Supplementary information

Estimating divergent carbon stocks and sinks via a simple hybrid algorithm approach

In the format provided by the authors and unedited

Supplementary Information

Supplementary Methods: The principle and practical guideline of the simple hybrid algorithm (SHA) in R programming language via cluster or cloud computing approach

1. Introduction

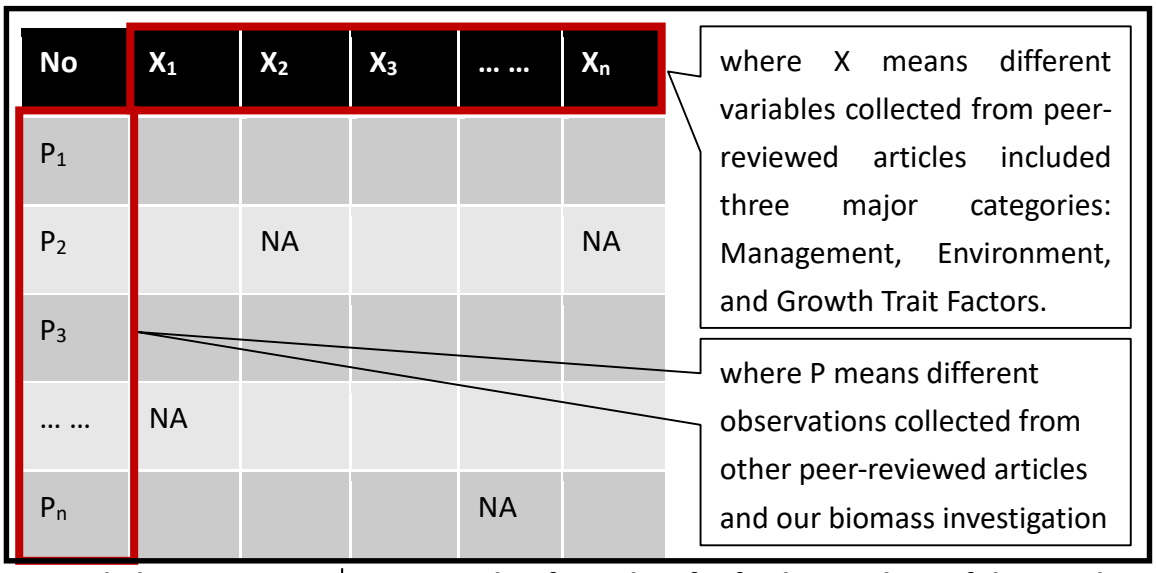
Target variables in the forest and natural environment are easily diverging and irregular, although the other target variables can be estimated via simple linear regression approaches. Furthermore, many studies have their own research aims, so the existing studies in the world often have different independent variables and the data sheets have the characteristics contained quite a lot of missing values. For conquering problems mentioned above, we design a simple hybrid algorithm (SHA) that contains multitaskable functions and can be operated easily. The flowchart of the SHA is described in the Figure 4 of the main text of "Estimating divergent carbon stocks and sinks via a simple hybrid algorithm". Below we elaborate a very simple and low-cost method to practice the process of the algorithm that combines linear regression, machine learning and climate classification tables to ensure that policymakers of any age in any country can fully understand and operate this algorithm by themselves easily.

2. Pre-treatment and Environment setting

2.1. Raw data collection

Raw data collected from google scholar, Nature publish group, Science (American Association for the Advancement of Science, AAAS), ScienceDirect and so on for English peer-reviewed journal articles. However, not all critical and reliable data are published in English. Therefore, according to different species that need to be estimated, look for peer-reviewed articles in different languages and countries, and it is better to have local native speakers in the research team to correctly understand and collect data. For instance, *Phyllostachys edulis* (Moso bamboo) forests are mainly distributed in China, Japan, South Korea and Taiwan, so our research team has native speakers of simplified Chinese, Japanese, Korean, and traditional Chinese. Further, we collected local peer-reviewed research articles from CNKI[1], J-stage[2], KISS[3], and airiti Library[4].

[1] Source: https://h5.cnki.net/#/home
[2] Source: https://www.jstage.jst.go.jp/browse/-char/ja
[3] Source: http://kiss.kstudy.com/search/sch-result.asp
[4] Source: https://www.airitilibrary.com/Search/alJnlbrowse



Extended Data Figure 1 | Frameworks of raw data for further analysis of the simple hybrid algorithm.

After collecting peer-reviewed articles, carefully read articles, input data in an excel file, save it as .csv for further analysis, and double check for prevent mistyping. The raw data used in this study is as Table S1, and the framework of template shown as Extended Data Figure 1. At the meanwhile of reading journal researches and making raw data sheet, record or define all variables in a table. (For instance, in the case of this research, common terms of variables defined as Table S2.)

2.2. Pre-treatment of *k*-fold cross validation

k-fold cross validation is a common method used to evaluate machine learning models on a limited raw datasheet. The method is to randomly divide the data into k sets, and then regard one set as testing data, and the rest of *k*-1 sets as training data, and repeat this process until each set is regarded as testing data once. And the prediction results are compared with the original raw data for performance comparison. Therefore, before the whole process starts, the value of *k* must be determined even though there is no hard rule for this value. Further, in previous studies, *k* was usually preset to 3 or 10 (divided into 3 or 10 testing data).

Under normal circumstances, studies rarely test the stability of a data matrix. However, because there are only 83 observations in this study, the simple test of the stability of the data matrix is carried out to determine the value of *k*.

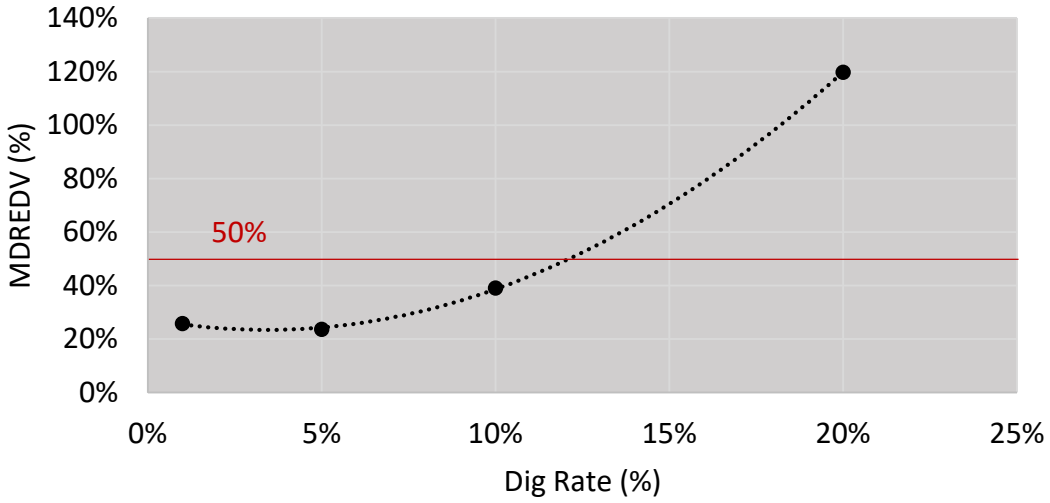
2.2.1. Stability test in practice

For determining the *k*, we designed a dig testing which is randomly digging original data sheet in different percentage (such as 2%, 5%, 10%, 20%, and so on), then calculate the mean absolute different ratio of each dug value (MADREDV) as Eq (S1),

and draw the correlation between dig rate and mean absolute different ratio of each dug value (MADREDV) (%).

$$\mathrm{MADREDV} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(dnrvPi - Oi)}{Oi}\right| \qquad \text{Eq(S1)}$$

where *dnrvPi* is the dug and re-estimated value; *Oi* is the original dug values collected from peer-reviewed articles; and n is the number of dug observations.



Extended Data Figure 2 | Correlation between dig rate and mean absolute different ratio of each dug value (%) in the original datasheet estimated via random forest with Gibbs sampling. Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a multivariate probability distribution, when direct sampling is difficult.

The results of stability test in this study show that if the dig rate over 10%, the MDREDV will dramatically increase over 50% (Extended Data Figure 2). Hence, we decided *k*=10 in this study and let the MDREDV will below 50%. (Of course, it is possible directly skip the above test, let *k*=10 or 3, depending on the number of observations. If observation size is large, *k*=3 will be better because of saving computing power.)

2.2.2. Separate data sheet into *k*-folds

Through the above test, it is possible to decide a suitable *k*. (For instance, in this study, we decided *k*=10.) Then randomly re-arrange orders of observations in raw datasheet, and evenly divided observations of it into 10 .csv files (10% observations per file) as testing data. Grab the rest of each testing data as training data (90% observations per file) prepared for further analysis.

2.3. Environment setting

Because most of the computer operating systems used are MS Windows, the environment settings in this process will all take MS Windows as an example and operate via R statistics and R studio.

2.3.1. Install R in cluster computing (Windows)

Download R from https://cran.r-project.org/mirrors.html and R studio from https://rstudio.com/products/rstudio/download/ , then install them in every single computer in the cluster used to run R codes and operate the following algorithm.

2.3.2. Install R in cloud computing (Amazon Web Services, AWS)

If research fund is enough, it is also possible using cloud computing to speed up all calculations. Below we introduce how to operate Amazon Web Services (AWS) for cloud computing.

2.3.2.1. Visit AWS web site https://aws.amazon.com/ , click Amazon EC2, apply a new account, and then log in.

2.3.2.2. Press “Launch instance” (Then you will see whole of the instance types)

2.3.2.3. Go to “Community AMIs” (AMIs means Amazon Machine Images).

2.3.2.4. Press “Ubuntu” and search “rstudio”

2.3.2.5. Choose “RStudio-1.3.1073_R-4.0.2_CUDA-10.1_cuDNN-7.6.5_ubuntu-18.04-LTS-64bit - ami-00b6908a3a1faec83 “

2.3.2.6. Choose an instance type you want. [If you need very strong computing power and have enough funding, it is also possible to choose 128 vCPUs and 3904 (GiB) Memory.]. Then, press “Next: Configure Instance Details”.

2.3.2.7. Check the instance details, then press “Next: Add storage”.

2.3.2.8. Check the detail of storage. (If your data matrix is really large, it is better to choose suitable storage size. e.g. 100 GiB), press “Next: Add Tags”, and then press “Next: Configure Security Group”.

2.3.2.9. Choose SSH for Type. Then, press “Add Rule”, choose “Custom TCP” for Type, set 80 for Port Range. Further, press “Add Rule”, choose “Custom TCP” for Type, set 8787 for Port Range. (Make sure the source of 80 and 8787 both are “0.0.0.0/0, ::/0”). In this page’s end, press “Review and Launch”.

2.3.2.10. After check the details of instance, press “Launch”. Then, the user interface will ask you to choose an existing key pair or create a new key pair. (The step is quite important because the key can use to connect via SSH). If it is the first time to launch instance, create a new key pair. Then, press “Launch Instance”.

2.3.2.11. Wait for 2 to 8 minutes until the Instance State turn yellow to green light.

2.3.2.12. Copy the Public DNS (IPv4) and paste to URL bar, then press enter for entering

the user interface of your new launched instance.

2.3.2.13. Input Username “rstudio” and Password which is preset your instance ID provided by AWS (such as this kind of format i-09d7c7c7e455d5612).

2.3.2.14. After enter the R studio user interface, the following steps are the same as usual computers for operating R. (Because the user interface of AWS is frequently changed, the latest process will be updated on GitHub from time to time, at https://github.com/gn03138868/WTF_sub-algorithm)

2.4. Input data

After collecting raw data, saving file as .csv, and building environment for running codes, open R studio software and run the code as below:

```
                    inputrawdata.R10.k.fold.X.training
                    <- read.csv("D:/Folder/rawdata k-fold X training.csv",
# Define the pathway of your input .csv files #
                    header=T, row.names = 1,
# It means the first row is title #
                    sep=",")
# Separate each value via comma #
                    head(inputrawdata.R10.k.fold.X.training)
# Show the input raw data #
                    summary(inputrawdata.R10.k.fold.X.training)
# Basic information of input raw data #
                    str(inputrawdata.R10.k.fold.X.training)
# Show the structure of input raw data #
```

The name defined after data entry does not have a mandatory form. It is possible named logically and easy to recognise as well. Furthermore, sometimes, different countries' pathway in R perhaps has different styles. If you are not sure about it, please google it and refer other pathways' styles.

2.5. Data observation

After finish to input data, observe the data for checking missing values.

```
                    md.pattern(inputrawdata.R10.k.fold.X.training)
# Observe missing values #
```

2.6. Install packages in R

Install packages in R before running the SHA.

```
                    install.packages("mice")
                    require(mice)
# Borrow the framework from MICE for our wandering through the random
forests (WTF) sub-algorithm #
```

At the beginning, we wanted to write the entire WTF architecture by ourselves, but found that there would be thousands or even tens of thousands lines of codes, which was very inconvenient for policymakers to understand and apply. Fortunately,

in the R community, there are packages that have been released and similar frameworks can be applied for our research purpose. MICE is one of them.

```
                    install.packages("Hmisc")
                    require(Hmisc)
# Package for Pearson's correlation test #

                    install.packages("corrplot")
                    require(corrplot)
# Package for correlation matrix #
```

If the total number of variables is below 50, it will be more convenient to use corrplot. If not, Hmisc must to be installed. We will elaborate more details about Pearson's correlation in section 3.

```
                    install.packages("ggplot2")
                    require(ggplot2)
                    install.packages("GGally")
                    require(GGally)
                    install.packages("ggfortify")
                    library(ggfortify)
# Install packages for visualisation #
```

It is possible to draw different kinds of figures after installing above packages in R.

3. Feature selection for linear regression (LR) models – 1st layer

3.1. Single features selecting (correlation tests)

We wanted to use stepwise for feature selection for making multilinear regression model initially. However, our data has a special character - a lot of missing values exist in data sheet, which means not every research investigate usual features in their own research. And after several times modification and rolling correction improved by conferences and seminars, we decided to construct simple linear regression models for estimating target variables which could be predicted by independent variables of the 1st layer of the SHA and must be in high accuracy and precision and to leave the irregular and divergent target variables to the next layer - Wandering through the random forests (WTF) sub-algorithm for estimating the target variables contained a lot

of unobserved variables. Hence, we use Pearson [5], Spearman [6], and Kendall [7] correlation methods to select features of each independent variable. The R codes describe as below:

```
                    cor.test(~Variable A + Variable B,
                    Data's name,
                    method = "spearman", "pearson" or "kendall")
# Correlation tests #
```

where the Variable A is target variable, the Variable B is independent variable, Data's name is the name of data you input in R, method = "_____" is for determining the method who want to use for correlation test.

If the Pearson correlation coefficient r > 0.85 and p<0.01, construct linear regression (LR) models for the target (dependent) variables. If not, transfer the estimation work to the next layer of the SHA.

3.2. Constructing linear regression models

The independent variables with high precision and accuracy for target variables can be screened out after analysing all the correlation between target and independent variables. Then construct a simple linear regression model for each target variable as the Eq (S2) and (S3) below.

$$Y = aX + b \qquad \text{Eq(S2)}$$

$$Y = c \times e^{aX} \qquad \text{Eq(S3)}$$

where *Y* is a target variable, *X* is an independent variable, *a* is a coefficient of the independent variable, *b* is constant, *e* is Euler's number, and *c* is the coefficient of *e*.

The above equations can be easily made in Microsoft Excel even if in R as the same, which software policymakers use depending on which software they are familiar with.

```
        LRmodelsName = lm(Variavle B~ Variable A,
                    data = Data's name,
                    na.rm = TRUE)
# Construct simple linear regression model as Eq(S1) #
```

Where the RLmodelsName is the Regression models you named, lm(Variavle B~

[5] Pearson, K., 1901. LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine.* 2(11), 559-572. http://pca.narod.ru/pearson1901.pdf

[6] Spearman, C., 1904. The Proof and Measurement of Association between Two Things. *Am. J. Psychol.* 15, 72–101. https://www.jstor.org/stable/pdf/1422689.pdf

[7] Kendall, M. G., 1938. A New Measure of Rank Correlation. *Biometrika* 30(1-2), 81–93. https://watermark.silverchair.com/30-1-2-81.pdf

Variable A) is a function code in R statistics software.

```
LRmodelsName = lm(Variavle B~ exp(Variable A),
                  data = Data's name,
                  na.rm = TRUE)
# Construct simple (exponential) linear regression model as Eq(S2) #
```

Where the exp(Variable A) is an exponential function code in R statistics software.

```
summary(LRmodelsName)
# Statistics summary #
# If R^2 < 0.85 or >-0.85, it is good for constructing linear regression models. #
```

Select independent variables which can be used to estimate target variable in high accuracy and precision via above simple correlation tests.

3.3. Diagnosis of linear regression models

After constructing leaner regression models, it is better to do some diagnosis than never.

Before operating the diagnosis, draw the figure of distribution first.

```
qplot(x = Variable B,
      y = Variable A,
      data = Data's name, na.rm=TRUE)
# Observing distributions of target and independent variables. #
```

where the Variable A is target variable, the Variable B is independent variable, Data's name is the name of data you input in R, na.rm=TRUE is for negating missing values.

Theoretically, after the regression model is fitted, its residual value must conform to the three assumptions of normality, independence, and homogeneity of variance, so the next three tests are used. Check whether these three conditions are satisfied or not.

```
shapiro.test(LRmodelsName$residual)
# Normality test #
```

where the LRmodelsName is the name of linear regression models constructed above.

If *p*-value is high (perhaps over 0.05 depended on the confidence level), it cannot reject the null hypothesis which means the assumption of normal distribution of residual value is reasonable. If *p*-value is low, the normal distribution of the residual value is unreasonable.

```
durbinWatsonTest(LRmodelsName)
# Independence test #
```

If *p*-value is high (perhaps over 0.05 depended on the confidence level), it cannot reject the null hypothesis which means the independence of the residual value is reasonable. If *p*-value is low, the independence of the residual value is unreasonable.

```
ncvTest(LRmodelsName)
# Homogeneity of variance test #
```

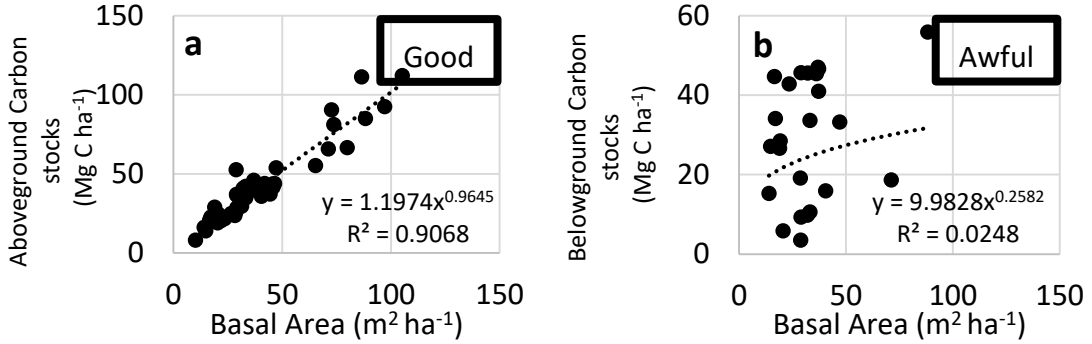
If *p*-value is high (perhaps over 0.05 depended on the confidence level), it cannot reject the null hypothesis which means the homogeneity of variance of the residual value is reasonable. If *p*-value is low, the homogeneity of variance of the residual value is unreasonable.

Through the above R codes and processes, simple linear regression models can be prepared for estimating target variables in 1st layer of the SHA if above conditions are satisfied. Input estimated values, which came from LR models, into the matrix output from 1st layer as an input matrix for the 2ed layer's training.

4. Wandering through the random forests (WTF) sub-algorithm – 2^{ed} layer

4.1. Background of the WTF sub-algorithm

A case of linear regression (LR) models has good performance in estimating aboveground carbon (AGC) stocks but awful in belowground (Extended Data Figure 3). The major reason is that target variables are always affected by a lot of environmental and anthropogenic factors (Table S2) caused the correlation between the dependent and independent variables are easily irregular even divergent.



Extended Data Figure 3 | Linear regression (LR) models for estimating aboveground carbon (AGC) and belowground carbon (BGC) stocks. a, A LR model for estimating AGC with the basal area as an independent variable. b, A LR model for estimating BGC with the basal area as an independent variable.

For conquering the problem, we designed a machine learning called wandering through the random forests (WTF) sub-algorithm, on the 2^{ed} layer to solve the problems that the 1st layer cannot handle. The WTF sub-algorithm is composed of random forests and a Markov chain Monte Carlo (MCMC) type algorithm - Gibbs sampling. The pseudocode of the WTF sub-algorithm describes in Table S5.

Table S5. The wandering through the random forests (WTF) sub-algorithm in the simple hybrid algorithm (SHA).

Algorithm | the WTF sub-algorithm in the SHA

1: **initialise** $X^m \sim X^{(0)}$

2: **for** duplicate m = 0, 1, 2, … **do**

1: **initialise** $X^{(0)} \sim q(X)$

2: **for** iteration *i* = 0, 1, 2, … **do**

1. $X_1^{(i)} \sim p(X_1 = x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)}, \ldots, X_D = x_D^{(i-1)})$
    1. **procedure** random forest
    2. **for** 1 to T **do**
    3. Draw 66.6% points from D for forming a new Di
    [The rest of 33.4% points from D for out of bag (OOB) test]
    4. Build full classification and regression trees on Di
    5. **end for**
    6. **average** all T trees
    7. **end procedure**
2. $X_2^{(i)} \sim p(X_2 = x_2 | X_1 = x_1^{(i)}, X_3 = x_3^{(i-1)}, \ldots, X_D = x_D^{(i-1)})$
    1. **procedure** random forest
    2. **for** 1 to T **do**
    3. Draw 66.6% points from D for forming a new Di
    [The rest of 33.4% points from D for out of bag (OOB) test]
    4.Build full classification and regression trees on Di
    5. **end for**
    6. **average** all T trees
    7. **end procedure**
3. …
4. $X_D^{(i)} \sim p(X_D = x_D | X_1 = x_1^{(i)}, X_2 = x_2^{(i)}, \ldots, X_D = x_{D-1}^{(i)})$
    1. **procedure** random forest
    2. **for** 1 to T **do**
    3. Draw 66.6% points from D for forming a new Di
    [The rest of 33.4% points from D for out of bag (OOB) test]
    4.Build full classification and regression trees on Di
    5. **end for**
    6. **average** all T trees
    7. **end procedure**

3. **end for**

3: **average** all X^m

4: **end for**

Through the WTF sub-algorithm, target variables with complex independent variables can be estimated more accurately and precision than simple linear regression models. Below we describe the process of implementing R statistical operations.

4.2. 2ed layer in practice

For finding the lowest loss function of combination in duplication, CARtrees, and iteration numbers, we input training data for running R codes via cluster computing or cloud computing. If we write the framework of the SHA sub-algorithm completely by ourselves, it will cost quite a lot of time and need to debug. Fortunately, “mice” package provides the corresponding Gibbs sampling architecture, and can be bridged to the “randomForest” package. So here combined these two packages as the basis for running the WTF sub-algorithms. We appreciate to the research team who wrote these two packages and give them respect. Now it is possible run this WTF sub-algorithm with just a few lines of codes.

4.2.1. After making some LR models, estimating the NA values, and inputting in original datasheet as a new data sheet for 2ed layer, load the new data sheet into Rstudio.

```
                    inputrawdata.R10.k.fold.X.training
                    <- read.csv("D:/Folder/rawdata k-fold X training for
                    L2.csv",
# Define the pathway of your input .csv files #
                    header=T, row.names = 1,
# It means the first row is title #
                    sep=",")
# Separate each value via comma #
                    head(inputrawdata.R10.k.fold.X.training)
# Show the input raw data #
                    summary(inputrawdata.R10.k.fold.X.training)
# Basic information of input raw data #
                    str(inputrawdata.R10.k.fold.X.training)
```

4.2.2. Then, start to training out the minimised loss function combination of duplication, CARtrees, and iteration numbers. The implemented R code is as follows.

```
## m=X; maxit=Y; trees=Z ##
## It is better to try different combinations of X, Y, and Z based on different data
## characteristics ##
mice.data.R10.k.fold.k.training.1.mXmYtZ
                <- mice(inputrawdata.R10.10.fold.2.training.1,
                m = X,                  # generate 200 data sheets #
                                # It is possible to generate more than 100 sheets if you like #
                maxit = Y,            # max iteration times#
                method = "rf",        # CART means Classification And Regression Tree #
                                        # rf means random forest #
        # (It is also possible choose another methods such as linear regresions #
        # logistic regression, cart, random forest, boostrap if you'd already
        # installed related packages) #
                ntree=Z,
        # ntree means CARtrees number in random forest #
                seed = 188)
        # set.seed()Let the sample survey data be the same every time #
summary(mice.data.R10.k.fold.k.training.1.m200m150t100)

save.image("Z:/… …/result_file_name.RData")
   # This is for defending the pathway of output results #
   ## Performanced by i5-9400F CPU @ 2.90GHz 2.90 GHz ##
   ## From 14Aug2020 13:53 to 16Aug2020 23:53 ETA 32hr06min ##
   ## These two lines are optional, but if there is a record, ##
   ## it is possible to record the calculation cost of the running algorithm. ##
```

4.2.3. After running the above codes, call results out for calculating.

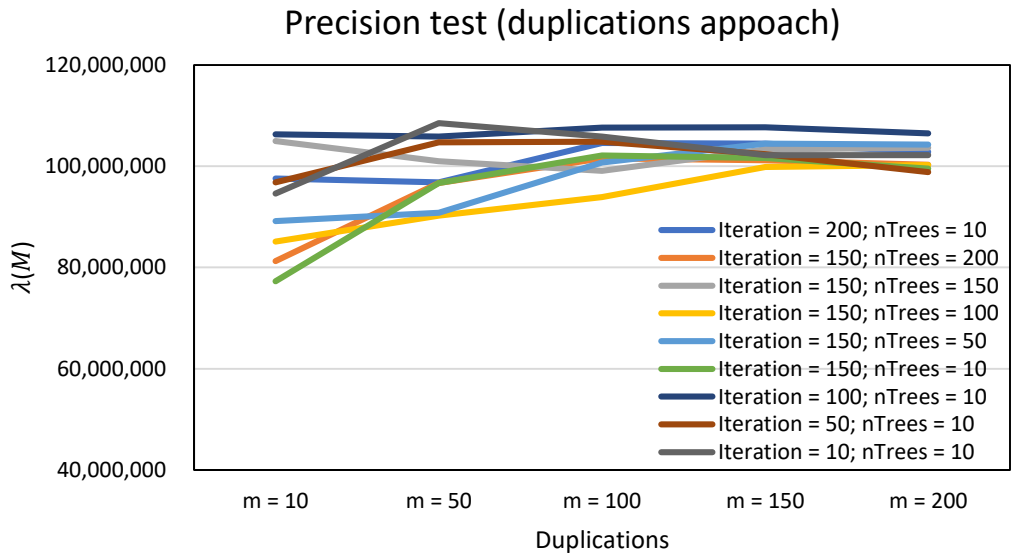
```
                        miceX.R10.k.fold.k.training.1.mXmYtZ <-
                        complete(mice.data.R10.k.fold.2.training.1.m200m10t10, X)
# first data sheet called as mice1#
# repeat this code to call all duplicates X = 1, 2, 3, …, X #
                        head(miceX.R10.k.fold.k.training.1.mXmYtZ)

                        miceX.R10.k.fold.k.training.1.mXmYtZ
                        [is.na(miceX.R10.k.fold.k.training.1.mXmYtZ)]<-0
# replace NA to 0 for further calculations #
```

4.2.4. After call out results, calculate MSE of different X.

```
                                mice.mean1_10.R10.k.fold.k.training.1.mXmYtZ<-
                                (mice1.R10.k.fold.k.training.1.mXmYtZ
                                +mice2.R10.k.fold.k.training.1.mXmYtZ
                                +mice3.R10.k.fold.k.training.1.mXmYtZ
                                + …
                                +miceX.R10.k.fold.k.training.1.mXmYtZ
                                   )/X
# Take the means of sum of the matrix, #
# which is the largest number of duplication X, #
# as the mean of maximum duplicates dataset values (Mi). #
                                head(mice.mean1_10.R10.k.fold.k.training.1.mXmYtZ)
```

4.2.5. After calculating the MSE of different combination of X, Y, and Z, then draw the relationships between $\lambda(M)$ and the different combination of them (Extended Data Figure 4).

4.2.6. Choose the combination with the lowest $\lambda(M)$ for further testing set after all combination of $\lambda(M)$ have converged. For instance, in the case of Extended Data Figure 4, the combination with the lowest $\lambda(M)$ is X = 150, Y=150, and Z=100.



Extended Data Figure 4 | Relationships between $\lambda(M)$ and the different combination of duplications, max iteration, and nTrees numbers in one of *k* folds.

After above process, we get a trained combination for one of *k* folds in training set.

5. *k*-fold cross validation[8]

5.1. Combine original training and testing data sheets as one .csv file after get the best combination of X, Y, and Z.

5.2. Then, input variables via LR models made from step 3.2. (In this step, it is possible to use Excel () or R which the operator prefers to use.)

5.3. Run the same process from step 4.2.1. to 4.2.4. and get the mean of output matrix from the 2^{ed} layer.

6. Climate classification table – 3rd layer

If the mean of output matrix has missing values inside, which means the output target variable have not pass the OBB test mentioned in Table S5, need to follow IPCC Guidelines for National Greenhouse Gas Inventories and make Table S4 prepared from Table S1. Then input the value into the output of 2^{ed} layer as the output of 3rd layer which is also the final output of the SHA. (At least, using the average of the global data can ensure that the accuracy of the estimated results is high.)

7. Final output compared with benchmark

7.1. Obtain the benchmark

Use the IPCC Guidelines for National Greenhouse Gas Inventories[9] and re-estimate the target variables to replace the original data matrix as the benchmark data matrix.

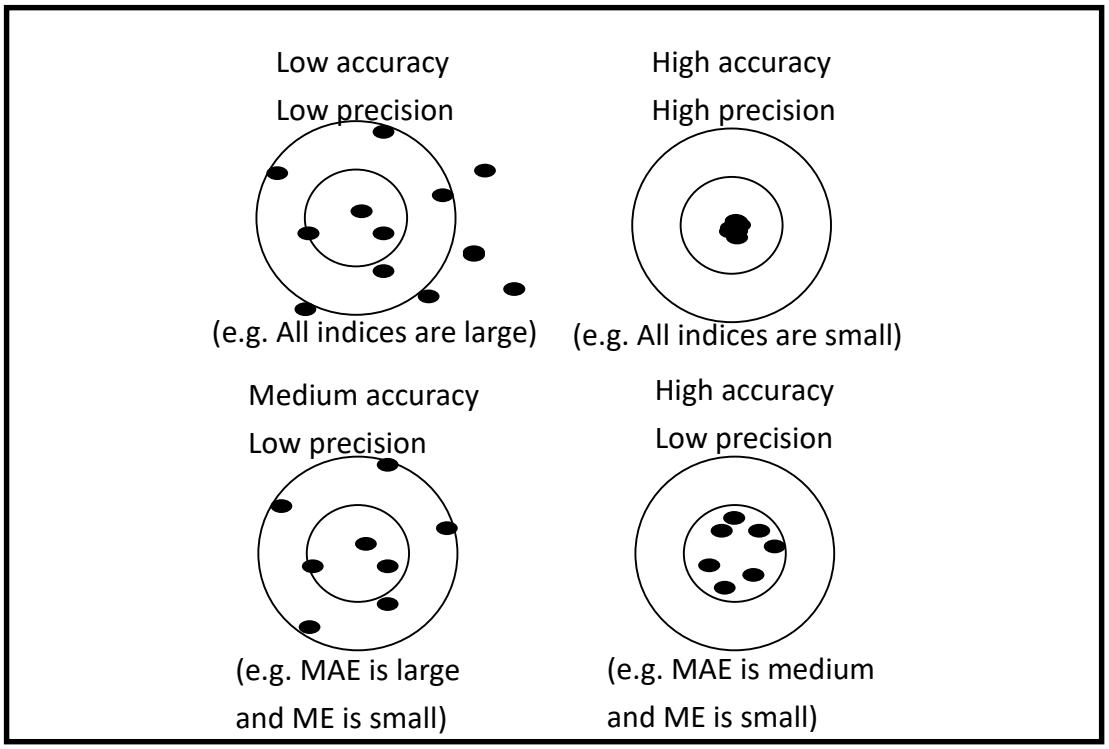
7.2. Performance calculating

Calculate the mean absolute percentage error (MAPE), root mean square error (RMSE), mean absolute error (MAE), mean error (ME), which mentioned in main text of Equation (2) to (4), of the benchmark and the estimated SHA output provided by *k*-fold cross validation. Of cause, it will also get each observation's absolute percentage error (APE), absolute error (AE), and error. For further statistical process. (Meaning of indices for the performance of algorithms describe in Extended Data Figure 5.)

---

[8] Do not to grab "testing data set" into the WTF sub-algorithm (the 2er layer) for training or to find parameters.

[9] Original sources of the IPCC's guideline: https://www.ipcc.ch/report/2006-ipcc-guidelines-for-national-greenhouse-gas-inventories/
Recently, it was refined here in 2019: https://www.ipcc.ch/report/2019-refinement-to-the-2006-ipcc-guidelines-for-national-greenhouse-gas-inventories/



Extended Data Figure 5 | **Meaning of indices for the performance of algorithms.**

7.3. Statistical process of different algorithms

After obtain the benchmark and performance calculating, save the results data as a .csv file. And choose one of the most familiar statistic software of yourself and input the data into the software for further process. The flowchart of statistics process of comparing the performance between different algorithms as shown in **Extended Data Figure 6**. In this study, we use R statistic software to operate it.

7.3.1. Shapiro-Wilk normality test

This step is for check that the group is normal distribution or not[10]. If *p*-value > 0.05, the data is normal distribution; if *p*-value < 0.05, the data is not normal distribution.

7.3.2. Homogeneity of variances' test in multiple groups

It is better to check the homogeneity of variances before recognising those each group is significant different or not via Bartlett's or Levene's test. If the result of homogeneity of variances' test is homogeneity, use one-way ANOVA; if not, use Welch's Heteroscedastic F Test.

[10] Royston, P. (1982). Algorithm AS 181: The W test for Normality. Applied Statistics, 31, 176-180. doi: 10.2307/2347986.

7.3.2.1. Bartlett's test

Bartlett's test is suit for normal distributions groups. If *p*-value > 0.05, variances are homogeneity; if *p*-value < 0.05, variances are heterogeneity.

7.3.2.2. Levene's test

Levene's test is suitable for other types' distribution. If *p*-value > 0.05, variances are homogeneity; if *p*-value < 0.05, variances are heterogeneity.

7.3.3. one-way ANOVA or Welch's Heteroscedastic F Test

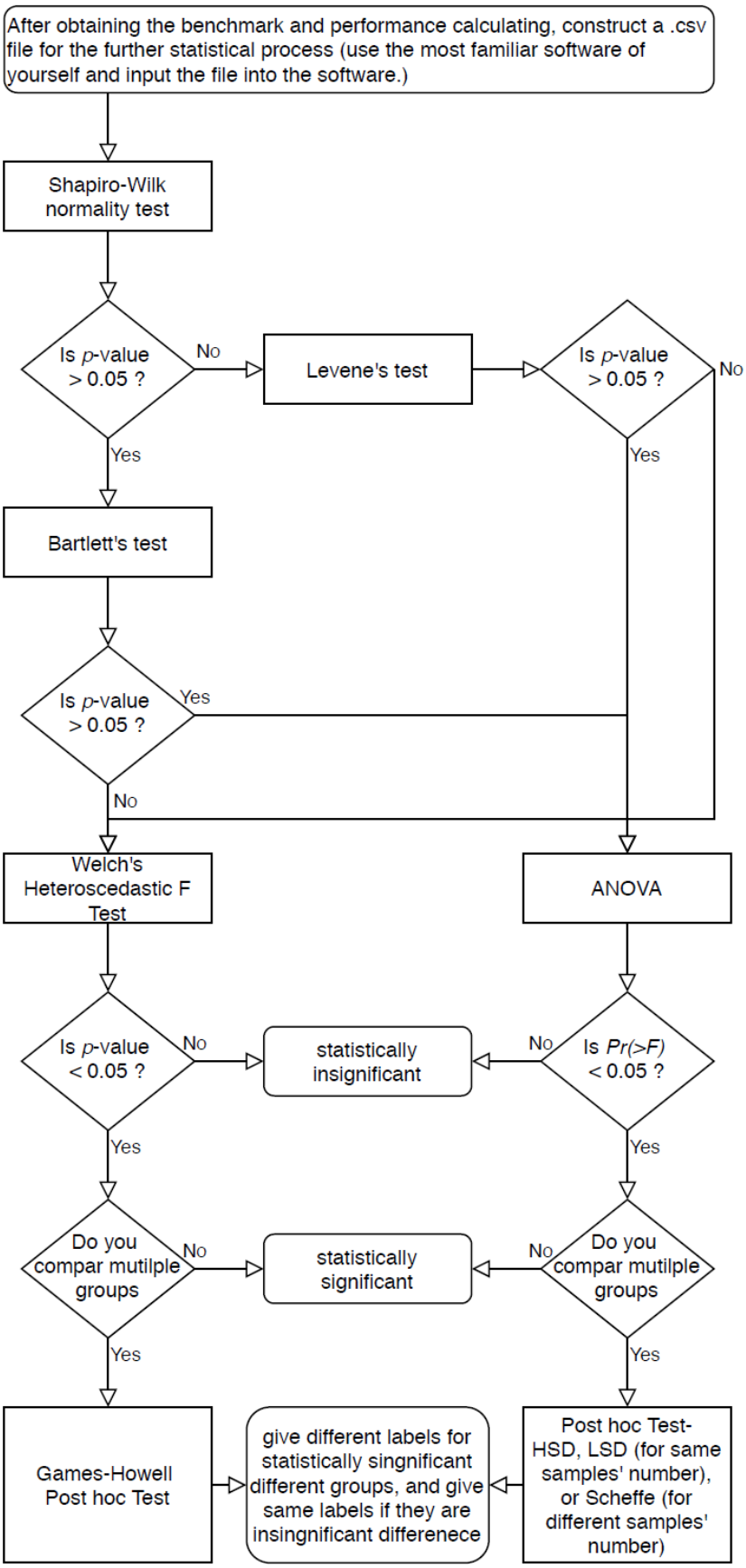
If *p*-value < 0.05, difference is statistically significant; if not, difference is statistically insignificant (alpha = 0.05).

7.3.4. Post-hoc test

If variances in one-way ANOVA are significant different, just operate HSD, LSD (for same samples' number) or Scheffe test (for different samples' number). Or if the variables are heterogeneity and the results of Welch's Heteroscedastic F Test is significant different, the post-hoc test is better to use Games-Howell method.

8. Final Visualisation

After running all the processes above, it is possible to visualise the result in table or box plot depended on the preference of the operator. In this study, we reveal the result in the main text of Table 2. The latest version of the process will continually updated on GitHub (https://github.com/gn03138868/WTF_sub-algorithm). If there are bugs in the code script, please report to the first author or corresponding author. We sincerely thank you.



Extended Data Figure 6 | Flowchart of statistical process of different algorithms.