benchmark

Release 0.3.0

Carlos De Oliveira
Nikhil Garge
Christopher Policastro

Nov 08, 2021

CONTENTS

1 Survey **1**
- 1.1 Abstract . . . 1
- 1.2 Background . . . 1
- 1.3 Contributions . . . 2
- 1.4 Algorithms . . . 3
- 1.5 Experiments . . . 6

2 Documentation **12**
- 2.1 benchmark.disaggregation . . . 12

Package Index **14**

Index **15**

CHAPTER

ONE

SURVEY

1.1 Abstract

We develop a method to transform low-frequency data to high-frequency data. Our extension of the Denton algorithm incorporates constraints to reflect subject matter expertise. We propose a relaxation of the quadratically constrained quadratic program to allow for a usable and scalable implementation. Experiments show the utility of the algorithm for up-sampling biannual data to annual data.

1.2 Background

We want to transform low-frequency data to high-frequency data. We observe a low-frequency dataset. We need to predict a high-frequency dataset. Before we can impute the missing values of the low-frequency dataset $(X_i)_{i=0}^{t-1}$, we need to observe a related high-frequency dataset

$$(Y_i)_{i=0}^{T-1} \tag{Indicator}$$

The indicator is k times more frequent than the benchmark. Since $t = \frac{T}{k}$, we can relabel the benchmark as

$$(X_{ik})_{i=0}^{\frac{T}{k}-1} \tag{Benchmark}$$

A subject matter expert must determine the indicator. We want to determine a high-frequency dataset

$$(Z_i)_{i=0}^{T-1} \tag{Interpolant}$$

such that Z_j for $j \bmod k \neq 0$ infers the missing value X_j. Note that the subject matter expert does not prescribe values of the interpolant.

Following the algorithm of F.T. Denton [1971] and P.A. Cholette [1984], we solve an optimization problem involving the benchmark and the indicator to determine the interpolant.

$$\begin{cases} \min\limits_{(Z_i)_{i=1}^{T-1}} \sum\limits_{i=1}^{T-1} \left(\dfrac{Z_t}{Y_t} - \dfrac{Z_{t-1}}{Y_{t-1}} \right)^2 & \text{(QP)} \\ X_{ik} = \sum\limits_{j=0}^{k-1} Z_{ik+j} & \text{for } i = 0, \ldots, t-1 \end{cases}$$

We discuss variants of the objective in Section 1.4.1 an variants of the constraints in Section 1.4.3.

Without the constraints, the interpolant would equal the indicator. However, the constraints require the interpolant to equal the low-frequency dataset under aggregation over a period. We are studying a disaggregation problem because we try to determine unobserved values that aggregate to observed values.

1.3 Contributions

We propose several contributions to the disaggregation problem.

1.3.1 Extensions

We suggest two extensions to the Denton algorithm. We modify the disaggregation problem through a modification of the linear constraint and the addition of a quadratic constraint.

1. Linear Constraint
    - Incorporate hyperparameters into the linear constraint
    - Propose a metric for hyperparameter tuning
2. Quadratic Constraint
    - Add a quadratic constraint to restrict the interpolant to estimates of subject matter experts
    - Relax the quadratically constrained quadratic program

The quadratic constraint will contain values of the interpolant specified by the subject matter expert. We refer to the values as annotations.

The quadratic constraint requires us to solve a quadratically constrained quadratic program. We replace the constraint with a penalization of the objective. The relaxation helps us provide a maintainable and scalable implementation of the algorithm.

1.3.2 Connections

By extending the Denton algorithm, we can capture solutions to problems outside the disaggregation problem. Through combinations of objectives and constraints, we can connect the disaggregation problem to seemingly unrelated problems like filtering.

See Section 1.5.2 for a combination of objective and constraints yielding the Hodrick-Precott algorithm. Note that the Hodrick-Prescott algorithm solves the filtering problem through a decomposition of the data into a trend component and cyclical component.

1.3.3 Usage

Some statistical software has support for the disaggregation problem. Within the R programming language, the `tempdisagg` library of C. Sax and P. Steiner [2013] implements several algorithms to solve the disaggregation problem including the Denton algorithm.

The Python programming language lacks support in standard libraries for the disaggregation problem. The `statsmodels` library of Seabold and Perktold [2010] does implement the Denton algorithm. However, the objective and constraints are hard-coded into the solver. We refactor the code in the `statsmodels` library to provide a flexible and usable implementation. We document the implementation in Chapter 2.

While the `benchmark` library implements a default solver, it has support for user-defined solvers involving different objective functions. The user-defined solvers can be exact solvers or approximate solvers. The default solver determines the exact solution to default objective which matches the objective from the `statsmodels` library. We focus on exact solvers for the default because we want to avoid the need for large datasets. Small datasets are more affordable, accessible and accurate than large datasets. Unlike an approximate solver, the exact solver does not require many batches of data for iterative search.

We implement a metric to compare datasets with different sizes. See Section 1.4.2. The metric allows us to compare the interpolant to the benchmark data. Based on low values and high values of the metric, we can tune hyperparameters in the algorithm. Since the `benchmark` library allows for user-defined linear constraints, we can programmatically determine the optimal linear constraints under the metric.

Since we use hyperparameter tuning to determine components of the algorithm like linear constraints, we want to leverage standard libraries such as `scikit-learn`. Following the developments of Pedregosa et al. [2011], researchers and practitioners have adopted the `scikit-learn` library to tasks like hyperparameter tuning. The `benchmark` library adheres to the `scikit-learn` application programming interface. So we can use tools like `GridSearchCV` or `RandomSearchCV` to tune hyperparameters.

1.4 Algorithms

We study the algorithm of F.T. Denton [1971]. However, other researchers have suggested solutions to the disaggregation problem. Following the treatment of C. Sax and P. Steiner [2013], we can summarize these algorithms through a heuristic.

$$\text{Interpolant} = \text{Indicator} + \text{Disaggregation}\left(\text{Benchmark} - \text{Aggregation}\left(\text{Indicator}\right)\right)$$

The indicator inhabits a high dimensional space. The benchmark inhabits a low dimensional space. We can map the indicator from the high dimension space to the low dimensional space through an aggregation operation like summation. Within the low dimensional space, we can determine the difference between the benchmark and the aggregated indicator.

We want to map the difference from the low dimensional space to the high dimensional space. We need to determine a disaggregation operation. The disaggregation operation should be the opposite of the aggregation operation. For example, if the aggregation operation is summation

$$(y_i)_{i=0}^{k-1} \mapsto \sum_{i=0}^{k-1} y_i$$

then the disaggregation operation could be

$$x \mapsto \left(\frac{x}{k}\right)_{i=0}^{k-1}$$

Note that the composition of disaggregation and aggregation is the identity. However the composition of aggregation and disaggregation is the average.

The algorithms of G. C. Chow and A.-L. Lin [1971], R. B. Litterman [1983] and R. B. Fernández [1981] propose different disaggregation operations. However, we will take another approach to the determination of the disaggregation operation. We focus on a constrained optimization problem yielding the interpolant through minimization of an objective.

1.4.1 Objective

We can replace the objective in (QP) with other objectives. Practitioners use different objectives to handle different datasets. Following A.M. Bloem, R.J. Dippelsman and N.O. Maehle [2001], we can identity some common objectives.

O1. $\displaystyle\sum_{i=1}^{T-1} \left(\frac{Z_i}{Y_i} - \frac{Z_{i-1}}{Y_{i-1}}\right)^2$

O2. $\displaystyle\sum_{i=1}^{T-1} \left(\log\left(\frac{Z_i}{Z_{i-1}}\right) - \log\left(\frac{Y_i}{Y_{i-1}}\right)\right)^2$

O3. $\sum_{i=1}^{T-1} \left(\frac{Z_i}{Z_{i-1}} / \frac{Y_i}{Y_{i-1}} \right)^2$

O4. $\sum_{i=1}^{T-1} \left(\frac{Z_i}{Y_i} / \frac{Z_{i-1}}{Y_{i-1}} - 1 \right)^2$

O5. $\sum_{i=1}^{T-1} \left((Z_i - Z_{i-1}) - (Y_i - Y_{i-1}) \right)^2$

While the `benchmark` library allows for any objective through a user-defined solver, we will focus on (O1) and (O5). We can categorize (O1) and (O5) among a general collection of objectives.

O1-h. $\left| (1-L)^h \left(\frac{Z_i}{Y_i} \right)_{i=0}^{T-1} \right|^2$ for $0 < h$

O5-h. $\left| (1-L)^h \left(Z_i - Y_i \right)_{i=0}^{T-1} \right|^2$ for $0 \leq h$

Here L denotes the lag operator which shifts indices of the data.

1.4.2 Metric

A subject matter expert must determine the indicator. Note that we cannot calculate the correlation between the benchmark and the indicator because the datasets have different sizes. Having a metric to compare the indicator and benchmark, would the help the determination of indictors.

Having solved the disaggregation problem, we cannot calculate the correlation between the benchmark and the indicator because the datasets have different sizes. We want to assess different hyperparameters through a comparison of the interpolant and benchmark. Having a metric to compare the interpolant and benchmark, would help hyperparameter tuning.

We propose canonical correlation analysis to provide a metric for comparison. We can use assess possible indicators.

$$\left((Y_{ik+j})_{j=0}^{k-1} \right)_{i=0}^{t-1} \qquad \text{(Reshaped Indicator)}$$

We can relabel the indicator. See Figure 1 for a diagram. Take the benchmark to be t samples of a random variable X with one component. Take the reshaped indicator to be t samples of a random variable $Y = (Y^{(i)})_{i=0}^{k}$ with k components. Form the sample cross-covariance matrix Σ_{XY} for the random variables X and Y. Note that the sample cross-covariance matrix estimates the cross-covariance matrix

$$\Sigma_{XY} = \begin{bmatrix} \mathbb{E}\left[(X - \mathbb{E}[X]) \cdot \left(Y^{(0)} - \mathbb{E}\left[Y^{(0)} \right] \right) \right] \\ \vdots \\ \mathbb{E}\left[(X - \mathbb{E}[X]) \cdot \left(Y^{(k-1)} - \mathbb{E}\left[Y^{(k-1)} \right] \right) \right] \end{bmatrix}^T$$

Form the sample covariance matrix Σ_{XX} for the random variables X and the sample covariance matrix Σ_{YY} for Y. Maximize over vectors u with one component and vectors $v = \left(v^{(i)} \right)_{i=0}^{k-1}$ with k components the ratio

$$\max_{u,\, v} \frac{u^T \, \Sigma_{XY} \, v}{\sqrt{u^T \, \Sigma_{XX} \, u} \sqrt{v^T \, \Sigma_{YY} \, v}}$$

We want to determine vectors u and v that maximize the correlation between $u^T \cdot X$ and $v^T \cdot Y$. In particular, we want to determine a projection

$$\sum_{i=0}^{k-1} v_i \cdot Y^{(i)} \qquad \text{(CCA Projection)}$$

of $(Y^{(i)})_{i=0}^{k}$ from k dimensions to 1 dimension that maximizes the correlation with X. If the projection of the reshaped indicator has a correlation close to 1 with the benchmark, then the subject matter expert has determined a comparable indicator.

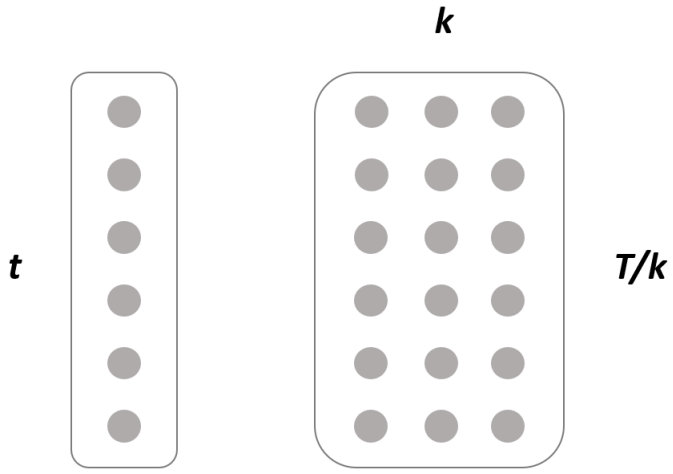

Fig. 1: Diagram of Benchmark and Reshaped Indicator

1.4.3 Linear Constraint

The linear constraint connects the interpolant to the benchmark through the aggregation operation. Note that X_{ik} represents the aggregate of $(Z_{ik+j})_{j=0}^{k-1}$. The general form of the linear constraint involves k hyperparameters $\mu_j \geq 0$.

$$X_{ik} = \sum_{j=0}^{k-1} \mu_j \cdot Z_{ik+j}$$

Here $i = 0, \ldots, t-1$. Following A.M. Bloem, R.J. Dippelsman and N.O. Maehle [2001], we can identity some common operations.

- $\sum_{j=0}^{k-1} Z_{ik+j}$ (Sum)
- $\frac{1}{k}\sum_{j=0}^{k-1} Z_{ik+j}$ (Average)
- Z_{ik} (First)
- $Z_{(i+1)k-1}$ (Last)

If X_{ik} reflects an aggregate value of $(Z_{ik+j})_{j=0}^{k-1}$, then we could use summation. If X_{ik} reflects a representative value of $(Z_{ik+j})_{j=0}^{k-1}$, then we could use average. If we want to peg the values of the interpolant to the benchmark at the start of the period, then we could use the first value. If we want to peg the values of the interpolant to the benchmark at the end of the period, then we could use the last value.

$$\sum_{i=0}^{k-1} v_i \cdot Z_{ik+j} \quad \text{(CCA)}$$

If X_{ik} reflects a projected value of Z_{ik+j} for $j = 0, \ldots, k-1$, then we could replace the sum in (QP) with the projections from Section 1.4.2. Note that the values $(v_i)_{i=0}^{k-1}$ are the values from the projection in canonical correlation analysis (CCA Projection).

Depending on the dataset, we might have to process the benchmark to have representative values. For example, we might replace

$$X_{ik} \mapsto \frac{X_{ik} + X_{i(k+1)}}{2}$$

to combine the value at the start of the period and end of the period.

1.4.4 Quadratic Constraint

The subject matter expert might have guesses for some of the missing values of the benchmark.

$$\left(\widehat{Z}_{i_j}\right)_{j=0}^{\widehat{T}-1} \tag{Annotations}$$

Note that the subject matter expect might have $\widehat{T} < T$ guesses. We propose a quadratic constraint to incorporate the annotations.

$$\begin{cases} \min\limits_{(Z_i)_{i=0}^{T-1}} \sum\limits_{i=1}^{T-1} \left(\dfrac{Z_t}{Y_t} - \dfrac{Z_{t-1}}{Y_{t-1}}\right)^2 & \text{(QCQP)} \\ X_{ik} = \sum\limits_{j=0}^{k-1} \mu_j \cdot Z_{ik+j} & \text{for } i = 0, \ldots, t-1 \\ \left(Z_{i_j} - \widehat{Z}_{i_j}\right)^2 \leq \eta_j & \text{for } j = 0, \ldots, \widehat{T}-1 \end{cases}$$

Here we have (O1-1) for the objective. However, we could have any objective in (QCQP). Since we have a quadratically constrained quadratic program, we could use solvers like the solver in S. Burer and R.D.C. Monteiro [2003] for the interpolant. However, we propose a relaxation of the quadratic constraint to provide a tractable optimization problem.

$$\begin{cases} \min\limits_{(Z_i)_{i=0}^{T-1}} \sum\limits_{i=1}^{T-1} \left(\dfrac{Z_t}{Y_t} - \dfrac{Z_{t-1}}{Y_{t-1}}\right)^2 + \sum\limits_{j=0}^{\widehat{T}-1} \epsilon_j \left(Z_{i_j} - \widehat{Z}_{i_j}\right)^2 \\ X_{ik} = \sum\limits_{j=0}^{k-1} \mu_j \cdot Z_{ik+j} & \text{for } i = 0, \ldots, t-1 \end{cases}$$

Note that we have a quadratic program with hyperparameters $\epsilon_0, \ldots, \epsilon_{\widehat{T}-1}$ and $\mu_0, \ldots, \mu_{k-1}$.

1.5 Experiments

We discuss several use-cases related to estimation of Gross Domestic Product (GDP). Please see the User's Guide of the `benchmark` library for the code.

1.5.1 Data

We have data from the World Bank (https://data.worldbank.org/). We collect annual reports of Australian GDP and Brazilian GDP. The data spans the 60 year period between 1960 and 2020. From the Brazilian dataset, we omit every other record. The Australian dataset becomes twice as frequent as the down-sampled Brazilian dataset.

- Indicator: Annual Australian GDP
- Benchmark: Bi-Annual Brazilian GDP

Note that Australia and Brazil are commodity countries. A subject matter expert might select Australia for the indicator among other commodity countries like South Africa. We can calculate correlation between the indicator and benchmark through canonical correlation analysis. We observe a 98.48% correlation between annual Australian GDP and bi-annual Brazilian GDP.

1.5.2 Analysis

We want to transform the bi-annual Brazilian GDP to annual Brazilian GDP.

Case A

- Indicator: Annual Australian GDP
- Benchmark: Bi-Annual Brazilian GDP
- Objective: O1-1
- Linear Constraint: Average
- Quadratic Constraint: None

Taking the indictor to be the annual Australian GDP and the benchmark to be the bi-annual Brazilian GDP, we can use the `benchmark` library to compute the interpolant. We use the O1-1 objective with average for the aggregation operation.

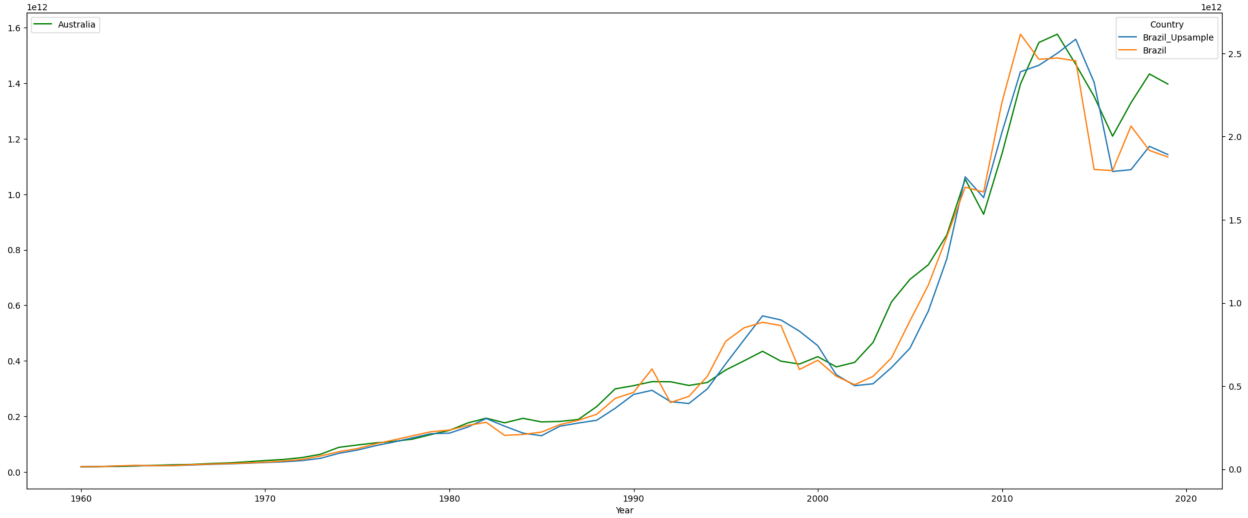

Fig. 2: Case A

Figure 2 shows the interpolant. Note that the benchmark library and the statsmodels library yield the same interpolant. We observe an approximate match between the interpolant and the Brazilian GDP. We calculate 99.09% correlation between the datasets.

Case B

- Indicator: Constant value of 1
- Benchmark: Bi-Annual Brazilian GDP
- Objective: O1-1
- Linear Constraint: Average
- Quadratic Constraint: None

If the indicator takes a constant value, then the indicator does not contribute to the disaggregation problem. With the (O1-1) objective, we attempt to limit the consecutive differences between values of the interpolant. We want to determine an approximately constant interpolant that on average matches the benchmark.

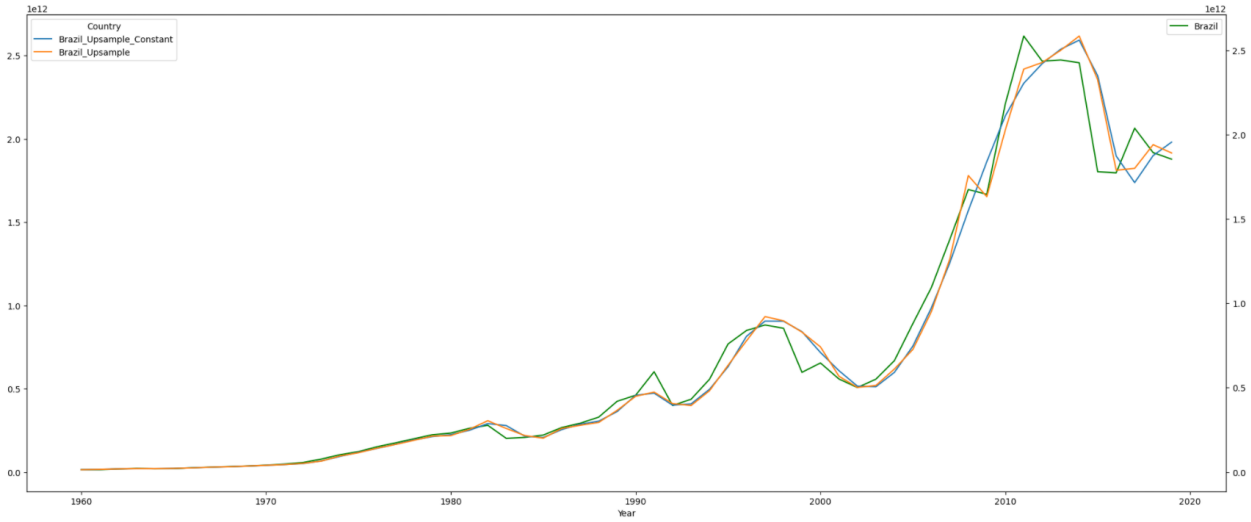

Fig. 3: Case B

Figure 3 shows the interpolant. We changed the frequency of the Brazilian GDP from annual to bi-annual. Since we have a value every other year, we can have the aggregation operation take the first value.

Case C

- Indicator: Australian
- Benchmark: Bi-Annual Brazilian GDP
- Objective: O1-1
- Linear Constraint: First
- Quadratic Constraint: None

Note that the interpolant for Case C more closely matches the Brazilian GDP than Case A.

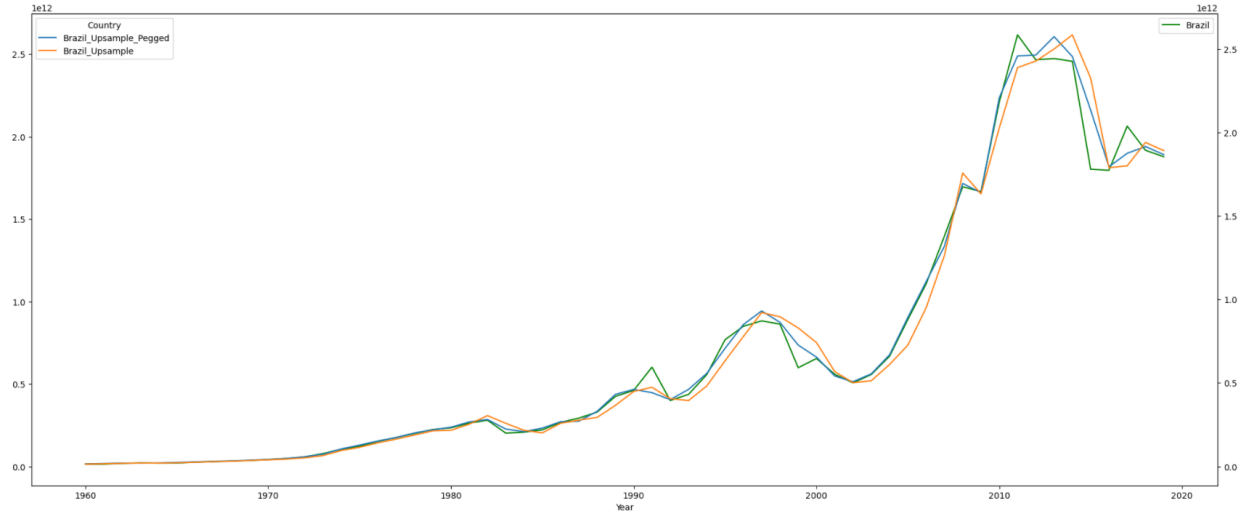

Fig. 4: Case C

Figure 4 shows the interpolant. We have pegged the values of the interpolant to the observed data.

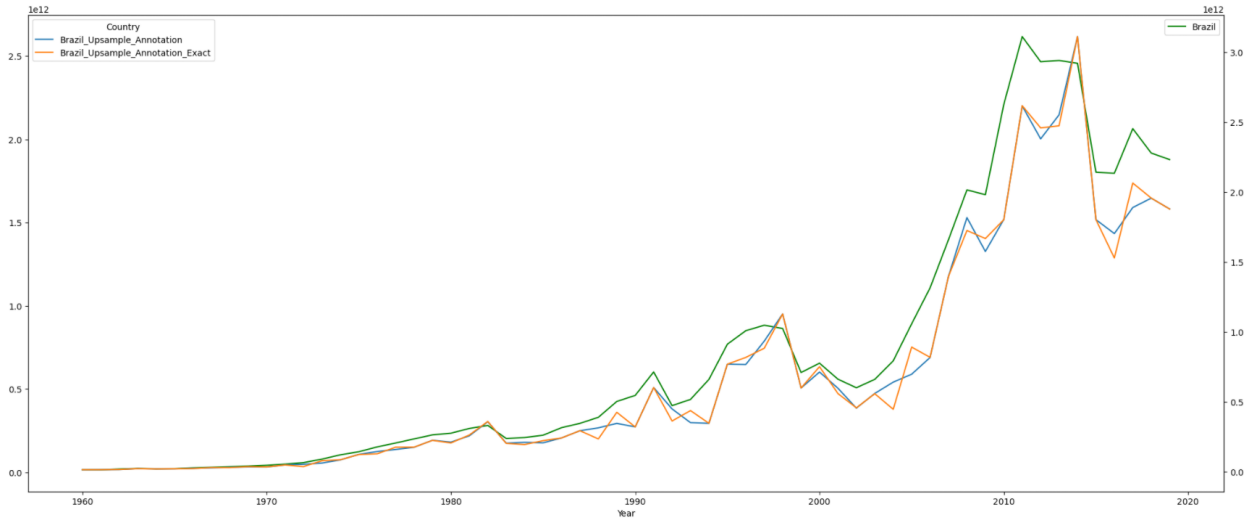

Fig. 5: Case D

Case D

- Indicator: Australian
- Benchmark: Bi-Annual Brazilian GDP
- Objective: O1-1
- Linear Constraint: Average
- Quadratic Constraint: Tetra-Annual Brazilian GDP with noise

If we want to include a quadratic constraint, then we need to specify guesses for the missing Brazilian GDP. We add a random value to the observed Brazilian GDP.

Figure 5 shows the interpolant. Since we have noisy values in the quadratic constraint, the interpolant for Case D matches the Brazilian GDP worse than the interpolant for Case A. However, we can repeat the experiment without adding a random value to the observed Brazilian GDP. In this case, the interpolant for Case D matches the Brazilian GDP better than the interpolant for Case A.

Case E

- Indicator: Australian
- Benchmark: Bi-Annual Brazilian GDP
- Objective: O5-1
- Linear Constraint: Average
- Quadratic Constraint: None

We can compare (O1-1) and (O5-1). If we switch from proportions to differences, then we obtain a different interpolant. Figure 6 shows the interpolant. Note that the interpolant for Case E matches the Brazilian GDP worse than the interpolant for Case A.

Case F

- Indicator: Constant value of 1
- Benchmark: Bi-Annual Brazilian GDP
- Objective: O5-2

- Linear Constraint: Constant value of 0
- Quadratic Constraint: Australian GDP

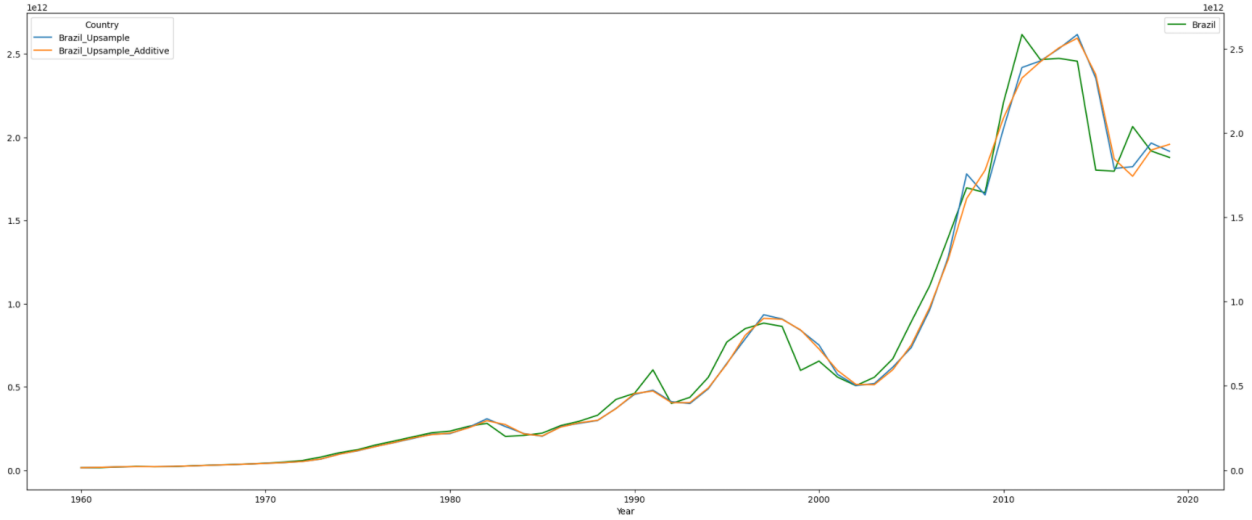

Fig. 6: Case E

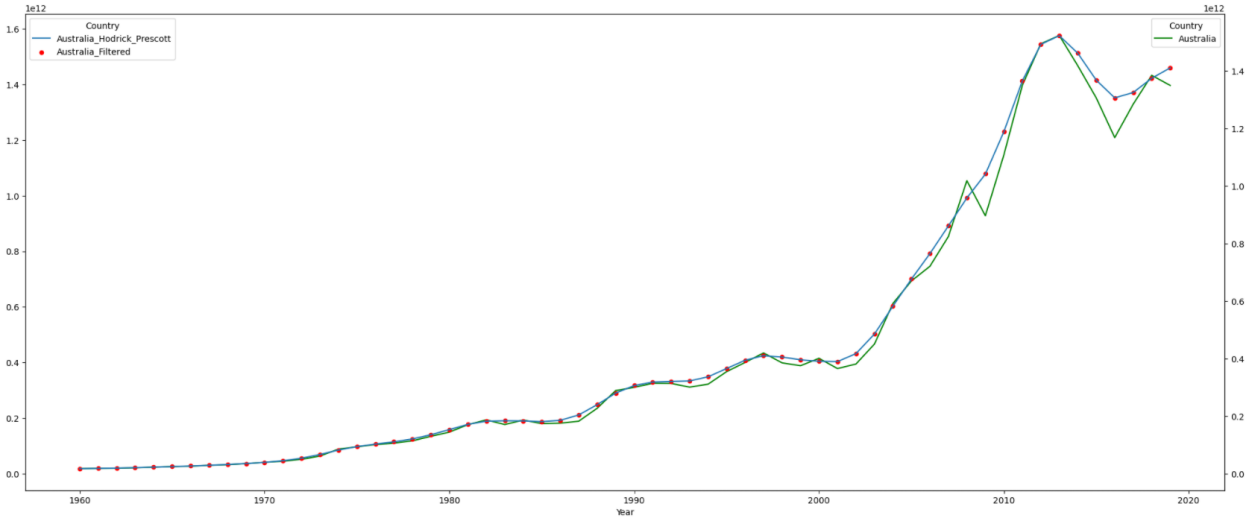

Fig. 7: Case F

Suppose we were interest in the filtering problem for the Australian GDP rather than the disaggregation problem for the Brazilian GDP.

$$\min_{(Z_i)_{i=0}^{T-1}} \sum_{i=1}^{T-2} (Z_{t+1} - 2Z_t + Z_{t-1})^2 + \epsilon \sum_{j=0}^{T} (Z_j - Y_j)^2$$

The Hodrick-Prescott algorithm filters the Australian GDP to decompose the data into a trend component and cyclical component.

Figure 7 shows the interpolant. Note that the interpolant matches the Australian GDP but the interpolant has less curvature. The expression $Z_{t+1} - 2Z_t + Z_{t-1}$ is a finite difference approximation to the second derivative. So the Hodrick-Prescott filter penalizes curvature in the data.

BIBLIOGRAPHY

A.M. Bloem, R.J. Dippelsman and N.O. Maehle. Quarterly national accounts manual–concepts, data sources, and compilation. 2001. URL http://www.imf.org/external/pubs/ft/qna/2000/Textbook/index.htm.

C. Sax and P. Steiner. Temporal disaggregation of time series. *The R Journal*, 5/2:80–87, 2013.

F.T. Denton. Adjustment of monthly or quarterly series to annual totals: An approach based on quadratic minimization. *Journal of the American Statistical Association*, 66.333:99–102, 1971.

G. C. Chow and A.-L. Lin. Best linear unbiased interpolation, distribution, and extrapolation of time series by related series. *The Review of Economics and Statistics*, 53 (4):80–83, 1971.

P.A. Cholette. Adjusting sub-annual series to yearly benchmarks. *Survey Methodology*, 10.1:35–49, 1984.

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.

R. B. Fernández. A methodological note on the estimation of time series. *The Review of Economics and Statistics*, 63 (3):471–476, 1981.

R. B. Litterman. A random walk, markov model for the distribution of time series. *Journal of Business Economic Statistics*, 1 (2):169–173, 1983.

S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via lowrank factorization. *Mathematical Programming*, 95.2:329–357, 2003.

Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

CHAPTER

TWO

DOCUMENTATION

2.1 benchmark.disaggregation

class benchmark.disaggregation.**Estimator**(*linear_constraint: Optional[Union[int, float, complex, str, bytes, numpy.generic, Sequence[Union[int, float, complex, str, bytes, numpy.generic]], Sequence[Sequence[Any]], numpy.typing._array_like._SupportsArray]] = None*, *quadratic_constraint_weight: Optional[Union[float, int]] = None*, *solver: Optional[Callable] = None*)

Bases: sklearn.base.BaseEstimator, sklearn.base.TransformerMixin

A scikit-learn compatible estimator for benchmarking.

See Also: statsmodels.tsa.interp.denton

linear_constraint: Optional[ArrayLike] = None linear constraint in optimization problem. default is average

quadratic_constraint_weight: Optional[Union[float,int]] = None weight of quadratic constraint in relaxation of optimization problem default is 0

solver: Optional[Callable] = None solver for custom objective in optimization default is consecutive differences of ratio between up-sampled data and high-frequency data

fit: Callable determine optimization problem with high frequency data and low frequency data along with annotations

transform: Callable projection of high frequency data through Canonical Correlation Analysis with low frequency data

predict: Callable solution of optimization problem

score: Callable correlation between low frequency data and up-sampled data in Canonical Correlation Analysis

linear_constraint: ArrayLike linear constraint in optimization problem

quadratic_constraint_weight: Union[float,int] weight of quadratic constraint in relaxation of optimization problem

solver: Callable solver for custom objective in optimiation

high_frequency_data: ArrayLike high frequency dataset

low_frequency_data: ArrayLike low frequency dataset

annotations: ArrayLike predictions of subject matter expert

interpolant: ArrayLike up-sampled dataset from solution to optimization problem

projection: ArrayLike projects from *transform* method

frequency: int Length of high_frequency_data divided by length of low_frequency_data

```
>>> high_frequency_data = np.array([50,100,150,100] * 5)
>>> low_frequency_data = np.array([500,400,300,400,500])
>>> estimator = benchmark.disaggregation.Estimator()
>>> estimator.fit(high_frequency_data, low_frequency_data)
>>> estimator.predict()
array([257.3392, 511.2246, 751.2952, 480.141 , 226.2556, 423.9027,590.0058, 359.
↪8359, 162.1888, 297.7839, 433.3789, 306.6484,171.0534, 376.5866, 613.6638, 438.
↪6962, 233.163 , 490.5022, 761.6564, 514.6784])
```

fit(*high_frequency_data: Union[int, float, complex, str, bytes, numpy.generic, Sequence[Union[int, float, complex, str, bytes, numpy.generic]], Sequence[Sequence[Any]], numpy.typing._array_like._SupportsArray]*, *low_frequency_data: Union[int, float, complex, str, bytes, numpy.generic, Sequence[Union[int, float, complex, str, bytes, numpy.generic]], Sequence[Sequence[Any]], numpy.typing._array_like._SupportsArray]*, *annotation_data: Optional[Union[int, float, complex, str, bytes, numpy.generic, Sequence[Union[int, float, complex, str, bytes, numpy.generic]], Sequence[Sequence[Any]], numpy.typing._array_like._SupportsArray]] = None*)

Fit estimator to high frequency data and low frequency data along with annotations

high_frequency_data: ArrayLike high frequency dataset

low_frequency_data: ArrayLike low frequency dataset

annotation_data: Optional[ArrayLike] = None predictions of subject matter expert

```
>>> high_frequency_data = np.array([50,100,150,100] * 5)
>>> low_frequency_data = np.array([500,400,300,400,500])
>>> estimator = benchmark.disaggregation.Estimator()
>>> estimator.fit(high_frequency_data, low_frequency_data)
```

predict(*X=None*)

return up-sampled data from solution to optimization problem

score(*X=None*, *y=None*)

Calculate correlation between up-sampled data and low frequency data through Canonical Correlation Analysis

See Also: sklearn.cross_decomposition.CCA

transform(*X=None*)

projection high frequency data to low frequency data through Canonical Correlation Analysis

See Also: sklearn.cross_decomposition.CCA

```
>>> high_frequency_data = np.array([50,100,150,100] * 5)
>>> low_frequency_data = np.array([500,400,300,400,500])
>>> estimator = benchmark.disaggregation.Estimator()
>>> estimator.fit(high_frequency_data, low_frequency_data)
>>> estimator.transform()
```

PACKAGE INDEX

b

benchmark.disaggregation, 12

INDEX

B

benchmark.disaggregation
    module, 12

E

Estimator (*class in benchmark.disaggregation*), 12

F

fit() (*benchmark.disaggregation.Estimator method*), 13

M

module
    benchmark.disaggregation, 12

P

predict() (*benchmark.disaggregation.Estimator method*), 13

S

score() (*benchmark.disaggregation.Estimator method*), 13

T

transform() (*benchmark.disaggregation.Estimator method*), 13