S09_01

1 SERIES TEMPORALES: MODELOS PREDICTIVOS

1.1 MODELOS DE SERIES JERARQUICAS

1.2 `v2.2`

1.3 Bibliografía

1.3.1 Básica:

- Rob J. Hyndman and George Athanasopoulos (2018). **“Forecasting principles and practice, Hierarchical and Grouped Series”**. https://otexts.com/fpp3/hierarchical.html.
- Olivares, K. Garza, F. Luo, D. Challú, C. and Mergenthaler, M. and A. Dubrawski, **“HierarchicalForecast: A Reference Framework for Hierarchical Forecasting in Python”**, https://arxiv.org/abs/2207.03517, Jun 2022.

1.3.2 Adicional:

- Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang, **“Optimal combination forecasts for hierarchical time series”** Computational Statistics & Data Analysis, vol. 55, no. 9, pp. 2579–2589, Sep. 2011.
- Panagiotelis, G. Athanasopoulos, P. Gamakumara, and R. J. Hyndman, **“Forecast reconciliation: A geometric view with new insights on bias correction”** International Journal of Forecasting, vol. 37, no. 1, pp. 343–359, Jan. 2021.
- Wickramasuriya, G. Athanasopoulos, and R. J. Hyndman, **“Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization”** Journal of the American Statistical Association, vol. 114, no. 526, pp. 804–819, Apr. 2019

1.4 Series Jerarquicas (Hierarchical)

- Las series de tiempo a menudo se pueden desagregar naturalmente por varios atributos de interés.
- Por ejemplo, el número total de coches vendidos por un fabricante puede desglosarse por tipo de producto, como coches turismo, SUV, electrico, …. Cada uno de estos puede ser desagregado en categorías más finas.
- Estas categorías están anidadas dentro de las categorías de grupos más grandes, por lo que la colección de series temporales sigue una estructura de agregación jerárquica. Por lo tanto, nos referimos a estos como “series temporales jerárquicas”.

- Las series de tiempo jerárquicas a menudo surgen debido a divisiones geográficas. Por ejemplo, las ventas totales de coches se pueden desglosar por país, luego dentro de cada país por estado, dentro de cada estado por región, y así sucesivamente hasta el punto de venta.

En tales escenarios, a menudo se requiere que los modelos proporcionen predicciones para todas las series desagregadas y agregadas. Un deseo natural es que esas predicciones sean **"coherentes"**, es decir, que la serie inferior se sume con precisión a los pronósticos de la serie agregada.

$n =$ total number of series in the hierarchy; $n = 1 + 2 + 5 = 8$

$m =$ the number of series at the bottom level; $m = 5$

Siempre $n > m$

$$y_t = y_{AA,t} + y_{AB,t} + y_{AC,t} + y_{BA,t} + y_{BB,t}$$
$$y_{A,t} = y_{AA,t} + y_{AB,t} + y_{AC,t} \qquad \text{and} \qquad y_{B,t} = y_{BA,t} + y_{BB,t}$$

$$y_t = y_{A,t} + y_{B,t}$$

[ ]:
```python
import warnings
warnings.filterwarnings('ignore')
```

1.5 Ejemplo Australia

$$y_{\text{Total},\tau} = y_{\beta_1,\tau} + y_{\beta_2,\tau} + y_{\beta_3,\tau} + y_{\beta_4,\tau} \tag{1}$$
$$\mathbf{y}_{[a],\tau} = \left[y_{\text{Total},\tau},\ y_{\beta_1,\tau} + y_{\beta_2,\tau},\ y_{\beta_3,\tau} + y_{\beta_4,\tau}\right]^\top \qquad \mathbf{y}_{[b],\tau} = \left[y_{\beta_1,\tau},\ y_{\beta_2,\tau},\ y_{\beta_3,\tau},\ y_{\beta_4,\tau}\right]^\top \tag{2}$$

Las restricciones de agregación se puede expresar como una matriz:

$$\mathbf{S}_{[a,b][b]} = \begin{bmatrix} \mathbf{A}_{\text{[a][b]}} \\ \\ \\ \mathbf{I}_{\text{[b][b]}} \\ \\ \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

donde $\mathbf{A}_{[a,b][b]}$ agrega las series de abajo a arriba arriba (bottom) y $\mathbf{I}_{\text{[b][b]}}$ es una matriz identidad. Entonce la serie jerarquica se puede representar como:

$$\mathbf{y}_{[a,b],\tau} = \mathbf{S}_{[a,b][b]}\mathbf{y}_{[b],\tau} \tag{4}$$

Para lograr la "coherencia", la mayoría de las soluciones estadísticas al desafío de la previsión jerárquica implementan un proceso de reconciliación en dos etapas.

1º obtenemos un conjunto del pronóstico base $\hat{\mathbf{y}}_{[a,b],\tau}$

2º los reconciliamos en pronósticos coherentes $\tilde{\mathbf{y}}_{[a,b],\tau}$.

La mayoría de los métodos de reconciliación jerárquica se pueden expresar mediante las siguientes transformaciones:

$$\tilde{\mathbf{y}}_{[a,b],\tau} = \mathbf{S}_{[a,b][b]}\mathbf{P}_{[b][a,b]}\hat{\mathbf{y}}_{[a,b],\tau} \tag{5}$$

Es necesario tener instalado Hierarchicalforecast

https://nixtla.github.io/hierarchicalforecast/

pip install hierarchicalforecast

conda install -c conda-forge hierarchicalforecast

pip install –target=$nb_path -U numba statsforecast datasetsforecast

[ ]:
```
#!pip install statsforecast datasetsforecast
```

[ ]:
```
#!pip install hierarchicalforecast
```

[ ]:
```
import numpy as np
import pandas as pd

#obtain hierarchical dataset
from datasetsforecast.hierarchical import HierarchicalData

# compute base forecast no coherent
from statsforecast.core import StatsForecast
from statsforecast.models import AutoARIMA, Naive

#obtain hierarchical reconciliation methods and evaluation
from hierarchicalforecast.core import HierarchicalReconciliation
from hierarchicalforecast.evaluation import HierarchicalEvaluation
from hierarchicalforecast.methods import BottomUp, TopDown, MiddleOut
```

Usamos el TourismSmall dataset.

La siguiente celda obtiene la serie temporal(Y_df) para los diferentes niveles de la jerarquía, la matriz de suma (S_df) que recupera el conjunto de datos completo de la jerarquía de nivel inferior y los índices de cada jerarquía indicados por etiquetas (tags).

[ ]:
```
# Load TourismSmall dataset
Y_df, S_df, tags = HierarchicalData.load('./data', 'TourismSmall')
Y_df['ds'] = pd.to_datetime(Y_df['ds'])
```

[ ]:
```
Y_df.tail()
```

[ ]:
```
            unique_id         ds    y
3199  nt-oth-noncity 2005-12-31   59
3200  nt-oth-noncity 2006-03-31   25
3201  nt-oth-noncity 2006-06-30   52
3202  nt-oth-noncity 2006-09-30   72
3203  nt-oth-noncity 2006-12-31  138
```

[ ]:
```
S_df.iloc[:5, :5]
```

[ ]:
```
       nsw-hol-city  nsw-hol-noncity  vic-hol-city  vic-hol-noncity  \
total           1.0              1.0           1.0              1.0
hol             1.0              1.0           1.0              1.0
vfr             0.0              0.0           0.0              0.0
bus             0.0              0.0           0.0              0.0
oth             0.0              0.0           0.0              0.0

       qld-hol-city
total           1.0
hol             1.0
vfr             0.0
bus             0.0
oth             0.0
```

[ ]:
```
# holidays, business, visiting other

tags
```

[ ]:
```
{'Country': array(['total'], dtype=object),
 'Country/Purpose': array(['hol', 'vfr', 'bus', 'oth'], dtype=object),
 'Country/Purpose/State': array(['nsw-hol', 'vic-hol', 'qld-hol', 'sa-hol', 'wa-
hol', 'tas-hol',
        'nt-hol', 'nsw-vfr', 'vic-vfr', 'qld-vfr', 'sa-vfr', 'wa-vfr',
        'tas-vfr', 'nt-vfr', 'nsw-bus', 'vic-bus', 'qld-bus', 'sa-bus',
        'wa-bus', 'tas-bus', 'nt-bus', 'nsw-oth', 'vic-oth', 'qld-oth',
        'sa-oth', 'wa-oth', 'tas-oth', 'nt-oth'], dtype=object),
 'Country/Purpose/State/CityNonCity': array(['nsw-hol-city', 'nsw-hol-noncity',
```

```
'vic-hol-city',
        'vic-hol-noncity', 'qld-hol-city', 'qld-hol-noncity',
        'sa-hol-city', 'sa-hol-noncity', 'wa-hol-city', 'wa-hol-noncity',
        'tas-hol-city', 'tas-hol-noncity', 'nt-hol-city', 'nt-hol-noncity',
        'nsw-vfr-city', 'nsw-vfr-noncity', 'vic-vfr-city',
        'vic-vfr-noncity', 'qld-vfr-city', 'qld-vfr-noncity',
        'sa-vfr-city', 'sa-vfr-noncity', 'wa-vfr-city', 'wa-vfr-noncity',
        'tas-vfr-city', 'tas-vfr-noncity', 'nt-vfr-city', 'nt-vfr-noncity',
        'nsw-bus-city', 'nsw-bus-noncity', 'vic-bus-city',
        'vic-bus-noncity', 'qld-bus-city', 'qld-bus-noncity',
        'sa-bus-city', 'sa-bus-noncity', 'wa-bus-city', 'wa-bus-noncity',
        'tas-bus-city', 'tas-bus-noncity', 'nt-bus-city', 'nt-bus-noncity',
        'nsw-oth-city', 'nsw-oth-noncity', 'vic-oth-city',
        'vic-oth-noncity', 'qld-oth-city', 'qld-oth-noncity',
        'sa-oth-city', 'sa-oth-noncity', 'wa-oth-city', 'wa-oth-noncity',
        'tas-oth-city', 'tas-oth-noncity', 'nt-oth-city', 'nt-oth-noncity'],
       dtype=object)}
```

Dividimos los datos en entrenamiento/test

[ ]:
```
#split train/test sets
Y_test_df  = Y_df.groupby('unique_id').tail(12)
Y_train_df = Y_df.drop(Y_test_df.index)
Y_test_df  = Y_test_df.set_index('unique_id')
Y_train_df = Y_train_df.set_index('unique_id')
```

[ ]:
```
Y_test_df
```

[ ]:
```
                       ds      y
unique_id
total          2004-03-31  85852
total          2004-06-30  66981
total          2004-09-30  73840
total          2004-12-31  70217
total          2005-03-31  85992
…                     …    …
nt-oth-noncity 2005-12-31     59
nt-oth-noncity 2006-03-31     25
nt-oth-noncity 2006-06-30     52
nt-oth-noncity 2006-09-30     72
nt-oth-noncity 2006-12-31    138

[1068 rows x 2 columns]
```

La siguiente celda calcula la predicción base para cada serie temporal utilizando los modelos auto_arima e ingenuo.

Y_hat_df contiene las predicciones pero no son coherentes.

[ ]:
```
# Compute base auto-ARIMA predictions
fcst = StatsForecast(df=Y_train_df,
                     models=[AutoARIMA(season_length=12), Naive()],
                     freq='M', n_jobs=-1)
Y_hat_df = fcst.forecast(h=12)
```

Los métodos utilizados para hacer **coherentes** los pronósticos son:

- **BottomUp**: La conciliación es una simple adición a los niveles superiores.
- **TopDown:** el segundo método restringe las predicciones de nivel base a la serie de nivel agregado superior y luego las distribuye a la serie desagregada mediante el uso de proporciones. Utiliza el método de predicióndе las proporciones.
- **MiddleOut**: las predicciones base son un nivel medio.

[ ]:
```
# Reconcile the base predictions
reconcilers = [
    BottomUp(),
    TopDown(method='forecast_proportions'),
    MiddleOut(middle_level='Country/Purpose/State',
              top_down_method='forecast_proportions')
]
hrec = HierarchicalReconciliation(reconcilers=reconcilers)
Y_rec_df = hrec.reconcile(Y_hat_df=Y_hat_df, Y_df=Y_train_df,
                          S=S_df, tags=tags)
```

El paquete HierarchicalForecast incluye la clase HierarchicalEvaluation para evaluar las diferentes jerarquías y también es capaz de calcular métricas escaladas en comparación con un modelo de referencia.

[ ]:
```
def mse(y, y_hat):
    return np.mean((y-y_hat)**2)

evaluator = HierarchicalEvaluation(evaluators=[mse])
```

Hierarchical Evaluation Method.

Parameters:
Y_hat_df: pd.DataFrame, Forecasts indexed by ‘unique_id’ with column ‘ds’ and models to evaluate.
Y_test_df: pd.DataFrame, True values with columns [‘ds’, ‘y’].
tags: np.array, each str key is a level and its value contains tags associated to that level.
Y_df: pd.DataFrame, Training set of base time series with columns [‘ds’, ‘y’] indexed by unique_id.
benchmark: str, If passed, evaluators are scaled by the error of this benchark.

Returns:
evaluation: pd.DataFrame with accuracy measurements across hierarchical levels.

[ ]:
```
evaluator.evaluate(Y_hat_df=Y_rec_df, Y_test_df=Y_test_df, tags=tags)
```

[ ]:
```
                                           AutoARIMA           Naive  \
level                             metric
Overall                           mse     1871861.665261   1953449.930712
Country                           mse     74199775.73822  70537977.416667
Country/Purpose                   mse     13612987.42582  15021186.854167
Country/Purpose/State             mse      775954.230324    858691.401786
Country/Purpose/State/CityNonCity mse       289593.64853    342695.709821

                                         AutoARIMA/BottomUp   Naive/BottomUp  \
level                             metric
Overall                           mse          1785268.951719   1953449.930712
Country                           mse         69738334.704122  70537977.416667
Country/Purpose                   mse         13042064.979425  15021186.854167
Country/Purpose/State             mse            741610.63441    858691.401786
Country/Purpose/State/CityNonCity mse            289593.64853    342695.709821

                                         AutoARIMA/TopDown_method-
forecast_proportions  \
level                             metric
Overall                           mse
1886451.103213
Country                           mse
74199775.73822
Country/Purpose                   mse
13335627.133582
Country/Purpose/State             mse
810258.568489
Country/Purpose/State/CityNonCity mse
315439.714209

                                         Naive/TopDown_method-
forecast_proportions  \
level                             metric
Overall                           mse
1953450.077714
Country                           mse
70537977.416667
Country/Purpose                   mse
15021189.419353
Country/Purpose/State             mse
858691.448579
Country/Purpose/State/CityNonCity mse
342695.736825

                                         AutoARIMA/MiddleOut_middle_level-
Country/Purpose/State_top_down_method-forecast_proportions  \
level                             metric
```

```
Overall                          mse
1767305.813749
Country                          mse
68067688.365514
Country/Purpose                  mse
12643020.185484
Country/Purpose/State            mse
775954.230324
Country/Purpose/State/CityNonCity mse
302209.461913

                                         Naive/MiddleOut_middle_level-
Country/Purpose/State_top_down_method-forecast_proportions
level                            metric
Overall                          mse
1953449.930712
Country                          mse
70537977.416667
Country/Purpose                  mse
15021186.854167
Country/Purpose/State            mse
858691.401786
Country/Purpose/State/CityNonCity mse
342695.709821
```

[ ]:
```
evaluator.evaluate(Y_hat_df=Y_rec_df, Y_test_df=Y_test_df, tags=tags,
 benchmark='Naive')
```

[ ]:
```
                                             AutoARIMA Naive  \
level                            metric
Overall                          mse-scaled  0.958234   1.0
Country                          mse-scaled  1.051912   1.0
Country/Purpose                  mse-scaled  0.906252   1.0
Country/Purpose/State            mse-scaled  0.903647   1.0
Country/Purpose/State/CityNonCity mse-scaled  0.845046   1.0

                                             AutoARIMA/BottomUp  \
level                            metric
Overall                          mse-scaled            0.913906
Country                          mse-scaled            0.988664
Country/Purpose                  mse-scaled            0.868245
Country/Purpose/State            mse-scaled            0.863652
Country/Purpose/State/CityNonCity mse-scaled            0.845046

                                             Naive/BottomUp  \
level                            metric
Overall                          mse-scaled             1.0
```

```
Country                            mse-scaled            1.0
Country/Purpose                    mse-scaled            1.0
Country/Purpose/State              mse-scaled            1.0
Country/Purpose/State/CityNonCity mse-scaled            1.0

                                             AutoARIMA/TopDown_method-
forecast_proportions  \
level                              metric
Overall                            mse-scaled
0.965702
Country                            mse-scaled
1.051912
Country/Purpose                    mse-scaled
0.887788
Country/Purpose/State              mse-scaled
0.943597
Country/Purpose/State/CityNonCity mse-scaled
0.920466

                                             Naive/TopDown_method-
forecast_proportions  \
level                              metric
Overall                            mse-scaled
1.0
Country                            mse-scaled
1.0
Country/Purpose                    mse-scaled
1.0
Country/Purpose/State              mse-scaled
1.0
Country/Purpose/State/CityNonCity mse-scaled
1.0

                                             AutoARIMA/MiddleOut_middle_level-
Country/Purpose/State_top_down_method-forecast_proportions  \
level                              metric
Overall                            mse-scaled
0.90471
Country                            mse-scaled
0.964979
Country/Purpose                    mse-scaled
0.841679
Country/Purpose/State              mse-scaled
0.903647
Country/Purpose/State/CityNonCity mse-scaled
0.881859
```

```
                                                  Naive/MiddleOut_middle_level-
Country/Purpose/State_top_down_method-forecast_proportions
level                              metric
Overall                            mse-scaled
1.0
Country                            mse-scaled
1.0
Country/Purpose                    mse-scaled
1.0
Country/Purpose/State              mse-scaled
1.0
Country/Purpose/State/CityNonCity mse-scaled
1.0
```

2 Ejemplo Población Penitenciaria Australiana (Grupos, no jerarquico)

En este cuaderno, explicaremos cómo producir pronósticos coherentes para la población penitenciaria australiana en diferentes grupos, replicando los resultados del libro Forecasting: Principles and Practice.

[ ]:
```
import numpy as np
import pandas as pd

# compute base forecast no coherent
from statsforecast.models import ETS
from statsforecast.core import StatsForecast

#obtain hierarchical reconciliation methods and evaluation
from hierarchicalforecast.utils import aggregate
from hierarchicalforecast.methods import BottomUp, MinTrace
from hierarchicalforecast.core import HierarchicalReconciliation
from hierarchicalforecast.evaluation import HierarchicalEvaluation
```

2.1 Agregación bottom time series

El conjunto de datos solo contiene la serie temporal en el nivel más bajo, por lo que debemos crear la serie temporal para todas las jerarquías.

[ ]:
```
from pandas.tseries.offsets import MonthEnd
Y_df = pd.read_csv('https://OTexts.com/fpp3/extrafiles/prison_population.csv')
Y_df = Y_df.rename({'Count': 'y', 'Date': 'ds'}, axis=1)
Y_df.insert(0, 'Country', 'Australia')
Y_df = Y_df[['Country', 'State', 'Gender', 'Legal', 'Indigenous', 'ds', 'y']]
Y_df['ds'] = pd.to_datetime(Y_df['ds'])+ MonthEnd(1)
Y_df.head()
```

```
[ ]:      Country State  Gender      Legal Indigenous         ds  y
     0  Australia   ACT  Female   Remanded       ATSI 2005-03-31  0
     1  Australia   ACT  Female   Remanded   Non-ATSI 2005-03-31  2
     2  Australia   ACT  Female  Sentenced       ATSI 2005-03-31  0
     3  Australia   ACT  Female  Sentenced   Non-ATSI 2005-03-31  5
     4  Australia   ACT    Male   Remanded       ATSI 2005-03-31  7
```

El conjunto de datos se puede agrupar en la siguiente estructura agrupada.

```
[ ]: hiers = [
         ['Country'],
         ['Country', 'State'],
         ['Country', 'Gender'],
         ['Country', 'Legal'],
         ['Country', 'State', 'Gender', 'Legal']
     ]
```

```
[ ]: hiers
```

```
[ ]: [['Country'],
      ['Country', 'State'],
      ['Country', 'Gender'],
      ['Country', 'Legal'],
      ['Country', 'State', 'Gender', 'Legal']]
```

Usando la función `aggregate` de `HierarchicalForecast` podemos obtener el conjunto completo de series de tiempo.

```
[ ]: Y_df, S_df, tags = aggregate(Y_df, hiers, lambda x: np.sum(x) / 1e3)
     Y_df = Y_df.reset_index()
```

```
[ ]: Y_df.head()
```

```
[ ]:    unique_id         ds      y
     0  Australia 2005-03-31  24296
     1  Australia 2005-06-30  24643
     2  Australia 2005-09-30  24511
     3  Australia 2005-12-31  24393
     4  Australia 2006-03-31  24524
```

```
[ ]: Y_df.tail()
```

```
[ ]:                         unique_id         ds     y
     2155   Australia/WA/Male/Sentenced 2015-12-31  3894
     2156   Australia/WA/Male/Sentenced 2016-03-31  3876
     2157   Australia/WA/Male/Sentenced 2016-06-30  3969
     2158   Australia/WA/Male/Sentenced 2016-09-30  4076
     2159   Australia/WA/Male/Sentenced 2016-12-31  4088
```

[ ]:
```
S_df.iloc[:5, :5]
```

[ ]:
```
                 Australia/ACT/Female/Remanded  Australia/ACT/Female/Sentenced  \
Australia                                  1.0                             1.0
Australia/ACT                              1.0                             1.0
Australia/NSW                              0.0                             0.0
Australia/NT                               0.0                             0.0
Australia/QLD                              0.0                             0.0

               Australia/ACT/Male/Remanded  Australia/ACT/Male/Sentenced  \
Australia                              1.0                           1.0
Australia/ACT                          1.0                           1.0
Australia/NSW                          0.0                           0.0
Australia/NT                           0.0                           0.0
Australia/QLD                          0.0                           0.0

               Australia/NSW/Female/Remanded
Australia                                1.0
Australia/ACT                            0.0
Australia/NSW                            1.0
Australia/NT                             0.0
Australia/QLD                            0.0
```

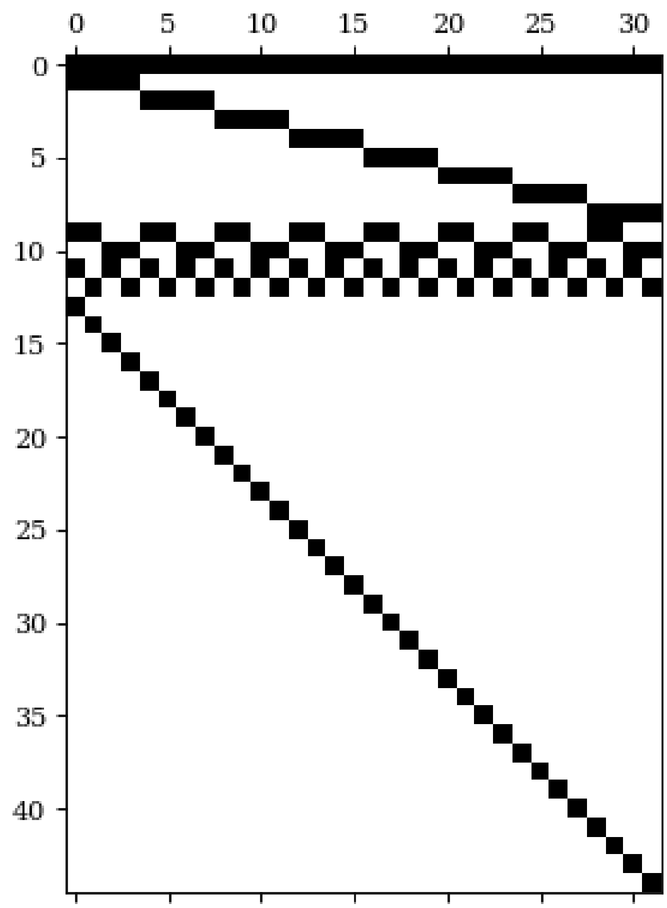
[ ]:
```
tags
```

[ ]:
```
{'Country': array(['Australia'], dtype=object),
 'Country/State': array(['Australia/ACT', 'Australia/NSW', 'Australia/NT',
'Australia/QLD',
        'Australia/SA', 'Australia/TAS', 'Australia/VIC', 'Australia/WA'],
       dtype=object),
 'Country/Gender': array(['Australia/Female', 'Australia/Male'], dtype=object),
 'Country/Legal': array(['Australia/Remanded', 'Australia/Sentenced'],
dtype=object),
 'Country/State/Gender/Legal': array(['Australia/ACT/Female/Remanded',
'Australia/ACT/Female/Sentenced',
        'Australia/ACT/Male/Remanded', 'Australia/ACT/Male/Sentenced',
        'Australia/NSW/Female/Remanded', 'Australia/NSW/Female/Sentenced',
        'Australia/NSW/Male/Remanded', 'Australia/NSW/Male/Sentenced',
        'Australia/NT/Female/Remanded', 'Australia/NT/Female/Sentenced',
        'Australia/NT/Male/Remanded', 'Australia/NT/Male/Sentenced',
        'Australia/QLD/Female/Remanded', 'Australia/QLD/Female/Sentenced',
        'Australia/QLD/Male/Remanded', 'Australia/QLD/Male/Sentenced',
        'Australia/SA/Female/Remanded', 'Australia/SA/Female/Sentenced',
        'Australia/SA/Male/Remanded', 'Australia/SA/Male/Sentenced',
        'Australia/TAS/Female/Remanded', 'Australia/TAS/Female/Sentenced',
        'Australia/TAS/Male/Remanded', 'Australia/TAS/Male/Sentenced',
        'Australia/VIC/Female/Remanded', 'Australia/VIC/Female/Sentenced',
```

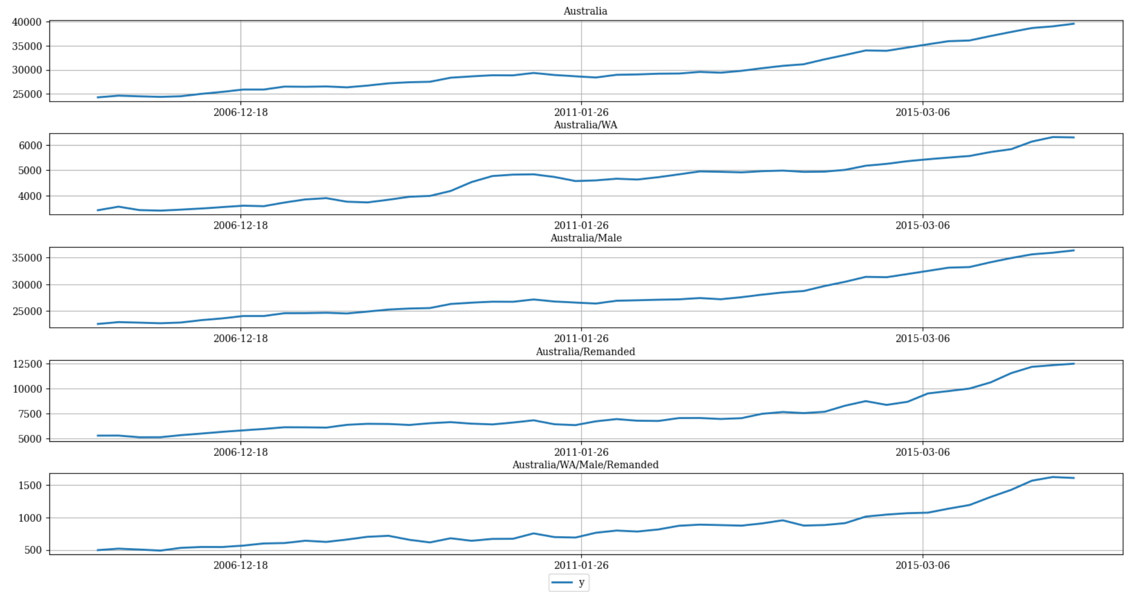
```
 'Australia/VIC/Male/Remanded', 'Australia/VIC/Male/Sentenced',
 'Australia/WA/Female/Remanded', 'Australia/WA/Female/Sentenced',
 'Australia/WA/Male/Remanded', 'Australia/WA/Male/Sentenced'],
dtype=object)}
```

[ ]:
```python
from hierarchicalforecast.utils import HierarchicalPlot
hplot = HierarchicalPlot(S=S_df, tags=tags)

hplot.plot_summing_matrix()
```



[ ]:
```python
hplot.plot_hierarchically_linked_series(
    bottom_series='Australia/WA/Male/Remanded',
    Y_df=Y_df.set_index('unique_id')
)
```



Conjuntos de entrenamiento/prueba

Usamos los últimos dos años (8 trimestres) como conjunto de prueba.

```
[ ]: Y_test_df = Y_df.groupby('unique_id').tail(8)
     Y_train_df = Y_df.drop(Y_test_df.index)
     Y_test_df = Y_test_df.set_index('unique_id')
     Y_train_df = Y_train_df.set_index('unique_id')
```

2.2 Cálculo de predicciones básicas

La siguiente celda calcula las **predicciones base** para cada serie temporal en `Y_df` utilizando los modelos `auto_ETS` y `naive`. Observe que `Y_hat_df` contiene los pronósticos pero no son coherentes.

```
[ ]: #https://nixtla.github.io/statsforecast/models.html#ets

     fcst = StatsForecast(df=Y_train_df,
                          models=[ETS(season_length=4, model='ZMZ')],
                          freq='Q', n_jobs=-1)
     Y_hat_df = fcst.forecast(h=8, fitted=True)
     Y_fitted_df = fcst.forecast_fitted_values()
```

2.3 Conciliar predicciones

La siguiente celda hace que las predicciones anteriores sean coherentes usando la clase `HierarchicalReconciliation`. Dado que la estructura de la jerarquía no es estricta, no podemos usar métodos como `TopDown` o `MiddleOut`. En este ejemplo usamos `BottomUp` y `MinTrace`.

[ ]:
```
reconcilers = [
    BottomUp(),
    MinTrace(method='mint_shrink')
]
hrec = HierarchicalReconciliation(reconcilers=reconcilers)
Y_rec_df = hrec.reconcile(Y_hat_df=Y_hat_df, Y_df=Y_fitted_df, S=S_df,␣
↪tags=tags)
```

`Y_rec_df` contiene las predicciones

[ ]:
```
Y_rec_df.head()
```

[ ]:
```
                   ds           ETS  ETS/BottomUp  \
unique_id
Australia 2015-03-31  34799.496094  34946.523438
Australia 2015-06-30  35192.636719  35410.093750
Australia 2015-09-30  35188.214844  35580.218750
Australia 2015-12-31  35888.628906  35951.203125
Australia 2016-03-31  36045.437500  36415.914062

           ETS/MinTrace_method-mint_shrink
unique_id
Australia                     34925.189672
Australia                     35434.882290
Australia                     35472.806757
Australia                     35939.136613
Australia                     36244.765912
```

2.4 Evaluación

El paquete `HierarchicalForecast` incluye la clase `HierarchicalE Evaluation` para evaluar las diferentes jerarquías y también es capaz de calcular métricas escaladas en comparación con un modelo de referencia.

[ ]:
```
def mase(y, y_hat, y_insample, seasonality=4):
    errors = np.mean(np.abs(y - y_hat), axis=1)
    scale = np.mean(np.abs(y_insample[:, seasonality:] - y_insample[:, :
↪-seasonality]), axis=1)
    return np.mean(errors / scale)

eval_tags = {}
eval_tags['Total'] = tags['Country']
eval_tags['State'] = tags['Country/State']
eval_tags['Legal status'] = tags['Country/Legal']
eval_tags['Gender'] = tags['Country/Gender']
eval_tags['Bottom'] = tags['Country/State/Gender/Legal']
eval_tags['All series'] = np.concatenate(list(tags.values()))
```

```
evaluator = HierarchicalEvaluation(evaluators=[mase])
evaluation = evaluator.evaluate(
    Y_hat_df=Y_rec_df, Y_test_df=Y_test_df,
    tags=eval_tags,
    Y_df=Y_train_df
)
evaluation = evaluation.reset_index().drop(columns='metric').drop(0).
 ↪set_index('level')
evaluation.columns = ['Base', 'BottomUp', 'MinTrace(mint_shrink)']
evaluation.applymap('{:.2f}'.format)
```

[ ]:

```
              Base BottomUp MinTrace(mint_shrink)
level
Total         1.36     1.07                  1.17
State         1.53     1.55                  1.59
Legal status  2.40     2.48                  2.38
Gender        1.09     0.82                  0.93
Bottom        2.16     2.16                  2.14
All series    1.99     1.98                  1.98
```

2.5 Plot Forecast

[ ]:

```
plot_df = pd.concat([Y_df.set_index(['unique_id', 'ds']),
                     Y_rec_df.set_index('ds', append=True)], axis=1)
plot_df = plot_df.reset_index('ds')
```

[ ]:

```
plot_df
```

[ ]:

```
                                   ds      y          ETS  ETS/BottomUp  \
unique_id
Australia                  2005-03-31  24296          NaN           NaN
Australia                  2005-06-30  24643          NaN           NaN
Australia                  2005-09-30  24511          NaN           NaN
Australia                  2005-12-31  24393          NaN           NaN
Australia                  2006-03-31  24524          NaN           NaN
…                                   …      …            …             …
Australia/WA/Male/Sentenced 2015-12-31   3894  3927.837646   3927.837646
Australia/WA/Male/Sentenced 2016-03-31   3876  3965.692139   3965.692139
Australia/WA/Male/Sentenced 2016-06-30   3969  4003.911621   4003.911621
Australia/WA/Male/Sentenced 2016-09-30   4076  4042.499512   4042.499512
Australia/WA/Male/Sentenced 2016-12-31   4088  4081.459229   4081.459229

                             ETS/MinTrace_method-mint_shrink
unique_id
Australia                                                NaN
Australia                                                NaN
Australia                                                NaN
```

```
Australia                                             NaN
Australia                                             NaN
…                                                       …
Australia/WA/Male/Sentenced                   3908.504409
Australia/WA/Male/Sentenced                   3902.781167
Australia/WA/Male/Sentenced                   3924.570718
Australia/WA/Male/Sentenced                   3949.118650
Australia/WA/Male/Sentenced                   4027.718115

[2160 rows x 5 columns]
```

[ ]:
```python
hplot.plot_series(
    series='Australia/WA/Male/Sentenced',
    Y_df=plot_df,
    models=['y', 'ETS', 'ETS/MinTrace_method-mint_shrink']
)
```