# ambrosia: An R package for calculating and analyzing food demand and in accordance with the Edmonds et al food demand model.

*5 October 2020*

## Summary

The `ambrosia` R package was developed to calculate food demand for staples and non-staple commodities that is responsive to changing levels of incomes and prices. `ambrosia` implements the framework to quantify food demand as established by Edmonds et al. (Edmonds et al. 2017) and allows the user to explore and estimate different variables related to the food demand system. Currently `ambrosia` provides 3 main functions:

(1) calculation of food demand for any given set of parameters including income levels and prices

(2) estimation of parameters within a given a dataset. Note: `ambrosia` is used to calculate parameters for the food demand model implemented in the Global Change Analysis Model(GCAM)(Calvin et al. 2019)

(3) exploration and preparation of raw data before starting a parameter estimation

### Summary of the Edmonds et al framework

The Edmonds et al. model represents a food demand model for staples and non-staple commodities at different levels of prices and incomes. Demand for staples is described as increasing when income is lower, eventually peaks at under 1000$ per person per capita, and then begins to decline as higher income ranges are approached. Demand for non-staples increases with income over all income ranges; however, total (staple + non-staple) demand saturates at high income level (Edmonds et al. 2017).

The Edmonds et al. approach uses an 11 parameter model where the parameters are fit using pooled cross-sectional-timeseries observations and a Bayesian Markov Chain Monte Carlo method(Hastings 1988). The framework represents demand for three categories of goods staples(s), non-staples(n) and materials (m) where materials represent everything in the economy other than staples and non-staple food commodities. The demand for these three categories changes with changes in income (Y) and prices (P). Expenditures on these three goods are assumed to exhaust income.

Demand for these three categories can be represented mathematically as,

(1)Staple demand: $q_s = A_s(x^{h_s(x)})(w_s^{e_{ss}(x)})(w_n^{e_sn(x)})$

(2)Non-staple food demand: $q_n = A_n(x^{h_n(x)})(w_s^{e_{ns}(x)})(w_n^{e_nn(x)})$

(3)Materials demand : $q_m = x - w_s q_s - w_n q_n$

Where $w_i$ is $P_i/P_m$, $x$ is $Y/P_m$ and $A_i$ are constants.

$e_{ij}$ is defined in a general way ,

(4)$e_{ij}(x) = g_{ij} * f_i(x)\alpha_j$

where $g_{i,j}$ are constants, $i >= j$ and $f_i(x) = (\delta ln(x_{h_i}(x)))/(\delta ln(x))$

If $h$ and $e$ were constants, $h$ would be an income elasticity as $x = Y/P_m$ and $e_{ij}$ would be own and cross price elasticity as $w_i = P_i/P_m$

The following functional forms are chosen for $h_s$ and $h_n$,

(5) $h_s(x) = (\lambda/x)(1 + (\alpha/ln(x)))$

(6) $h_n(x) = \nu/(1 - x)$

In addition to the above, two other scaling parameters are applied when normalizing the demand values to that of materials. These are *psscl* for staples and *pnscl* for non-staples.

The table in appendix I summarizes all of the parameters described above.

The parameters are fit using a weighted least square log likelihood function (Caroll and Ruppert 1988) described below.

(7) $ln(L) = \Sigma_{i=1}^{N}(w_i(y_i - \hat{y_i})^2)/2\sigma^2$

where, $y_i$ is the $i$th data value and $\hat{y_i}$ is the corresponding model output and $w_i$ is the weight assigned to the data point. Since the parameters were fit based on regional data, the regional population was used as the weight.

By applying the 11 parameters to the equations described above, the user can generate estimates of demand for staples and non staple commodities in thousand calories across different income levels and prices.

## Statement of need

`ambrosia` is part of an ecosystem of tools within the Global Change Intersectoral Modeling System (GCIMS) that help users computationally explore science and policy questions related to different dimensions of human-earth systems. The parameters calculated from `ambrosia` are utilized directly in the agricultural and economic modules of the Global Change Analysis Model(Calvin et al. 2019). `ambrosia` ensures that the parameters that are used within GCAM are scientifically and empirically sound and also ensures reproducibility of the parameters for validation to comply with the commitment of GCIMS to FAIR guiding principles for scientific data management(Wilkinson et al. 2016).The code is structured to ensure that the parameters can be updated and tested effectively with changes in underlying data.

Just as `ambrosia` is used to explore questions related to food consumption, other tools within GCIMS are used to explore other relevant dimensions of human-earth systems(Di Vittorio, Vernon, and Shu 2020),(Hartin et al. 2015),(Li et al. 2017),(Vernon et al. 2018).

In addition to providing scientifically sound parameters for GCAM, `ambrosia` has been developed to help researchers explore questions related to trends in food demand empirically. Since the equations of the model are grounded in peer reviewed research while the code itself is written in R (which increases usability), the tool is useful to researchers interested in,

a) analyzing trends in food demand with a model that is responsive to changes in incomes and prices, that can easily be implemented on any time series (dataset),
b) incorporating a detailed food demand model in their own earth system/economic model.

Another motivation to develop ambrosia is functionalizing and separating out the different components of the sophisticated food demand framework described above into simple R functions that can be easily parameterized and customized by the user. The sections below contain a detailed discussion of the different functions and customization options available within the tool. In this way, the model not only enables easy use and future development, but also enables easy modularization of the code within other systems.

## Functions and customization

The `ambrosia_vignette.rmd` provides usable examples for the major functions within the code. As described in the summary statement, the functions within ambrosia can be classified into three distinct categories

1) Functions to explore food demand variables
2) Functions to estimate parameters using custom data
3) Functions to process raw data for parameter estimation

In addition to these, the code contains a number of helper functions that may be useful to the user depending upon their research requirements.

### Functions to explore demand variables

The ambrosia package can be easily loaded as a standard R package after installation from Github. The user can calculate demand for staples and non-staples using the `food.dmnd()` function. The user will have to pass in a dataset with the price of staples ($Ps$), price of non-staples ($Pn$), incomes ($Y$) in GDP per capita in thousand USD. In addition to the dataset, the user must pass a vector of 11 parameters. In order to functionalize the parameters, the code contains a function called `vec2param()` that will generate a parameter structure that can be used by the food demand function. The usage of the functions are described in the example below. The example makes use of the vector of parameters directly from Edmonds et al. The food demand function is implemented using equations (1),(2),(3) desbcribed above.

```r
#Example 1: Creating estimates of demand

#Parameter values are taken from Edmonds et al (Pg 12, Table 3 )

original_param_vector <- c(1.28,1.14,-0.19,0.21,-0.33,0.5,0.1,16,5.06,100,20)

#Names of the parameters above are as follows. These are the same as the names in Table 1 in Appendix I

parameter_names <- c('A_s', 'A_n', 'xi_ss', 'xi_cross', 'xi_nn', 'nu1_n',
                     'lambda_s', 'k_s', 'Pm', 'psscl','pnscl')

#Generate parameter set for the food demand model

params <- vec2param(original_param_vector)

#Create a sample dataset
Test_Data<-data.frame(Y=seq(0.1,30, by=0.1))
Test_Data %>% mutate(Ps=0.1,Pn=0.2)->Test_Data

Food_Demand<-food.dmnd(Ps = Test_Data$Ps,
                       Pn = Test_Data$Pn,
                       Y = Test_Data$Y,
                       params = params)
```

Using the function above will create a dataframe with estimates of demand for each level of price and incomes and also the budget shares (shares of incomes spent) for staples and non-staples. Plotting these result in Figure 1 and Figure 2 below respectively

The demand code iteratively solves for the budget shares using a Broyden solver(Broyden 1969) with changing incomes and re-calculates income and price elasticities for changes in budget shares.The user can separately

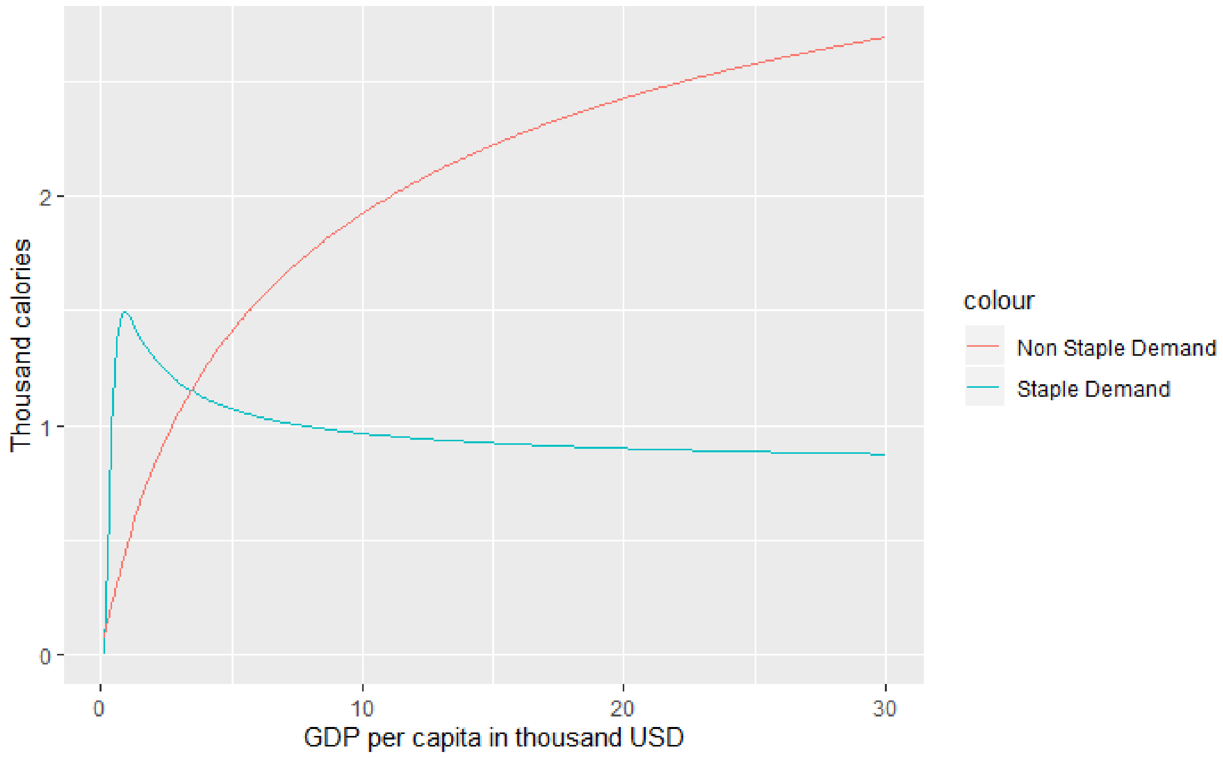

Figure 1: Figure 1:Food Demand in thousand calories for a range of incomes with constant prices.

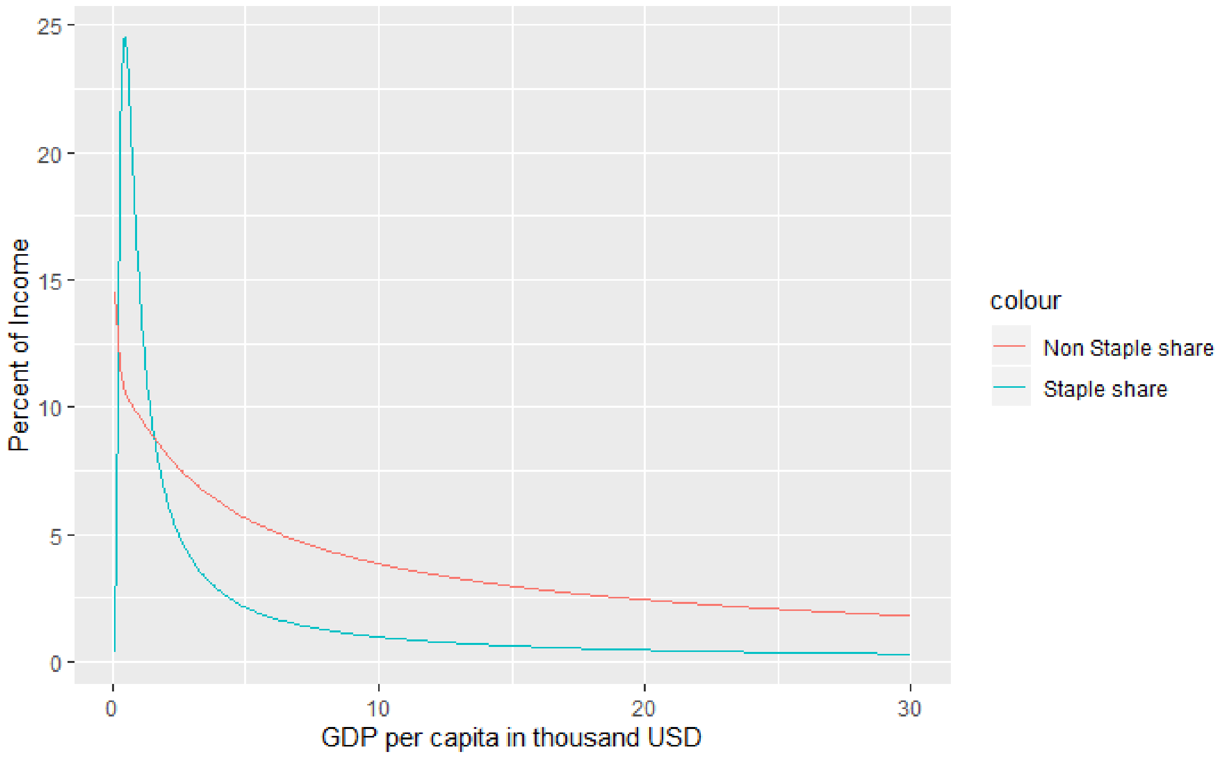

Figure 2: Figure 2:Budget shares as a percent for a range of incomes with constant prices.

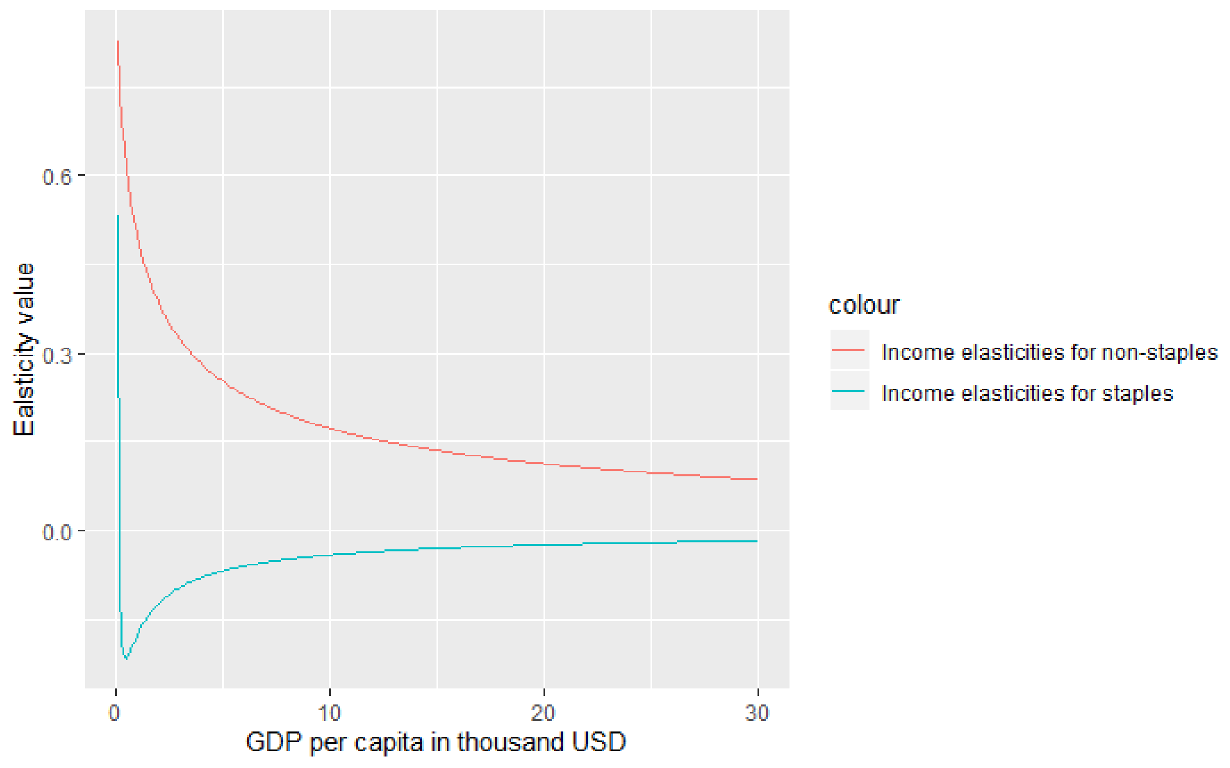

Figure 3: Figure 3:Income elasticities relative to income

analyze the income elasticities by using two functions (one for staples and other for non-staples) from within the parameter structure generated above. These functions implement equations (5) and (6) described above for a given level of income (Y). This is explained in the example below

```
#Example 2: Calculating/Analyzing income elasticities

#As explained in the documentation of these functions, setting the second parameter to TRUE will genera
#the Y term (Y ^ elasticity) as opposed to the elasticity itself.

#Get income elasticities for staples (1st function)
Food_Demand$eta.s <- params$yfunc[[1]](Y=Food_Demand$Y,FALSE)

#Get income elasticities for staples (1st function)
Food_Demand$eta.n <- params$yfunc[[2]](Y=Food_Demand$Y,FALSE)
```

The results from the above can be plotted to create the following plot that shows the relationship of income elasticities to income levels.

Similar to the income elasticities, the user can also calculate and analyze price elasticities. These elasticities are calculated in accordance with equation (4) described above. The function `calc1eps()` can be used to recalculate price elasticities by passing the following parameters:

a)different budget shares (alphas)

b)income elasticities (calculated in example 2)

c)A matrix of values for elasticities (This is a part of the parameter structure set up in example 1)

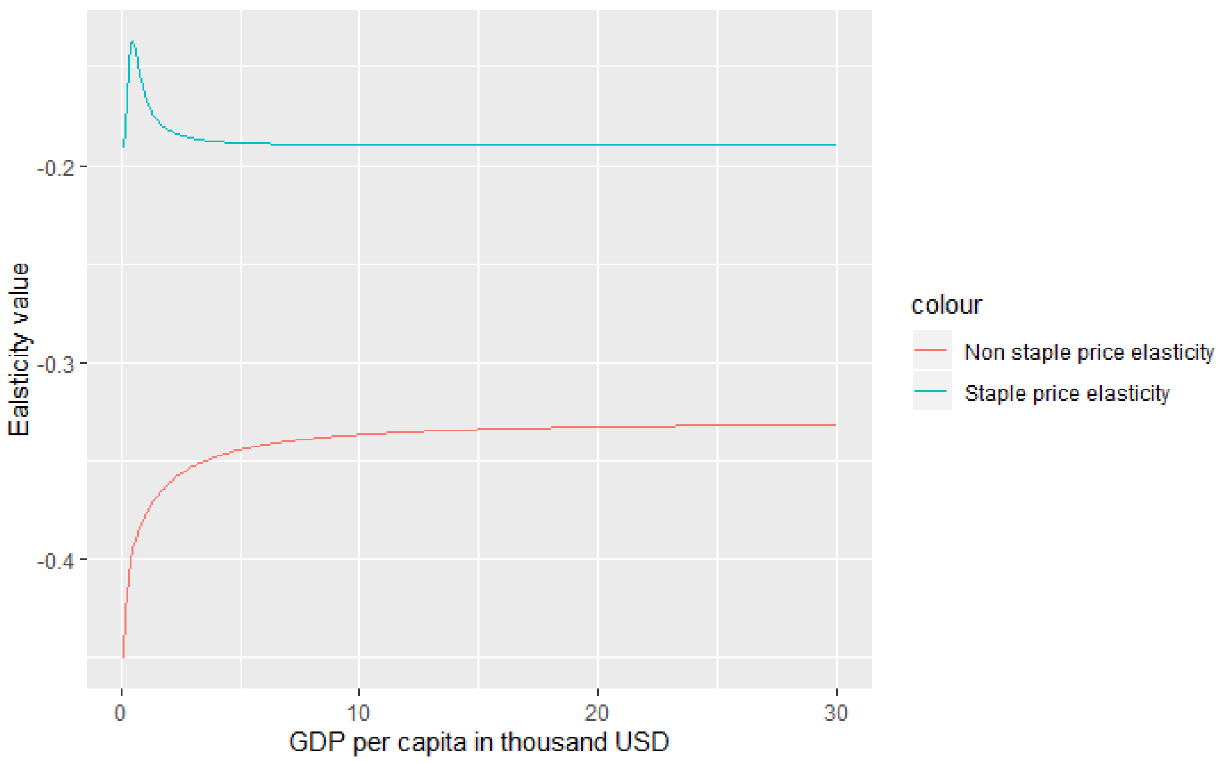

Figure 4: Figure 4:Price elasticities relative to income

The functions to derive price elasticities are described in the example below.

```
#Example 3: Calculating/Analyzing price elasticities

Food_Demand$staple_price_elasticity <- calc1eps(Food_Demand$alpha.s,Food_Demand$alpha.n,Food_Demand$eta


Food_Demand$non_staple_price_elasticity <- calc1eps(Food_Demand$alpha.s,Food_Demand$alpha.n,Food_Demand
```

Similar to the income elasticities, these can now be plot against incomes as seen in Fig 6. below

## Functions to process raw data for parameter estimation

As mentioned in the statement of need, one of the benefits of using `ambrosia` is that a user can estimate their own parameters with a custom data set using the log-likelihood maximization approach. To enable this, `ambrosia` is equipped with a function (`create_dataset_for_parameter_fit()`) that will help a user generate a dataset that is appropriate for parameter estimation. There are a few steps that the function will perform on a sample dataset.

1) It will ensure that the user's dataset contains all columns required for parameter estimation
2) It will filter out anomalies and outliers using parameter values selected by the user. This step is necessary since data on food consumption and prices are often incomplete which may lead to unrealistically high or low values of consumption or prices in the dataset.

3) After this, the function will create clusters of observations from the dataset based on income levels, and prices of staples and non-staples. This step is necessary because this being economic data, the observational error can only be calculated within different clusters. The code will also check for a user specified minimum number of clusters(If there are anomalies within the dataset, the clustering can be incorrect leading in a small number of clusters). The clustering is implemented using the Divisive Analysis Clustering Algorithm (DIANA)(Kaufman and Rousseeuw. 2009).
4) Once the clustering is completed, the code will calculate the observational error which is the variance in food demand for staples ($\sigma^2 Qs$) and non-staples ($\sigma^2 Qn$) .Note that the user can chose a lower limit on the observational error calculated. The default value of the lower limit is 0.01.

In addition to a data frame, the function will return a CSV file output called "Processed_Data_for_MC.csv" that is stored in the outputs folder that will be used for the parameter estimation. The example below illustrates how to use the function on a raw dataset.

```
#Example 3: Creating a dataset for parameter fitting

parameter_data <- create_dataset_for_parameter_fit(data=Sample_Data,
                                                   min_clusters = 100,
                                                   min_price_pd = 20,
                                                   min_cal_fd = 1000,
                                                   lower_limit_sigma = 0.01)
```

The dataset returned by this function can now be used for parameter estimation. The user can also plot the observational errors for staples and non-staples to ensure there is a valid distribution and the data is not skewed (as seen in Figure 6 below for Staples).

## Functions for parameter estimation

Finally, the user can complete the parameter estimation on the dataset returned by `create_dataset_for_parameter_fit()` with a call to the `calculate_ambrosia_params()` function. `ambrosia` builds on the Edmonds et al. approach by maximizing the log-likelihood score using the `optim()` function. The following steps are involved in the parameter estimation function,

1) First a log-likelihood function is set up with the data returned by the function above. This function is the same as equation (7) described above.
2) Now, the value returned by this function will be maximized using `optim()`. The user can provide a seed of initial parameters to begin the optimization process (the lowest possible seed would be the lowest values of all 11 parameters). The default seed is set to the original parameters from Edmonds et al. The user can now specify the optimization method to be used. The default is set to the "BFGS" method, but the user can also run the optimization using methods such as "Neldor-Mead" etc.
3) The function will now return a vector of parameters that can be used to derive estimates of food demand (Similar to Example 1 above). The function also prints out the maximized value of the log-likelihood function, so that the user can verify the efficiency and effectiveness of the parameter estimation.

The example below illustrates the use of this function along with all parameters.

```
#Example 4: Estimating parameters

parameter_data <- calculate_ambrosia_params(optim_method = "BFGS",
                                            original_param_vector= c(1.28,1.14,-0.19,0.21,-0.33,0.5,0.1,16,5
                                            datadir = "outputs/Processed_Data_for_MC.csv")
```

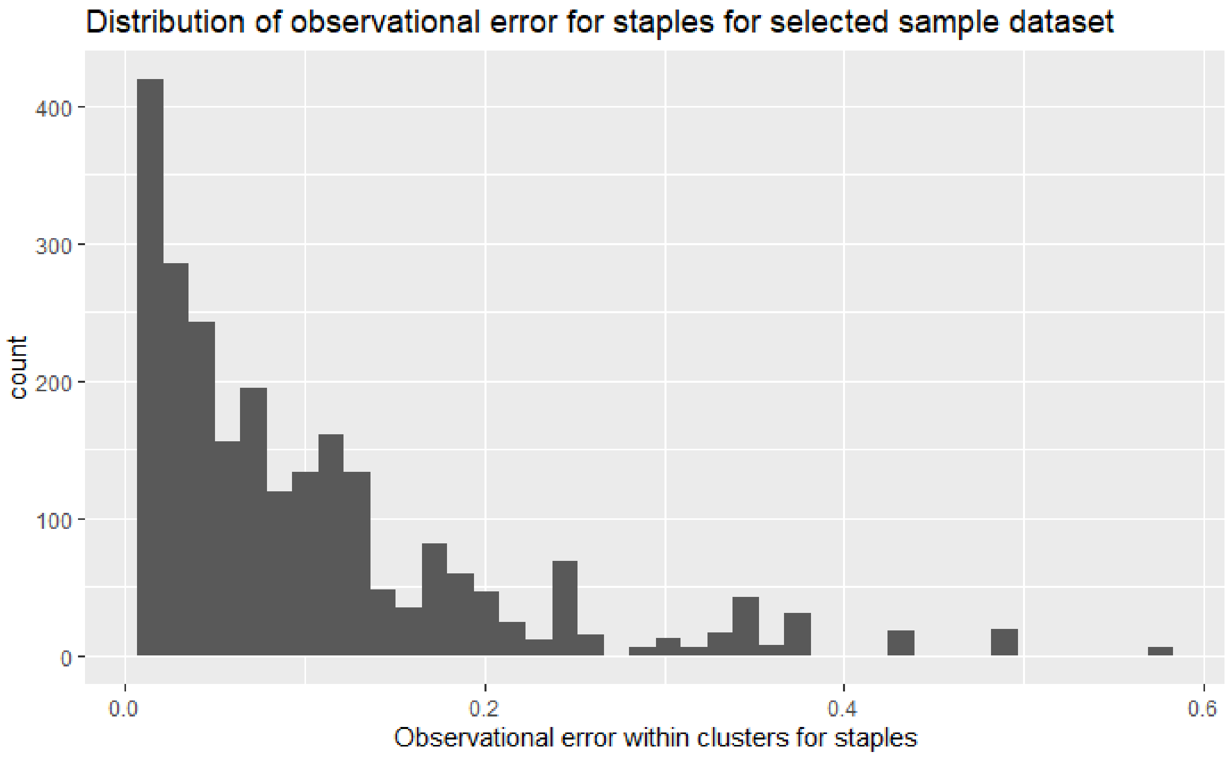

Figure 5: Figure 5:Distribution of observational error for staples for sample data

## Other functions

In the addition to the main functions described above, ambrosia also includes a number of supplementary functions that are used within the main functions themselves that the user can customize to suit their research needs. These functions are summarized in the table below.

| Category | Function name | Description |
|---|---|---|
| Parameter estimation | mc.eval.fd.likelihood() | The function will return the log likelihood value for a set of parameters and a given set of observational data. |
| | mc.setup() | Function returned by mc.setup() is used to fit parameters using the maximization approach |
| Data processing | recursive.partition() | A function that is used for clustering using the DIANA approach using any criteria variables |
| | assign.sigma.Q() | Computes observational errors within clusters for staple and non staple food demand |
| | calc.pop.weight() | Calculates weights used in equation (7) for the least squares log likelihood function |
| Food demand functions | calc.elas.actual() | Calculates elasticities using numerical derivatives for given prices and incomes. |
| | calc.hicks.actual() | Calculates Hicks elasticities using a Slutsky equation |

## Other outputs

In addition to the outputs described above, the model will generate 2 outputs as csv files for testing and validation. Firstly, the `Processed_Data_for_MC.csv` is saved as an output of the `create_dataset_for_parameter_fit()` function. This dataset is used to fit the parameters and is also used by the testing suite to ensure outputs are valid.`parameter_data.csv` is saved as an output of the `calculate_ambrosia_params()` function. This is the final set of parameters with their scientifc names that is passed to GCAM.

## Availability

### Operating system

Mac OS X; Linux; Windows

### Programming language

R (>= 3.3.0)

### Dependencies

dplyr (>= 0.7)

nleqslv (>= 3.2)

reshape2 (>= 1.4.3)

ggplot2 (>= 2.2.1)

cluster (>= 2.0)

tidyr (>= 0.7.1)

### Code repository

*Name*- Github

*Identifier*- https://github.com/JGCRI/ambrosia/tree/master

*License*- Apache License 2.0

## Acknowledgements

This research was supported by the U.S. Department of Energy (DOE), Office of Science, Office of Biological and Environmental Research, as a part of research in the MultiSector Dynamics, Earth and Environmental System Modeling Program.

## Appendix I

Table describing all 11 parameters for the food demand model with values from the latest iteration.

| parameter name | Description | value (from the latest iteration of the model) |
|---|---|---|
| A_s | Scaling parameter for staples (constant) | 1.13 |
| A_n | Scaling parameter for non-staples (constant) | 1.24 |
| xi_ss | Price elasticity for staples | -0.024 |
| xi_cross | Cross price elasticity | -0.010 |
| xi_nn | Price elasticity for non_staples | -0.143 |
| nu1_n | Income elasticity for non-staples | 0.5 |
| lambda_s | Income elasticity for staples | 0.0476 |
| k_s | Value of income (Y) at which elasticity is 0 | 16.0 |
| Pm | Price of materials | 5.13 |
| psscl | Additional scaling parameter for staples | 100.0 |
| pnscl | Additional scaling parameter for non-staples | 20.1 |

## References

Broyden, C. G. 1969. "A New Method of Solving Nonlinear Simultaneous Equations." *The Computer Journal* 12 (1): 94–99.

Calvin, Katherine, Pralit Patel, Leon Clarke, Ghassem Asrar, Ben Bond-Lamberty, Ryna Yiyun Cui, Alan Di Vittorio, et al. 2019. "GCAM V5. 1: Representing the Linkages Between Energy, Water, Land, Climate, and Economic Systems." *Geoscientific Model Development (Online)* 12 (PNNL-SA-137098).

Caroll, R.J., and D. Ruppert. 1988. "Transformation and Weighting in Regression." *CRC Press* 30.

Di Vittorio, Alan, Christopher R Vernon, and Shijie Shu. 2020. "Moirai Version 3: A Data Processing System to Generate Recent Historical Land Inputs for Global Modeling Applications at Various Scales." *Journal of Open Research Software* 8 (PNNL-SA-142149).

Edmonds, James A, Robert Link, Stephanie T Waldhoff, and Ryna Cui. 2017. "A Global Food Demand Model for the Assessment of Complex Human-Earth Systems." *Climate Change Economics* 8 (04): 1750012.

Hartin, Corinne, Pralit Patel, A Schwarber, Robert Link, and Ben Bond-Lamberty. 2015. "A Simple Object-Oriented and Open-Source Model for Scientific and Policy Analyses of the Global Climate System – Hector V1.0." *Geoscientific Model Development* 8 (gmd-8-939-2015).

Hastings, W. K. 1988. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika* 1.

Kaufman, Leonard, and Peter J. Rousseeuw. 2009. "Finding Groups in Data: An Introduction to Cluster Analysis" 344.

Li, Xinya, Chris R Vernon, Mohamad I Hejazi, Robert P Link, Leyang Feng, Yaling Liu, and Lynn T Rauchenstein. 2017. "Xanthos–a Global Hydrologic Model." *Journal of Open Research Software* 5 (PNNL-SA-126584).

Vernon, Chris R, Yannick Le Page, Min Chen, Maoyi Huang, Katherine V Calvin, Ian P Kraucunas, and Caleb J Braun. 2018. "Demeter–a Land Use and Land Cover Change Disaggregation Model." *Journal of Open Research Software* 6 (PNNL-SA-131044).

Wilkinson, Mark D, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. 2016. "The Fair Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3 (1): 1–9.