indicating whether the seasonal component should be multiplicative or additive. If anything else than "multiplicative" is specified, the mode defaults to additive.

divisor (int, optional):

A scaling parameter for adjusting the standard deviation of the distribution of p. The standard deviation of p is set to p/divisor. Defaults to 20.

deviation (float, optional):

Parameter specifying the standard deviation of the beta for the seasonal component. Defaults to 0.2.

Example:

```
Pc.fit(p1 = (7, 2),
    p2 = (365.25, 2),
    p1_mode = "additive",
    p2_mode = "multiplicative",
    divisor = 15,
    deviation = 0.3)
```

```
PyCipio.sample_mod(posterior_draws = 2000, post_pred_draws = 1000,
prior_pred_draws = 1000, random_seed = 42, chains = 2):
```

Description:

Sample the posterior, the posterior predictive, the prior predictive distribution and generate predictions on the test data.

Arguments:

posterior_draws (int, optional):

Number of draws for the posterior. Defaults to 2000.

prior_pred_draws (int, optional):

Number of draws for the prior predictive distribution. Defaults to 1000.

random_seed (int, optional):

Random seed for ensuring reproducibility. Defaults to 42.

chains (int, optional):

Number of chains used for sampling the posterior. Defaults to 2.

Example:

```
Pc.sample_mod(posterior_draws = 3000,
post_pred_draws = 1500,
prior_pred_draws = 1500,
random_seed = 13,
chains = 4)
```

```
PyCipio.plot_fit_idx(idx = None, path = False):
```

Description:

Plots the posterior predictive distribution overlayed on the training data.

Arguments:

idx (list, optional):

List of strings containing the names of the groups to plot. Defaults to None.

path (str, optional):

String specifying the path for saving the plot.

File-extension is automatically inserted and is hard set to .png. Defaults to False.

Example:

```
Pc.plot_fit_idx(idx = ["group1", "group2"], path = "my_path/my_plot")
```

Note:

The same functionality exists for plotting the posterior predictive distribution overlayed on the training data. This method called *plot_prediction_idx* is identical in inputs and outputs, but only differs on this point. For more details, see the full docstrings on Github.

```
PyCipio.plot_residuals(idx = None, path = False):
```

Description:

Plots the residuals from predictions generated by the model.

Arguments:

idx (list, optional):

List of strings containing the names of the groups to plot. Defaults to None.

path (str, optional):

String specifying the path for saving the plot.

File-extension is automatically inserted and is hard set to .png. Defaults to False.

Example:

```
Pc.plot_residuals(idx = ["group1", "group2"], path = "my_path/my_plot")
```

2.3 Workflow

3 Related work and differences

3.1 fpp3

3.2 Facebook Prophet

4 Examples

4.1 Example 1

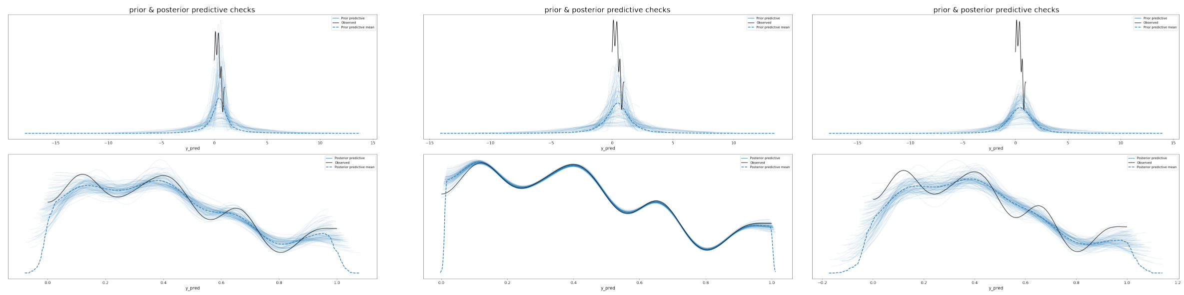
Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something,

(a) First plot (b) Second plot (c) Third Plot

Figure 1: Predictions in one and 2 groups

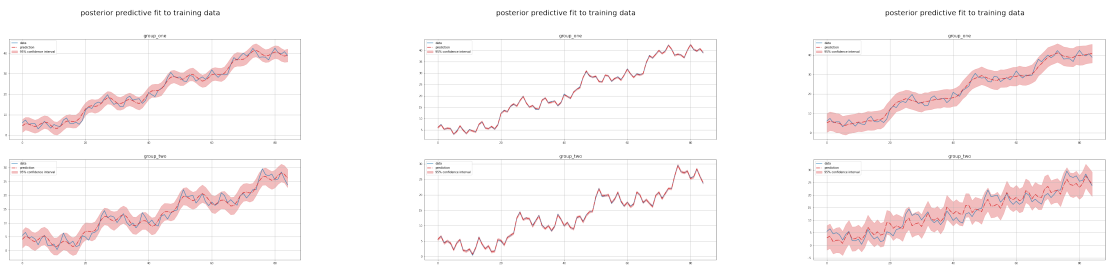
Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something,

Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something,



(a) First plot (b) Second plot (c) Third Plot

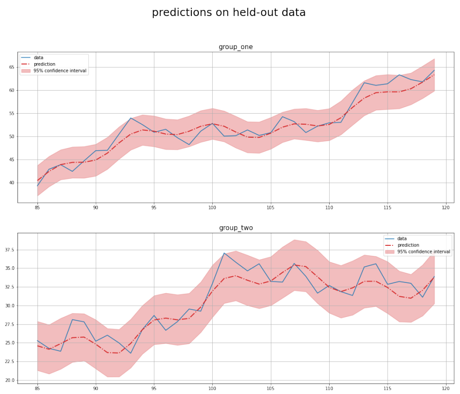
Figure 2: Predictions in one and 2 groups



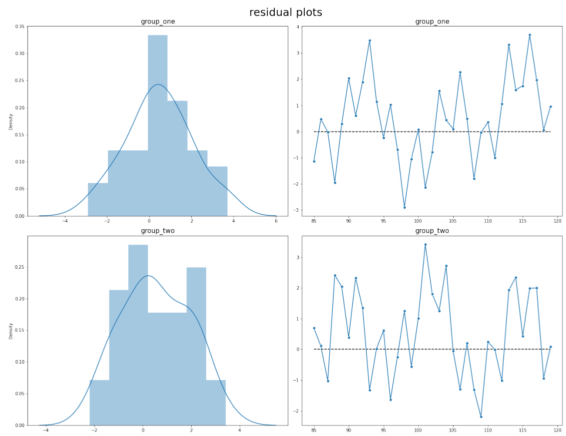
(a) First plot (b) Second plot (c) Third Plot

Figure 3: Predictions in one and 2 groups

Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something,



(a) First plot

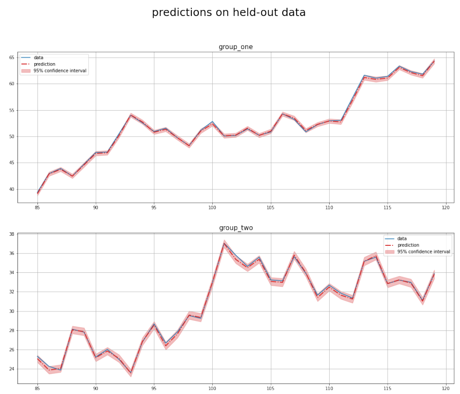


(b) Second plot

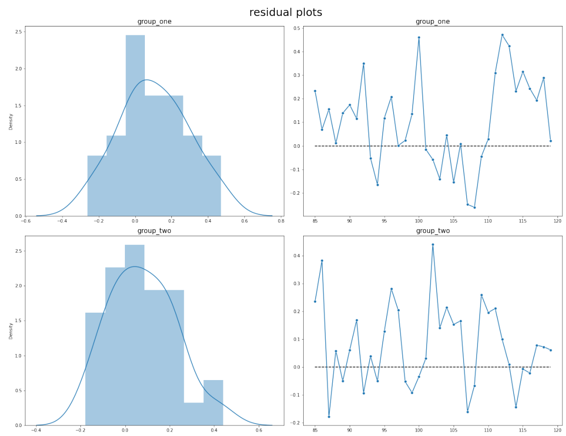
idx	RMSE	MAE	MAPE	sMAPE
group_one	1.62	1.27	2.46	2.41
group_two	1.5	1.21	4.01	3.98

(c) Third Plot

Figure 4: Predictions in one and 2 groups



(a) First plot



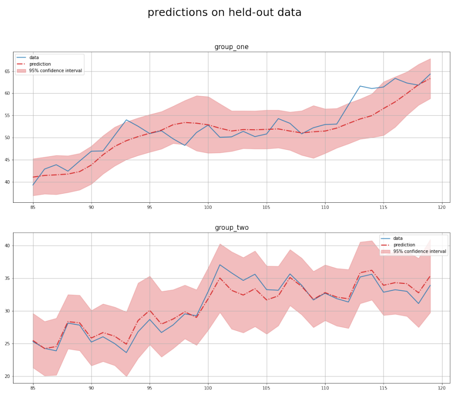
(b) Second plot

idx	RMSE	MAE	MAPE	sMAPE
group_one	0.21	0.17	0.33	0.32
group_two	0.17	0.13	0.44	0.45

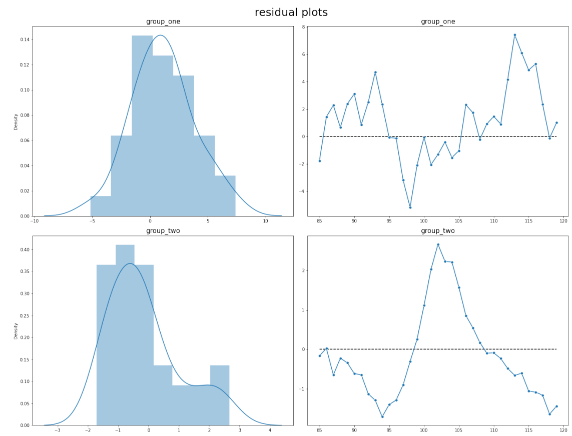
(c) Third Plot

Figure 5: Predictions in one and 2 groups

Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something, Obviously here you write something,



(a) First plot



(b) Second plot

idx	RMSE	MAE	MAPE	sMAPE
group_one	2.89	2.23	4.31	4.25
group_two	1.16	0.94	3.13	3.03

(c) Third Plot

Figure 6: Predictions in one and 2 groups

4.2 Example 2

4.3 Example 3

5 Limitations and future work

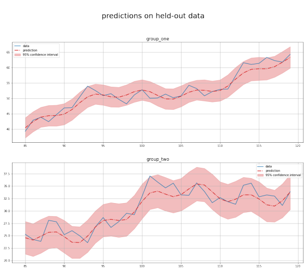
5.1 The Goal of PyCipio

5.2 Flexibility

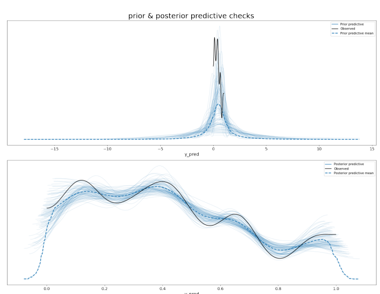
5.3 Hierarchical

5.4 Prediction on unseen data set

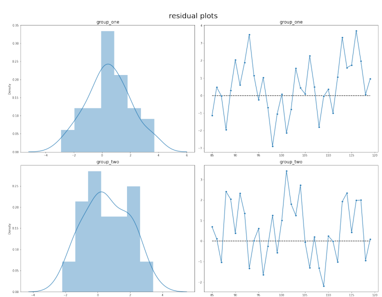
5.5 Why min-max scaling is great



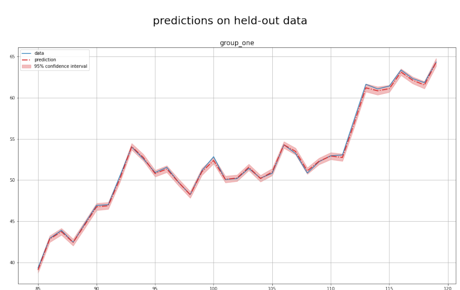
(a) First plot



(b) Second plot



(c) Third Plot



(d) Fourth plot

Figure 7: Predictions in one and 2 groups