

Microsoft Research Terrestrial Carbon Model Package: User's Guide

M. J. Smith, D. W. Purves, M. C. Vanderwel, V. Lyutsarev and S. Emmott, Computational Science Laboratory, Microsoft Research Cambridge, 21 Station Road, Cambridge, CB1 2FB, UK

This user's guide accompanies the research publication "The climate dependence of the terrestrial carbon cycle, including parameter and structural uncertainties" Details of that publication at <http://research.microsoft.com/apps/pubs/default.aspx?id=180603> and www.biogeosciences.net/10/583/2013/

referred to below as Smith *et al.* (2013).

Please email queries to Matthew.Smith@Microsoft.com

Contents

1. Introduction	3
2. System requirements	3
3. Install Microsoft Visual C# 2010 Express or Microsoft Visual Studio 2010 if you want to work with the code	4
4. Download and unpack the solution to obtain the code and the executable	4
5. Study the solution structure	6
6. If you have a 64 bit operating system then change the default build to 64 bit	7
7. Skim-read Program.cs	8
8. Study the structure and contents of the data folder	8
9. Running the s fully data-constrained global terrestrial carbon model from command line arguments	15
10. Repeating the methods used in the paper	15
10.1. FULL	16
10.2. MAPS	19
10.3. EQCARBDIST	20
10.4. BUILD-UP <n>	22

10.5.	ALL-DUMMY <n>.....	23
10.6.	ONE-DUMMY <n>.....	24
10.7.	OMIT-DATA<n>	25
10.8.	ANALYSE-PARAMS.....	25
10.9.	SIMULATE.....	27
10.10.	ANALYSE-SIMULATIONS.....	29
10.11.	FINAL-TEST	30
11.	R Scripts to produce final publication graphs	30
12.	Conducting novel studies.....	31
13.	Using the Dmitrov libraries within the code.....	33
14.	Obtaining DataSet Viewer.....	33
15.	Using Filzbach.....	33
16.	Using FetchClimate	34

1. Introduction

The study of Smith *et al.* (2013) reports the development and analysis of the fully data-constrained global terrestrial carbon model within a prototype framework for rapid modeling engineering and refinement.

At present the fully data-constrained global terrestrial carbon model and the framework are both contained within the same Microsoft Visual Studio solution, written principally in the C# programming language (we composed some of the graphs using the statistical package "R" and provide that code with the solution package). This user guide provides instructions on how to use framework to repeat the analyses of Smith *et al.* (2013). A separate download is needed to run the future carbon cycle projections (Fig. 3) in the analysis of Smith *et al.* (2013) because simulating the inferred models under different climate change scenarios was not part of the prototype framework for model engineering and refinement. The relevant code is also available for downloading and we provide instructions here on how to modify the code to perform simulations.

The study of Smith *et al.* (2013) was performed through interacting with the raw C# source code within the Microsoft Visual Studio solution, principally by enabling or disabling calls to procedures corresponding to different experiments or analyses. We have thoroughly commented the code to help users understand what it does. We have also made it possible for users to run specified analyses through command line arguments, without needing to use Visual Studio. We have not yet added a graphical user interface.

2. System requirements

The study of Smith *et al.* (2013) can be implemented directly as an executable binary file or through Microsoft Visual Studio. For the former, user's will not need to have Microsoft Visual C# 2010 Express or Microsoft Visual Studio 2010 installed on their computer to run the framework, but will not be able to implement alterations to the code (although the code can still be viewed using a text reader). Although you do not have to install any version of Visual Studio you still have to ensure you have the following components installed on your computer for the executable file to run:

- Microsoft .NET Framework 4.0 Client Profile <http://microsoft.com/net/download> Microsoft Visual C++ 10.0 Redistributable (x86 or x64 depending on processor architecture and operating system of your computer), available from the Microsoft downloads site. We recommend you search for "Microsoft Visual C++ 2010 SP1 Redistributable Package" from the <http://www.microsoft.com/download> website.

If users use Microsoft Visual C# 2010 Express or Microsoft Visual Studio 2010 to work with the code then they will benefit from being able to read the code, navigate the solution structure and implement any modifications.

3. Install Microsoft Visual C# 2010 Express or Microsoft Visual Studio 2010 if you want to work with the code

If you do not already have Microsoft Visual C# 2010 Express or Microsoft Visual Studio 2010 on your computer then you will need to install one of these to be able to navigate the solution structure and implement modifications to the code.

Microsoft Visual C# 2010 Express is free to download from <http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express>. It provides the basic functionality needed to load, run and edit the solution. After a period of time you will probably have to register your use of Visual C# 2010 Express to continue to use it. Additional functionality (source control, multiple .NET languages) can be obtained using Microsoft Visual Studio 2010 (<http://www.microsoft.com/visualstudio/en-us>) although this is generally not freely available. This user guide refers to using the solution in Microsoft Visual C# 2010 Express.

4. Download and unpack the solution to obtain the code and the executable

If you do not want to read or modify the code used in the study of Smith *et al.* (2013) but only want to run it then you will still need to download and unpack the solution to obtain the executable file that will implement the study.

The Microsoft Visual Studio Solution is packaged as a .zip file and can be downloaded from <http://research.microsoft.com/en-us/downloads/8c51f0b5-17a1-413e-90c4-43c61c7e4843/default.aspx>. After downloading the file, unpack the .zip file to a folder on your computer.

This should generate a single folder that contains all of the files in the solution: i.e.

YOUR_ROOT_DIRECTORY/Package

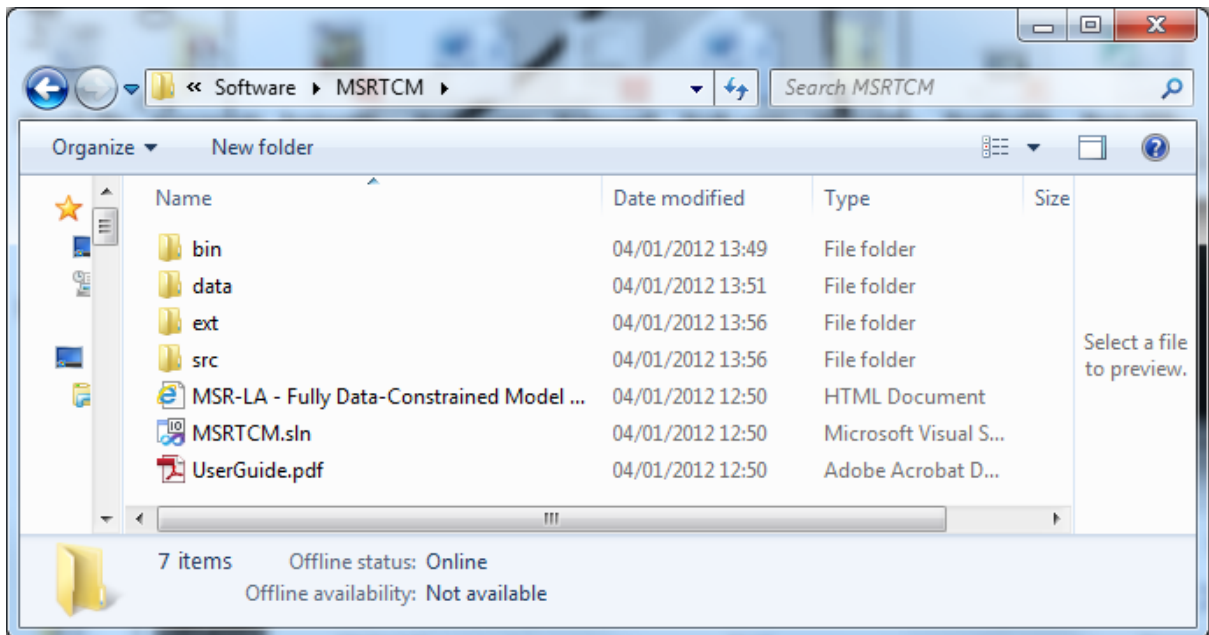


Figure 1 | You should see something like this when you open the Package folder

Within the Package folder (Fig. 1) should be:

- A “bin” folder: contains the executable MSRTCM.exe and associated libraries necessary to run the fully data-constrained global terrestrial carbon model without having Visual Studio or Visual C# express installed.
- a “data” folder: contains a hierarchical set of folders for holding the input and output data, some of which already contain data
- an “ext” folder: containing compiled binaries for Filzbach (Parameter Inference), FetchClimate (remote data access), Scientific DataSet (facilitating the handling of datasets), as well as some standard scripts for running the statistical package “R” to produce the graphs used in Smith *et al.* (2013). More details on these packages are provided below.
- an "src" folder: containing all of the source code used in the study
- "MSR-LA - Fully Data-Constrained Model for Global Vegetation.htm": containing the legal terms of use for the package and citations to all of the data providers who kindly agreed to us releasing derivatives of their data along with the study of Smith et al (2013) to enable users to recreate our results.
- A “MSRTCM.sln” file - The Microsoft Visual Studio Solution description, this file can be opened using Microsoft Visual Studio or Microsoft Visual C# 2010 Express
- "UserGuide.pdf": This user guide

5. Study the solution structure

If either Microsoft Visual C# 2010 Express or Microsoft Visual Studio 2010 is installed on your computer then you can open the “MSRTCM.sln” file to load the solution. Double click the “MSRTM.sln” file. If Visual Studio doesn’t open then right-click the file and choose Open With > Microsoft Visual C# 2010 Express.

The main entry point for standard use of the solution for conducting the study of Smith *et al.* (2013) was the Program.cs file. This is listed at the bottom of the solution structure in the Solution Explorer window (Fig. 2).

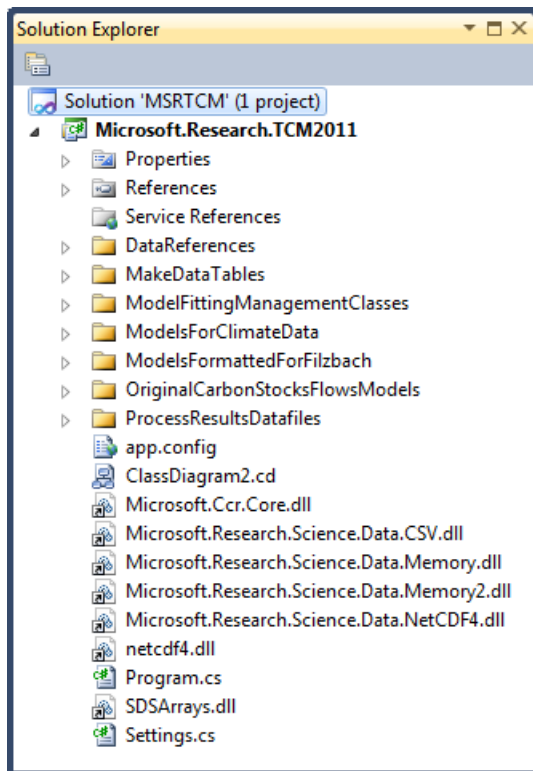


Figure 2 | This is what you should see when you open the solution explorer window. The Program.cs file contains the functions for the highest level operations of the code. Other classes are grouped into folders.

In the Solution Explorer window you can click the little arrows on the left of folders or file names to expand or contract lists of files as was done to give the image on the right. This shows the overall structure of the solution which basically divides up the references (to code libraries), raw C# scripts and some other files into different folders corresponding to different categories of use in the solution.

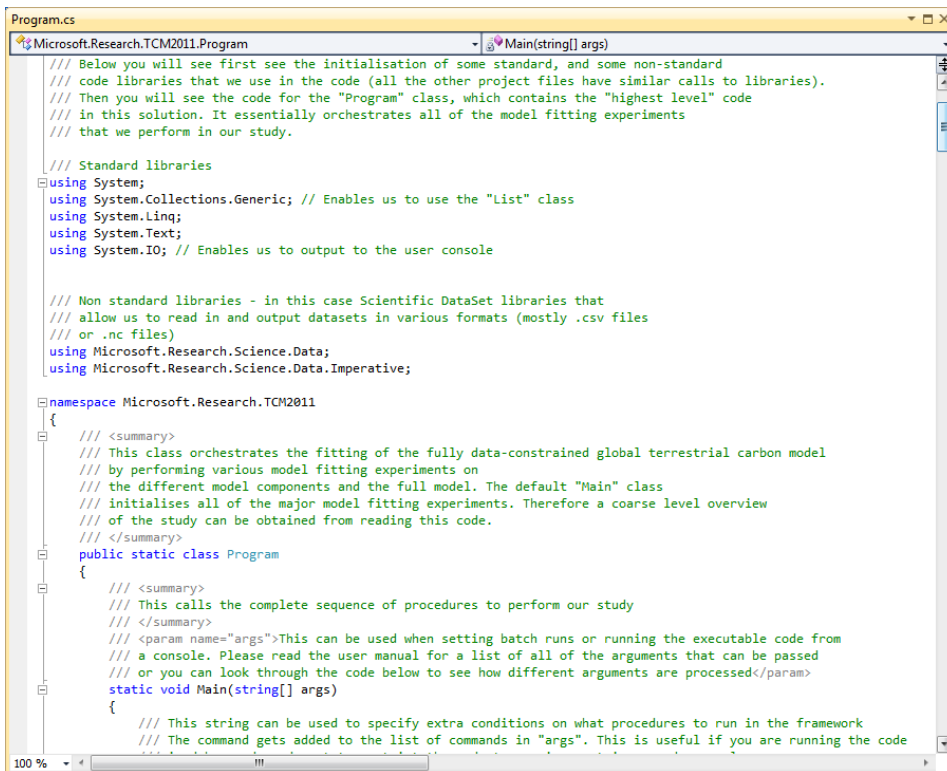
The different folders in the solution structure correspond to different folders in the YOUR_ROOT_DIRECTORY/Package/src folder. In summary, these are

- DataReferences: Contains text files detailing the sources of, and giving citations for, all of the non-Microsoft datasets used in the Smith *et al.* (2013) study.
- MakeDataTables: A set of C# scripts for reading in the different ecological and climatological datasets for use in the Smith *et al.* (2013) study.
- ModelFittingManagementClasses: A set of C# scripts to enable Bayesian parameter estimation for arbitrary combinations of models and datasets
- ModelsForClimateData: Some C# scripts to enable the calculation of environmental variables related to water balance (evapotranspiration, soil water content, fire frequency)
- ModelsFormattedForFilzbach: C# scripts to handle the conversion of the ecological models used in the study into a format suitable for Bayesian parameter estimation.
- OriginalCarbonStocksFlowsModels: C# scripts of the ecological models used in the study.
- ProcessResultsDatafiles: A set of C# scripts for post-processing data resulting from Bayesian parameter estimation. It also includes code for mapping predictions.

6. If you have a 64 bit operating system then change the default build to 64 bit

We have set the default configuration to be for 32 bit operating systems but if your processor is 64 bit then you should get improved performance (faster running program and access to more memory) if you switch to 64 bit. To do this in either Visual Studio or Visual C# Express right-click the “Solution ‘MSRTCM’ (1 project)” node in Solution Explorer window and select “Configuration Manager...” in the corresponding context menu. The Configuration Manager window then appears. In the “Active solution platform:” box select “64bit” and close the window. You can also use Configuration Manager to switch between Release and Debug configurations. In Release configuration the C# compiler does more code optimizations and so the program runs faster. Visual C# will save these preferences when you close the window. .

7. Skim-read Program.cs



```
Program.cs
Microsoft.Research.TCM2011.Program
Main(string[] args)

/// Below you will see first see the initialisation of some standard, and some non-standard
/// code libraries that we use in the code (all the other project files have similar calls to libraries).
/// Then you will see the code for the "Program" class, which contains the "highest level" code
/// in this solution. It essentially orchestrates all of the model fitting experiments
/// that we perform in our study.

/// Standard libraries
using System;
using System.Collections.Generic; // Enables us to use the "List" class
using System.Linq;
using System.Text;
using System.IO; // Enables us to output to the user console

/// Non standard libraries - in this case Scientific DataSet libraries that
/// allow us to read in and output datasets in various formats (mostly .csv files
/// or .nc files)
using Microsoft.Research.Science.Data;
using Microsoft.Research.Science.Data.Imperative;

namespace Microsoft.Research.TCM2011
{
    /// <summary>
    /// This class orchestrates the fitting of the fully data-constrained global terrestrial carbon model
    /// by performing various model fitting experiments on
    /// the different model components and the full model. The default "Main" class
    /// initialises all of the major model fitting experiments. Therefore a coarse level overview
    /// of the study can be obtained from reading this code.
    /// </summary>
    public static class Program
    {
        /// <summary>
        /// This calls the complete sequence of procedures to perform our study
        /// </summary>
        /// <param name="args">This can be used when setting batch runs or running the executable code from
        /// a console. Please read the user manual for a list of all of the arguments that can be passed
        /// or you can look through the code below to see how different arguments are processed.</param>
        static void Main(string[] args)
        {
            /// This string can be used to specify extra conditions on what procedures to run in the framework
            /// The command gets added to the list of commands in "args". This is useful if you are running the code
        }
    }
}
```

Figure 3 | Program.cs contains the highest level operations of the code with Main() being the initial entry point.

Program.cs has been thoroughly commented to make it as readable as possible (Fig. 3), a principle which applies to all of the solution code.

It starts with references to standard and non-standard (specific to the solution) namespaces which you will normally just ignore.

The main function you need to look at is called Main(). It begins at the bottom of the image below with the text `static void Main(string[] args)`.

The main function orchestrates calls to the highest level operations to be conducted by the solution. These are i) the Bayesian parameter estimation of models given datasets, and ii) post parameter estimation steps such as the simulation and mapping of model predictions.

Note that a convenient way to navigate through functions is to click on the function you are interested in and press "F12". This should automatically take you to the code for the function.

8. Study the structure and contents of the data folder

The data files necessary to repeat the study of Smith *et al.* (2013) are included with the package and reside in the YOUR_ROOT_DIRECTORY/Package/data folder.

This folder has several subfolders

OutputData

The output directory for all outputs from the solution (other than training, evaluation or test data). This contains four additional folders to subdivide the data into that produced from performing parameter inference (ModelFittingOutputData), from post-processing all of the results from the different model fitting experiments (ProcessedFittingOutputData), from assessing the model using the final test data (ProcessedReservedTestOutputData) and from simulating the model to study its predictions (SimulationOutputData). This folder is initially blank except for one file (ProcessedFittingOutPutData\ProcessErrorValues.csv) needed to recreate our results. However all of the output data resulting from our study are available for download from <http://research.microsoft.com/en-us/downloads/a1281531-df37-4489-a556-56799fd252b4/default.aspx> and <http://download.microsoft.com/download/1/F/D/1FD1F550-69C4-4503-B2FE-B47F94607A7F/MSRTCMSIMData.zip>.

RawData

This was used to hold the raw datafiles used to produce the training, evaluation and test datafiles. We do not have permission to redistribute all of these datafiles and some of them are quite large so this folder only contains the data mask that we used to partition the raw data into training/evaluation and test data (cru20DataMask.nc). These datafiles can be obtained from the sources listed in Table 1.

Table 1: Data sources for the study of Smith *et al.* (2013)

Data set name	Data set description	Data source	Citation
Global biomass carbon map in the year 2000	The amount of carbon held in terrestrial vegetation, tonnes carbon ha-1	Carbon Dioxide Information and Analysis Centre http://cdiac.ornl.gov/epubs/ndp/global_carbon/carbon_documentation.html	Ruesch, Aaron, and Holly K. Gibbs. 2008. New IPCC Tier-1 Global Biomass Carbon Map For the Year 2000. Available online from the Carbon Dioxide Information Analysis Center [http://cdiac.ornl.gov], Oak Ridge National Laboratory, Oak Ridge, Tennessee.
Global litter producti	Litter production rates, g dry	Matthews, E. Global litter production, pools and turnover times: Estimates	Matthews, E. Global litter production, pools and turnover times: Estimates from measurement data and regression

on data	matter m-2 yr-1	from measurement data and regression models. J. Geophys. Res. 102, 18771-18800 (2003).	models. J. Geophys. Res. 102, 18771-18800 (2003).
Global gridded surfaces of selected soil characteristics (IGBP-DIS)	Soil carbon density (kg m-2) at a depth interval of 0-100 cm.	Oak Ridge National Laboratory Distributed Active Archive Centre (ORNL-DAAC) http://daac.ornl.gov/SOILS/guides/igbp-surfaces.html	Global Soil Data Task Group. 2000. Global Gridded Surfaces of Selected Soil Characteristics (IGBP-DIS). [Global Gridded Surfaces of Selected Soil Characteristics (International Geosphere-Biosphere Programme - Data and Information System)]. Data set. Available on-line [http://www.daac.ornl.gov] from Oak Ridge National Laboratory Distributed Active Archive Center, Oak Ridge, Tennessee, U.S.A. doi:10.3334/ORNLDAAC/569.
"Class B site" net primary productivity (NPP)	Net primary productivity (kg carbon m-2 yr-1)	Oak Ridge National Laboratory Distributed Active Archive Centre (ORNL-DAAC). http://daac.ornl.gov/NPP/html_docs/EMDI_des.html ,	Olson, R. J., J. M. O. Scurlock, S. D. Prince, D. L. Zheng, and K. R. Johnson (eds.). 2001. NPP Multi-Biome: NPP and Driver Data for Ecosystem Model-Data Intercomparison. Data set. Available on-line [http://www.daac.ornl.gov] from the Oak Ridge National Laboratory Distributed Active Archive Center, Oak Ridge, Tennessee, U.S.A.
GLOPNET Leaf Traits data	Estimated lifespan (in months) of leaves and whether they are evergreen or deciduous	Authors of Wright, I. <i>et al.</i> The worldwide leaf economics spectrum. Nature, 428, 821 (2004).: Ian Wright and Peter Reich	Wright, I. <i>et al.</i> The worldwide leaf economics spectrum. Nature, 428, 821 (2004).
Global root turnover data	Mean root turnover (yr-1)	Gill, R. & Jackson, R. B. Global Patterns of root turnover for terrestrial ecosystems. New Phytologist 81:275-280 (2000). published by John Wiley & Sons Ltd. http://onlinelibrary.wiley.com/doi/10.1046/j.1469-8137.2000.00681.x/abstract	Gill, R. & Jackson, R. B. Global Patterns of root turnover for terrestrial ecosystems. New Phytologist 81:275-280 (2000). published by John Wiley & Sons Ltd.

Forest turnover data	Forest turnover rates (yr-1) from different sites worldwide.	Stephenson, N.L. & van Mantgem, P.J. Forest turnover rates follow global and regional patterns of productivity. <i>Ecol. Lett.</i> 8, 524-531 (2005).published by John Wiley & Sons Ltd. http://onlinelibrary.wiley.com/doi/10.1111/j.1461-0248.2005.00746.x/supinfo	Stephenson, N.L. & van Mantgem, P.J. Forest turnover rates follow global and regional patterns of productivity. <i>Ecol. Lett.</i> 8, 524-531 (2005).published by John Wiley & Sons Ltd.
Global Fire Data	Percentage of a grid cell burned per year for 100 years (1900-2000)	Florent Mouillot Data obtained from the author's web-page	Mouillot,F. & Field, C. B., Fire history and the global carbon budget. <i>Global Change Biology</i> , 11(3): 398-420 (2005).
Metabolic fraction of carbon in terrestrial vegetation	Fraction of leaf and fine root carbon that is decomposed quickly by soil organisms (fraction)	Ise, T. & Moorcroft, P.R. The global-scale temperature and moisture dependencies of soil organic carbon decomposition: an analysis using a mechanistic decomposition model. <i>Biogeochem.</i> 80, 217-231 (2006). Published by Springer	Ise, T. & Moorcroft, P.R. The global-scale temperature and moisture dependencies of soil organic carbon decomposition: an analysis using a mechanistic decomposition model. <i>Biogeochem.</i> 80, 217-231 (2006). Published by Springer
Global land cover for the year 2000	Discrete classifications of land cover types represented as integer codes.	European Commission , http://bioval.jrc.ec.europa.eu/products/glc2000/data-access.php	Bartholome, E. M., & Belward A. S., GLC2000; a new approach to global land cover mapping from Earth Observation data, <i>International Journal of Remote Sensing</i> , 26 , 1959-1977 (2005). The Global Land Cover Map for the Year 2000, 2003.
CRU CL 2.0 Global gridded climate data	Monthly values of a range of environmental variables obtained by averaging over the period 1961-	Climatic Research Unit at the University of East Anglia http://www.cru.uea.ac.uk/cru/data/hrg/	New, M., Lister, D., Hulme, M. & Makin, I. A high-resolution data set of surface climate over global land areas. <i>Climate Research</i> 21 , 1-25 (2002).

	1990 and has a spatial resolution of 10 arc-minutes.		
Soil available water capacity data	Total available water capacity (mm water per 1 m soil depth) at 0.5 degree resolution	Oak Ridge National Laboratory Distributed Active Archive Centre (ORNL-DAAC). http://daac.ornl.gov/SOILS/guides/IsricGrids.html	Batjes, N. H. (ed.). 2000. Global Data Set of Derived Soil Properties, 0.5-Degree Grid (ISRIC-WISE). [Global Data Set of Derived Soil Properties, 0.5-Degree Grid (International Soil Reference and Information Centre - World Inventory of Soil Emission Potentials)]. Data set. Available on-line [http://www.daac.ornl.gov] from Oak Ridge National Laboratory Distributed Active Archive Center, Oak Ridge, Tennessee, U.S.A. doi:10.3334/ORNLDAAC
Global Vegetation Types, 1971-1982	Classification of vegetation types at 1 degree resolution	Oak Ridge National Laboratory Distributed Active Archive Centre (ORNL-DAAC). http://daac.ornl.gov/VEGETATION/guides/matthews_global_veg.html	Matthews, E. 1999. Global Vegetation Types, 1971-1982 (Matthews). Data set. Available on-line [http://daac.ornl.gov] from Oak Ridge National Laboratory Distributed Active Archive Center, Oak Ridge, Tennessee, U.S.A. doi:10.3334/ORNLDAAC/419.
Potential vegetation data	Classification of potential vegetation across the global land surface	Center for Sustainability and the Global Environment (SAGE), part of the Nelson Institute for Environmental Studies at the University of Wisconsin - Madison http://www.sage.wisc.edu/atlas/data.php?incdataset=Potential Vegetation	Ramankutty, N., and J.A. Foley (1999). Estimating historical changes in land cover: North American croplands from 1850 to 1992. <i>Global Ecology and Biogeography</i> 8, 381-396.

ProcessedRawData

This was used to hold the results of processing the raw-datafiles into a standard format. We do not have permission to redistribute all of these datafiles so this folder is blank. These files are produced by the MakeDataTables scripts

TrainingEvaluationData

The sample of the datafiles in the “RawData” folder used as training and evaluation data, with associated climatic data. We have permission to distribute this derived data from all data providers so all of the training and evaluation data is contained within that folder.

Part of the study of Smith *et al.* (2013) was to divide the raw data (in the *ProcessedRawData* folder) into training and test data. This involves using a random geographical mask to assign approximately 25% of the terrestrial land surface to final test data (the *cru20DataMask.nc* file in the *RawData* folder), with the remaining being training data. The training data is then assigned “fold” numbers and a weighting inversely proportional to its relative frequency of data from that type of climate in the data. Initially the test data has no modifications and is simply copied to corresponding files in the *Test* folder. However the final step in the study of Smith *et al.* (2013) was to assess model performance using this data, and it has been assigned associated climate data as a result of that process.

The *TrainingEvaluationData* folder contains two sorts of datafiles: the {DATA_CODENAME}SetData.csv files simply contain the subset of the data in *ProcessedRawData* that was selected as training/evaluation data with added fold numbers and the Holdridge zone classification (Holdridge, L. R. Life Zone Ecology. Tropical Science Centre, San Jose, Costa Rica, 1967). The {DATA_CODENAME}SetClimateData.csv files contain the same data but with added climate variables obtained by referring to the environmental datasets (Table 1). We recommend you open one of these files in DataSetViewer to inspect and explore the data (you can download DataSet Viewer from <http://research.microsoft.com/projects/sds>).

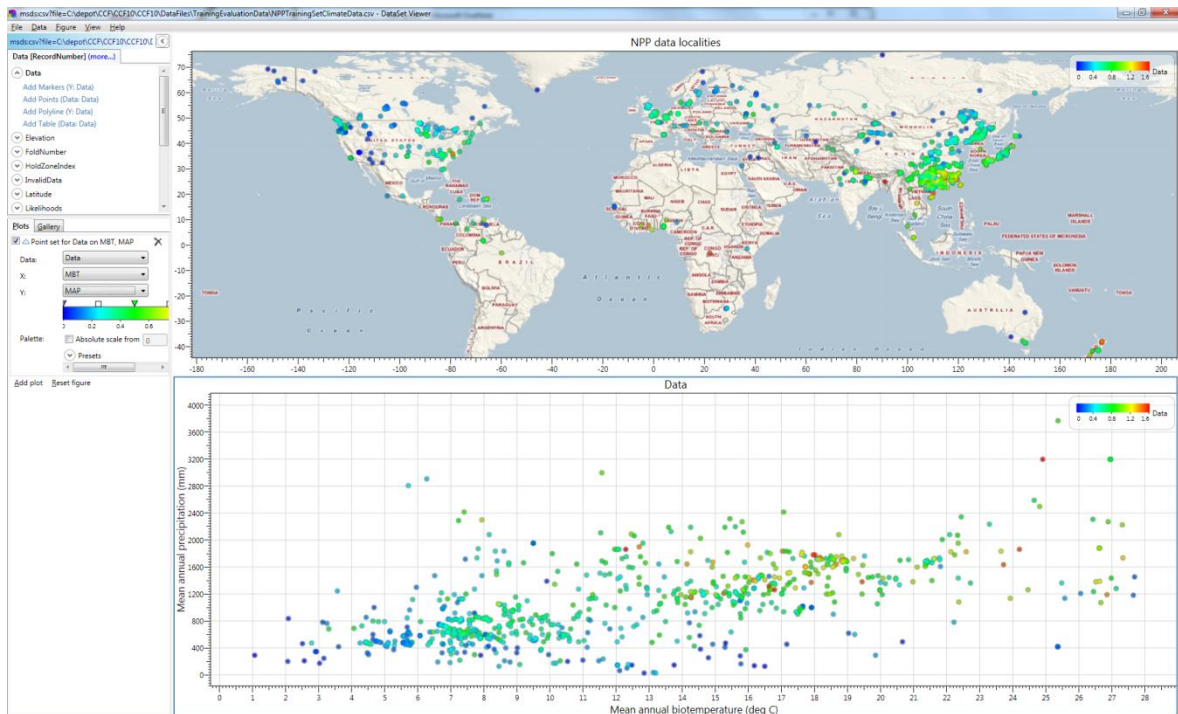


Figure 4 | Using Dataset Viewer to inspect the Net Primary Productivity (NPP) training and evaluation data. Top panel shows a global map of the sample localities. Bottom panel plots the mean annual biotemperature of localities against mean annual precipitation. The colour of the points indicates NPP ($\text{kg m}^{-2} \text{yr}^{-1}$).

ReservedTestData

The sample of the datafiles in “RawData” used as final test data, with associated climatic data. We have permission to distribute this derived data from all data providers so all of the test data is contained within that folder.

A note on data file format

Smith *et al.* (2013) used two main file formats for input and output data: NetCDF (extension “.nc”) and CSV (extension “.csv”). CSV file format is used for containing files with arbitrary numbers of 1 or 2 dimensional data arrays because it can be conveniently read using several commonly used programs (Notepad, Excel and DataSetViewer). NetCDF file format is more convenient for handling N-dimensional data structures (e.g. 2-dimensional space through time), which are also usually accompanied by large file sizes (say, >50Mb).

9. Running the fully data-constrained global terrestrial carbon model from command line arguments

A compiled 32 bit version of our code is included in the YOUR_ROOT_DIRECTORY/Package/bin folder. You can set this running by navigating to this folder using your console and typing “**MSRTCM**”. However we recommend initially that you run

```
MSRTCM FULL -test
```

as described below. Other command line arguments are explained in the subsequent sections.

10. Repeating the methods used in the paper

Implementing the methods used in the study of Smith *et al.* (2013) can be done with certain run-time flags, or commands. The commands come from a "command" string in the Main program function, or they can be specified as command line parameters when you invoke the program. The former method is more convenient when you start the program from Visual Studio. The latter method aids the implementation of parametric sweep jobs on a computation cluster or implementing the code independently from Visual Studio.

We recommend that you begin with all of the original data files in place because i) the code should definitely run if it is provided these data files, ii) it takes less time for you to see the parameter estimation algorithms in process and iii) these are the exact data files that were used to generate the results of Smith *et al.* (2013).

If you do not begin with the original data files then the code will look for the raw data files and build new training and test datasets. The code will throw an exception during this process (i.e. crash) if it cannot find the original data files, or find them in the correct format. To obtain the raw data files please refer to the sources above (Table 1) or contact the authors for assistance (we cannot guarantee that these original source data files will always be available from the initial locations).

We will now describe the most typical ways in which a user will use our system to recreate our results.

NOTE: Fitting all but the simplest models using 10-fold cross validation with multiple components can take minutes to hours and running all the experiments and simulations performed by Smith *et al.* (2013) would take days or even weeks on a single processor with a standard personal computer. It will therefore be more practical to run the different experiments in stages. Moreover, you can restrict the Markov Chain length used for parameter approximation initially to verify that all of the

different procedures run and produce results (although the results themselves will be useless). This can be done by specifying `-mcmc:10,1000` in the relevant command string

10.1. FULL

Fitting the full Microsoft Data-Constrained Model of Global Vegetation

The first experiment we run is to fit the full model. This partly serves to verify that all of the model components are set up correctly. The command line command is:

```
MSRTCM FULL
```

or

```
MSRTCM FULL -mcmc:10,1000
```

or

```
MSRTCM FULL -test
```

Note that the latter two commands are equivalent. Use the latter command if you simply want to check it is working right (this restricts the Markov Chain length to 10 burn-in steps and 1000 sampling steps) - we recommend you run this first just to make sure that the code is working fine.

Through Visual Studio you can implement the same commands through specifying "FULL" or "FULL -test" for the "command" string:

```
static void Main(string[] args)
{
    string command = "FULL -test";
```

The computational framework will then perform the following operations

- Identify all of the names of all sub-component models for the full model.
- Add to a "SetOfModels" class all of the subcomponents including: The name; the data distribution type (e.g. normal, lognormal, logistic); the function to initialize the parameters to be estimated in Filzbach; the function used to make predictions given parameters and data; the function used to estimate the error about the predictions; the function used to make the datafiles from the raw data.
- If the training datafiles do not exist for all of the models then remake the training datafiles from the raw data (using the function identified previously). This involves transforming

and/or sampling the original source data files and dividing into training/evaluation data and final test data, classifying each location into a Holdridge Life Zone and assigning fold numbers to the data points.

- For each model component initialize all of the parameters to be estimated.
- For each fold perform Markov Chain Monte Carlo estimation of the parameter space to estimate the parameter probability distributions.
- Post-process the results of 10-fold model parameter estimation.

This produces the following files in the DataSets folder

Table 2 | Output data from the FULL model fitting procedure

File name and location DataFiles\	Description	Variables
OutputData\ModelFitting OutputData\LastOutput.csv	Updated after every training fold has completed providing an opportunity to visually inspect plots of predictions versus observations (this was useful in debugging)	Shows the last set of predictions for every set of observations from the last iteration of training.
OutputData\ModelFitting OutputData\FullModelResultsCompilation.csv	A compilation of results from N-Fold parameter estimation for the full model. This is a key file allowing for visual inspections of parameter values, summary statistics and performance metrics for training and evaluation data. It contains the raw Markov Chains for each parameter value, the median and 95th percentile for each fold of each parameter and the mean of that data, model component performance assessment metrics by fold and their means, examples of predictions versus observations.	<p>For each parameter</p> <p>Samples from each Markov Chain</p> <p>Median 5th and 95th percentile credibility intervals for each fold and averages across folds</p> <p>Parameter probability distributions from each fold and on average</p> <p>Prior parameter settings</p> <p>For each model, each fold, for each of the sampled parameter values given the Training (TL) or Evaluation (VL) data</p> <p>Likelihoods given sampled parameter values</p> <p>Probability distributions for the likelihoods</p> <p>Correlation coefficient (CC)</p>

		<p>Coefficient of Determination (CD)</p> <p>Mean root-mean squared error (MRMSE)</p> <p>Mean relative error (MRE)</p> <p>Mean coefficient of variation (CV)</p> <p>Deviance information criterion (DIC-training data only)</p> <p>Mean 5th and 95th percentiles of the above metrics for each fold and averaged across folds</p> <p>Mean 5th and 95th percentiles of the model predictions for each data point given the sampled Markov Chain</p> <p>A copy of the empirical data and count of the number of training and validation datapoints</p>
OutputData\ProcessedFittingOutputData\ProcessErrorValues.csv	A compilation of the process error values inferred for each model. This is used to estimate the likelihoods for the evaluation data in the data-omit parameter inference experiments.	The mean median estimate of the process error parameter for each model component in the full model

We found it convenient to visualize the results in these files using Dataset Viewer. It allows us to rapidly inspect the parameter probability distributions and performance metrics for multiple models (e.g. Fig. 5)

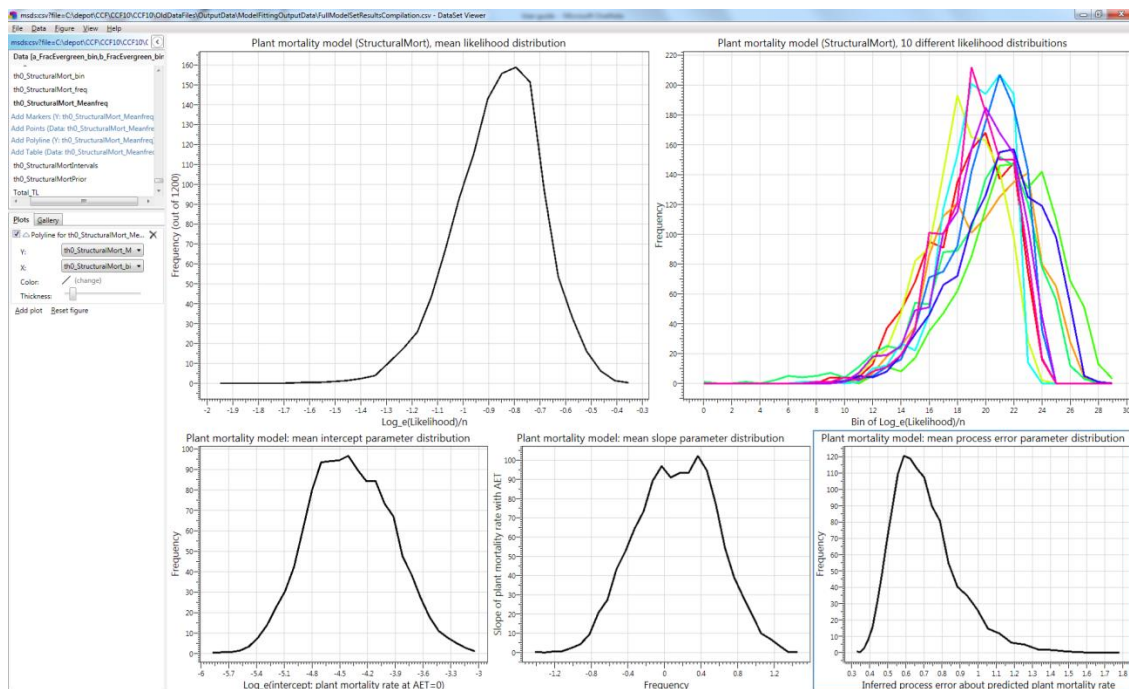


Figure 5 | Using Dataset Viewer to inspect the inferred probability distributions for the plant mortality model likelihood (using evaluation data) and parameters.

10.2. MAPS

Map the predictions of equilibrium carbon stocks and flows for the global land surface and simulate a global re-vegetation event

This functionality is provided simply to produce predictions from the model for the global land surface at 0.5 degree resolution. It completes two main operations - the first is to solve the equilibrium equations for the global land surface. The second is to initialize all the carbon pools across the global land surface at the same out-of-equilibrium values and simulate 100 years of dynamics under constant climate conditions at each site.

The maps require the OutputData\ModelFittingOutputData\FullModelResultsCompilation.csv datafile to have been produced by FULL model parameter estimation. Then the maps can be produced by the command line command

MSRTCM MAPS

Or if you want to produce maps immediately after parameter estimation you can write

MSRTCM FULL MAPS

Alternatively you can alter the "command" string in the Main function of Program.cs to "MAPS" or "FULL MAPS" using Visual Studio

Table 3 describes the results of running the MAPS procedure.

Table 3 | Output data from mapping the equilibrium carbon stocks and re-vegetation

File name and location	Description	Variables
DataFiles\ OutputData\SimulationOutputData\EquilibriumMapForFullModelSet.csv	Contains results of estimating equilibrium carbon stocks and flows for the global land surface at 0.5 degree resolution.	All carbon stocks and flows for all land points with accompanying latitude and longitude coordinates
OutputData\SimulationOutputData\SimulationMapForFullModelSet.csv	Contains results of simulating the recovery of equilibrium carbon stocks from low levels over a 100 year time period under constant climate conditions at 0.5 degree resolution.	Plant and soil carbon for all land points through time (100 years)

We find it convenient to inspect the results in these datafiles using DataSet Viewer (e.g. Fig. 6)

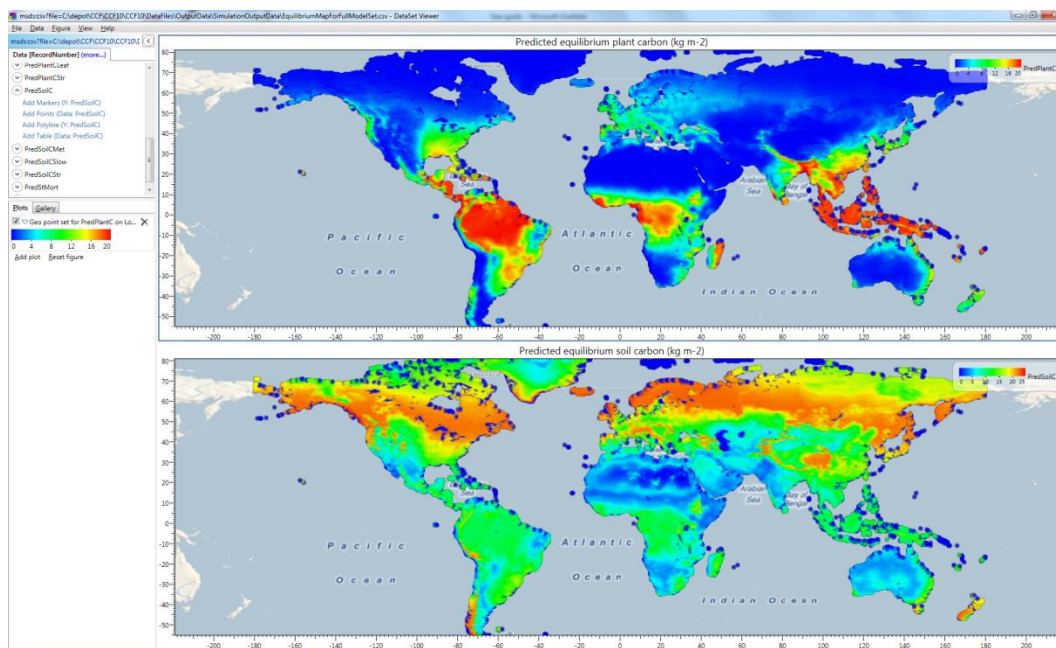


Figure 6 | Using DataSet Viewer to inspect the predicted maps of equilibrium plant (top) and soil carbon (bottom) using the full model.

10.3. EQCARBDIST

Map the probabilistic predictions of equilibrium carbon stocks for the global land surface at 10 arc minute spatial resolution (used to produce Fig. 2 of the manuscript)

A useful feature of the global terrestrial carbon model is that it enables probabilistic of equilibrium carbon stocks and flows to be made for anywhere on earth. The EQCARBDIST routine makes such predictions for every land surface point on earth at 10 arc-minute resolution (approx. 18km), outputting maps of plant and soil carbon in terms of the mean median prediction and 5th and 95th percentiles over 10-folds of parameter inference.

The maps require the OutputData\ModelFittingOutputData\FullModelResultsCompilation.csv datafile to have been produced by FULL model parameter estimation. Then the maps can be produced by the command line command

MSRTCM EQCARBDIST

Or if you want to produce maps immediately after parameter estimation you can write

MSRTCM FULL EQCARBDIST

Alternatively you can alter the "command" string in the Main function of Program.cs to "EQCARBDIST" or "FULL EQCARBDIST" using Visual Studio

This procedure first produces a dataset called

OutputData\SimulationOuputData\EnvironmentsHighResBackup.csv using the New *et al* (2002) and Batjes, N. H. (ed.). 2000 datasets (Table 1) if they do not already exist in that folder. A copy of that file is packaged with the software. The code then calculates equilibrium plant and soil carbon for each of the 1200 parameter samples from the 10 markov chains in FullModelResultsCompilation.csv. This procedure takes several hours on a reasonably fast processor.

Table 4 describes the results of running the MAPS procedure.

Table 4 | Output data from making probabilistic maps of terrestrial plant and soil carbon

File name and location DataFiles\	Description	Variables
OutputData\SimulationOutputData\ HighResGridsSoilL95.csv HighResGridsSoilMed.csv HighResGridsSoilU95.csv HighResGridsPlantL95.csv HighResGridsPlantMed.csv HighResGridsPlantU95.csv	Contains 2 dimensional grid representations of the median 5 th and 95 th percentile estimates of plant and soil carbon	Either of the 5 th , 95 th or median estimates for plant or soil carbon with accompanying latitude and longitude coordinates

OutputData\SimulationOutputData\ HighResEnvironmentsColumnsSoil.csv HighResEnvironmentsColumnsPlant.csv	As above but rearranging the data into columns and combining medians with credibility intervals.	Combined the 5 th , 95 th and median estimates for plant or soil carbon with accompanying latitude and longitude coordinates
OutputData\SimulationOutputData\HighResEnvironmentsMapForFullModelSet.nc	As above but all data combined into a single NetCDF file.	Combined the 5 th , 95 th and median estimates for plant and soil carbon with accompanying latitude and longitude coordinate

These results can be inspected using Dataset Viewer or plotted using the statistical package R with the code in the using the ext\RScripts\MainManuscript\Fig2.R script

10.4. BUILD-UP <n>

Build-up parameter estimation experiments

Performs the parameter estimation operations described in FULL above, but for subsets of the model structure. The command line command is

MSRTECM BUILD-UP ALL

Or alternatively you can specify BUILD-UP ALL in the "command" string in the Main function of Program.cs using Visual Studio. This command will result in 10-fold parameter estimation for all substructures in the full model. Alternatively you can estimate the parameters for a particular substructure by specifying a number after BUILD-UP. For example,

MSRTECM BUILD-UP 1

Will result in the NPP model being fit. The integers correspond to the following sub-model structures

This functionality is useful if you want to distribute the model fitting experiments on a computer cluster (as we did).

Table 5 | Integer codes for performing parameter inference on different subsets of the model structure

Experiment Number	Models in the experiment
-------------------	--------------------------

[1]	NPP
[2]	FracEvergreen
[3]	LeafMortEvergreen
[4]	LeafMortDeciduous
[5]	FRootMort
[6]	StructuralMort
[7]	FracAlloc
[8]	NPP + Fire
[9]	NPP + FracStruct
[10]	FracAlloc + FRootMort + LeafMortEvergreen + LeafMortDeciduous + FracEvergreen + StructuralMort + NPP + Fire + FracStruct + PlantC
[11]	[10] + LitterTot
[12]	[10] + LitterTot + SoilC

Completion of a BUILD-UP parameter estimation experiment results in a `OutputData\ModelFittingOutputData\ModelSet<n>ResultsCompilation.csv` file, where n is the integer code corresponding to the model fitting experiment. It contains the same parameter and model performance summary as the `OutputData\ModelFittingOutputData\FullModelResultsCompilation.csv` file detailed in Table x above, but for the specific BUILD-UP parameter estimation experiment.

10.5. ALL-DUMMY <n>

Perform the BUILD-UP fitting experiments but where every model component is replaced with a DUMMY (a null-model)

Performs the parameter estimation operations described above but where every model component is replaced with one having a single parameter for estimation of the empirical data plus a process error parameter. This is useful for comparing the performance of the models fitted above to that of a null model. The commands and outcomes are exactly as in the BUILD-UP experiments but for the DUMMY models. The command line to fit all of the dummy model experiments is

MSRTCM ALL-DUMMY ALL

Or for a single experiment (see code above) replace ALL with an integer corresponding to the experiment (see Table 5)

MSRTCM ALL-DUMMY 1

Or alternatively you can specify ALL-DUMMY ALL or ALL-DUMMY {Experiment Number} in the "command" string in the Main function of Program.cs using Visual Studio.

Completion of a ALL-DUMMY parameter estimation experiment results in a OutputData\ModelFittingOutputData\DummyModelSet<n>ResultsCompilation.csv file, where n is the integer code corresponding to the model fitting experiment.

10.6. ONE-DUMMY <n>

Perform parameter estimation for the full model but with one model component replaced with a DUMMY (a null model)

Performs the parameter estimation operations described above for the full model but when a specific model component has been replaced with one having a single parameter for estimation of the empirical data plus a process error parameter. The command line to perform this sequentially for all model components is

MSRTCM ONE-DUMMY ALL

Or for a single experiment (see code above) replace ALL with an integer corresponding to the experiment (see Table 6)

MSRTCM ONE-DUMMY 1

Or alternatively you can specify ONE-DUMMY ALL or ONE-DUMMY {Experiment Number} in the "command" string in the Main function of Program.cs using Visual Studio.

The integers correspond to the following model being replaced by a dummy:

Table 6 | Integer codes to use to specify which model to replace with a dummy

1 – FracAlloc	2 - FRootMort	3 - LeafMortEvergreen	4 - LeafMortDeciduous
5 - FracEvergreen	6 - StructuralMort	7 - NPP	8 - Fire
9 - FracStruct	10 - LitterTot	11 - PlantC	12 - SoilC

Completion of a ONE-DUMMY parameter estimation experiment results in a OutputData\ModelFittingOutputData\FullModelSetReplace<ModelOmitted>ResultsCompilation.csv

file , with <ModelOmitted> corresponding to the specific model component that had been replaced by a null model.

10.7. OMIT-DATA<n>

Perform parameter estimation for the full model but omitting an entire empirical dataset each time

Performs the parameter estimation operations described above for the full model but with a specific dataset omitted during the parameter estimation procedures. The command line to perform this sequentially for all datasets is

MSRTCM OMIT-DATA ALL

Or for a single experiment (see code above) replace ALL with an integer corresponding to the experiment (see Table 7)

MSRTCM OMIT-DATA 1

Or alternatively you can specify OMIT-DATA ALL or OMIT-DATA {Experiment Number} in the "command" string in the Main function of Program.cs using Visual Studio.

The integers correspond to the following datasets being removed:

Table 7 | Integer codes to use to specify which dataset to remove when inferring the parameters for the full model

1 - FracAlloc	2 - FRootMort	3 - LeafMortEvergreen	4 - LeafMortDeciduous
5 - FracEvergreen	6 - StructuralMort	7 - NPP	8 - Fire
9 - FracStruct	10 - LitterTot	11 - PlantC	12 - SoilC

Completion of an **OMIT-DATA** parameter estimation experiment results in a OutputData\ModelFittingOutputData\NFoldOmitSpecific<ModelOmitted>ResultsCompilation.csv file , with <ModelOmitted> corresponding to the specific dataset that had been removed during model training (although it is still used in model evaluation).

10.8. ANALYSE-PARAMS

Analyses the results of the BUILD-UP, ONE-DUMMY and OMIT-DATA parameter estimation experiments

To run this procedure you will need to have run FULL and at least one complete set of the BUILD-UP, ONE-DUMMY or OMIT-DATA parameter estimation experiments (each of these produces 12

*ResultsCompilation.csv files). These files must be in the OutputData\ModelFittingOutputData\ folder. Table 8 summarizes these requirements and what ANALYSE produces using these files

Table 8 | Output data from analyzing the outputs of the model parameter inference experiments

Experiments required	Files produced	Description
FULL, BUILD-UP (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ProcessedLikelihoodsBuildUpVL.csv, ProcessedLikelihoodsBuildUpTL.csv	Assembles and summarizes the model performance assessment metrics for the training (TL) and evaluation datasets (VL) arising from the 12 BUILD-UP experiments.
FULL, BUILD-UP (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ProcessedParametersBuildUp.csv	Assembles and summarizes the inferred parameter values arising from the 12 BUILD-UP experiments.
FULL, BUILD-UP (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ProcessedPredictionsBuildUp.csv	Assembles and summarizes predictions versus observations plots arising from the 12 BUILD-UP experiments.
FULL, BUILD-UP (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ExampleOutputsBuildUp.csv	Produces component functions using posterior parameter probability distributions arising from the 12 BUILD-UP experiments.
FULL, ONE-DUMMY (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ProcessedLikelihoodsReplaceDummyVL.csv, ProcessedLikelihoodsReplaceDummyTL.csv	Assembles and summarizes the model performance assessment metrics for the training (TL) and evaluation datasets (VL) arising from the 12 BUILD-UP experiments.
FULL, OMIT-DATA (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ProcessedLikelihoodsOmittedVL.csv, ProcessedLikelihoodsOmittedTL.csv	Assembles and summarizes the model performance assessment metrics for the training (TL) and evaluation datasets (VL) arising from the 12 OMIT-DATA experiments.
FULL, OMIT-DATA (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ProcessedParametersOmitted.csv	Assembles and summarizes the inferred parameter values arising from the 12 OMIT-DATA experiments.
FULL, OMIT-DATA (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ProcessedPredictionsOmitted.csv	Assembles and summarizes predictions versus observations plots arising from the 12 OMIT-DATA experiments.
FULL, OMIT-DATA (all 12), ALL-DUMMY (all 12)	OutputData\ProcessedFittingOutputData\ExampleOutputsOmitted.csv	Produces component functions using posterior parameter probability distributions arising from the 12 OMIT-DATA experiments.

10.9. SIMULATE

Simulates the full model using climate data from different climate model simulation outputs and different parameter values for the plant mortality model

The simulation experiments conducted in the study of Smith *et al.* (2013) were performed using separate code to that of the prototype framework for model engineering and refinement. To conduct the simulations conducted by Smith *et al.* (2013) you will need to download the necessary .cs file to run the simulations, insert it into the source code of MSRTCM and then recompile the code. Specifically

- Download the MSRTCMSim.zip package from <http://research.microsoft.com/en-us/downloads/49ad471e-7411-4f65-910a-2a541f946575/default.aspx>
- Unzip the package and find the ClimateChangeSimulatorImpl.cs file in the YOUR_ROOT_DIRECTORY/MSRTCMSim/src/ProcessResultsDatafiles folder.
- Replace the ClimateChangeSimulatorImpl.cs file in the YOUR_ROOT_DIRECTORY/MSRTCM/src/ProcessResultsDatafiles folder with that datafile
- Recompile the MSRTCM.exe solution.

The climate change prediction data processed using the new code, which were used to force the model under changing climate scenarios, were obtained from the following source

Table 9: Climate data source used in the climate change simulations of Smith *et al.* (2013)

Data set name	Data set description	Data source	Citation
Simulation outputs from the HadCM3 model for the AR4 SRES scenarios	Predicted monthly values of environmental variables for the surface of the earth gridded at a 2.5x3.75 degree resolution from the year 2000 through to 2199 for the AR4 A1F1 and B1 anthropogenic emissions	The IPCC Data Distribution Centre. <i>AR4 GCM Data.</i> http://www.mad.zmaw.de/IPCC_DDC/html/SRES_AR4/index.html http://cera-www.dkrz.de/WDCC/ui/Compact.jsp?acronym=UKMO_HadCM3_SRESA1B_1 http://cera-	Lowé, 2005: IPCC DDC AR4 UKMO-HadCM3 SRESA1B run1. World Data Center for Climate. CERA-DB "UKMO_HadCM3_SRES A1B_1" Lowé, 2005: IPCC DDC AR4 UKMO-HadCM3 SRESB1 run1. World Data Center for Climate. CERA-DB "UKMO_HadCM3_SRES

	scenarios.	www.dkrz.de/WDCC/ui/Compact.jsp?acronym=UKMO_HadCM3_SRESB1_1	B1_1"
--	------------	--	-------

The downloadable package also contains the results of our simulation experiments. These are contained in the folder

YOUR_ROOT_DIRECTORY/MSRTCMSim/data/OutputData/SimulationOutputData

The simulation experiments were the process that took up the most compute-time in the study of Smith *et al.* (2013). A simulation was performed for each sample of the Markov Chain (1200 samples) for each model training fold (10 folds) under two different climate change scenarios (2 scenarios) and for 3 different parameterizations of the plant mortality model (3 mortality models). This equates to $1200 \times 10 \times 2 \times 3 = 72,000$ simulations. Each simulation took a couple of minutes on a reasonably fast computer however to complete all simulations we divided the jobs by fold, scenario and mortality model. In other words we simulated separately each lot of 1200 samples of parameter values.

The procedures involved in conducting the simulations are

- Set up an instance of the full model: adds to a "SetOfModels" class all of the subcomponents including: The name, the data distribution type (e.g. normal, lognormal, logistic), the function to initialize the parameters to be estimated in Filzbach, the function used to make predictions given parameters and data, the function used to estimate the error about the predictions, the function used to make the datafiles from the raw data.
- Initialize the parameters in Filzbach
- Read in the previously estimated parameter values from OutputData\ModelFittingOutputData\FullModelResultsCompilation.csv
- If an instruction has been given to get different parameters for the mortality model (details below) then replace those parameters
- Create a file containing average environmental variables for all terrestrial land points at 0.5 degree resolution using the CRU CL 2.0 Global gridded climate data dataset - if it doesn't already exist
- Create a file containing the differences to apply to the above environmental variables under a specific climate change scenario (details below) if it doesn't already exist.

- Simulate the model for each parameter set in FullModelResultsCompilation.csv, saving the results in \OutputData\SimulationOutputData\OutputExperiment<scenario><Mortality Model><FoldNumber>.csv

Once the solution has been re-built to allow simulations to be performed, the command for specifying a particular simulation to run is

MSRTECM SIMULATE <FoldNumber> <Scenario> <MortalityModel>

Where

- FoldNumber is an integer from 1-10,
- Scenario is "A1F1" or "B1", corresponding to the A1F1 climate change scenario or the B1 scenario respectively, and
- Mortality model is "NM", "M" or "ZM" - corresponding to not replacing the mortality model parameters, replacing the parameters with those in OutputData\ModelFittingOutputData\ModelSet6ResultsCompilation.csv, or replacing the parameters with those in OutputData\ModelFittingOutputData\FullModelSetReplaceStructuralMortResultsCompilation.csv file, respectively

For example,

MSRTECM SIMULATE 3 A1F1 NM

Simulates the full model using parameters from fold number 3 using the A1F1 climate change scenario and using the inferred mortality model parameters for the full model. Alternatively you can specify the commands in the "command" string in the Main function of Program.cs using Visual Studio.

We performed all of the simulations using a computer cluster through a cluster job manager.

Each job creates a datafile called

YOUR_ROOT_DIRECTORY/Package/OutputData/SimulationOutputData/OutputExperiment<scenario><Mortality Model><FoldNumber>.csv containing the estimated global plant and soil carbon pools as well as a detailed breakdown of carbon in different pools through time for 6 different spatial locations.

10.10. ANALYSE-SIMULATIONS

Analyses the simulations arising from the SIMULATE command

This procedure was used to combine the simulation results for the different training data folds to produce 6 summary results files: 3 different mortality model parameterizations x 2 scenarios. For each combination of scenario and mortality model you need to have all 10 OutputExperiment<scenario><Mortality Model><FoldNumber>.csv files in the YOUR_ROOT_DIRECTORY/Package/data/OutputData/SimulationOutputData folder.

The procedure checks each scenario-mortality model combination and if all 10 corresponding OutputExperiment files exists it produces a YOUR_ROOT_DIRECTORY/Package/data/OutputData/SimulationOutputData <scenario><Mortality Model>Processed.csv file, containing mean, median and 95th percentiles of the model predictions across the 10 sets of parameter values.

The simulation analysis files produced during the Smith *et al.* (2013) study are in the MSRTCMSimData.zip package available from <http://download.microsoft.com/download/1/F/D/1FD1F550-69C4-4503-B2FE-B47F94607A7F/MSRTCMSIMData.zip>

10.11. FINAL-TEST

Assesses the predictive performance of the full model using the final test data

It is good practice to only perform this final step once the full model has been finalized, as we did in the study of Smith *et al.* (2013).

This procedure reads in the inferred parameter distributions for the full model from YOUR_ROOT_DIRECTORY/Package/data/OutputData\ModelFittingOutputData\FullModelResultsCompilation.csv and uses them to predict the data held in the ReservedTestData folder. This firstly results in the compiled model performance assessment metrics in the YOUR_ROOT_DIRECTORY/Package/data/OutputData/ProcessedReservedTestOutputData/TestDataResultsCompilation.csv file. It is then post-processed to result in the YOUR_ROOT_DIRECTORY/Package/data/OutputData/ProcessedReservedTestOutputData/ProcessedLikelihoodsFullModelTest.csv file which is used when graphs are plotted using R.

11. R Scripts to produce final publication graphs

We used the statistical package R to produce some of the final graphs for Smith *et al.* (2013) and include the scripts that did this in the

YOUR_ROOT_DIRECTORY/Package/ext/RScripts folder. The files are divided into those used for the main manuscript (in the "MainManuscript" folder) and those used in the supplementary information (in the "SupplementaryInformation" folder). They do not use any additional libraries and so should work with most versions of R. You will need to alter the scripts to refer to the correct file path containing the source datafiles. The function of the scripts is obvious from the file names.

The script used to produce Fig. 9 of the main manuscript is included with the simulation output data and simulation code in the MSRTCMSim.zip package available from

<http://research.microsoft.com/en-us/downloads/49ad471e-7411-4f65-910a-2a541f946575/default.aspx> (the file is Fig9.R). The file is

YOUR_ROOT_DIRECTORY/MSRTCMSim/ext/RScripts/MainManuscript/Fig9.R.

12. Conducting novel studies

We strongly encourage scientists to work with our code to conduct novel studies. At present we cannot promise dedicated technical support for this although please do email matthew.smith@microsoft.com with queries. Please include MSRTCM SUPPORT in the subject line of the email.

At present users will have to work with the raw code to conduct novel studies. We anticipate that most users will want to work with the automated parameter estimation capabilities of the code. We therefore highlight the key elements of the code that you may need to change in order to implement a new model.

- Specifying a new model component. Examples of how model components are specified are in the "OriginalCarbonStocksFlowsModels" folder. Different models were specified as different object-oriented classes with specific fields to store parameter values. We recommend users look at `MiamiNPPModel.cs` to see a detailed breakdown for a specific model.
- Formatting the model component for Filzbach. In order for parameter estimation to be performed on specific model components we write a class that handles the interface between Filzbach and the model component. Examples are in the "ModelsFormattedForFilzbach" folder. This contains
 - a "SetupParameters()" function that initialises parameter values in Filzbach,
 - a MakePrediction() function that makes predictions for a list of sites by obtaining the required climate data or predictions from another component, setting up an

instantiation of the model object with the current parameter values in Filzbach, and then making predictions for each site using the model prediction functions.

- An ErrorFunction() that predicts the process error associated with the predictions
- "Dummy" functions - which are alternatives to a, b and c for implementing a null model.
- Formatting a new source dataset for use in parameter inference. Source data can vary considerably in format and so we generally found that we had to write a separate function for reading in each source datafile. There are a range of examples of this in the "MakeDataTables" folder. Ultimately the code must result in the production of a datafile containing latitude, longitude and the data to be predicted - with the inclusion of elevation data optional. These are combined into one datafile using the ClimateLookup.CombineDatasets() function.
- Registering the model component as one for parameter estimation. At present this is handled by the MakeSetOfModels() or MakeSetOfModels2() functions in the CCFittingStudy.cs class. These add model components to a SetOfModels class that stores a list of model components for parameter inference. If these functions are passed a string array containing the name of a model in their list then they attempt to add the model component to the set of models. The difference between the two functions is that MakeSetOfModels() registers the normal model component whereas MakeSetOfModels2() registers the null model for the component. To add a new model name you can specify

```
if (ModelsToInclude.Contains("YOUR_NEW_MODEL"))
    NewSetOfModels.AddModelToModelSet("YOUR_NEW_MODEL",
    <StringOfDataDistributionType>, YourNewModelClass.SetupParameters,
    YourNewModelClass.MakePrediction, YourNewModelClass.ErrorFunction,
    YourNewModelClass.ProcessData);
```

- Decide on the type of fitting. Fitting a single multi-component or single model is best implemented by modifying the FullModel() function in the Program.cs file. This simply outputs an array of strings indicating the model components to be used for parameter inference. If you want to fit a sequence of model structures then this is best implemented by modifying the IdentifyIndividualModelExperiments() function in the Program.cs file. This specifies a list of string arrays representing different combinations of model components.

- The likelihood function used for different models is specified in the LikelihoodAnalysis.cs file. These return log-likelihood values as well as other variables given data and the results of the prediction equation from the parameterised model for an assumed data distribution type (normal, lognormal or logistic). Calls to the likelihood functions are orchestrated by the CalculateLikelihoodFilzbach() function in the SetOfModels.cs class.
- The different model fitting experiments are called in the Program.cs file but are specified in the NFoldFitting() function of the CCFFittingStudy.cs class.
- The different model performance assessment metrics are called from the CalculateStatisticsandAddToFold() function of the SetOfModels.cs class although the statistics themselves are specified in the MakeSummaryStatistics.cs file.

13. Using the Dmitrov libraries within the code

We use the Dmitrov (also known as Scientific DataSet) libraries to manage the use of datasets throughout our C# code; we developed the software to facilitate the use of multidimensional datasets in diverse formats and sizes from within code. We included libraries from one specific version of Dmitrov - version 1.2.12907. The use of this version outside of the MSRTSM solution will not be supported by the Dmitrov team. The full version of project Dmitrov libraries and tools can be obtained from <http://research.microsoft.com/projects/sds>.

14. Obtaining DataSet Viewer

We find it especially convenient to view model inputs and outputs using DataSet Viewer. We developed DataSet Viewer as a simple standalone menu-driven tool for quickly exploring and comparing time series, geographic distributions and other patterns within scientific data. DataSet Viewer combines selection, filtering and slicing tools, with various chart types (scatter plots, line graphs, heat maps, as well as tables), and geographic mapping (using Bing Maps). It is freely available as part of the Dmitrov tools and utilities package available from

<http://research.microsoft.com/en-us/um/cambridge/groups/science/tools/dmitrov/dmitrov.htm>

15. Using Filzbach

Filzbach is a flexible, fast, robust, parameter estimation engine that allows you to parameterize arbitrary, non-linear models, of the kind that are necessary in biological sciences, against multiple, heterogeneous data sets. Filzbach allows for Bayesian parameter estimation, maximum likelihood analysis, priors, latents, hierarchies, error propagation and model selection, from just a few lines of code. It includes a set of libraries to allow interoperability between C# and Filzbach, which we use in

this study and these are included with the package. Please consult the Filzbach user manual to obtain full details of how to use Filzbach or see <http://research.microsoft.com/filzbach/>.

16. Using FetchClimate

FetchClimate is a set of libraries and web service to facilitate access to various climatic datasets. The climate data for our study was not obtained through FetchClimate. Instead we used a local copy of the New *et al.* (2002)¹⁸ gridded monthly climate dataset. However we have also enabled access to exactly the same dataset using our FetchClimate data service, which returns exactly the same data. This can be implemented by including `-fetchclimate:true` when running the program from command line, or by altering the UseFetchClimate setting to True in the solution properties window in Visual Studio.

We include FetchClimate as a prototype of how users might obtain standard environmental or other datasets through a cloud-based data provider (in this case run in Azure) to avoid the burden of having to have local copies of all the necessary files. All of the calls to the FetchClimate data service are contained in the ClimateLookup.cs file in the MakeDataTables folder. Please consult the FetchClimate user manual to obtain full details of how to use FetchClimate or see <http://research.microsoft.com/fetchclimate/>.