NIXTLA

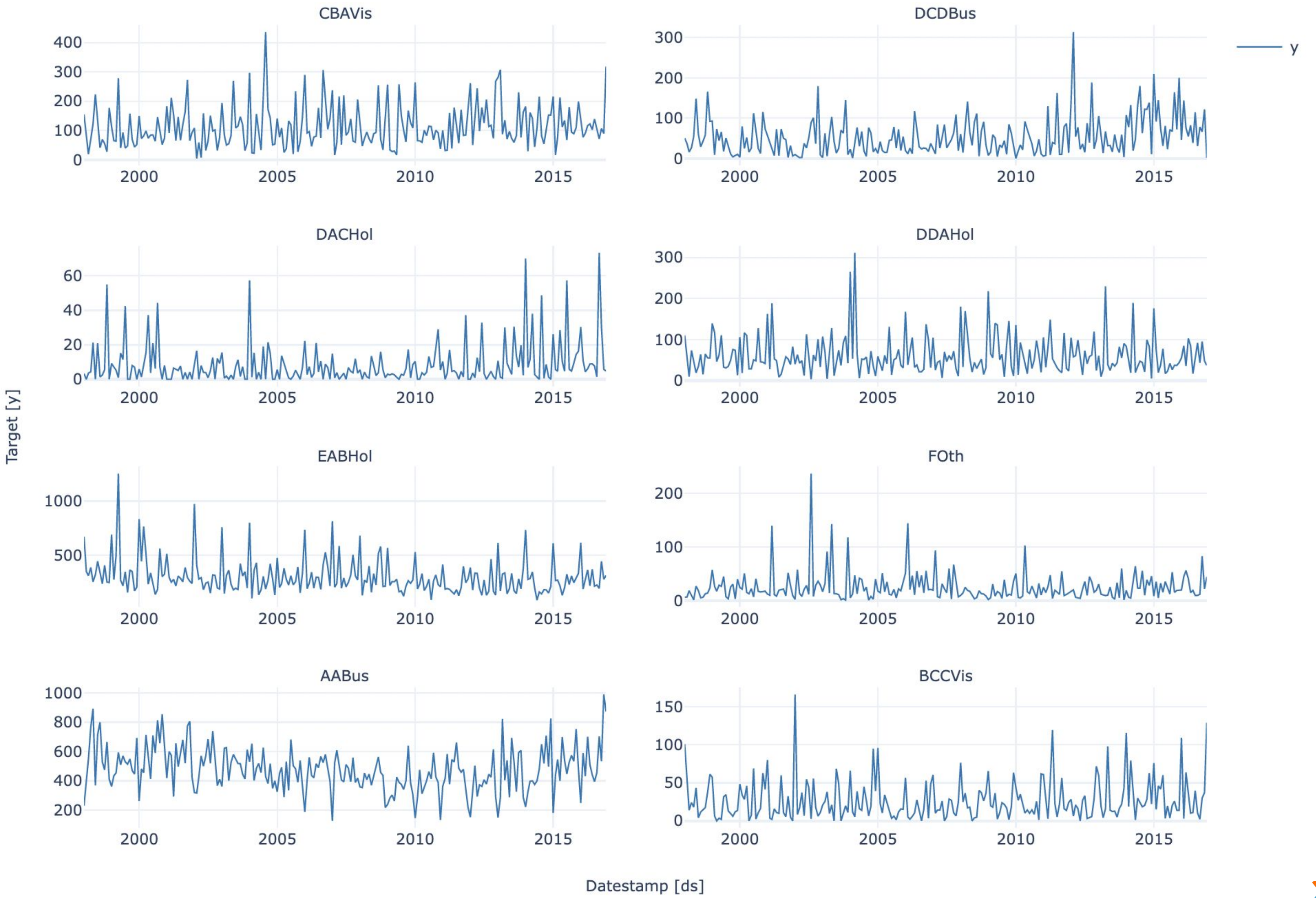
Time Series Research and Deployment

Hierarchical Forecasting in Python

nixtla

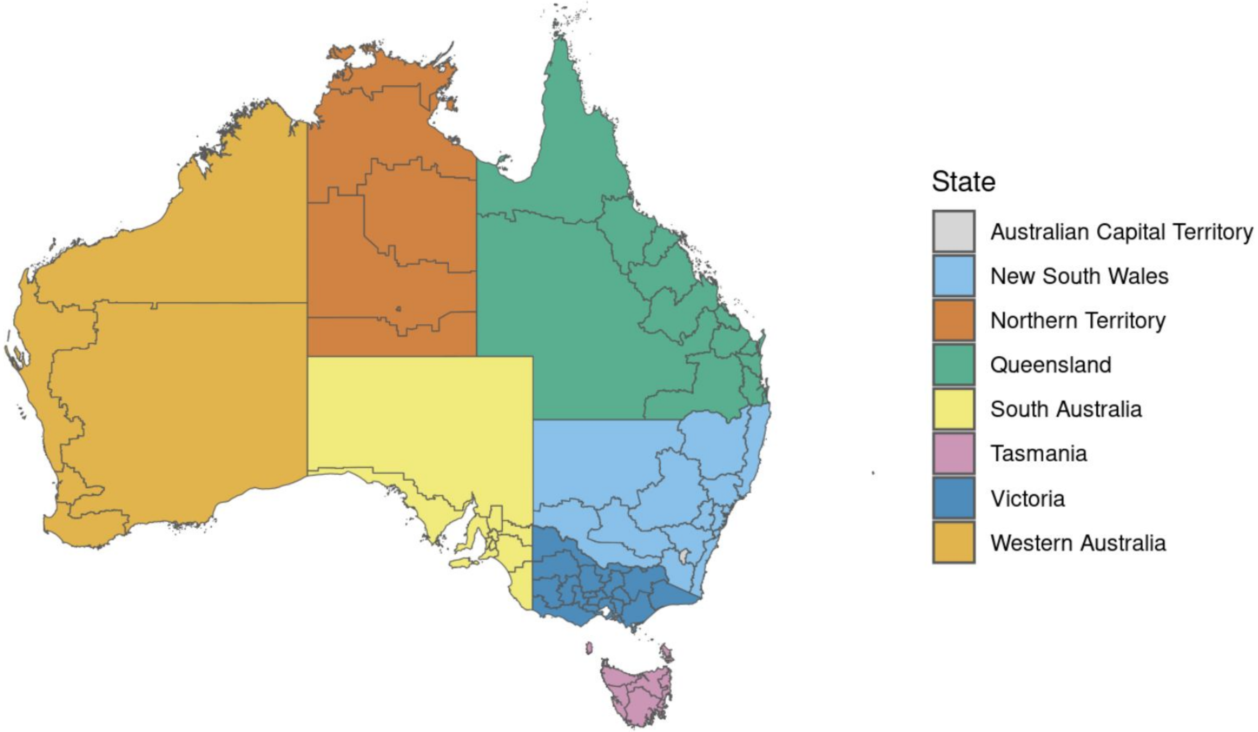
Setting: you are the minister of tourism in Australia

Question: how many tourists are going to visit Australia?


CBAVis
DCDBus
DACHol
DDAHol
EABHol
FOth
AABus
BCCVis
y
Target [y]
Datestamp [ds]

- How should we staff the airports?
- How will the economy grow?

Australia's has: 7 states, 27 zones and 76 regions.



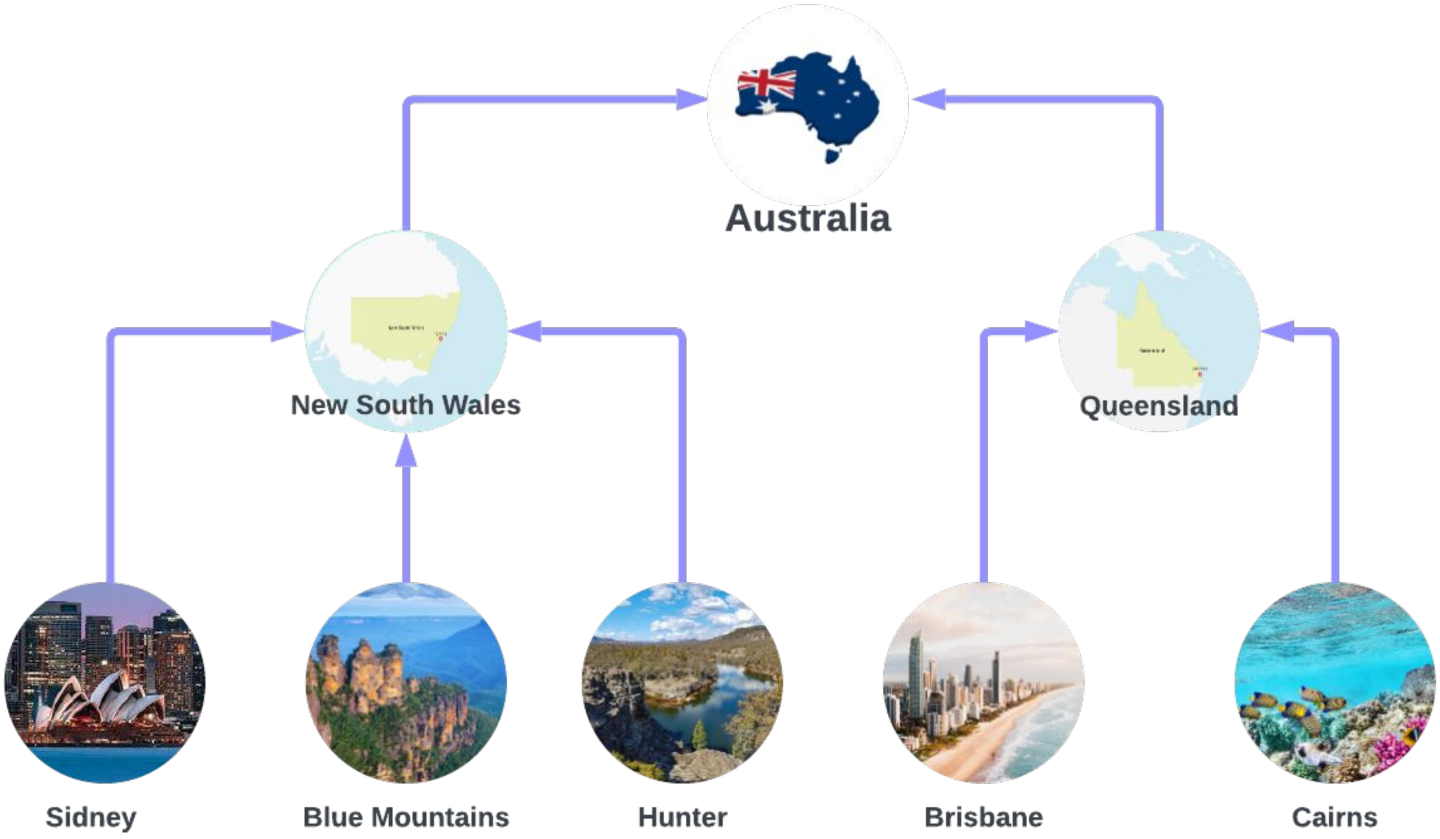
nixtla

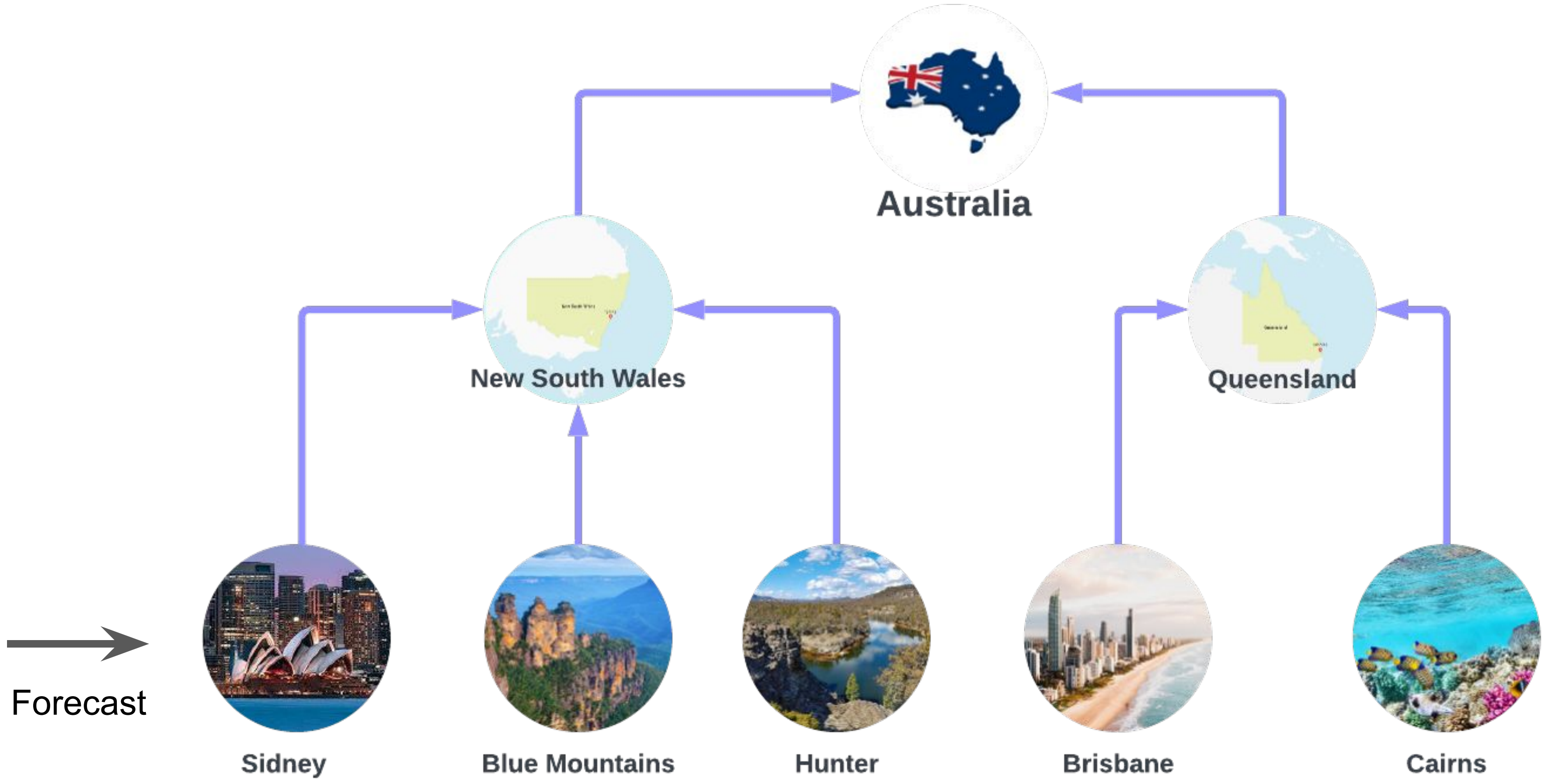
We need forecasts for the whole of Australia, for each of the states, territories, and regions.

Table A2: Australian Tourism flows.

Geographical Level	**Series per Level**	**Series per Level & Purpose**	**Total**
Australia	1	4	5
States	7	28	35
Zones	27	108	135
Regions	76	304	380
Total	111	444	555

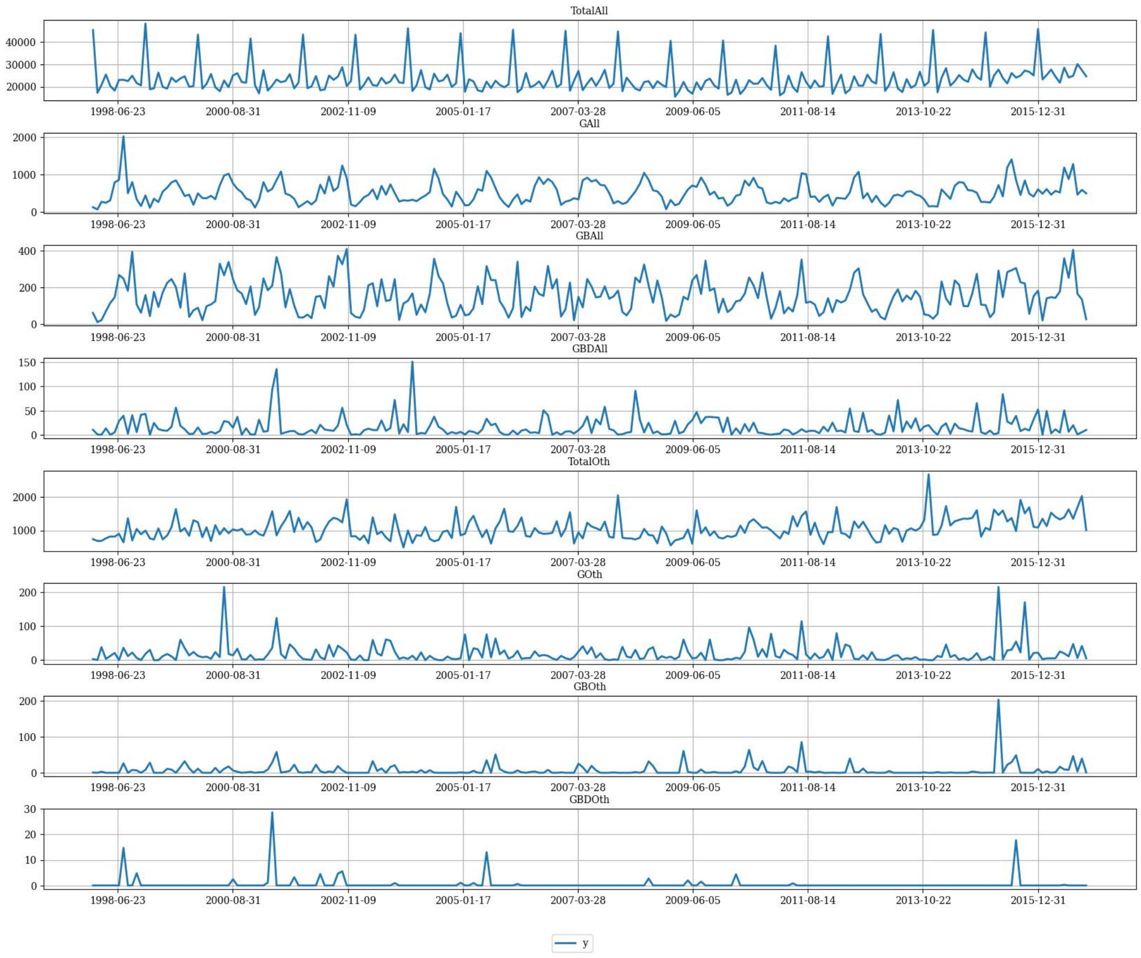
Approach 1: Bottom UP


Australia
New South Wales
Queensland
Sidney
Blue Mountains
Hunter
Brisbane
Cairns

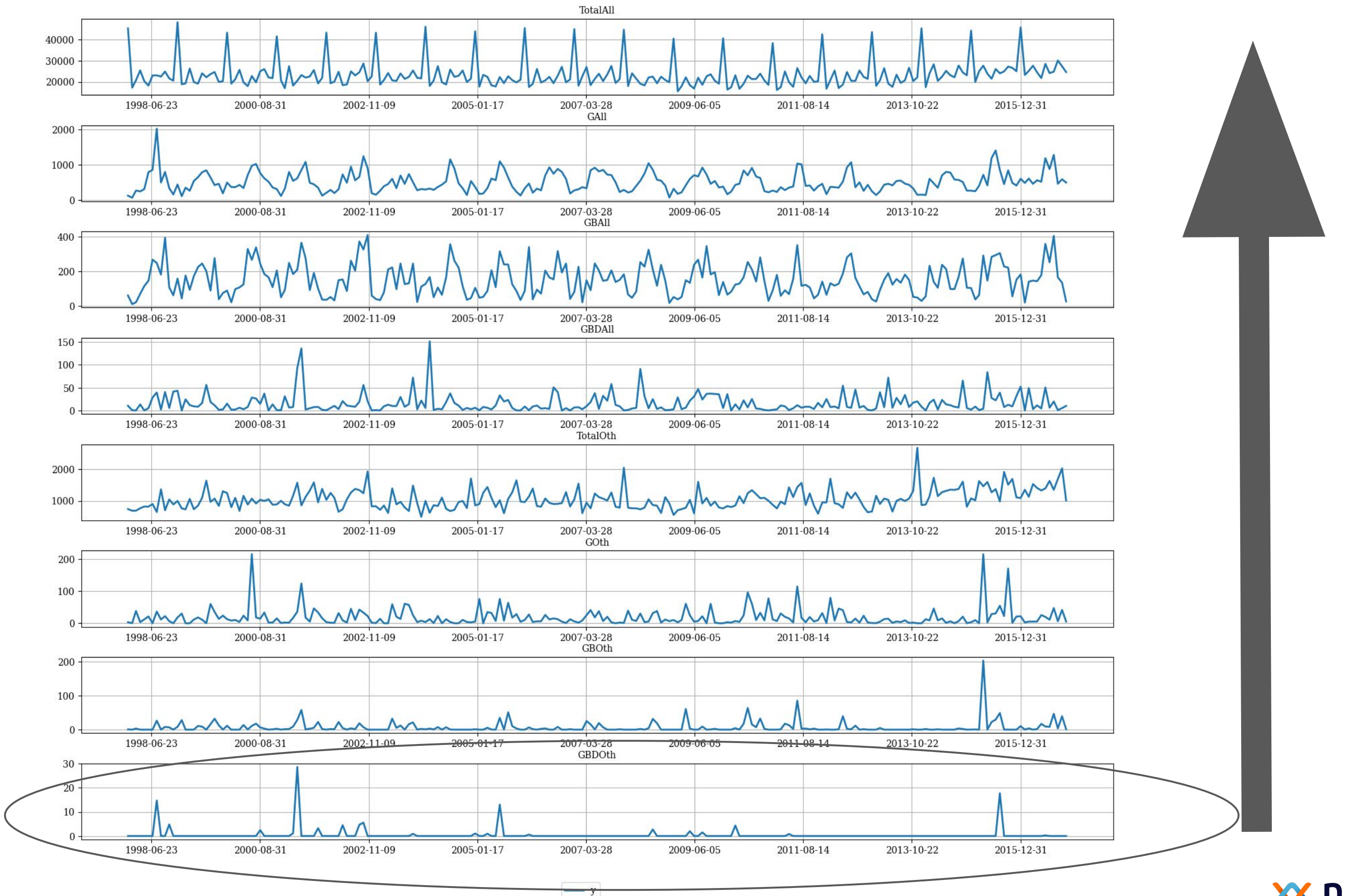


Australia
New South Wales
Queensland
Forecast
Sidney
Blue Mountains
Hunter
Brisbane
Cairns

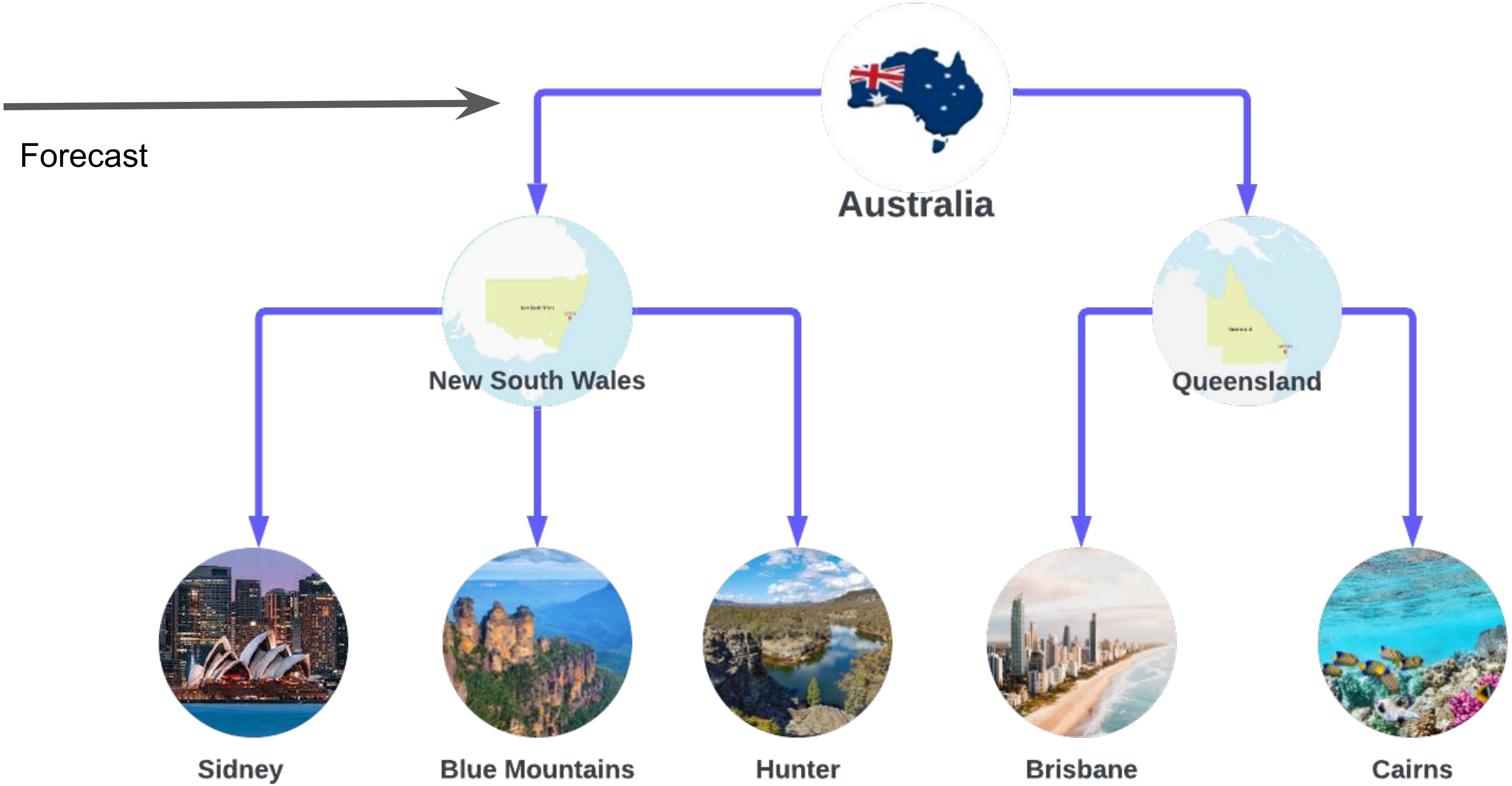
nixtla



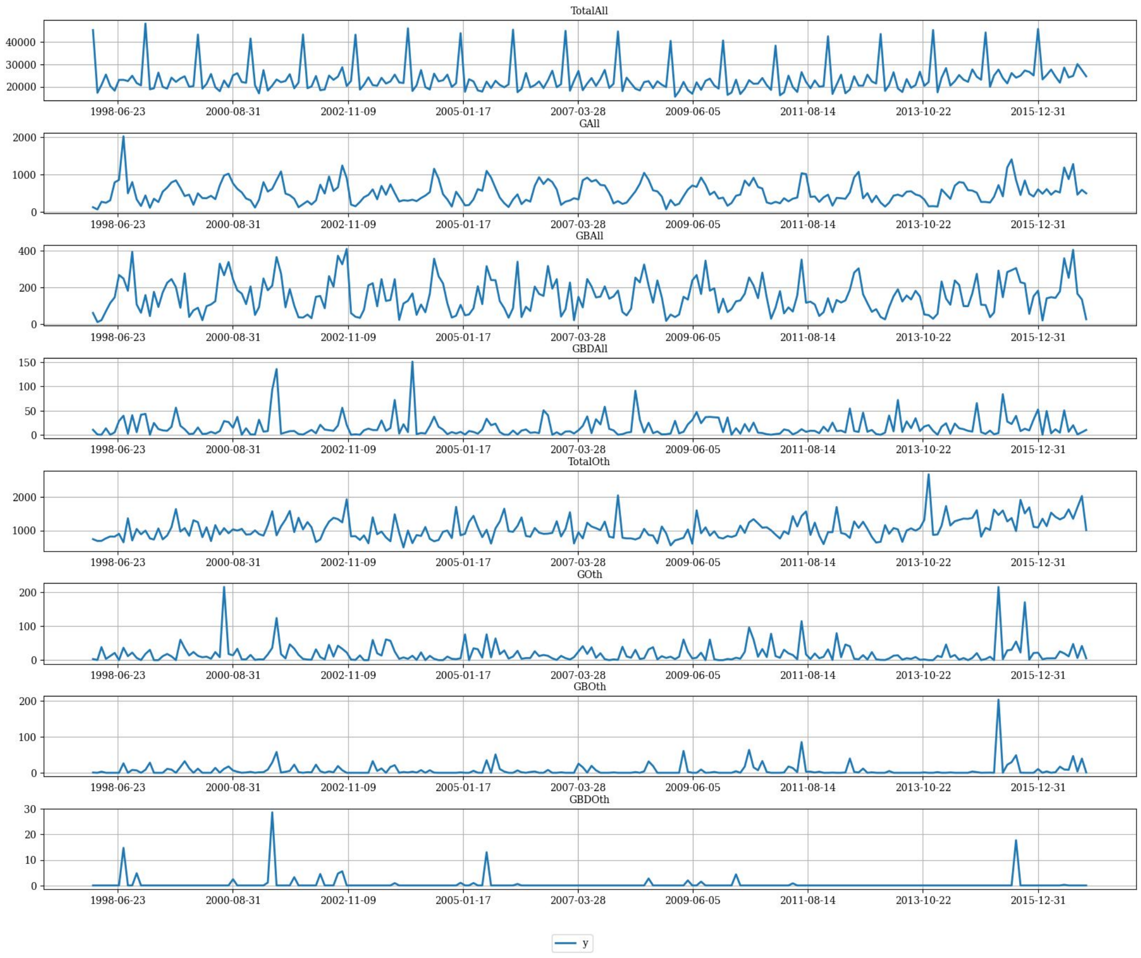
TotalAll
40000
30000
20000
GAll
2000
1000
0
GBAll
400
200
0
GBDAll
150
100
50
0
TotalOth
2000
1000
GOth
200
100
0
GBOth
200
100
0
GBDOth
30
20
10
0
1998-06-23
2000-08-31
2002-11-09
2005-01-17
2007-03-28
2009-06-05
2011-08-14
2013-10-22
2015-12-31
y


TotalAll
GAll
GBAll
GBDAll
TotalOth
GOth
GBOth
GBDOth
1998-06-23
2000-08-31
2002-11-09
2005-01-17
2007-03-28
2009-06-05
2011-08-14
2013-10-22
2015-12-31
y

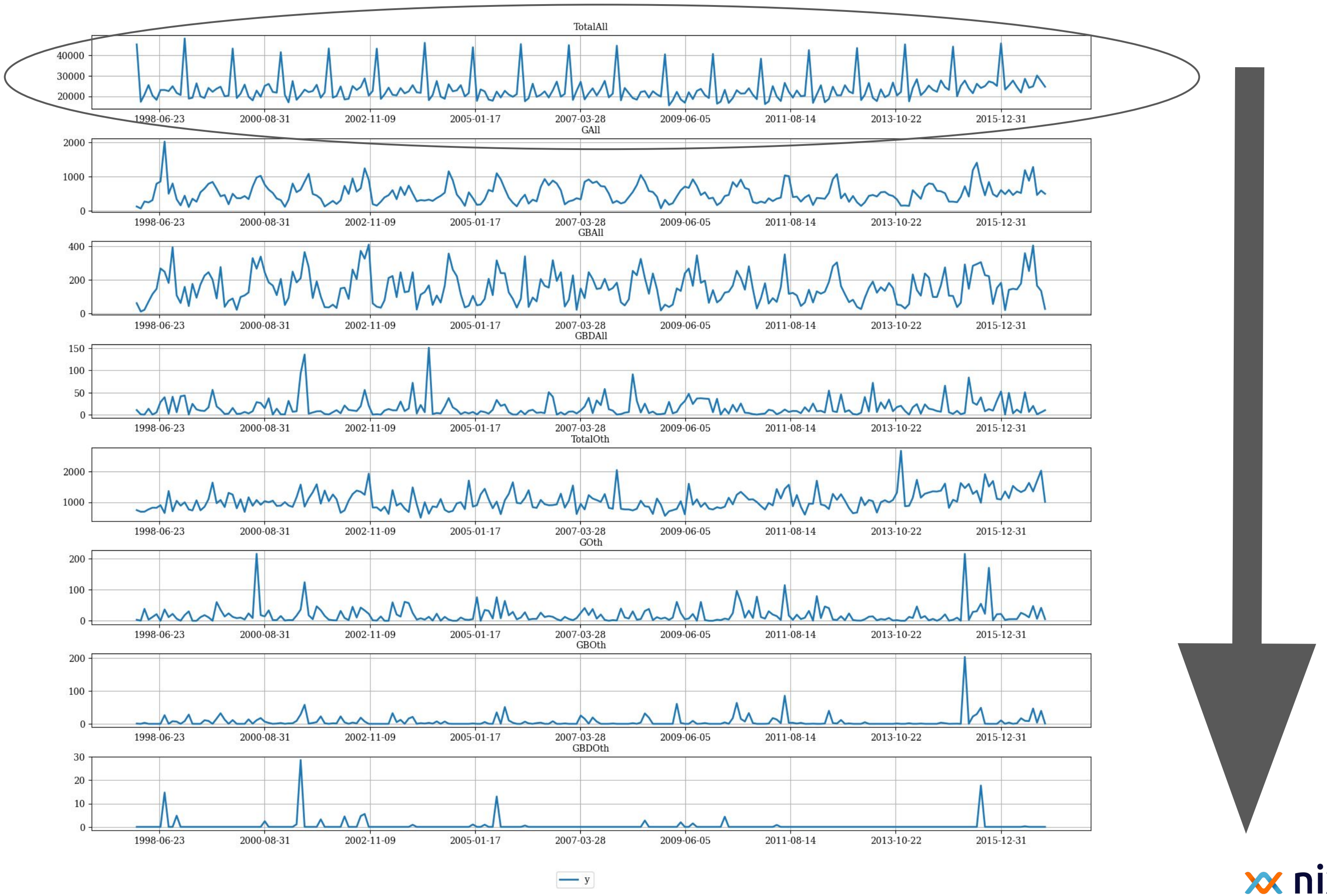
Approach 2: Top Down


Forecast
Australia
New South Wales
Queensland
Sidney
Blue Mountains
Hunter
Brisbane
Cairns

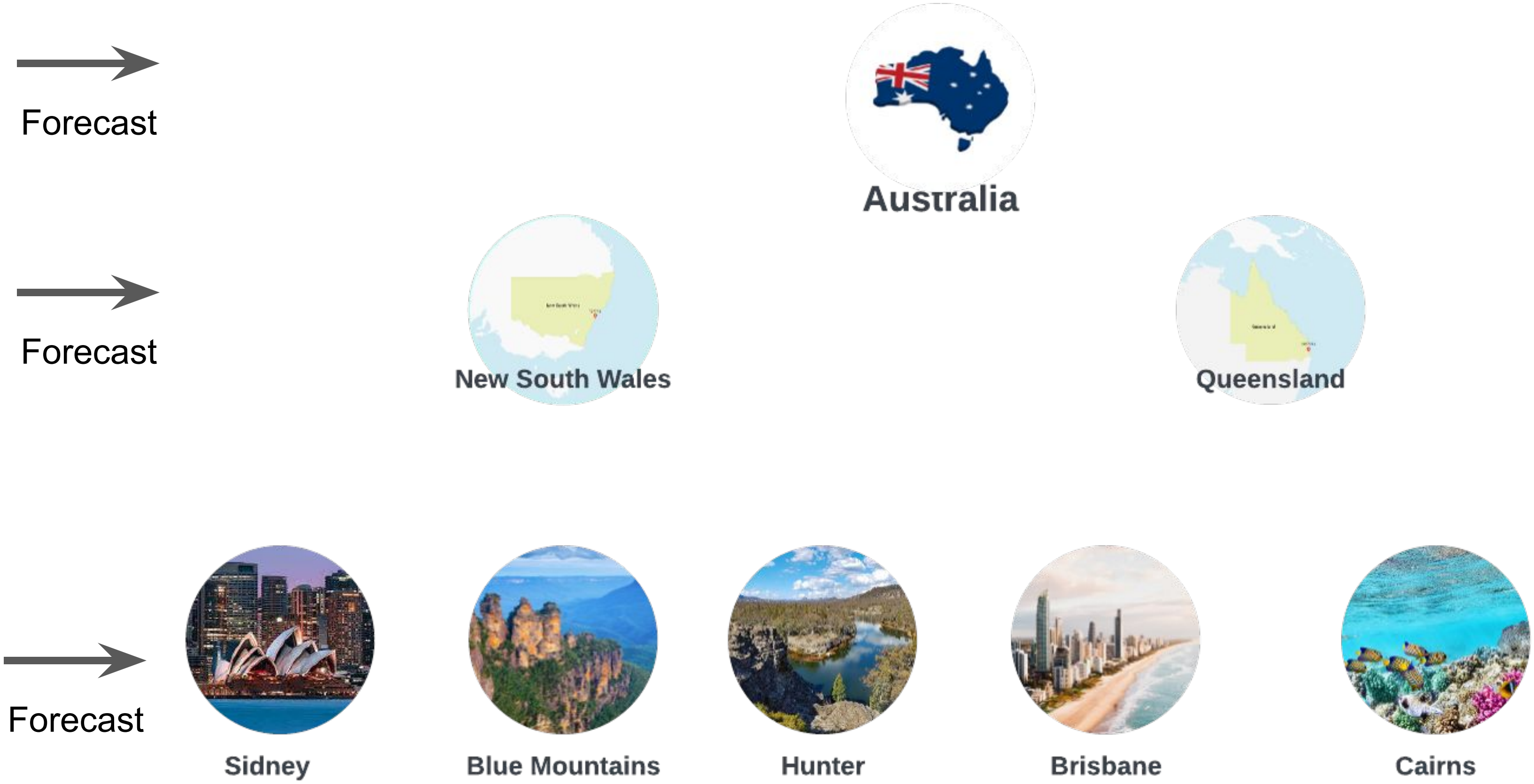
nixtla



TotalAll
40000
30000
20000
GAll
2000
1000
0
GBAll
400
200
0
GBDAll
150
100
50
0
TotalOth
2000
1000
GOth
200
100
0
GBOth
200
100
0
GBDOth
30
20
10
0
1998-06-23
2000-08-31
2002-11-09
2005-01-17
2007-03-28
2009-06-05
2011-08-14
2013-10-22
2015-12-31
y


TotalAll
40000
30000
20000
GAll
2000
1000
0
GBAll
400
200
0
GBDAll
150
100
50
0
TotalOth
2000
1000
GOth
200
100
0
GBOth
200
100
0
GBDOth
30
20
10
0
1998-06-23
2000-08-31
2002-11-09
2005-01-17
2007-03-28
2009-06-05
2011-08-14
2013-10-22
2015-12-31
y

Approach 3: forecast all series!


Forecast
Australia
Forecast
New South Wales
Queensland
Forecast
Sidney
Blue Mountains
Hunter
Brisbane
Cairns

How to forecast hundreds or millions of series?

nixtla

nixtla

Open Source Time Series Ecosystem

Stars 5k

⚡ StatsForecast

Lightning fast forecasting with statistical and econometric models.

Github

🤖 MLForecast

Scalable machine learning for time series forecasting.

Github

🧠 NeuralForecast

Scalable and user friendly neural forecasting algorithms for time series data.

Github

```
pip install mlforecast
```

```python
import lightgbm as lgb
from mlforecast import MLForecast
from window_ops.expanding import expanding_mean
from window_ops.rolling import rolling_mean

# Create an instance of MLForecast with specified models, frequency, differences, lags, and lag_transforms
mlf = MLForecast(
    models=[lgb.LGBMRegressor()],   # List of models to use, in this case, a LightGBM Regressor
    freq='MS',                      # Frequency of the time series data (monthly start)
    differences=[12],               # List of differences to apply, in this case, a seasonal difference of 12 periods
    lags=[1, 12],                   # List of lags to use as features, in this case, lags of 1 and 12 periods
    lag_transforms={                # Dictionary of lag transformations to apply on the selected lags
        1: [expanding_mean],        # Apply expanding mean transformation on the 1-period lag
        12: [(rolling_mean, 24)],   # Apply rolling mean transformation with a window size of 24 on the 12-period lag
    },
)
```

```python
# Fit the MLForecast model to the tourism dataset
mlf.fit(
    tourism_df,                          # Input dataset (Tourism)
    id_col='unique_id',                  # Column containing unique time series identifiers
    time_col='ds',                       # Column containing timestamps
    target_col='y'                       # Column containing the target variable to forecast
)

# Generate forecasts using the MLForecast model for 12 steps ahead
base_forecasts_df = mlf.predict(12)
```

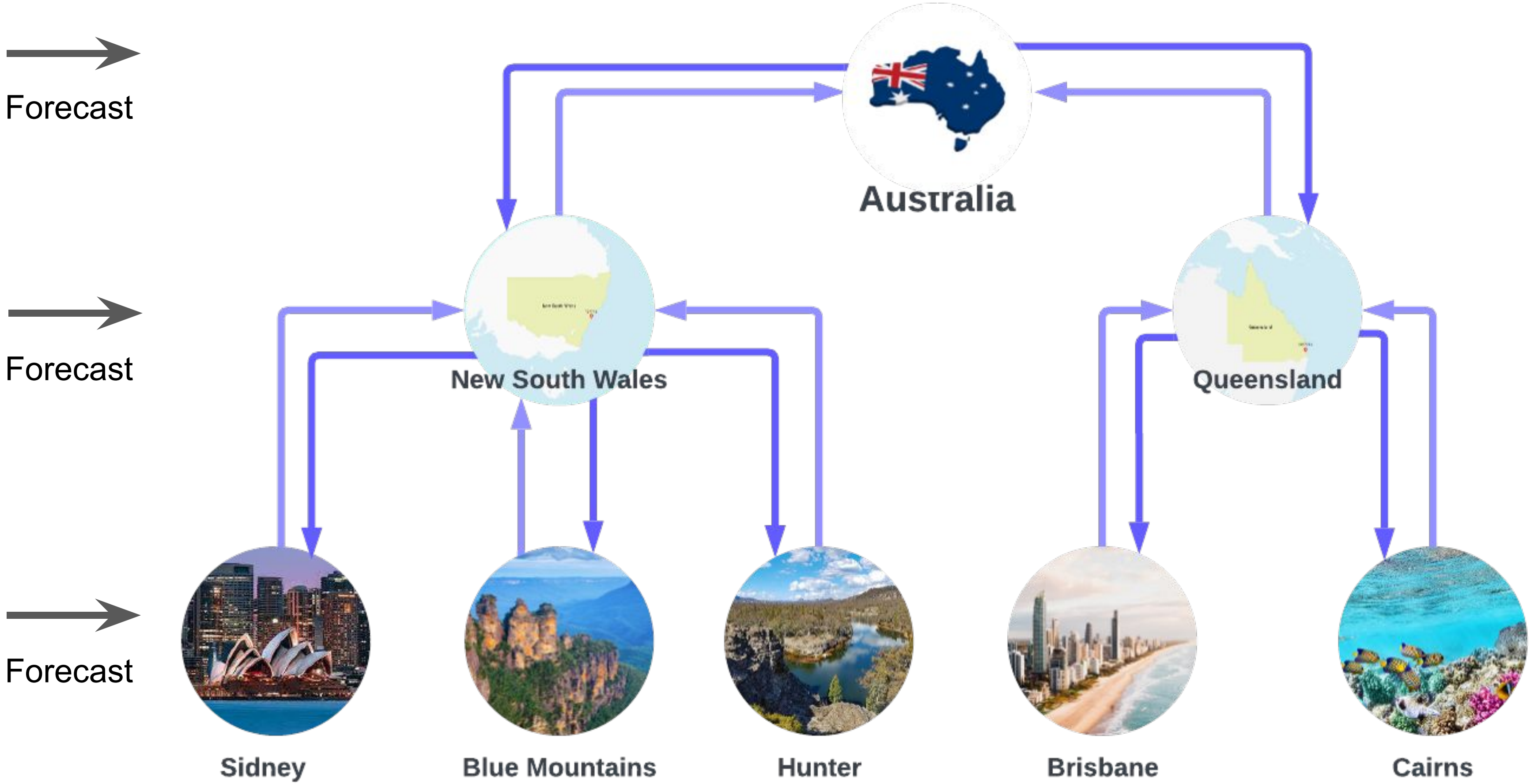

TotalAll
GAll
GBAll
GBDAll
TotalOth
GOth
GBOth
GBDOth
1997-05-19
2000-02-13
2002-11-09
2005-08-05
2008-05-01
2011-01-26
2013-10-22
2016-07-18
y
LGBMRegressor

nixtla

Why is the national forecast different from the sum of all the states!?

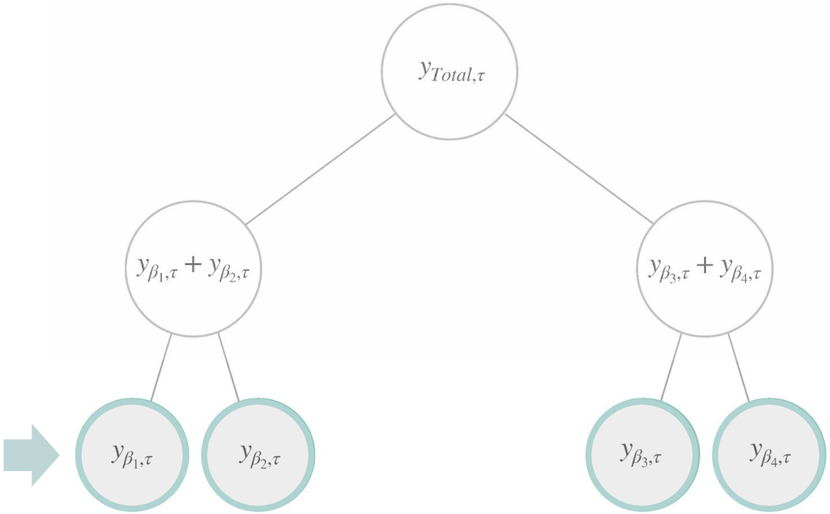
Problem: the forecast at different levels don’t add up.

What is the optimal solution?

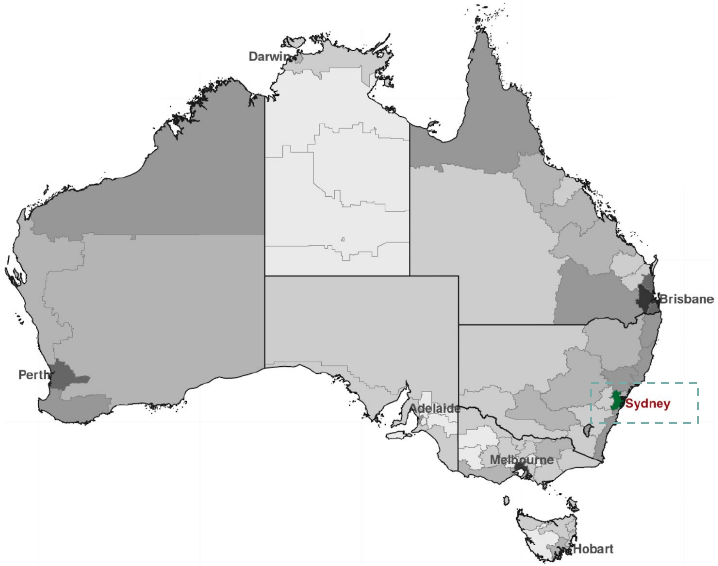

Forecast
Forecast
Forecast
Australia
New South Wales
Queensland
Sidney
Blue Mountains
Hunter
Brisbane
Cairns

nixtla

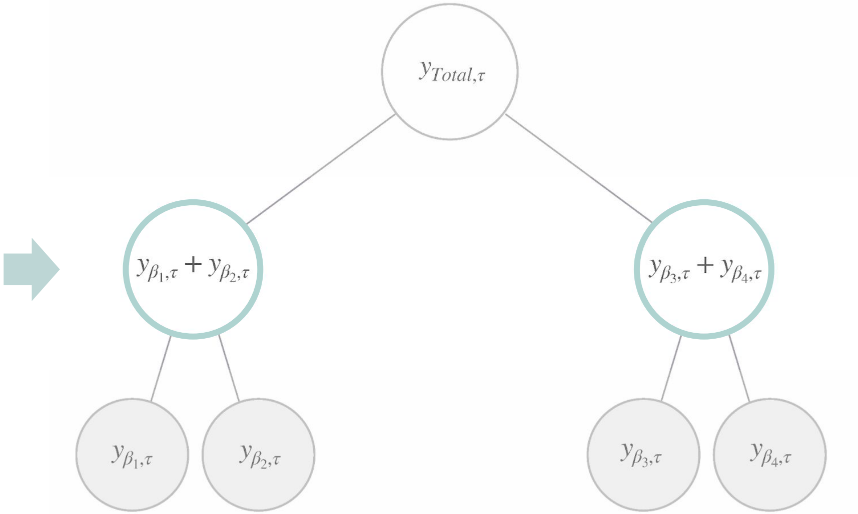
Hierarchical Reconciliation

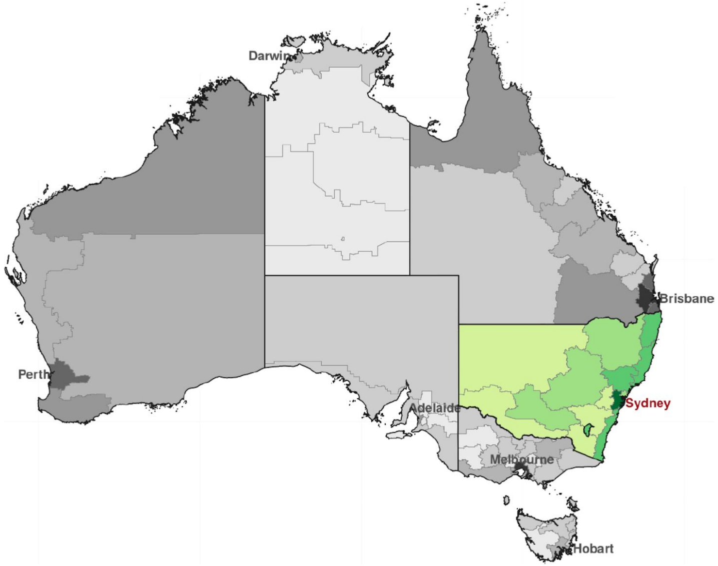


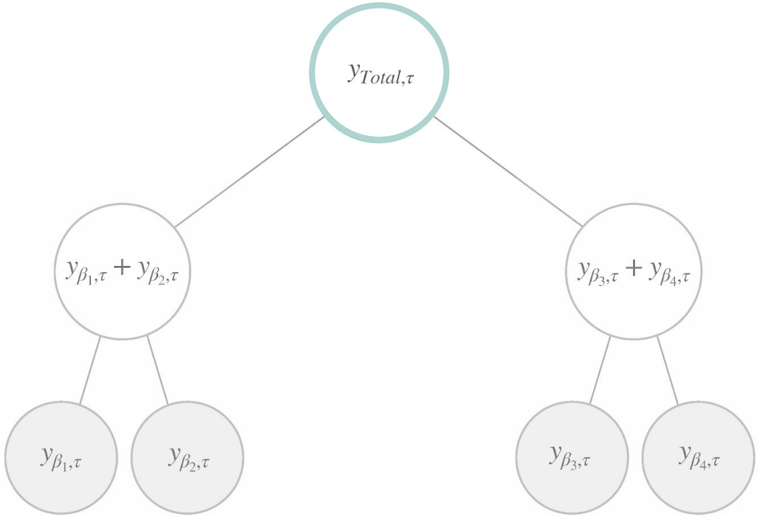
$y_{Total,\tau}$
$y_{\beta_1,\tau} + y_{\beta_2,\tau}$
$y_{\beta_3,\tau} + y_{\beta_4,\tau}$
$y_{\beta_1,\tau}$
$y_{\beta_2,\tau}$
$y_{\beta_3,\tau}$
$y_{\beta_4,\tau}$

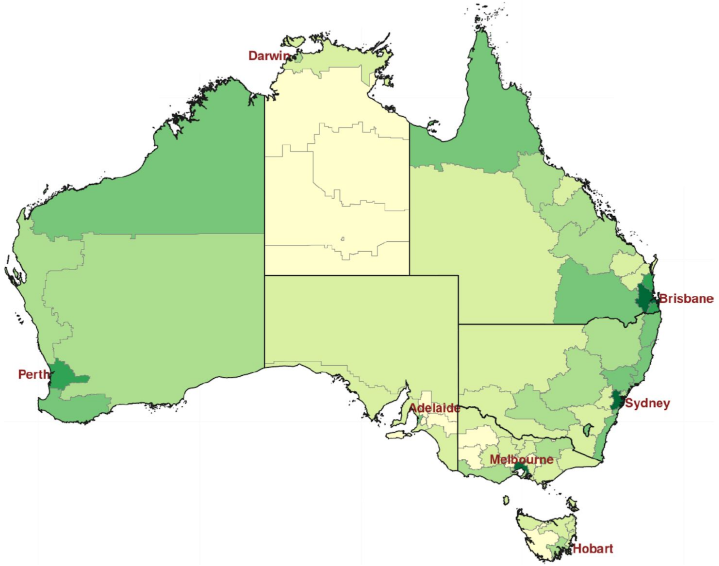


Darwin
Brisbane
Perth
Sydney
Adelaide
Melbourne
Hobart

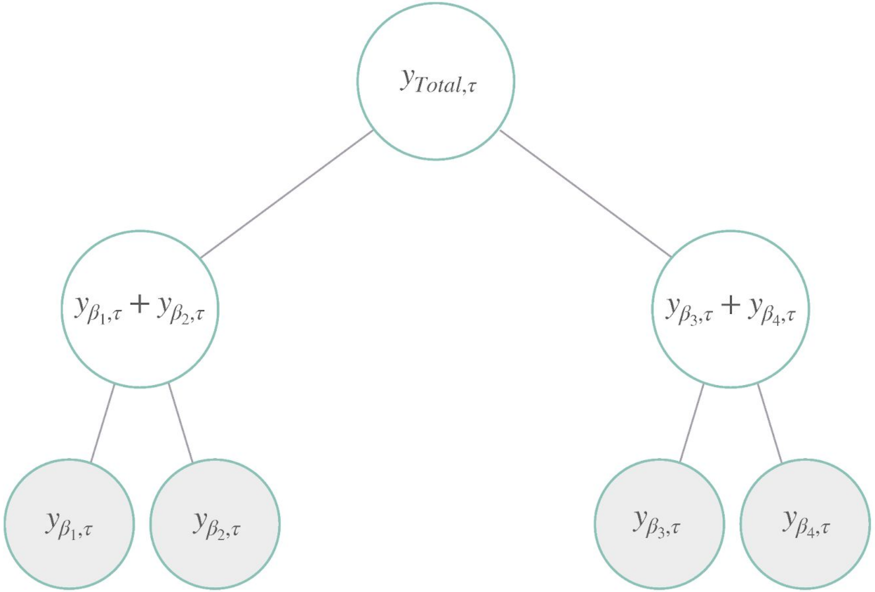

$y_{Total,\tau}$
$y_{\beta_1,\tau} + y_{\beta_2,\tau}$
$y_{\beta_3,\tau} + y_{\beta_4,\tau}$
$y_{\beta_1,\tau}$
$y_{\beta_2,\tau}$
$y_{\beta_3,\tau}$
$y_{\beta_4,\tau}$


Darwin
Brisbane
Perth
Sydney
Adelaide
Melbourne
Hobart


$y_{Total,\tau}$
$y_{\beta_1,\tau} + y_{\beta_2,\tau}$
$y_{\beta_3,\tau} + y_{\beta_4,\tau}$
$y_{\beta_1,\tau}$
$y_{\beta_2,\tau}$
$y_{\beta_3,\tau}$
$y_{\beta_4,\tau}$


Darwin
Brisbane
Perth
Sydney
Adelaide
Melbourne
Hobart

Notation



The example's hierarchical, aggregated and base series are:

$$y_{\text{Total},\tau} = y_{\beta_1,\tau} + y_{\beta_2,\tau} + y_{\beta_3,\tau} + y_{\beta_4,\tau}$$

$$\mathbf{y}_{[a],\tau} = \left[y_{\text{Total},\tau},\; y_{\beta_1,\tau} + y_{\beta_2,\tau},\; y_{\beta_3,\tau} + y_{\beta_4,\tau}\right]^{\intercal}$$

$$\mathbf{y}_{[b],\tau} = \left[y_{\beta_1,\tau},\; y_{\beta_2,\tau},\; y_{\beta_3,\tau},\; y_{\beta_4,\tau}\right]^{\intercal}$$

The summing matrix of the example can be written as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{S}_{[a][b]} \\ \\ \mathbf{I}_{[b][b]} \end{bmatrix} = \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

nixtla

2 Step Reconciliation Strategies

Two-stage process, first a set of base forecasts $\hat{\mathbf{y}}_{[a,b],\tau} \in \mathbb{R}^{N_a+N_b}$ is created and

then adapted into coherent forecasts $\tilde{\mathbf{y}}_{[a,b],\tau}$

They can be expressed by:

$$\tilde{\mathbf{y}}_{[a,b],\tau} = \mathbf{H}\mathbf{P}\hat{\mathbf{y}}_{[a,b],\tau}$$

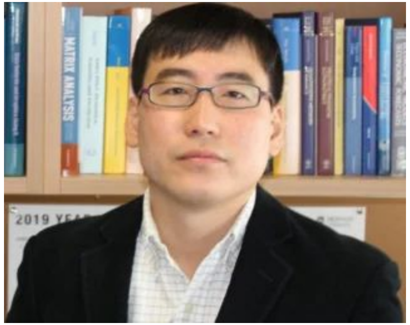
With the hierarchical constraints matrix and a projection matrix $\mathbf{P} \in \mathbb{R}^{N_b \times (N_a+N_b)}$

Gross, C. W., & Sohl, J. E. (1990). Disaggregation methods to expedite product line forecasting. Journal of Forecasting, 9, 233–254. Available at https://onlinelibrary.wiley.com/doi/abs/10.1002/for.3980090304.

nixtla

Statistical Approach: Minimize Variance

MONASH University



MinTrace

Wickramasuriya et al. (2019) show that the variance-covariance matrix of the h-step-ahead coherent forecast errors is given by

$$V_h = Var\left(\mathbf{y}_{T+h} - \tilde{\mathbf{y}}_h\right) = HPW_hP^TH^T$$

Where $W_h = Var\left(\mathbf{y}_{T+h} - \hat{\mathbf{y}}_h\right)$ is the variance-covariance matrix of the corresponding base forecast errors.

Wickramasuriya et al. (2019) show that the matrix P which minimizes the trace of V_h (the sum of all the error variances) such that $HPH = P$, is given by,

$$P = (H^TW_h^{-1}H)^{-1}H^TW_h^{-1}$$

Therefore, the optimally reconciled forecasts are given by,

$$\tilde{\mathbf{y}}_h = HG\hat{\mathbf{y}}_h = H(H^TW_h^{-1}H)^{-1}H^TW_h^{-1}\hat{\mathbf{y}}_h$$

To use this in practice, we need to estimate W_h, the forecast error variance of the h-step-ahead base forecasts. Usually, the matrix can be approximated (for example $W_h = k_hI$ recovers the ols method).

ERM

The Empirical Risk Minimization reconciliation strategy (Taieb et al., 2019) relaxes the unbiasedness assumptions from previous reconciliation methods like MinT and optimizes square errors between the reconciled predictions and the validation data to obtain an optimal reconciliation matrix P.

The exact solution for P follows the expression:

$$P^{*} = (H^{\intercal}H)^{-1}\,\mathbf{y}_{h}^{\intercal}\hat{\mathbf{y}}_{h}\,(\hat{\mathbf{y}}_{h}\hat{\mathbf{y}}_{h})^{-1}$$

The alternative Lasso regularized $\mathbf{P}$ solution is useful when the observations of validation data is limited or the exact solution has low numerical stability.

$$P^{*} = \text{argmin}_{P}||\mathbf{y}_{h} - HP\hat{\mathbf{y}}_{h}||_{2}^{2} + \lambda||P - P_{\text{BU}}||_{1}$$

nixtla

That sounds hard.

nixtla

Open Source Time Series Ecosystem

Stars 5k

⚡ StatsForecast

Lightning fast forecasting with statistical and econometric models.

Github

🤖 MLForecast

Scalable machine learning for time series forecasting.

Github

🧠 NeuralForecast

Scalable and user friendly neural forecasting algorithms for time series data.

Github

👑 Hierarchical Forecast

Probabilistic Hierarchical forecasting with statistical and econometric methods.

Github

🔧 TS features

Calculates various features from time series data. Python implementation of the R package *tsfeatures*.

Github

```
pip install hierarchicalforecast
```

```python
# Import aggregate function to construct hierarchies,
# summing matrix (H), and tags
from hierarchicalforecast.utils import aggregate

# Define different hierarchy levels
hierarchies = [['Country'], ['Country', 'State'], ['Country', 'State', 'Region']]

# Use aggregate function to create hierarchical time series dataframe, summing matrix (H), and tags
tourism_df, H_df, tags = aggregate(
    bottom_tourism_df,  # Input bottom-level time series
    spec=hierarchies    # Specify hierarchies to be created
)
```

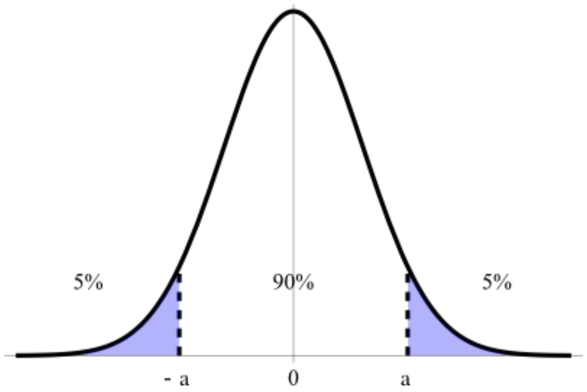
```python
# Import HierarchicalReconciliation class
from hierarchicalforecast.core import HierarchicalReconciliation

# Instantiate the HierarchicalReconciliation object with a list of reconcilers
hrec = HierarchicalReconciliation(
    reconcilers=[
        MinTrace(method='ols', nonnegative=True),  # Minimum trace method using OLS with nonnegative constraints
        ERM(method='closed'),                     # Empirical Risk Minimization using the closed form solution
    ]
)
```

nixtla

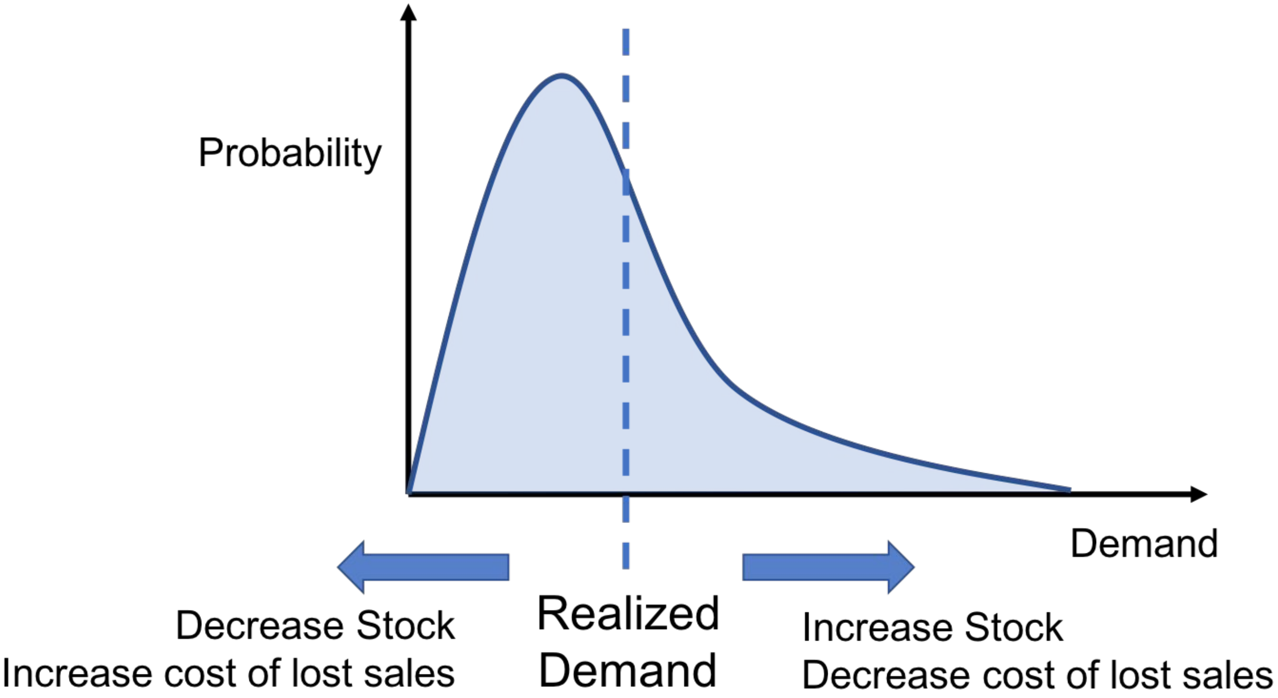
```python
# Reconcile the base forecasts using the HierarchicalReconciliation object
reconciled_fcsts_df = hrec.reconcile(
    base_forecasts_df,  # Base forecasts dataframe
    summing_matrix_df,  # Summing matrix dataframe
    tags,               # Hierarchy tags
    fitted_values_df    # Fitted values dataframe
)
```

What about uncertainty quantification?



nixtla

Not all errors are the same.



nixtla

Ok… that sounds hard.

```python
# Fit the model to the tourism_df dataset
mlf.fit(
    tourism_df,
    id_col='unique_id',                        # Unique identifier column
    time_col='ds',                             # Timestamp column
    target_col='y',                            # Target variable column
    prediction_intervals=PredictionIntervals( # Prediction intervals configuration
        n_windows=4,                           # Number of cross validation windows (conformal scores)
        window_size=12,                        # forecast horizon (1 year)
    )
)

# Generate predictions for the next 12 months and produce 80, and 90 prediction intervals
base_forecasts_df = mlf.predict(12, level=[80, 90])
```

nixtla

Confidence Intervals with Conformal Prediction

```python
# Fit the model to the tourism_df dataset
mlf.fit(
    tourism_df,
    id_col='unique_id',                          # Unique identifier column
    time_col='ds',                               # Timestamp column
    target_col='y',                              # Target variable column
    prediction_intervals=PredictionIntervals( # Prediction intervals configuration
        n_windows=4,                              # Number of cross validation windows (conformal scores)
        window_size=12,                           # forecast horizon (1 year)
    )
)

# Generate predictions for the next 12 months and produce 80, and 90 prediction intervals
base_forecasts_df = mlf.predict(12, level=[80, 90])
```

```python
# Fit the model to the tourism_df dataset
mlf.fit(
    tourism_df,
    id_col='unique_id',                     # Unique identifier column
    time_col='ds',                          # Timestamp column
    target_col='y',                         # Target variable column
    prediction_intervals=PredictionIntervals( # Prediction intervals configuration
        n_windows=4,                          # Number of cross validation windows (conformal scores)
        window_size=12,                       # forecast horizon (1 year)
    )
)

# Generate predictions for the next 12 months and produce 80, and 90 prediction intervals
base_forecasts_df = mlf.predict(12, level=[80, 90])
```

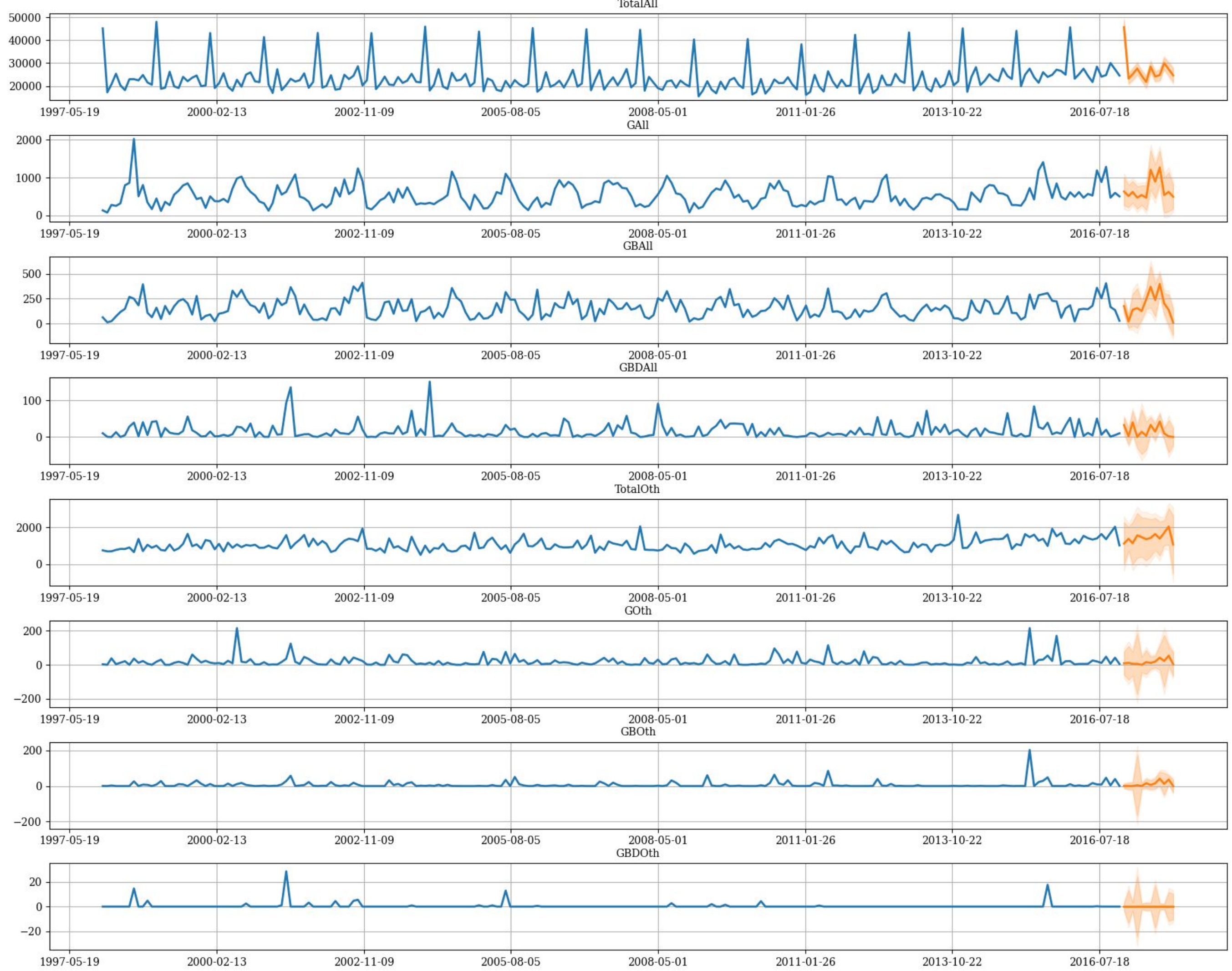

GABAll
ECABus
BDHol
AFABus
CVis
BDABus
BDDHol
CBBVis
y
LGBMRegressor
LGBMRegressor_level_80
LGBMRegressor_level_90
Target [y]
Datestamp [ds]
2013
2014
2015
2016
2017

nixtla

```python
# Reconcile the base forecasts using the HierarchicalReconciliation object
reconciled_fcsts_df = hrec.reconcile(
    base_forecasts_df,  # Base forecasts dataframe
    summing_matrix_df,  # Summing matrix dataframe
    tags,               # Hierarchy tags
    fitted_values_df,   # Fitted values dataframe
    level=[80, 90],     # Prediction intervals
)
```

```python
# Reconcile the base forecasts using the HierarchicalReconciliation object
reconciled_fcsts_df = hrec.reconcile(
    base_forecasts_df,  # Base forecasts dataframe
    summing_matrix_df,  # Summing matrix dataframe
    tags,               # Hierarchy tags
    fitted_values_df,   # Fitted values dataframe
    level=[80, 90],     # Prediction intervals
)
```

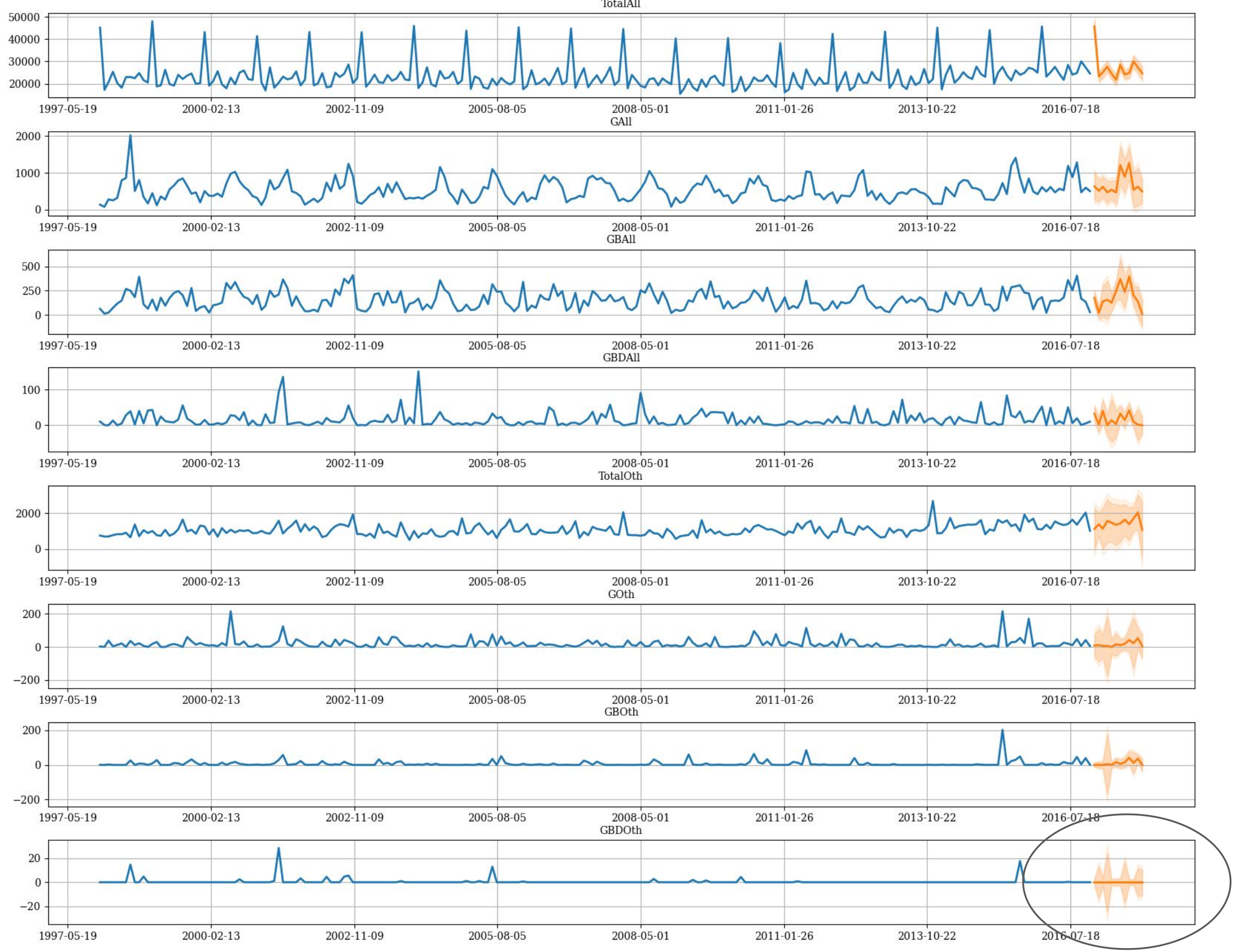
nixtla


TotalAll
50000
40000
30000
20000
GAll
2000
1000
0
GBAll
500
250
0
GBDAll
100
0
TotalOth
2000
0
GOth
200
0
−200
GBOth
200
0
−200
GBDOth
20
0
−20
1997-05-19
2000-02-13
2002-11-09
2005-08-05
2008-05-01
2011-01-26
2013-10-22
2016-07-18

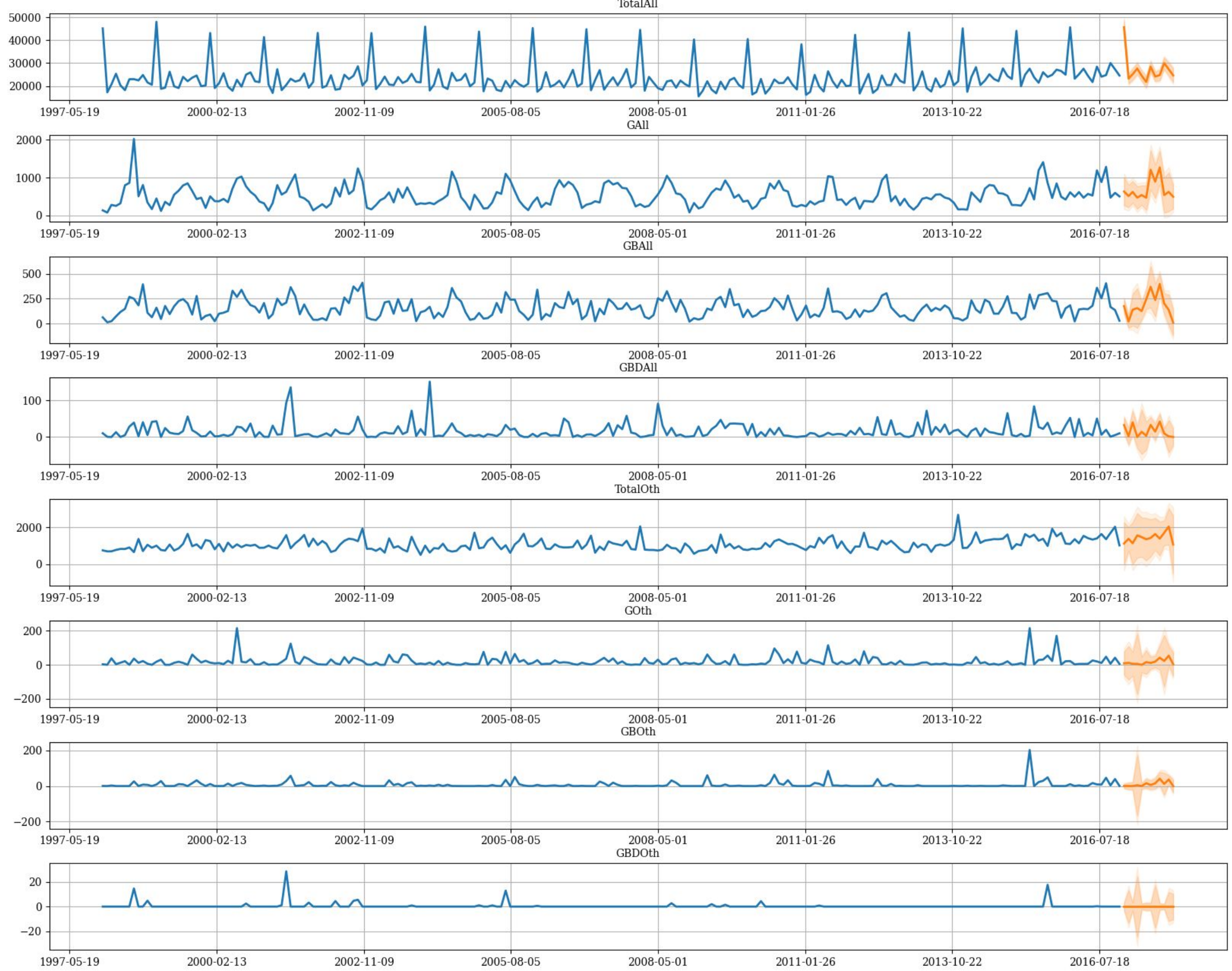
Warning!
Scientists at work

Problems with classical approaches

Normality assumption


TotalAll
50000
40000
30000
20000
GAll
2000
1000
0
GBAll
500
250
0
GBDAll
100
0
TotalOth
2000
0
GOth
200
0
−200
GBOth
200
0
−200
GBDOth
20
0
−20
1997-05-19
2000-02-13
2002-11-09
2005-08-05
2008-05-01
2011-01-26
2013-10-22
2016-07-18
nixtla

Univariate approach



TotalAll
50000
40000
30000
20000
GAll
2000
1000
0
GBAll
500
250
0
GBDAll
100
0
TotalOth
2000
0
GOth
200
0
−200
GBOth
200
0
−200
GBDOth
20
0
−20
1997-05-19
2000-02-13
2002-11-09
2005-08-05
2008-05-01
2011-01-26
2013-10-22
2016-07-18

Complex pipeline (2 steps)

nixtla

Solution?

Hierarchical Mixture Networks (HINT)

nixtla

Flexible (mixture) and efficient (composite likelihood) multivariate probability.

Expands vast collection of neural forecasting methods in a single framework.

Mixtures Mesh

Probabilistic Coherent Distribution

A probabilistic hierarchical coherent distribution, is a multivariate forecasting system that satisfies that for a set of random variables (A, B, C) with $C = A + B$.

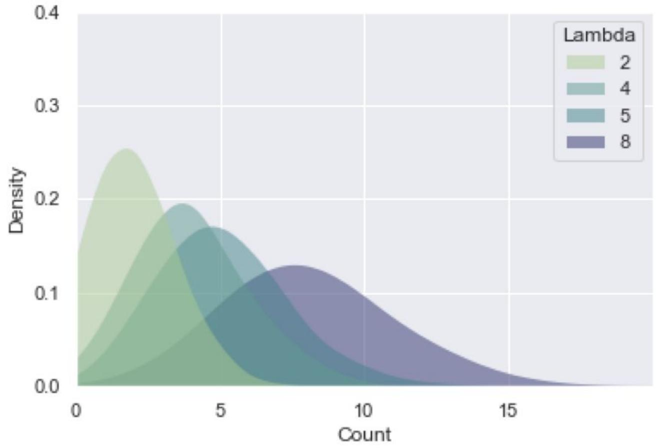
The marginal distributions satisfy:

$$\mathrm{P}(A) = \sum_{B} \mathrm{P}(A, B) \quad \text{and} \quad \mathrm{P}(B) = \sum_{A} \mathrm{P}(A, B)$$

$$\mathrm{P}(C) = \sum_{A,B} \mathrm{P}(A, B)\, \mathbb{1}(C = A + B)$$

Poisson Mixtures Mesh

The foundation of HINT models is the assumption that the joint distribution of a a time series $\mathbf{y}_{[b][t+1:t+H]}$ is described by a Mixture distribution with the **component matching assumption**, that achieves by construction probabilistic hierarchical coherence.

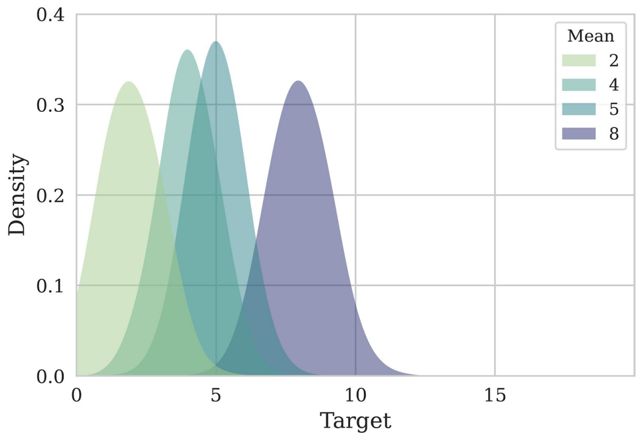


$$\mathbb{P}(\mathbf{y}_{[b][t+1:t+H]}) = \sum_{k=1}^{N_k} w_k \prod_{(\beta,\tau)\in[b][t+1:t+H]} \text{Poisson}(y_{\beta,\tau} \mid \lambda_{\beta,\tau})$$

nixtla

Gaussian Mixture Mesh

The foundation of HINT models is the assumption that the joint distribution of a a time series is described by a Mixture distribution with the **component matching assumption**, that achieves by construction probabilistic hierarchical coherence.

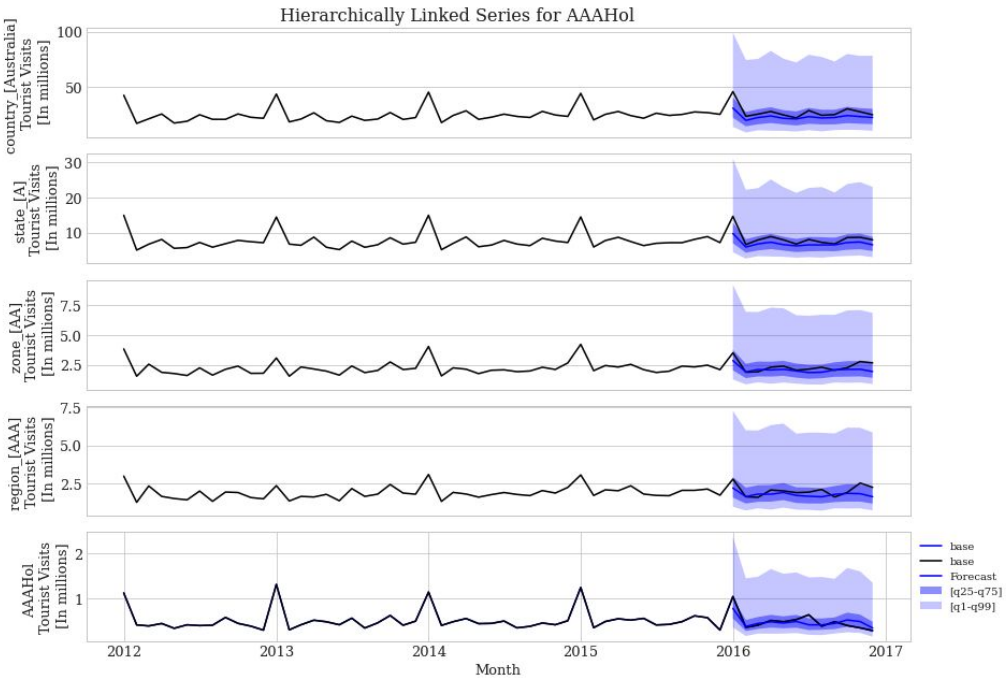


$$\mathbb{P}(\mathbf{y}_{[b][t+1:t+H]}) = \sum_{\kappa=1}^{K} w_{\kappa} \prod_{(\beta,\tau)\in[b][t+1:t+H]} \mathrm{Normal}(y_{\beta,\tau} \mid \mu_{\beta,\tau,\kappa},\ \sigma_{\beta,\tau,\kappa})$$

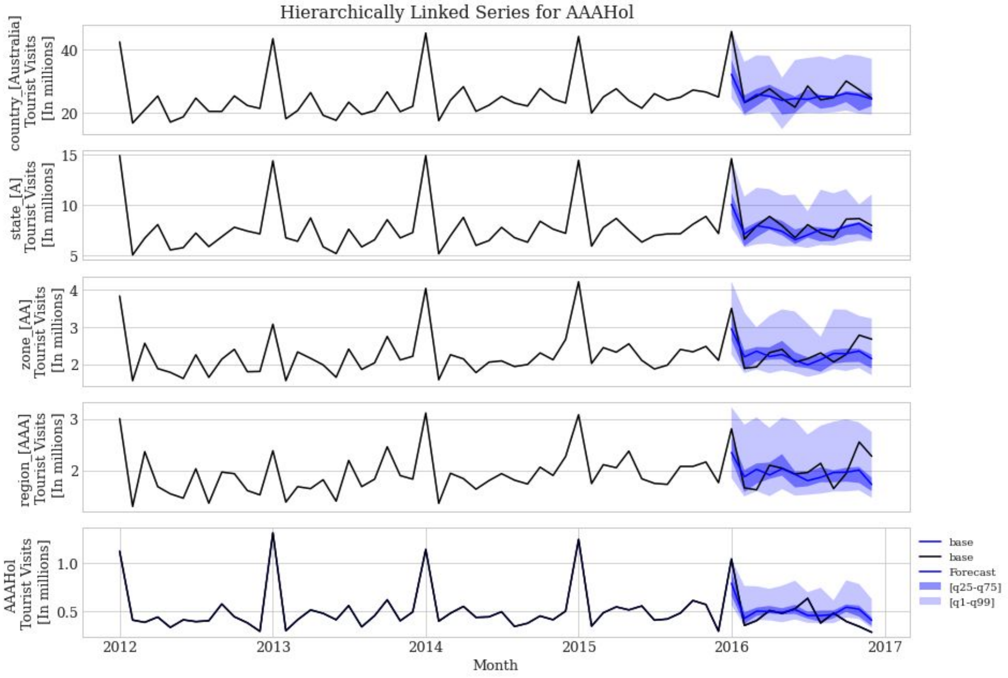
We reached the maximum number of allowed equations…

nixtla

Naive Approach (Independence)



Informed approach (Correlation Groups)



nixtla

Why is this cool?

Now you can do hierarchical forecasting with your favorite deep learning model

nixtla

Ok… but this time it must be hard…

nixtla

nixtla

Open Source Time Series Ecosystem

Stars 5k

⚡ StatsForecast

Lightning fast forecasting with statistical and econometric models.

Github

🤖 MLForecast

Scalable machine learning for time series forecasting.

Github

🧠 NeuralForecast

Scalable and user friendly neural forecasting algorithms for time series data.

Github

```
pip install neuralforecast
```

Let’s fit a hierarchical coherent LSTM with a PMM

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint], # Define models
    freq='MS',                    # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

nixtla

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint], # Define models
    freq='MS',                      # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

nixtla

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint], # Define models
    freq='MS',                     # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

nixtla

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint], # Define models
    freq='MS',                      # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint], # Define models
    freq='MS',                    # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

nixtla

Let’s include a hierarchical coherent NHiTS with a GMM

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

#  NHITS with Gaussian Mixture
nhits = NHITS(h=12,
              input_size=24,
              loss=GMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')
# Hierarchical Reconciliation using NHITS model
nhits_hint = HINT(h=12, model=nhits, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint, nhits_hint], # Define models
    freq='MS',                      # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

#  NHITS with Gaussian Mixture
nhits = NHITS(h=12,
              input_size=24,
              loss=GMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')
# Hierarchical Reconciliation using NHITS model
nhits_hint = HINT(h=12, model=nhits, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint, nhits_hint], # Define models
    freq='MS',                      # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

nixtla

```python
from neuralforecast import NeuralForecast
from neuralforecast.models import NHITS, LSTM, HINT
from neuralforecast.losses.pytorch import GMM, PMM

# Base models
# LSTM with Poisson Mixture
lstm = LSTM(h=12,
            loss=PMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

#  NHITS with Gaussian Mixture
nhits = NHITS(h=12,
              input_size=24,
              loss=GMM(n_components=2, num_samples=100, quantiles=list(np.arange(100)/100)))

# Hierarchical Reconciliation using LSTM model
lstm_hint = HINT(h=12, model=lstm, H=H_df, group_level=1, reconciliation='bottom_up')
# Hierarchical Reconciliation using NHITS model
nhits_hint = HINT(h=12, model=nhits, H=H_df, group_level=1, reconciliation='bottom_up')

# Fit and Predict
fcst = NeuralForecast(
    models=[lstm_hint, nhits_hint], # Define models
    freq='MS',                      # Monthly frequency
)
fcst.fit(df=tourism_df) # Fit neuralforecast
forecasts = fcst.predict() # Predict using trained model
```

Wrap Up


FORECASTING ALL LEVELS
RECONCILING HIERACHIES
PROBABILISTIC RECONCILATION
HIERARCHICAL MIXTURE NETWORKS (HINT)
imgflip.com

Show some love

nixtla

@nixtlainc

👑 Hierarchical Forecast

Probabilistic Hierarchical forecasting with statistical and econometric methods.

Github

🧠 NeuralForecast

Scalable and user friendly neural forecasting algorithms for time series data.

Github

🤖 MLForecast

Scalable machine learning for time series forecasting.

Github

HierarchicalForecast: A Reference Framework for Hierarchical Forecasting in Python

Kin G. Olivares* KDGUTIER@CS.CMU.EDU
Federico Garza* FEDERICO@NIXTLA.IO
David Luo DJLUO@CS.CMU.EDU
Cristian Challu CCHALLU@CS.CMU.EDU
Max Mergenthaler-Canseco MAX@NIXTLA.IO
Souhaib Ben Taieb SOUHAIB.BENTAIEB@UMONS.AC.BE
Shanika L. Wickramasuriya S.WICKRAMASURIYA@AUCKLAND.AC.NZ
Artur Dubrawski AWD@CS.CMU.EDU

Reference

Forecasting: Principles and Practice (3rd ed)

Rob J Hyndman and George Athanasopoulos

Monash University, Australia

References

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Tarkmen, and Yuyang Wang. GluonTS: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. URL `http://jmlr.org/papers/v21/19-820.html`.

Oren Anava, Vitaly Kuznetsov, and (Google Inc. Sponsorship). Web traffic time series forecasting, forecast future traffic to wikipedia pages. Kaggle Competition, 2018. URL `https://www.kaggle.com/c/web-traffic-time-series-forecasting/`.

Australian Bureau of Statistics. Labour Force, Australia. Accessed Online, 2019. URL `https://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/6202.0Dec%202019?OpenDocument`.

Souhaib Ben Taieb and Bonsoo Koo. Regularized regression for hierarchical forecasting without unbiasedness conditions. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1337–1347, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330976. URL `https://doi.org/10.1145/3292500.3330976`.

Souhaib Ben Taieb, James W. Taylor, and Rob J. Hyndman. Coherent probabilistic forecasts for hierarchical time series. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3348–3357. PMLR, 06–11 Aug 2017. URL `http://proceedings.mlr.press/v70/taieb17a.html`.

David Bolin and Jonas Wallin. Local scale invariance and robustness of proper scoring rules, 2019. URL `https://arxiv.org/abs/1912.05642`.

Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Daniel M. Dunn, William H. Williams, and T. L. Dechaine. Aggregate versus subaggregate models in local area forecasting. *Journal of the American Statistical Association*, 71(353): 68–71, 1976.

Gene Fliedner. An investigation of aggregate variable time series forecast strategies with specific subaggregate time series statistical correlation. *Computers and Operations Research*, 26(10–11):1133–1149, September 1999. ISSN 0305-0548. doi: 10.1016/S0305-0548(99)00017-9. URL `https://doi.org/10.1016/S0305-0548(99)00017-9`.

Federico Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G. Olivares. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022. URL `https://github.com/Nixtla/statsforecast`.

Tilmann Gneiting. Quantiles as optimal point forecasts. *International Journal of Forecasting*, 27(2):197–207, 2011. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2009.12.015. URL `https://www.sciencedirect.com/science/article/pii/S0169207010000063`.

Charles W. Gross and Jeffrey E. Sohl. Disaggregation methods to expedite product line forecasting. *Journal of Forecasting*, 9(3):233–254, 1990. doi: 10.1002/for.3980090304. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/for.3980090304`.

Xing Han, Sambarta Dasgupta, and Joydeep Ghosh. Simultaneously reconciled quantile forecasting of hierarchically related time series. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 190–198. PMLR, 13–15 Apr 2021. URL `http://proceedings.mlr.press/v130/han21a.html`.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

Julien Herzen, Francesco Lassig, Samuele Giuliano Piazzetta, Thomas Neuer, Lao Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Koacisz, Dennis Bader, Frederick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gael Grosch. DARTS: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6, 2022. URL `http://jmlr.org/papers/v23/21-1177.html`.

Tao Hong, Pierre Pinson, and Shu Fan. Global Energy Forecasting Competition 2012. *International Journal of Forecasting*, 30(2):357–363, 2014. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2013.07.001. URL `https://www.sciencedirect.com/science/article/pii/S0169207013000745`.

Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software, Articles*, 27(3):1–22, 2008. ISSN 1548-7660. doi: 10.18637/jss.v027.i03. URL `https://www.jstatsoft.org/v027/i03`.

Tim Januschowski, Jan Gasthaus, and Yuyang (Bernie) Wang. Open-source forecasting tools in python. *Foresight Journal of Applied Forecasting*, 2019. URL https://www.amazon.science/publications/open-source-forecasting-tools-in-pytho

Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodriguez, Chao Zhang, and B. Prakash. PROFHIT: Probabilistic robust forecasting for hierarchical time-series. *Computing Research Repository*, 06 2022. URL https://arxiv.org/abs/2206.07940.

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340052. doi: 10.1145/2833157.2833162. URL https://doi.org/10.1145/2833157.2833162.

Markus Löning, Anthony J. Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Király. SKTIME: A unified interface for machine learning with time series. *Computing Research Repository*, abs/1909.07872, 2019. URL http://arxiv.org/abs/1909.07872.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M5 competition: Background, organization, and implementation. *International Journal of Forecasting*, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.07.007. URL https://www.sciencedirect.com/science/article/pii/S0169207021001187.

Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Zhi Chen, Anil Gaba, Ilia Tsetlin, and Robert L. Winkler. The M5 uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting*, 38(4):1365–1385, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.10.009. URL https://www.sciencedirect.com/science/article/pii/S0169207021001722. Special Issue: M5 competition.

Carlo Mazzaferro. *scikit-hts: Hierarchical Time Series Forecasting with a familiar API*, 2019. URL https://scikit-hts.readthedocs.io/en/latest/. Python package.

Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

Kin G. Olivares, Nganba Meetei, Ruijun Ma, Rohan Reddy, Mengfei Cao, and Lee Dicker. Probabilistic hierarchical forecasting with deep poisson mixtures. *International Journal of Forecasting, submitted*, Working Paper version available at arXiv:2110.13179, 2021. URL https://arxiv.org/abs/2110.13179.

Guy H. Orcutt, Harold W. Watts, and John B. Edwards. Data aggregation and information loss. *The American Economic Review*, 58(4):773–787, 1968. ISSN 00028282. URL http://www.jstor.org/stable/1815532.

Anastasios Panagiotelis, Puwasala Gamakumara, George Athanasopoulos, and Rob J. Hyndman. Probabilistic forecast reconciliation: Properties, evaluation and score optimisation. *European Journal of Operational Research*, 306(2):693–706, 2023. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2022.07.040. URL https://www.sciencedirect.com/science/article/pii/S0377221722006087.

Biswajit Paria, Rajat Sen, Amr Ahmed, and Abhimanyu Das. Hierarchically Regularized Deep Forecasting. In *Submitted to Proceedings of the 39th International Conference on Machine Learning*. PMLR. Working Paper version available at arXiv:2106.07630, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Gregory Piatetsky. Python eats away at R: Top software for analytics, data science, machine learning in 2018: Trends and analysis. https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learni 2018. Accessed: 2022-07-05.

Syama Sundar Rangapuram, Lucien D. Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In Maria Florina Balcan and Marina Meila, editors, *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 06–11 Aug 2021.

Julien Siebert, Janek Groß, and Christof Schroth. A systematic review of python packages for time series analysis. *Engineering Proceedings*, 5, 2021. doi: https://doi.org/10.3390/engproc2021005022. URL https://arxiv.org/abs/2104.07406.

Tourism Australia, Canberra. Tourism Research Australia (2005), Travel by Australians. https://www.kaggle.com/luisblanche/quarterly-tourism-in-australia/, Sep 2005.

Tourism Australia, Canberra. Detailed tourism Australia (2005), Travel by Australians, Sep 2019. Accessed at https://robjhyndman.com/publications/hierarchical-tourism/.

Shanika L. Wickramasuriya. Probabilistic forecast reconciliation under the Gaussian framework. *Accepted at Journal of Business and Economic Statistics*, 2023.

Shanika L. Wickramasuriya, George Athanasopoulos, and Rob J. Hyndman. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, 114(526):804–819, 2019. doi: 10.1080/01621459.2018.1448825. URL https://robjhyndman.com/publications/mint/.

Bohan Zhang, Yanfei Kang, and Feng Li. *pyths: A python package for hierarchical forecasting*, 2022. URL https://angelpone.github.io/pyhts/tutorials/Tutorials.html. Python package.