OpenBench 用户手册

The Open Source Land Surface Model Benchmarking System Users' Guide

目录

第一部分 引言

1 引言

在地球系统科学和气候变化研究中，陆面模型(Land Surface Models，LSMs)扮演着至关重要的角色。这些模型模拟陆地表面与大气之间的能量、水分和碳循环交换，为我们理解和预测全球气候变化提供了重要工具。然而，随着模型复杂性的增加和应用范围的扩大，如何有效评估和比较不同陆面模型的性能成为了一个日益突出的挑战。基于此，我们开发了 The Open Source Land Surface Model Benchmarking System（OpenBench）软件。OpenBench 是一个专门设计用于陆面模型评估的开源基准测试系统。OpenBench 的目标是为研究人员和模型开发者提供一个标准化、全面且灵活的工具，用于验证模型输出、比较不同模型的性能，并深入分析模型在各种条件下的表现。通过提供一套统一的评估标准和工具，OpenBench 旨在促进陆面模型的持续改进和科学社区的协作。

OpenBench 的设计秉承着开放、标准、灵活、全面和高效的核心理念。开放性体现在 OpenBench 作为一个开源项目，鼓励科学社区的广泛参与和贡献，致力于提高评估方法的透明度和可复现性。标准化方面，OpenBench 通过提供统一的评估指标和数据处理流程，确保了不同研究之间结果的可比性。灵活性则体现在系统设计允许用户根据特定需求自定义评估流程，并支持各种类型的陆面模型和数据格式。全面性方面，OpenBench 不仅关注单一变量的表现，还提供了多变量、多尺度的综合评估能力。最后，效率方面，OpenBench 通过并行计算和优化的数据处理流程，能够高效处理大规模数据集。

OpenBench 的功能强大且全面，为陆面模型评估提供了一站式解决方案。它允许同时处理多个模型的输出，方便研究人员直接比较不同模型在相同条件下的表现，这对于模型比较、集成和不确定性分析等研究至关重要。OpenBench 支持对众多陆面过程变量进行评估，涵盖了生态系统碳循环、水文循环、能量平衡和生物物理参数等方面，并为每个变量提供了一套全面的评估指标，例如总初级生产力、蒸散发、土壤湿度等，全方位评估模型性能。不仅如此，OpenBench 还提供了基本统计指标和高级评分方法，例如 Kling-Gupta 效率、Nash-Sutcliffe 效率等，并支持时间序列分析和空间模式评估，能够深入分析模型在不同时空尺度上的表现。OpenBench 支持在小时到年际的时间尺度、站点到全球的空间尺度以及不同生态系统类型上进行评估，使用户能够全面了解模型在各种条件下的表现。为了确保评估结果的可靠性，OpenBench 集成了自动数据格式转换、时空插值、缺失数据处理和异常值检测等功能，保证了输入数据的质量和一致性。此外，OpenBench 还提供了丰富的可视化工具，包括时间序列图、空间分布图、Taylor 图等，不仅有助于结果解释，还能生成高质量的图表用于报告和发表。最后，OpenBench 实现了并行计算，用户可以根据硬件资源配置计算核心数，从而高效处理大规模数据集，显著提高处理速度。

OpenBench 在陆面模型研究和应用中用途广泛。它可以帮助模型开发者验证新开发的模型模块，评估参数化方案的改进，识别模型的优势和不足，从而推动模型的改进和完善。OpenBench 也适用于多模型比较研究，例如气候变化影响评估。它能够确保结果的可比性，揭示模型间的差异，并支持不同类型的比较分析，为研究人员提供了一个理想的工具。此外，OpenBench 还可以集成到模型校准和参数优化流程中，为优化算法提供目标函数，评估不同参数设置下的

模型性能，并分析参数敏感性，从而帮助研究人员找到最佳的模型参数。在数据同化和模型-数据融合应用中，OpenBench 可以评估同化前后的模型性能，为同化算法提供误差统计信息，并验证同化结果的时空一致性，最终提高数据同化的效果。OpenBench 也能够用于气候变化影响评估，它可以评估模型在历史期的表现，分析未来情景下的模型一致性，并量化预测的不确定性，为气候变化影响评估提供可靠的依据。总而言之，OpenBench 为陆面模型研究提供了全面的评估和分析工具，促进了模型的发展和应用，并为天气和气候等相关研究提供了有力支持。

第二部分 快速入门

2 快速运行一个评估实例

2.1 代码基本结构

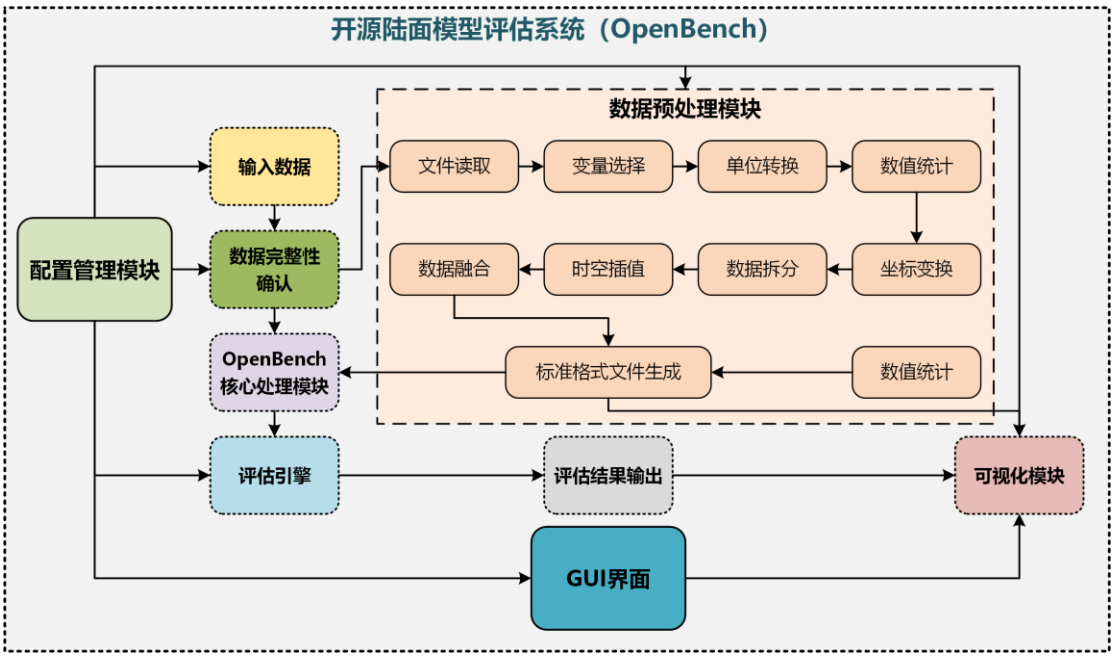

图 1：OpenBench 的总体流程图

OpenBench 采用模块化设计，其核心组件包括**配置管理模块**、**数据预处理模块**、**评估引擎**（包括**评估模块**、**比较分析模块**、**统计分析模块**）、**可视化模块**和**日志生成器**。配置管理模块负责读取和解析用户配置文件，设置评估参数。数据预处理模块处理模型输出和参考数据，进行格式转换、对齐和质量控制。评估引擎是实现评估的核心模块，下面包括评估模块、比较分析模块和统计分析模块。评估模块用于计算各种评估指标和评分方法，比较分析模块用于多模型比较和集成分析，统计分析模块提供高级统计分析功能。可视化模块生成各种图表和可视化结果，日志生成器汇总评估过程细节生成评估日志。

OpenBench 的工作流程高度自动化，用户只需准备配置文件和输入数据，即可获得全面的评估结果。系统首先读取配置，加载并预处理数据，然后执行指定的评估指标计算。如果开启比较功能，系统会进行多模型比较分析。接下来，系统会根据配置执行统计分析，最后生成可视化结果和评估日志。这种模块化设计使不同功能模块之间的耦合度降到最低，便于系统的维护和扩展。在使用时，主要对配置管理模块进行处理。

2.2 安装与运行

OpenBench 的安装与运行分为评估系统安装、软件环境的配置、数据的准备和评估系统运行四个主要步骤。

（1）评估系统安装与配置

OpenBench 当前可在 Linux 系统下运行。当前版本已托管在 github.com。安装指令为：

```
git clone git@github.com:zhongwangwei/OpenBench.git
```

假设代码放置的根目录为**$OpenBench**。

（2）软件环境的配置

OpenBench 在 Linux 操作系统下的软件需求为：

```
Python（版本≥3.10）
```

主要软件依赖为：

```
NetCDF4（版本≥1.5.7）
xarray（版本≥0.19.0）
NumPy（版本≥1.21.0）
SciPy（版本≥1.7.0）
Matplotlib（版本≥3.4.0）
Pandas（版本≥1.3.0）
joblib（版本≥1.1.0）
Dask（版本≥2022.1.0）
Cartopy（版本≥0.20.0）
```

flox（版本≥0.5.0）

以下文件内容给出了一个典型环境下的软件配置，文件路径为**$OpenBench/requirement.yml**

```yaml
name: openbench
channels:
  - conda-forge
  - defaults
channel_priority: strict
dependencies:
  - python>=3.10
  - netCDF4>=1.5.7
  - xarray>=0.19.0
  - numpy>=1.21.0
  - scipy>=1.7.0
  - matplotlib>=3.4.0
  - pandas>=1.3.0
  - joblib>=1.1.0
  - dask>=2022.1.0
  - cartopy>=0.20.0
  - flox>=0.5.0
```

以上文件对 Python 解释器和 Python 第三方库进行了配置，列出了评估系统所需的 Python 包及其版本。

如当前环境已安装 conda，可使用 conda 安装并激活新环境（openbench）：

```
conda env create -f $OpenBench/requirements.yml
conda activate openbench
```

也可使用 pip 进行安装：

```
pip install -r $OpenBench/requirements.txt
```

（3）准备数据

准备好 OpenBench 运行所需的两种必要数据：模拟数据集和参考数据集。

（4）运行评估系统

OpenBench 通过以下三个文件对一个评估实例进行设置：

- **$OpenBench/main.json**：主配置文件

- **$OpenBench/sim.json**：模型数据配置文件
- **$OpenBench/ref.json**：参考数据配置文件

以上三个文件可随意命名。评估系统中已给出一个配置好的案例对应文件：**main-Debug.json** 、 **sim-Debug.json** 、 **ref-Debug.json** ， 均 位 于 **$OpenBench** 目录下。详细配置可参照本手册的第三部分。

进入根目录**$OpenBench**，运行配置案例：

```
python script/openbench.py nml/main-Debug.json
```

需注意的是，其中运行所跟随目录为所使用的主配置文件（此处案例为**main-Debug.json**）,模型数据配置文件（此处案例为 **sim-Debug.json**）与参考数据配置文件（此处案例为 **ref-Debug.json**）在主配置文件中设置。

以下截取自 **main-Debug.json**，

```
    "reference_nml": "./nml/ref-Debug.json", //模型数据配置文件路径
    "simulation_nml": "./nml/sim-Debug.json",//参考数据配置文件路径
```

评估系统输出结果位于**$OpenBench/output/**下。

第三部分 系统配置

针对系统配置，OpenBench 提供了基于 Debug 案例的全流程配置注释文件，分别为：

（1）主配置文件：

```
$OpenBench/nml/main-Debug_comment_ZH.json
```

（2）模型数据配置文件：

```
$OpenBench/nml/sim-Debug_comment_ZH.json
```

（3）参考数据配置文件：

```
$OpenBench/nml/ref-Debug_comment_ZH.json
```

（4）模型数据 1 配置：

```
$OpenBench/nml/user/debug/grid_case_comment_ZH.json
```

模型数据 2 配置：

```
$OpenBench/nml/user/debug/station_case_comment_ZH.json
```

（5）参考数据 1 配置：

```
$OpenBench/nml/Ref_variables_defination/Debug/GLEAM_hybrid_PLUMBER2_com
ment_ZH.json
```

参考数据 2 配置：

```
$OpenBench/nml/Ref_variables_defination/Debug/ILAMB_monthly_comment_ZH.
json
```

参考数据 3 配置：

```
$OpenBench/nml/Ref_variables_defination/Debug/PLUMBER2_comment_ZH.json
```

参考数据 4 配置：

```
$OpenBench/nml/Ref_variables_defination/Debug/GLEAM4.2a_monthly_comment
_ZH.json
```

以上文件对所有参数设置加入注释解析，由于 json 格式文件无法加入注释，请使用对应无注释版本（**xxx.json**）运行推荐案例。其配置顺序如图 2 所示。

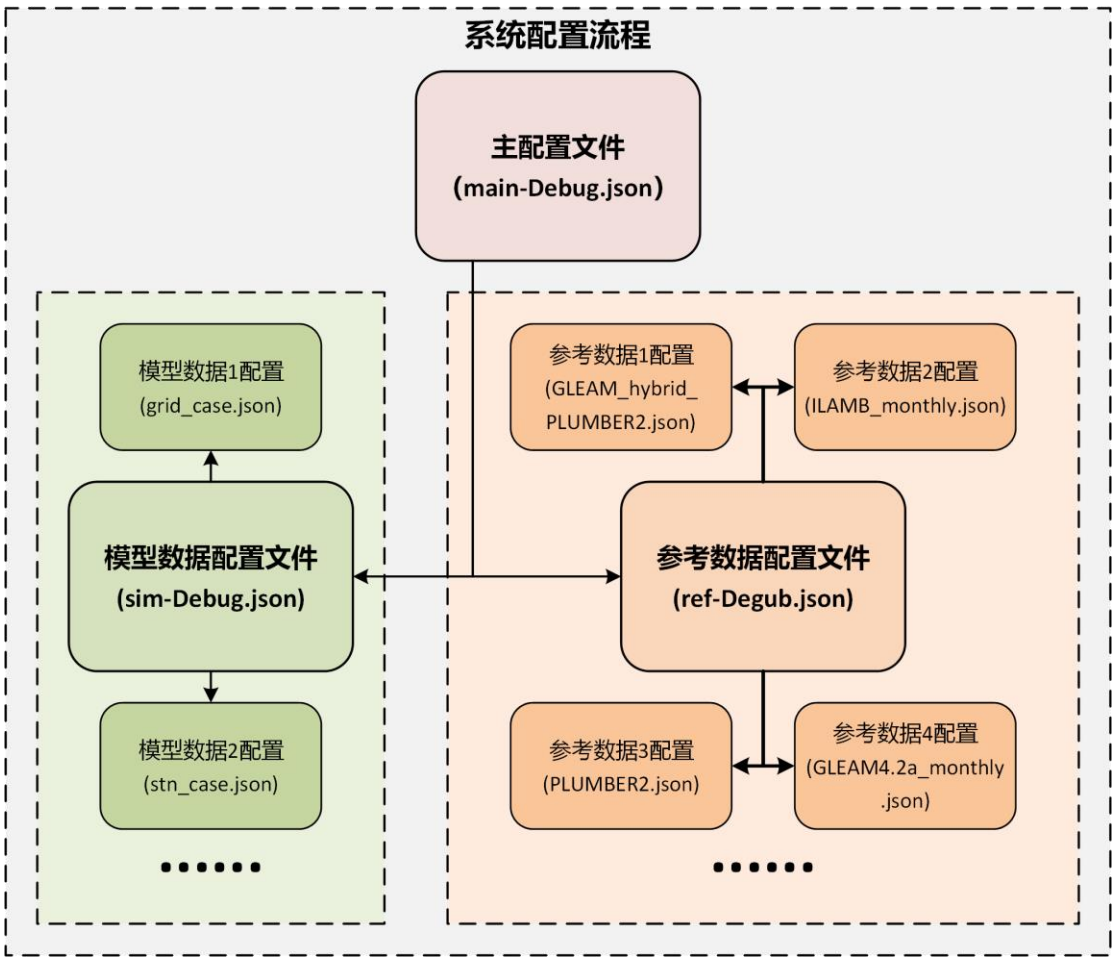
系统配置流程
主配置文件
(main-Debug.json)
模型数据1配置
(grid_case.json)
模型数据配置文件
(sim-Debug.json)
模型数据2配置
(stn_case.json)
参考数据1配置
(GLEAM_hybrid_
PLUMBER2.json)
参考数据2配置
(ILAMB_monthly.json)
参考数据配置文件
(ref-Degub.json)
参考数据3配置
(PLUMBER2.json)
参考数据4配置
(GLEAM4.2a_monthly
.json)

图 2：系统配置流程

3 主配置文件（main.json）

主配置文件（**main.json**）采用 json 格式，定义了整个评估过程的参数,包括基本设置、评估项目、评估指标、比较方法和统计分析等。

main-Debug_comment_ZH.json 示例：

```
{
/*主配置文件：定义整个评估过程参数，
包括基础设置、评估项目、评估指标、
评分方法、比较分析和统计分析等*/
  /*基础设置：评估的基本设置，
  如此处设定一个名为Debug 的案例，
  其中模型数据配置来源于sim-Debug.json，
  参考数据配置来源于ref-Debug.json，
  评估模块、比较模块、统计模块全部开启*/
  "general": {
    "basename": "Debug",          //评估案例名称
    "basedir": "./output",        //评估结果输出路径
    "compare_tim_res": "month",   //比较的时间分辨率 （如, "month",
"day"）
    "compare_tzone": 0.0,         //时区设置
    "compare_grid_res": 2.0,      //空间网格分辨率（度）
    "syear": 2004,                //评估的起始年份
    "eyear": 2005,                //评估的结束年份
    "min_year": 1.0,              //评估最小间隔年限（小于1 年不评估）
    "max_lat": 90.0,              //评估区域的最大纬度（-90~90）
    "min_lat": -90.0,             //评估区域的最小纬度（-90~90）
    "max_lon": 180.0,             //评估区域的最大经度（-180~180）
    "min_lon": -180.0,            //评估区域的最小经度（-180~180）
    "reference_nml": "./nml/ref-Debug.json", //模型数据配置文件路径
    "simulation_nml": "./nml/sim-Debug.json",//参考数据配置文件路径
    "statistics_nml": "./nml/stats.json",    //统计分析配置文件路径
    "figure_nml": "./nml/figlib.json",       //可视化配置文件路径
    "num_cores": 48,              //并行计算使用的核心数
    "evaluation": true,           //是否开启评估模块
    "comparison": true,           //是否开启比较模块
    "statistics": true,           //是否开启统计模块
    "debug_mode": true,           //是否debug 模块（增加输出信息）
    "weight": "None",             //是否加权计算（"area", "mass", "None"）
```

```
    "IGBP_groupby": true,          //是否进行IGBP分类聚合
    "PFT_groupby": true,           //是否进行PFT分类聚合
    "unified_mask": true           //当模型数据均为grid时，是否只评估重合区
域
  },
  /*评估项目：定义可评估变量及其开关，
  基础设置中评估模块开启，
  此处所有定义为true的变量均被设为评估项目，
  包括Evapotranspiration、Latent_Heat和Sensible_Heat*/
  "evaluation_items": {
    "Evapotranspiration": true,
    "Latent_Heat": true,
    "Sensible_Heat": true,
    "Surface_Net_LW_Radiation": false,
    "Surface_Net_SW_Radiation": false,
    /*…*/
  },
  /*评估指标：定义评估指标及其开关，
  基础设置中评估模块开启，
  所有评估项目均会计算以下被设为true的评估指标，
  包括RMSE、correlation和KGESS*/
  "metrics": {
    "RMSE": true,
    "correlation": true,
    "KGESS": true,
    "kappa_coeff": false,
    "rv": false,
    /*…*/
  },
  /*评分方法：定义评分指标及其开关，
  基础设置中评估模块开启，
  所有评估项目均会计算以下被设为true的评分指标，
  包括Overall_Score*/
  "scores": {
    "nBiasScore": false,
    "nRMSEScore": false,
    "nPhaseScore": false,
    "nIavScore": false,
    "nSpatialScore": false,
    "Overall_Score": true,
    "The_Ideal_Point_score": false
  },
  /*比较分析：定义多模式比较时所使用比较分析方法，
  基础设置中比较模块开启，
```

```
  所有评估项目的所有统计指标和评分方法均会使用以下定义为true 的比较方法，
  包括HeatMap、Kernel_Density_Estimate、Single_Model_Performance_Index、
  Ridgeline_Plot 和Standard_Deviation*/
  "comparisons": {
    "HeatMap": true,
    "Taylor_Diagram": false,
    "Target_Diagram": false,
    "Kernel_Density_Estimate": true,
    "Whisker_Plot": false,
    "Parallel_Coordinates": false,
    "Portrait_Plot_seasonal": false,
    "Single_Model_Performance_Index": true,
    "Relative_Score": false,
    "Ridgeline_Plot": true,
    "Diff_Plot": false,
    "Mean": false,
    "Median": false,
    "Min": false,
    "Max": false,
    "Sum": false,
    "Mann_Kendall_Trend_Test": false,
    "Correlation": false,
    "Standard_Deviation": true,
    "Functional_Response": false
  },
  /*统计分析：定义统计分析方法，
  基础设置中统计模块开启，
  所有评估项目的所有统计指标和评分方法均会使用以下定义为true 的统计方法，
  此案例虽在基础设置中开启统计模块，
  但未设定任何一种统计方法*/
  "statistics": {
    "Z_Score": false,
    "Hellinger_Distance": false,
    "Three_Cornered_Hat": false,
    "Partial_Least_Squares_Regression": false,
    "False_Discovery_Rate": false,
    "ANOVA": false
  }
}
```

4 模型数据配置文件（sim.json）

模型数据配置文件（**sim.json**），定义了模型输出数据的位置、格式、变量名称等信息,是系统正确读取和处理模型数据的关键。同主配置文件一致，文件采用 json 格式。

sim-Debug_comment_ZH.json 示例：

```
{
/*模型数据配置文件：定义模型输出数据的名称和位置，
具体数据的格式、变量名称等需到对应模型输出数据的配置文件中修改，
如此处需修改对应grid_case.json 和stn_case.json*/
  /*基础设置：定义模型输出数据，对应评估项目的模型模拟数据源，
  只有在main 中打开的评估项目需要给出模型模拟数据源，
  如此处有两个模型输出数据分别为区域模型输出案例（grid_case），
  单点模型输出案例（stn_case）*/
  "general": {
    "Evapotranspiration_sim_source": [
      "grid_case",
      "station_case"
    ],
    "Latent_Heat_sim_source": [
      "grid_case",
      "station_case"
    ],
    "Sensible_Heat_sim_source": [
      "grid_case",
      "station_case"
    ]
  },
  /*数据源设置：定义模型输出数据的配置文件，
  详细配置请按照对给定路径文件进行设定*/
  "def_nml": {
    "grid_case": "./nml/user/debug/grid_case.json",
    "station_case": "./nml/user/debug/station_case.json"
  }
}
```

其中“def_nml“中所给出“grid_case”和“station_case“是模式运行的

实例，可以是任意名称，只要和“general”里面的名称保持一致即可。每个实例都有自己的配置文件,定义该模型输出数据的详细信息。如以上案例给出两个实例数据配置。

grid_case_comment_ZH.json 示例:

```
{
/*具体模型数据配置文件（grid_case）：
定义某一所需比较的具体模型模拟结果格式*/
  "general": {
    "model_namelist": "./nml/Mod_variables_defination/CoLM.json",
    //所使用模式，此处为CoLM 模式结果，已提前预设好多种模式格式，
    可参考./nml/Mod_variables_defination/下对应模式
    "timezone": 0.0,
    //模式结果所使用时区
    "data_type": "grid",
    //数据类型（"grid"或"stn"）
    "data_groupby": "month",
    //数据聚合类型（如grid 则为"year","month","day"，
    如stn 则为"single"）
    "fulllist": "",
    //stn 专属，数据的全部列表，如grid 则设为""
    "tim_res": "month",
    //时间分辨率（"year","month","day"）
    "grid_res": 2.0,
    //空间分辨率（度）
    "suffix": "",
    //数据名称时间后缀（此处数据全称格式为grid_case_hist_2004-01.nc，
    无后缀）
    "prefix": "grid_case_hist_",
    //数据名称时间前缀（此处数据全称格式为grid_case_hist_2004-01.nc，
    前缀为grid_case_hist_）
    "syear": 2004,
    //开始年份（开始于该年1 月）
    "eyear": 2005,
    //结束年份（结束于该年12 月）
    "root_dir": "./data/simulation/debug/grid"
    //数据存放路径
  }
}
```

5 参考数据配置文件（ref.json）

参考数据配置文件（**ref.json**），定义了用于评估模型性能的参考数据的来源、格式、变量名称等信息。这些参考数据可能包括实地观测、卫星数据、再分析数据等。同样，文件采用 json 格式.

ref-Debug_comment_ZH.json 示例：

```
{
/*参考数据配置文件：定义用于评估模型性能的参考数据的名称和位置，
具体数据的格式、变量名称等需到对应参考数据的配置文件中修改，
OpenBench 中已预设好大量模型参考数据配置，
如使用对应参考数据可直接使用预设参考数据配置，
详细可参考 ref.json 中参考数据预设路径及对应配置，
同时可对任意评估项目设置一个或多个参考数据，
参考数据可包括实地观测、卫星数据、再分析数据等*/
  /*基础设置：定义每个评估项目的参考数据源，
  只有在 main 中打开的评估项目需要给出参考数据源，
  不同评估项目的来源可以不一致*/
  "general": {
    "Evapotranspiration_ref_source": [
      "GLEAM4.2a_monthly",
      "GLEAM_hybird_PLUMBER2"
    ],
    "Latent_Heat_ref_source": [
      "ILAMB_monthly",
      "PLUMBER2"
    ],
    "Sensible_Heat_ref_source": [
      "ILAMB_monthly",
      "PLUMBER2"
    ]
  },
  /*数据源设置：定义上述所给定参考数据源的具体配置，
  详细配置请按照对给定路径文件进行设定*/
  "def_nml": {
    "GLEAM_hybird_PLUMBER2":
"./nml/Ref_variables_defination/Debug/GLEAM_hybird_PLUMBER2.json",
    "ILAMB_monthly":
"./nml/Ref_variables_defination/Debug/ILAMB_monthly.json",
```

```
    "PLUMBER2": "./nml/Ref_variables_defination/Debug/PLUMBER2.json",
    "GLEAM4.2a_monthly":
"./nml/Ref_variables_defination/Debug/GLEAM4.2a_monthly.json"
  }
}
```

参考数据集都有自己的配置块定义该数据集的详细信息，默认保存在 **$OpenBench/nml/Ref_variables_defination** 文件夹内。如以上案例中给出四个参考数据配置。

ILAMB_monthly_comment_ZH.json 示例：

```
{
/*具体参考数据配置文件（ILAMB_monthly）：
定义某一参考数据格式，
统一参考数据可包括多个评估项目，
此处比较Sensible_Heat 和Latent_Heat*/
  /*基础设置：设置该参考数据格式*/
  "general": {
    "root_dir": "./data/reference/debug/ILAMB",//数据存放路径
    "timezone": 0.0,                          //数据所使用时区
    "data_type": "grid",                      //数据类型（"grid"或"stn"）
    "data_groupby": "Single",                 //数据聚合类型（如grid 则为
"year","month","day"，如所有时间聚合在一起或stn 为"single"）
    "tim_res": "Month",                       //时间分辨率
（"year","month","day"）
    "grid_res": 2.0                           //空间分辨率（度），单点可给
为""
  },
  /*参考数据变量设置：
  设置该参考数据中所有比较变量的格式，
  如此处为grid 需设置Sensible_Heat 和Latent_Heat 的
  路径前缀、变量名、变量单位、文件名称、起止年份等*/
  "Sensible_Heat": {
    "sub_dir": "Sensible_Heat/FLUXCOM",
    "varname": "sh",
    "varunit": "w m-2",
    "prefix": "sh",
    "suffix": "",
    "syear": 2004,
    "eyear": 2005
```

```
  },
  "Latent_Heat": {
    "sub_dir": "Latent_Heat/FLUXCOM",
    "varname": "le",
    "varunit": "w m-2",
    "prefix": "le",
    "suffix": "",
    "syear": 2004,
    "eyear": 2005
  }
}
```

如果需要加入新的自定义参考数据集,则需要创建一个自定义的变量定义文件，命名可以随意，内容可以参考上述 ILAMB_monthly 变量定义文件。

6 输出数据与图表

评估结果将保存于**$OpenBench/output/basename** 下，以 Debug 案例为例，其输出结果保存于**$OpenBench/output/Debug** 文件夹下，包括:

（1）log/：包含 OpenBench 评估过程日志

（2）ouput/comparisons/：包含比较分析结果

（3）output/data/：包含所使用各模型数据和参考数据针对评估项目的数据处理结果，若为站点数据则包含站点的时间序列折线图

（4）output/metrics/：包含各种评估指标结果

（5）output/scores/：包含各种评分指标结果

（6）scratch/：预处理临时文件，如按年分割的数据

（7）tmp/：中间处理临时文件，如重采样后的数据

output 目录下包含相应结果图表和数据文件。

第四部分 研发用户指南

7 高级用法

7.1 自定义评估指标

用户可以通过修改 **Mod_Metrics.py** 和 **Mod_Scores.py** 来添加新的评估指标和评分方法。

7.2 并行计算

通过在 **main.json** 中设置 num_cores，可以启用并行计算以提高性能。

7.3 自定义数据处理

如需添加对新数据格式的支持，可以修改 **Mod_DatasetProcessing.py**。

8 常见问题

Q: 如何添加新的评估变量?

A: 在 main.json 的 “evaluation_items” 部分添加新变量，并确保在 “simulation_nml” 和 “reference_nml” 中提供相应的数据路径。

Q: 如何处理不同时间分辨率的数据?

A: OpenBench 会自动将数据重采样到 “compare_tim_res” 指定的时间分辨率。确保在配置文件中正确设置了原始数据的时间分辨率。

Q: 安装系统依赖后，第一次运行评估系统时提示 Cartopy 错误?

A：安装 Cartopy 依赖后，第一次运行请在联网状态下运行，如无法联网，则可按故障排除中教程手动下载。

9 故障排除

- **如果遇到内存错误，尝试减少 num_cores 或使用更小的数据集进行测试。**
- **确保所有输入数据路径在配置文件中正确设置。**
- **检查 Python 版本和所有依赖库是否已正确安装。**
- **OpenBench 在第一次运行时， Cartopy 需下载相应地理数据，如无法联网，请遵照以下操作流程手动下载：**

（1）首先确定 Cartopy 数据存储目录：

```
import cartopy
print(cartopy.config['data dir'])

# 通常路径可能是：/home/username/.local/share/cartopy
```

（2）在可联网的机器上访问 Natural Earth 官网下载数据(Cultural、Physical)：

Natural Earth » Downloads - Free vector and raster map data at 1:10m, 1:50m, and 1:110m scales

（3）需要下载的核心文件（110m 分辨率）：

- 海岸线：**ne_110m_coastline.zip**
- 国界线：**ne_110m_admin_0_boundary_lines_land.zip**
- 陆地：**ne_110m_land.zip**
- 海洋：**ne_110m_ocean.zip**
- 湖泊：**ne_110m_lakes.zip**

（4）目录结构准备：

```
└── cartopy_data_dir/  # 上一步打印的目录
```

```
    ├── shapefiles/
    │   ├── natural_earth/
    │   │   ├── cultural/
    │   │   └── physical/
    └── raster/
        └── natural_earth/
```

（5）手动安装步骤：

```
# （请替换实际路径）：

cd /home/username/.local/share/cartopy
mkdir -p shapefiles/natural_earth/physical
mkdir -p shapefiles/natural_earth/cultural
# 解压下载的文件到对应目录
unzip ne_110m_coastline.zip -d shapefiles/natural_earth/physical/
unzip ne_110m_admin_0_boundary_lines_land.zip -d shapefiles/natural_eart
h/cultural/
```

（6）在代码中强制使用本地数据：

```
import cartopy
import cartopy.feature as cfeature
# 强制使用本地数据（避免尝试下载）
coastline = cfeature.NaturalEarthFeature(
    category='physical',
    name='coastline',
    scale='110m',
    facecolor='none'
)
ax.add_feature(coastline, edgecolor='black')
```

（7）验证安装：

```
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111, projection=ccrs.PlateCarree())
ax.coastlines()
plt.show()
```

（8）对于完全无网络环境，建议额外下载：

- **ne_110m_graticules_*.zip**（经纬网格）
- **ne_110m_geography_*.zip**（地理标记）

- **ne_110m_admin_*.zip**（行政区划）

注意事项：

- 保持文件目录结构与 Cartopy 预期一致
- 所有解压后的.shp 文件应直接在 **physical/cultural** 目录下
- 团队使用时可将该目录打包共享，统一解压到各用户的 cartopy 数据目录
- 110m 分辨率适合全球绘图，如需更高精度可下载 50m 或 10m 数据