

# Cobwood: Enhancing Forest Economics Model Reusability Through Labeled Panel Data Structures

2025

## Summary

Managing forest ecosystems effectively requires long-term foresight into global wood markets. This planning relies on macroeconomic panel data spanning multiple countries over extended time periods. The cobwood package introduces a data structure to manipulate panel data for forecasting and scenario analysis. By implementing country and time indexes, the package improves model readability, as the source code closely mirrors mathematical equations used in research publications. To demonstrate cobwood's practical application, we present a reimplementation of the Global Forest Products Model (GFPMx). Our data structure leverages the Xarray package, which provides labeled N-dimensional arrays and robust capabilities including NetCDF file storage for model outputs. This approach offers two key advantages: enhanced source code clarity that facilitates model inspection, and comprehensive metadata for country, product, and time coordinates along with units metadata in the dataset attributes. These features position cobwood as an ideal component for integration into broader modeling toolchains.

## Statement of need

Forest management requires a long-term perspective, as trees mature over decades or centuries, while wood markets operate on a global scale. Decision makers in the forestry sector need reliable long-term forecasts of global consumption and trade patterns for forest products to manage ecosystems effectively. They particularly value the ability to explore potential future wood harvest developments under various demand and supply scenarios. This need has prompted forest economists to develop macroeconomic models of the forest sector. Several global forest sector models currently exist, including the Global Forest Products Model (GFPM)[@buongiorno2003global], the

European Forest Institute Global Trade Model (EFI-GTM) [kallio2004global], the Global Forest and Agriculture Model (G4M) [gusti2020g4m], and the Global Forest Trade Model (GFTM) [jonsson2015global]. Variants such as Timba (a GFPM adaptation) [tifs2025] complement numerous regional and national models. Model transparency is crucial for determining whether a particular model is suitable for analyzing specific policy questions or can be appropriately modified for new purposes. While research papers provide the conceptual foundation for these models, reading the source code of the model implementation offers a more comprehensive understanding of the modeling system. Detailed knowledge on the source code implementation of a model becomes essential when extending a model to address novel research questions.

Macroeconomic forest sector models typically organize market datasets along two key dimensions: country and time. In econometrics, this structure is known as panel data. These market datasets contain information on production, consumption, and trade for specific forest products such as roundwood, sawnwood, wood panels, pulp, and paper products. Current modeling software often lacks a consistent, well-labeled panel data structure. Instead, these programs use partial labeling approaches—such as matrices names or vector names within data frames, or simple column names in spreadsheets. While this approach can make programs more concise, it creates challenges for newcomers trying to understand the models. Variable names are often unclear, and the limited data labeling makes the code difficult to interpret for those unfamiliar with the specific implementation. Examples of readability issue can be found in the source code of models like GFTM, GFPM, and Timba. That source code for other forest sector models, such as EFI-GTM and G4M-GLOBIOM-Forest, is not yet publicly available for review.

The cobwood model has been used to produce scenario analysis for technical reports [mubareka2025] and [rougieux2024]. The first model programmed inside cobwood is only a reimplementaion of an existing model, the main value of this python package doesn't lie in the model itself, but in the panel data structure that can be used to implement many models.

The next sections describe input output data and the data structure.

## Input, output

Cobwood can process **input** data from any tabular source that Python supports. For instance, the GFPMx model uses a single Excel spreadsheet containing separate sheets for consumption, production, import, export, and prices of major forest products from FAOSTAT. The implementation first converts these sheets to CSV files, which the `gfpmx_data.py` module then transforms into an Xarray data structure. Remember that Xarray datasets can be converted to pandas data frame very easily.

**Output** data is saved in NetCDF format, which serves as Xarray's disk

representation. While not commonly used in economics, this format is standard in earth systems modeling, making it ideal for integrated modeling systems where economic and biophysical models exchange data. The `write_datasets_to_netcdf` method adds a third dimension coordinate called “product” before saving datasets to NetCDF files. These files include metadata labels for units, establishing a foundation for reproducible analysis across research teams.

## Data structure and implementation

Cobwood leverages Xarray’s labeled data arrays to represent panel data, enabling a more intuitive approach to economic modeling. This design allows developers to write Python functions that closely mirror the mathematical equations found in academic literature, with explicit time and country dimensions. The package is designed for extensibility across different models, though the initial release focuses on implementing the Global Forest Products Model (GF-PMx) [buongiorno2021gfpmx]. The core of cobwood is the GFPMX object, which organizes global forest product data including consumption, production, trade flows, and prices.

Data organization follows a logical hierarchy:

- Each forest product is stored as a separate Xarray dataset (e.g., `gfpmx[“sawn”]` for sawnwood)
- Within each product dataset, specific variables are accessible as two-dimensional arrays (e.g., `gfpmx[“sawn”][“cons”]` for consumption)
- These arrays maintain panel data structure with clear country and year dimensions
- Some variables, like demand elasticities, use only the country dimension

To explore available variables and their units for any product, users can access the `variables` property (e.g., `gfpmxb2021[“sawn”].variables`). For example this prints the unit used by the roundwood product for the production variable:

```
gfpmxb2021[“indround”][“prod”].unit
# '1000m3'
```

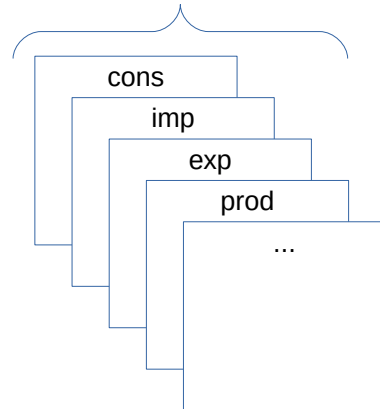
A key advantage of Xarray’s approach is the automatic dimension alignment when performing operations between arrays, which simplifies mathematical operations across different data elements. As Figure 1 illustrates, the labeled panel data structure creates a clean, organized data representation that makes the modeling system more accessible to new users.

## Model run

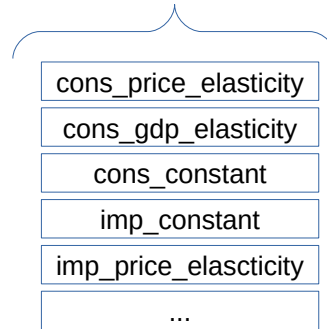
Load the input data into a GFPMX model object. The `rerun=True` argument gives the instruction to erase previous model runs of this scenario. When run-

The dataset `gfmpx["sawn"]` contains many data arrays

Two dimensional variables with countries and time coordinates



One dimensional variables with country coordinates



`gfmpx["sawn"]["cons"]` is a 2 dimensional variable

	1995	1996	...	2099	2100
Algeria					
Angola					
...					
Ukraine					
Uzbekistan					

`gfmpx["sawn"]["cons_price_elasticity"]` is a one dimensional variable

Algeria	
Angola	
...	
Ukraine	
Uzbekistan	

``gfmpx.all_products_ds`` contains 34 variables along 3 coordinates

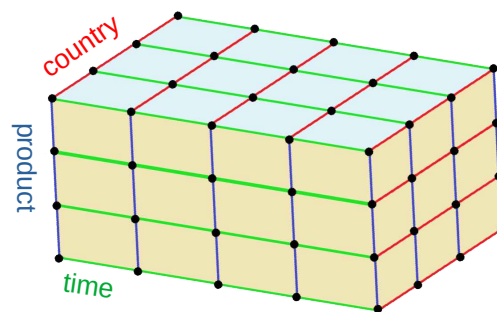


Figure 1: Data structure

ning the model, `compare=True` argument makes it compare the model output with a reference model run from the external model that we use as a reference in this case the Excel implementation of GFPMx:

```
from cobwood.gfpmx import GFPMX
gfpmxb2021 = GFPMX(scenario="base_2021", rerun=True)
gfpmxb2021.run(compare=True, strict=False)
```

After a model run, the scenario output data is automatically saved inside the model's `output_dir` directory. When re-loading the model later, specify the argument `rerun=False` (default) to load both input and output data without the need to run the model.

## Visualisation

The following python code draws a faceted plot of industrial roundwood consumption, import, export, production and price. We don't need to re-run the model this time since we can simply reload the model's output data from the previous run above.

```
from cobwood.gfpmx import GFPMX
gfpmxb2021 = GFPMX(scenario="base_2021", rerun=False)
```

The first plot draws coloured lines by continents. It is the default plot.

```
gfpmxb2021.facet_plot_by_var("indround")
```

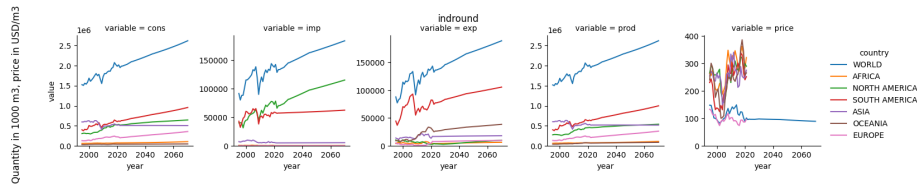


Figure 2: Industrial roundwood variables by continent

The country argument can specify one coloured line by country:

```
gfpmxb2021.facet_plot_by_var("indround", countries=["Canada", "France", "Japan"])
```

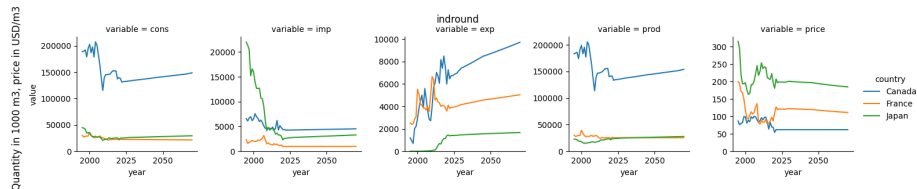


Figure 3: Industrial roundwood variables by country

Plots are visible in the model's output directory `gfpmxb2021.output_dir`. Since the method returns a plot object, the output of the `facet_plot_by_var()` method can also be displayed in a Jupyter notebook directly.

The following code draws a plot of Forest area and forest stock.

```
gfpmxb2021.facet_plot_by_var("indround", countries=["Canada", "France", "Japan"])
```

Xarray objects also have a plot method which provides built-in visualisation capabilities.

## Conclusion

The cobwood package introduces a new way to represent macroeconomic forest products market data using N-dimensional labeled data arrays. This data structure, built on Xarray, improves source code readability by allowing equations with time and country coordinates to be easily identified in the code. Units are stored as metadata attributes within the data structure. Model outputs are saved to NetCDF files, which preserve Xarray's data model, including dimensions and attributes. We believe this new structure makes Cobwood an excellent foundation for implementing various forest sector models. Additionally, the scenario configuration file enables easy comparison of multiple model implementations across a wide range of configuration parameters. # References