Binning Across a Series of AssembLies Toolkit (BASALT)

配置及使用中文说明（BASALT-1.2.0 版，2026 年 01 月）

✧ **BASALT GitHub 页面：https://github.com/EMBL-PKU/BASALT**

✧

✧ **BASALT 在线文档：https://basalt-guide.readthedocs.io/en/latest/**

✧

✧ **PKU EMBL 课题组的英文官方网站: https://embl-pku.github.io/**

(一) BASALT 更新日志

[2025/12/16]:🤗 我们在 MIT 许可证下正式发布 BASALT V1.2.0。

- Python 版本升级： 将 Python 版本要求从 3.8 升级至 3.12，并提供了全新的、更友好的安装脚本。
- 新增分箱算法： 在 BASALT 工具包的 Extra Binner 模块中加入了 LorBin [Nature Communications, 2025]，这是目前最前沿的分箱（Binning）工具。
- 质控软件变更： 将默认的质量控制（QC）评估软件从 CheckM 更改为 CheckM2 v1.1.0。
- GPU 加速支持： 支持对 BASALT 中的 Semibin2 和深度学习模型进行 GPU 加速。
- 权重路径自定义： 通过配置 ~/.bashrc 文件，BASALT 中深度学习模型的权重文件现在可以存放在任意路径，不再强制局限于 ~/.cache 目录。

[2024/06/12]:🤗 我们在 MIT 许可证下正式发布 BASALT V1.1.0。

[2024/03/11]:🤗 研究论文发表。 题为 “BASALT refines binning from metagenomic data and increases resolution of genome-resolved metagenomic analysis”（BASALT 优化宏基因组数据分箱并提高基因组解析宏基因组分析的分辨率）的文章已在 《Nature Communications》 上发表。

[2023/8/18]:🤗 我们在 MIT 许可证下正式发布 BASALT V1.0.0。

(二) BASALT 支持的数据类型及特点

BASALT 是一个多功能工具包，支持从（1）纯二代序列的组装序列、（2）纯三代序列组装序列（现阶段主要支持 PacBio-HiFi 数据，其他纯三代数据如 ONT 因覆盖率计算的问题仍在评估中，约在稍晚开放支持）及（3）二代加上三代数据的混合组装的一系列数据中获取高质量 MAGs。BASALT 具有以下几个的特点：

1） BASALT 支持多组装文件的相互精修

已有的软件如 MetaWRAP 及 DASTool 仅支持单一组装文件的分箱，并在多组装文件出现时利用去冗余软件，如 dRep 进行去冗余。部分分箱软件，如 VAMB 等，也具有多组装文件共同输入的功能，但我们的实测结果表明这些软件做多组装文件时的分箱效果，不如它们做单一分箱的效果。在总结我们过往的测试结果的基础上，BASALT 开发了一套通过多个组装文件同时分箱后，利用我们自主开发的覆盖率一致性算法识别冗余基因组，挑选质量最好

的基因组，并且进一步将冗余基因组对选定的最好质量的基因组进行修正。这里面包括通过冗余基因组中存在的片段补全被选定的基因组，及利用冗余基因组对选定基因组进行补洞（gap）。故理论上使用了 BASALT 后不需要再用 dRep 进行去冗余，但用户可根据自己的需求进行修改。

2） 基于深度学习的修正（refinement）模块可大幅度的提高基因组质量

结合自主开发的神经网络的算法，BASALT 可高效率的识别潜在错误片段，并进过自动评估后去除潜在的错误片段。在过去我们对 Binning refinner 和 RefineM 评估中，BASALT 的修正效果均大幅度好于这两个软件。由于 metaWRAP 内置的修正软件所用的是 Binning refiner, metaWRAP 在修正上差于 BASALT。DASTool 与 BASALT 比较的结果也显示 BASALT 修正的效果好于 DASTool。并且，这一模块可独立使用，如用户有一组已经完成分箱的基因组，可通过 BASALT 内的 data feeding 脚本将基因组和原始数据提供于 BASALT，进行覆盖率等计算，随后用 refinement 模块进行修正。

3） BASALT 对 polish 流程及重组装进行大幅度优化

在二代及三代数据存在的情况下，BASALT 强化了修正功能，及我们整合了利用 pilon 等软件的 polish 的功能。在这一过程中，BASALT 进行了大量 polishing 的效率优化，在运行相同的 polish 次数（run）的情况下，BASALT 可以比用户单独使用 polish 软件节省约 90% 的 polish 的时间。BASALT 对各种情况下的数据进行重组装的优化。尤其是对 2/3 代混装的效率和质量均达到了已知最好的水平。纯 3 代数据的优化仍在进行中，该功能会在稍晚释放给用户使用。我们仍需提醒用户一点，即重组装可大幅度提高基因组质量，但同时也会导致获取基因组的时间增长非常多。我们在 2+3 代混合重组装时测试了 Opera-MS/Unicycler 及 SPAdes 等软件，最终选择 SPAdes 为默认的重组装软件，这也是基于效率的考虑。事实上，Unicycler 在单菌 2+3 代混装下的重组装效果远远好于 SPAdes，但 Unicycler 重组装的耗时为 SPAdes 数倍。故在重组装过程中，我们选择了 SPAdes 作为默认软件，Unicycler 为可选的软件。

BASALT 已经可以比较好的支持纯 2 代数据的分箱分析及 2+3 的混装数据的分析，但基于纯 3 代测序数据组装文件的分析仍在开发中。BASALT 仍有大量改进空间，我们将进一步解决 3 代测序数据的分析问题及效率问题。感谢用户的耐心，期望大家和我们一同提出 BASALT 的问题，如遇到问题，请直接和我们联系，可发邮件至 yuke.sz@pku.edu.cn 进行反馈，我们将陆续完善整个软件包的功能。谢谢！

(三) 依赖程序：

1) 系统及脚本语言

Linux x64 系统, Python (>=3) , Perl, Java (>=1.7)

Python 附加脚本包：biopython, pandas, numpy, scikit-learn, copy, multiprocessing, collections，pytorch, tensorboardx

2) 内置的自动分箱软件

MetaBAT2, Maxbin2, CONCOCT, Semibin2, Metabinner, LorBin（可选，默认不使用）

*VAMB 也存在于 BASALT 环境中，但内部测试的结果显示 VAMB 的效果并不是很好，故我们暂时阻隔了通过 BASALT 对该软件的调用。若想使用 VAMB 的用户，可以自行安装 VAMB 进行分箱后，使用 data feeding 将分箱结果输入至 BASALT 中。由于 Python 版本支持的问题, MetaBinner 在 BASALT v1.2.0 中将不再支持, 如果想要继续使用 MetaBinner 可以使用 BASALT v1.1.0 的版本。Lorbin 属于 extrabinner 的部分如果选择使用 Lorbin 的话请参考 Lorbin 的 GitHub 官方文档 https://github.com/LorMeBioAI/LorBin 进行下载部署。

3) 映射、解析、预测及比对软件

Bowtie2, BWA, SAMtools, Prodigal, BLAST+, HMMER, Minimap2

4) 组装及碱基修正软件

SPAdes, IDBA-UD, Pilon, Unicycler（可选，默认不使用）

5) 基因组评估软件

CheckM：BASALT v1.1.0 默认使用 CheckM., 可选使用 CheckM2. CheckM 数据库, pplacer

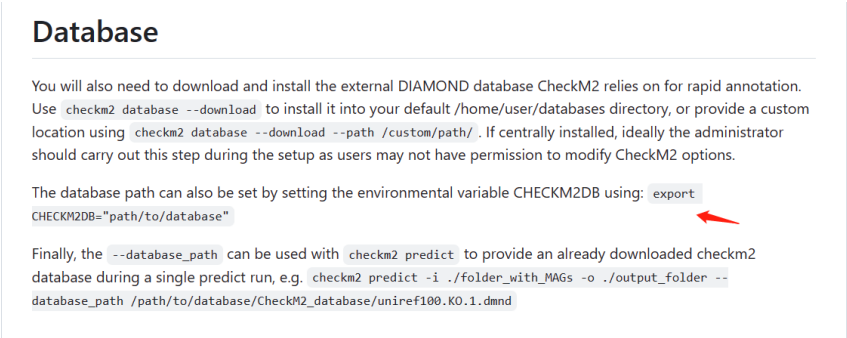
CheckM2：BASALT v1.2.0 默认使用 CheckM2, 可选使用 CheckM

BASALT 中已包含 CheckM2 的主程序，但 BASALT 环境中没有默认下载 CheckM2 数据库。

*由于 CheckM2 为基于神经网络设计的程序；我们在以标准基因组进行评估的过程中发现 CheckM2 与 CheckM 有大量的偏差，故现阶段我们仍倾向于使用 CheckM 而非 CheckM2 进行质量评估。未来在 CheckM2 更加成熟，或有可替代软件的情况下将会进行修正。用户如需使用 CheckM2, 可使用参数-qc 进行选择，详见下文说明

*BASALT 环境中没有配置 CheckM2 数据库。用户需手动配置 CheckM2 的数据库

（https://github.com/chklovski/CheckM2，可参考 CheckM2 页面）。



Database

You will also need to download and install the external DIAMOND database CheckM2 relies on for rapid annotation. Use `checkm2 database --download` to install it into your default /home/user/databases directory, or provide a custom location using `checkm2 database --download --path /custom/path/`. If centrally installed, ideally the administrator should carry out this step during the setup as users may not have permission to modify CheckM2 options.

The database path can also be set by setting the environmental variable CHECKM2DB using: `export CHECKM2DB="path/to/database"`

Finally, the `--database_path` can be used with `checkm2 predict` to provide an already downloaded checkm2 database during a single predict run, e.g. `checkm2 predict -i ./folder_with_MAGs -o ./output_folder --database_path /path/to/database/CheckM2_database/uniref100.KO.1.dmnd`

6) 工作站硬件要求

8+内核，128GB+RAM；暂不支持显卡加速，该功能仍在开发中，将在后续版本推出

(四) BASALT v1.2.0 安装方法

1) BASALT v1.2.0 conda 安装指令

```
git clone https://github.com/EMBL-PKU/BASALT.git

cd BASALT

conda create -n basalt_env -c conda-forge -c bioconda \
    python=3.12 \
    megahit metabat2 maxbin2 concoct prodigal semibin \
    bedtools blast bowtie2 diamond checkm2 \
    unicycler spades samtools racon pplacer pilon \
    ncbi-vdb minimap2 miniasm idba hmmer entrez-direct \
    biopython uv --yes

conda activate basalt_env

uv pip install tensorflow torch torchvision tensorboard tensorboardx \
    lightgbm scikit-learn numpy==1.26.4 scipy pandas matplotlib \
    cython biolib joblib tqdm requests checkm-genome
```

2) 下载 BASALT 深度学习模型权重

```
# please chanage the download path according to your computer

python download.py --path "your_model_folder"
```

这里的 path 是自定权重下载的路径，这里的路径与 4)的设置 BASALT 的~/.bashrc 文件直接相关请确认好一致性, path 默认路径是当前文件夹./BASALT_WEIGHT/

3) 下载 BASALT 源文件并给源文件赋权

```
chmod +x install.sh

bash install.sh
```

chmod +x /path/to/basalt/bin/*

4) 设置 BASALT 环境变量在~/.bashrc 文件当中

nano ~/.bashrc

export CHECKM2DB=/path/to/checkm2db/CheckM2_database/uniref100.KO.1.dmnd

export CHECKM_DATA_PATH=/path/to/checkmdb

export BASALT_WEIGHT=/path/to/BASALT

source ~/.bashrc

CHECKM2DB, CHECKM_DATA_PATH 和 BASALT_WEIGHT 位置根据自己服务器的路径设计即可, CHECKM2DB, CHECKM_DATA_PATH 的数据库请参考 CheckM2 的 GitHub 官方文档 https://github.com/chklovski/CheckM2 和 CheckM 的 GitHub 官网文档 https://github.com/Ecogenomics/CheckM

我们也提供了 CHECKM2DB CHECKMDB BASALT_WEIGHT 的 Google Drive 多权重链接:

https://drive.google.com/drive/folders/1d0e_2FpYRBAZLwKXl8fA-yDK4b5PBA_E?usp=sharing

5) 关于 Lorbin 支持问题

Lorbin 属于 extrabinner 的部分如果选择使用 Lorbin 的话请参考 Lorbin 的 GitHub 官方文档 https://github.com/LorMeBioAI/LorBin 进行下载部署

6) [另外一种使用方式] 利用 singularity 运行预安装好的 BASALT v1.2.0

为方便用户使用，我们提供了无需安装的 BASALT v1.2.0 singularity 版本，用户只需下载预安装好的 BASALT，利用 singularity 即可直接运行。

- 通过 Google Drive 的网址下载 basalt.sif 文件:

  https://drive.google.com/drive/folders/1d0e_2FpYRBAZLwKXl8fA-yDK4b5PBA_E?usp=sharing

- 运行命令：

  singularity run basalt.sif BASALT -h

- 若 BASALT.sif 文件不在/home 目录下，则需要添加-B 进行挂载，命令如下：

  singularity run -B /path/to/folder basalt.sif BASALT -h

- 运行 singularity 版本时，利用集群 PBS 等提交任务的方法可以正常运行，但直接利用 nohup 后台挂载运行时程序可能会意外退出。如果直接

运行的话，可以使用 screen 方式运行，如：

```
screen -dmS [SESSION_NAME] bash -c 'bash BASALT.sh > BASALT.log'
```

- basalt.sif 中已安装各种需要的环境及软件，因此除运行 BASALT 外，还可以调用各种软件，如：

```
singularity run basalt.sif bowtie2 -h
singularity run basalt.sif checkm2 -h
```

(五) BASALT v1.1.0 安装方法

1) 外网的用户推荐使用 BASALT_setup_international.py 安装

- 下载 BASALT_setup.py
- 通过该命令下载软件：`python BASALT_setup.py`（该过程耗时较长，请耐心等待）

*注：该脚本已内置了 BASALT 的环境配置、主程序及模型的下载及安装

2) 内网且无法翻墙的用户推荐使用 BASALT_setup_China_mainland.py 安装

- 下载 BASALT_setup_China_mainland.py
- 通过该命令下载软件：`python BASALT_setup_China_mainland.py`（该过程耗时较长，请耐心等待）
- 下载神经网络训练模型:

从腾讯微云下载 BASALT.zip 并上传至您的 Linux 路径：https://share.weiyun.com/r33c2gqa ，随后依次执行以下三个命令

```
mv BASALT.zip ~/.cache
cd ~/.cache
unzip BASALT.zip
```

*注：由于国内访问 Github 时常受限，国内用户可使用以下的手动安装方式进行 BASALT 的安装。

3) 手动配置 BASALT 环境（选择一）（国内用户推荐用该方式安装）

由于需要安装许多依赖项，推荐使用 conda 安装和管理 BASALT。安装 Miniconda 或 Anaconda 后，创建一个 conda 环境，并在特定环境中安装 BASALT。

- （可选）添加镜像以提高下载速度，如国内用户可用清华的 conda 镜像进行安装：

  site=https://mirrors.tuna.tsinghua.edu.cn/anaconda

  conda config --add channels ${site}/pkgs/free/

  conda config --add channels ${site}/pkgs/main/

  conda config --add channels ${site}/cloud/conda-forge/

  conda config --add channels ${site}/cloud/bioconda/

- 从 Github 克隆 BASALT 仓库并配置 BASALT 环境

  git clone https://github.com/EMBL-PKU/BASALT.git

  cd BASALT

  conda env create -n BASALT --file basalt_env.yml

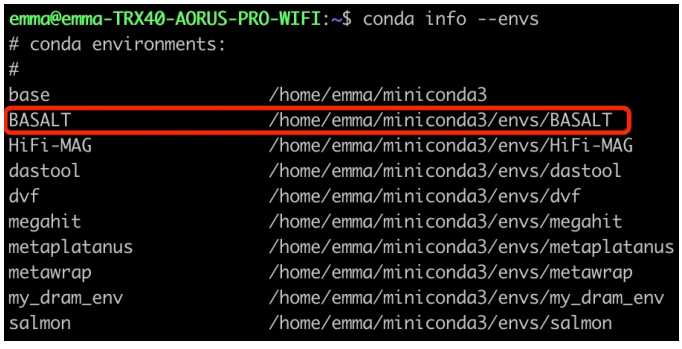
  conda create 的命令耗时较长，需要耐心等待

  * 注：因为 github 在国内访问不稳定，我们同时将环境文件：basalt_env.yml 存至微云之中，国内用户可通过共享文件夹下载 basalt_env.yml 文件：https://share.weiyun.com/xXdRiDkl，在下载完成之后，可参考上面缩写的 cd BASALT 命令，及 conda create 命令进行操作

- 授权 BASALT 脚本。一般来说 conda BASALT 环境位于 conda 安装的子目录 envs 中（例如：/home/anaconda2/envs）

  chmod -R 777 your_conda/envs/BASALT/bin/*

  *注：命令举例：conda 中 BASALT 环境所在地址可以使用以下方式进行查看：conda info –envs，在其中找到 BASALT 所在的地址，如下图：



  其安装的子目录为/home/emma/miniconda3/envs/BASALT，故在这一情况下使用的命令为：chmod -R 777 /home/emma/miniconda3/envs/BASALT /bin/*

- 下载神经网络训练模型:

  预先下载好 BASALT_models_download.py 脚本，并执行下列命令

python BASALT_models_download.py

*注：BASALT_models_download.py 可通过 github 直接获得，为保证国内用户可以下载该文件，可访问 https://share.weiyun.com/xXdRiDkl，从中下载该文件。

BASALT_models_download.py 命令在国内可能会失效，国内安装推荐使用以下的方法一进行安装，国外安装也可用以下方法二进行手动安装：

方法一（内网）：

从腾讯微云下载 BASALT.zip 并上传至您的 Linux 路径：https://share.weiyun.com/r33c2gqa

mv BASALT.zip ~/.cache

cd ~/.cache

unzip BASALT.zip

方法二（外网）：

wget https://figshare.com/ndownloader/files/41093033

mv 41093033 BASALT.zip

mv BASALT.zip ~/.cache

cd ~/.cache

unzip BASALT.zip

4) 手动配置 BASALT 环境（选择二）（国内用户推荐用该方式安装）

由于需要安装许多依赖项，推荐使用 conda 安装和管理 BASALT。安装 Miniconda 或 Anaconda 后，创建一个 conda 环境，并在特定环境中安装 BASALT。

- 从 Github 克隆 BASALT 仓库

  git clone https://github.com/EMBL-PKU/BASALT.git

  cd BASALT

- 修改环境配置文件内容

  删除 basalt_env.yml 文件 dependencies 部分的 checkm-genome=1.1.3=py_1 和 metabinner=1.4.4=hdfd78af_0 两行内容并保存

- 配置 BASALT 环境

  conda env create -n BASALT --file basalt_env.yml

  conda create 的命令预计耗时 20 分钟，请耐心等待

- 下载 checkm 数据库文件，并于 BASALT 环境目录下创建 checkm_data 文件夹，

然后在路径下解压 checkm 数据库。

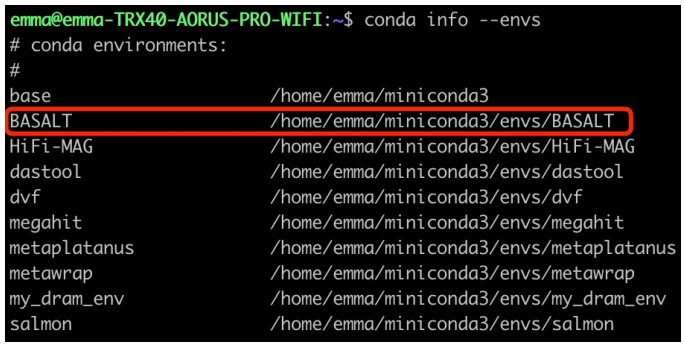
wget

https://data.ace.uq.edu.au/public/CheckM_databases/checkm_data_2015_01_16.tar.gz

cd your_conda/envs/BASALT/checkm_data

tar -zxvf checkm_data_2015_01_16.tar.gz

*注：一般来说 conda BASALT 环境位于 conda 安装的子目录 envs 中（例如：/home/anaconda2/envs）conda 中 BASALT 环境所在地址可以使用以下方式进行查看：conda info –envs，在其中找到 BASALT 所在的地址，如下图：



其安装的子目录为/home/emma/miniconda3/envs/BASALT

- 在激活的 BASALT 环境里安装 metabinner

  conda activate BASALT

  conda install metabinner=1.4.4=hdfd78af_0 -y

  *注：因为 metabinner 软件依赖 checkm-genome 运行，所以安装 metabinner 的过程中也会安装 checkm-genome

- 授权 BASALT 脚本

  unzip BASALT_script.zip

  mv BASALT_script.zip/* your_conda/envs/BASALT/bin/

  chmod -R 777 your_conda/envs/BASALT/bin/*

- 下载和配置神经网络训练模型

  mv BASALT.zip ~/.cache

  cd ~/.cache

  unzip BASALT.zip

5) 使用 mamba 配置 BASALT 环境（国内用户推荐用该方式安装，速度较快）

- 从 Github 克隆 BASALT 仓库

```
git clone https://github.com/EMBL-PKU/BASALT.git
cd BASALT
```

- （可选）修改镜像以提高下载速度，如国内用户可用清华的 pip 镜像

```
pip config set global.index-url https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple
```

- 配置 BASALT 环境

```
mamba env create -n BASALT --file basalt_env.yml
```

mamba create 的命令预计耗时 10 分钟，请耐心等待

*注：安装过程中可能会遇到 semibin 依赖 python-igraph 下载不完全的问题，可以通过下面的命令手动下载：

```
pip install python-igraph
```

- 授权 BASALT 脚本

```
unzip BASALT_script.zip
mv BASALT_script.zip/* your_conda/envs/BASALT/bin/
chmod -R 777 your_conda/envs/BASALT/bin/*
```

- 下载和配置神经网络训练模型

```
mv BASALT.zip ~/.cache
cd ~/.cache
unzip BASALT.zip
```

6) 利用 singularity 运行预安装好的 BASALT

为方便用户使用，我们提供了无需安装的 BASALT singularity 版本，用户只需下载预安装好的 BASALT，利用 singularity 即可直接运行。

- 通过腾讯微云的网址下载 BASALT.sif 文件:

https://share.weiyun.com/xKmoBmrF

- 运行命令：

```
singularity run BASALT.sif BASALT -h
```

- 若 BASALT.sif 文件不在/home 目录下，则需要添加-B 进行挂载，命令如下：

```
singularity run -B /path/to/folder BASALT.sif BASALT -h
```

- 运行 singularity 版本时，利用集群 PBS 等提交任务的方法可以正常运行，但直接利用 nohup 后台挂载运行时程序可能会意外退出。如果直接运行的话，可以使用 screen 方式运行，如：

screen -dmS [SESSION_NAME] bash -c 'bash BASALT.sh > BASALT.log'

- BASALT.sif 中已安装各种需要的环境及软件，因此除运行 BASALT 外，还可以调用各种软件，如：

  singularity run BASALT.sif bowtie2 -h

  singularity run BASALT.sif checkm2 -h

7) 测试文件

我们准备了测试文件以测试 BASALT 是否成功安装并且可以顺利运行。测试文件分为三个部分：测试数据（Data.tar.gz）、用作对照看软件是否安装成功的测试结果（Final_bestbinset.tar.gz）和执行命令脚本（basalt.sh）

- 下载测试文件并运行程序

  从 https://figshare.com/articles/dataset/BASALT_demo_files/22323424 下载测试文件（共三个：Data.tar.gz、Final_bestbinset.tar.gz、basalt.sh），然后执行下面的命令运行程序

  bash basalt.sh （该过程耗时大约六小时，请耐心等待）

(六) 软件使用

1) 参数简述

以 python 执行 BASALT.py 脚本进行操作；BASALT.py 为 BASALT 主程序。如果使用 conda 环境，运行时直接输入 BASALT 及相关参数；如果单独使用环境，使用 python BASALT.py 及相关参数运行程序。以 conda 环境为例，若需帮助，可使用 BASALT -h 查看运行参数。

a) 必须参数：

-a 输入组装文件，组间以英文逗号分隔‘,’不可出现空格。

BASALT 支持用户以.fa, .fna, .fasta, .gz, .tar.gz, .zip 等格式如数序列文件。

示例： -a assembly1.fa,assembly2.fa

- BASALT 支持单一组装文件或多组装文件进行 binning；在样品的物种非常相似，但群落中物种丰度不同的情况下，如时间序列样品，多组装文件可大幅度提高 binning 的效果，但也可能会导致运算时间过长，但我们依然推荐大家用多组装文

件模式进行分箱分析。如果运算资源不足，推荐使用多组数据进行合并组装的组装文件进行后续的分箱分析。详细可参考下文中的应用案例。

-s 输入二代测序 paired-end 数据，多组数据间要用‘/’间隔不同组的文件，‘/’或‘，’前后不可出现空格; BASALT 支持用户以.gz, .tar.gz, .zip 等格式如数序列文件

示例：-s d1_r1.fq,d1_r2.fq/d2_r1.fq,d2_r2.fq

✧ BASALT 支持.tar.gz 等文件的输入，但是最好直接输入未压缩的序列文件

-l 输入三代测序数据，组间以英文逗号分隔，不可出现空格；BASALT 支持用户以.gz, .tar.gz, .zip 等格式如数序列文件

示例：-l lr1.fq,lr2.fq

✧ BASALT 支持.tar.gz 等文件的输入，但是最好直接输入未压缩的序列文件

-t 输入线程数，示例： -t 32

-hf 输入 HiFi 数据，组间以英文逗号分隔 ’,’ 不可出现空格

示例： -hf hifi1.fq,hifi2.fq

-m 输入内存量，示例：-m 128，最小内存量建议设置为 32G

b) 可选参数

-q 或--quality-check 设定 quality check 软件的参数运行。

默认软件为 CheckM，若需要使用，可用以下命令调用

示例：-q checkm2

✧ 虽然 checkm2 是新出的评估软件，但我们发现 checkm2 与 checkm 在评估一些新类群时差异非常巨大，在和很多国际同行，顶尖团队的成员，包括 GTDB 的开发者交流之后，我们认为现阶段 BASALT 的默认程序仍然用 CheckM 为默认软件更为妥当。用户如果需要使用 checkm2, 可以直接使用-q 参数进行设定

-e 设定额外分箱工具。

可选参数 m，表示设定额外分箱工具为 Metabinner

示例：-e m，代表 BASALT 将会在使用 MetaBAT2，Maxbin2，CONCOCT 及 Semibin2 作为分箱工具的同时也添加 Metabinner 对数据进行分箱分析

✧ 其他工具我们仍在评估中，估计 2024 年 7 月份评估结束会更新至 BASALT

-o 或--out，设定输出文件夹的名称。

BASALT 的默认输出文件夹名称为 Final_bestbinset,可通过-o 参数输出文件的名称。

示例：-o Human_gut，则输出文件最终的保存文件夹为 Human_gut_final_bestbinset

-d 或--data-feeding-folder，设定额外输入的文件夹的名称。

如用户用其他 binner 生成了一组基因组，BASALT 可通过-d 参数将这组基因组整合到 BASALT binning 的最终的基因组中。假设用户有一个文件夹，名为 Human_gut_microbiome, BASALT 将自动化的读取文件夹中的基因组，将其与 BASALT 的得到的基因组进行去冗余、选择最优基因组并进行修正，进而获得最终的精修基因组。

示例：-d Metawrap_bins

✧ -d 参数除了可以支持 BASALT 跑出的基因组直接和-d 提供的基因组进行融合。同时也支持用户对自己之前已经跑完的基因组利用 BASALT 进行修正。用户可以在将其他的基因组提供给 BASALT 之后，利用-b 参数对多个基因组文件夹进行去冗余；或利用-r 参数对这一文件夹内的基因组进行修正。详见下文示例。

-b 或--binsets-list，利用 BASALT 对多个 binsets 进行基因组的去冗余。

使用时需要将多个基因组文件夹输入给 BASALT。-b 参数需输入至少两个含有基因组的文件夹

示例：-b Metawrap_bins,DAStools_binset

✧ -b 参数功能类似于 drep。相比于 drep，BASALT 在近缘样品的去冗余上有统治性的优势。具体使用时，先需要将基因组利用 BASALT -d 参数将基因组文件做成 BASALT 所需要的文件夹，之后再用-b 文件去冗余。详见下文示例。

-r 或--refinement-binset，利用 BASALT 对多个 binsets 进行修正。

使用时需要将多个基因组文件夹输入给 BASALT。-r 参数只可输入单一文件夹。

示例：-r refinement_binset

✧ -r 参数功能用于包括 BASALT 自主产生或外源产生的基因组的修正工作。使用前先需要用-d 参数将这些基因组文件提交给 BASALT 进行基因组的 feeding,而后输入特定文件进行修正工作。详见下文示例。

--sensitive 基因获取及修正的敏感程度，包含三个模式：quick/sensitive/more-sensitive。

示例：--sensitive more-sensitive

✧ 现阶段 quick 模块包含的 binner 为 metabat2 及 semibin2；sensitive 模式含有 metabat2、semibin2 及 concoct；more-sensitive 模式中含有 metabat2、semibin2、concoct 及 maxbin2，并且在 more-sensitive 下的 refinement 功能模块中含有一个深度修正的

模块。虽然 more-sensitive 模式可以得到最多且质量最高的基因组，但因样品的差异我们很难确定和 sensitive 模式相比，more-sensitive 模式是否更好。在我们的测试里，在 more-sensitive 模式下，我们的数据可以获得更多的稀有古菌的基因组，但这存在很强的随机性，更多的时候，sensitive 模式与 more-sensitive 模式获得的基因组数量差别不大。但 sensitive 模式在复杂样品的情况下，耗时仅为 more-sensitive 模式不足一半，可大幅度的提高速度。近期我们将对软件，包括各种 binner 进行评测并整合至 BASALT 之中，预计在 2024 年 7 月的版本会进行一次较大的更新。现阶段，BASALT 的默认为 sensitive 模式。

--module 设定 BASALT 的工作模块。

BASALT 可单独进行 autobinning + bin selection，refinement 及这三个部分的独立运行，也可以以 all 进行完整的自动化 binning，默认模式为 all。

示例：--module autobinning

--min-cpn 设定保留的基因组的最小完整度，默认值为 35， 示例：--min cpn 30

--max-ctn 设定保留的基因组的最大污染度，默认值为 20，示例：--max-ctn 25

--mode 设定运行模式，预设 new 及 continue 两种模式。New 模式为重新开始 binning 项目，continue 为断点续跑跑上一次任务未完成的任务，默认模式为 continue

--refinepara 设定 refinement 模块的运行参数，预设 deep 和 quick 两个参数，deep 参数可将基因组修正的更好，但会相对耗费更多时间，默认参数为 deep

-h 帮助文档

2) 使用范例：

1）单独使用二代测序数据进行 binning 及修正

- BASALT -a as1.fa,as2.fa,as3.fa -s ds1_r1.fq,d1_r2.fq/d2_r1.fq,d2_r2.fq -t 60 -m 250

* 组装文件的顺序和序列的顺序并不重要，顺序可以随意放置

2）使用二代测序数据及三代测序数据进行 binning 及修正

- BASALT -a as1.fa,as2.fa,as3.fa -s ds1_r1.fq,d1_r2.fq/d2_r1.fq,d2_r2.fq -l lr1.fq,lr2.fq -t 60 -m 250

3）使用 hifi 数据进行 binning 及修正

- BASALT -a as1.fa -hf hifi1.fq -t 60 -m 250

4）使用 hifi 数据及二代测序数据进行 binning 及修正

- BASALT -a as1.fa -hf hifi1.fq -s ds1_r1.fq,d1_r2.fq -t 60 -m 250

5) 在 BASALT 使用 checkm2：

- BASALT -a as1.fa,as2.fa,as3.fa -s ds1_r1.fq,d1_r2.fq/d2_r1.fq,d2_r2.fq -t 60 -m 250 -q checkm2

6）使用 BASALT 进行基因组的获取，并且融合之前已有的基因组数据

- BASALT -a as1.fa,as2.fa,as3.fa -s ds1_r1.fq,d1_r2.fq/d2_r1.fq,d2_r2.fq -d m_binset -t 60 -m 250

7）使用 BASALT 进行外源数据基因组的去冗余:

a) 数据输入（Data feeding）

- BASALT -s sample1.R1.fq, sample1.R2.fq/sample2.R1.fq, sample2.R2.fq -d mbin,dbin -t 60 -m 250

-d 参数会产生一个 Date_feeded 文件夹，其中含有 BASALT 做后续工作需要的修改了 index 的组装文件，如下文中的 500_mbin.fa；基因组文件夹，如 500_mbin.fa_BestBinsSet; 覆盖率文件，如 Coverage_matrix_for_binning_500_mbin.fa.txt;及修改过格式的 reads 文件。

b) 如冗余（De-replication）

填入 BASALT 所需的多组基因组文件夹、多个组装文件、多对 reads 及覆盖率文件。该过程将产生一个去冗余后的基因组文件夹 BestBinset。

- BASALT -b 500_mbin.fa_BestBinsSet,501_dbin.fa_BestBinsSet -c Coverage_matrix_for_binning_500_mbin.fa.txt, Coverage_matrix_for_binning_501_mbin.fa.txt -a 500_mbin.fa,501_mbin.fa -s sample1.R1.fq, sample1.R2.fq/ sample2.R1.fq, sample2.R2.fq -t 60

8）使用 BASALT 进行基因组的修正:

在数据输入之后（data feeding），或者去冗余完成后,将基因组文件通过-r 参数输入到 BASALT 内。完成后脚本会生成 BestBinset_outlier_refined 文件夹

- BASALT -r BestBinset -c Coverage_matrix_for_binning_500_mbin.fa.txt, Coverage_matrix_for_binning_501_mbin.fa.txt -a 500_mbin.fa,501_mbin.fa -s PE_r1_sample1.R1.fq,PE_r2_sample1.R2.fq/PE_r1_sample2.R1.fq, PE_r2_sample2.R2.fq -t 60

3) 使用举例

a) **项目一**：水产养殖肠道样本组装及分箱策略

项目背景：采集三种不同水产养殖生物池塘中：养殖生物肠道、养殖水环境、养殖塘沉积物样本共计 27 份，进行宏基因组二代测序及三代测序。

组装策略：首先，通过 OPERA-MS 软件对二代测序数据及三代测序数据进行单个样本的混合组装；同时，对所获取的相同类型样品平行样的双端数据及三代测序数据合并后进行多个样本的混合组装，以获取该类型样本更多维度信息，便于后续分箱处理；在获取单个样本及多个平行样本混合的组装文件后，进行分箱处理。

示例——以 A 养殖塘水环境样本为例：

- 样本数据：

  sampleA_w1_R1.fq, sampleA_w1_R2.fq,

  sampleA_w2_R1.fq, sampleA_w2_R2.fq,

  sampleA_w3_R1.fq, sampleA_w3_R2.fq

  sampleA_w1_lr.fq

  sampleA_w2_lr.fq

  sampleA_w3_lr.fq

- 组装结果：

  sampleA_w1_assembly.fa

  sampleA_w2_assembly.fa

  sampleA_w3_assembly.fa

  sampleA_w1+A_w2+A_w3_assembly.fa

- 分箱命令：

  BASALT -a

sampleA_w1_assembly.fa,sampleA_w2_assembly.fa,sampleA_w3_assembly.fa,sampleA_w1+A_w2+A_w3_assembly.fa -s sampleA_w1_R1.fq,sampleA_w1_R2.fq/sampleA_w2_R1.fq,sampleA_w2_R2.fq/sampleA_w3_R1.fq,sampleA_w3_R2.fq -l sampleA_w1_lr.fq,sampleA_w2_lr.fq,sampleA_w3_lr.fq -t 60 -m 250

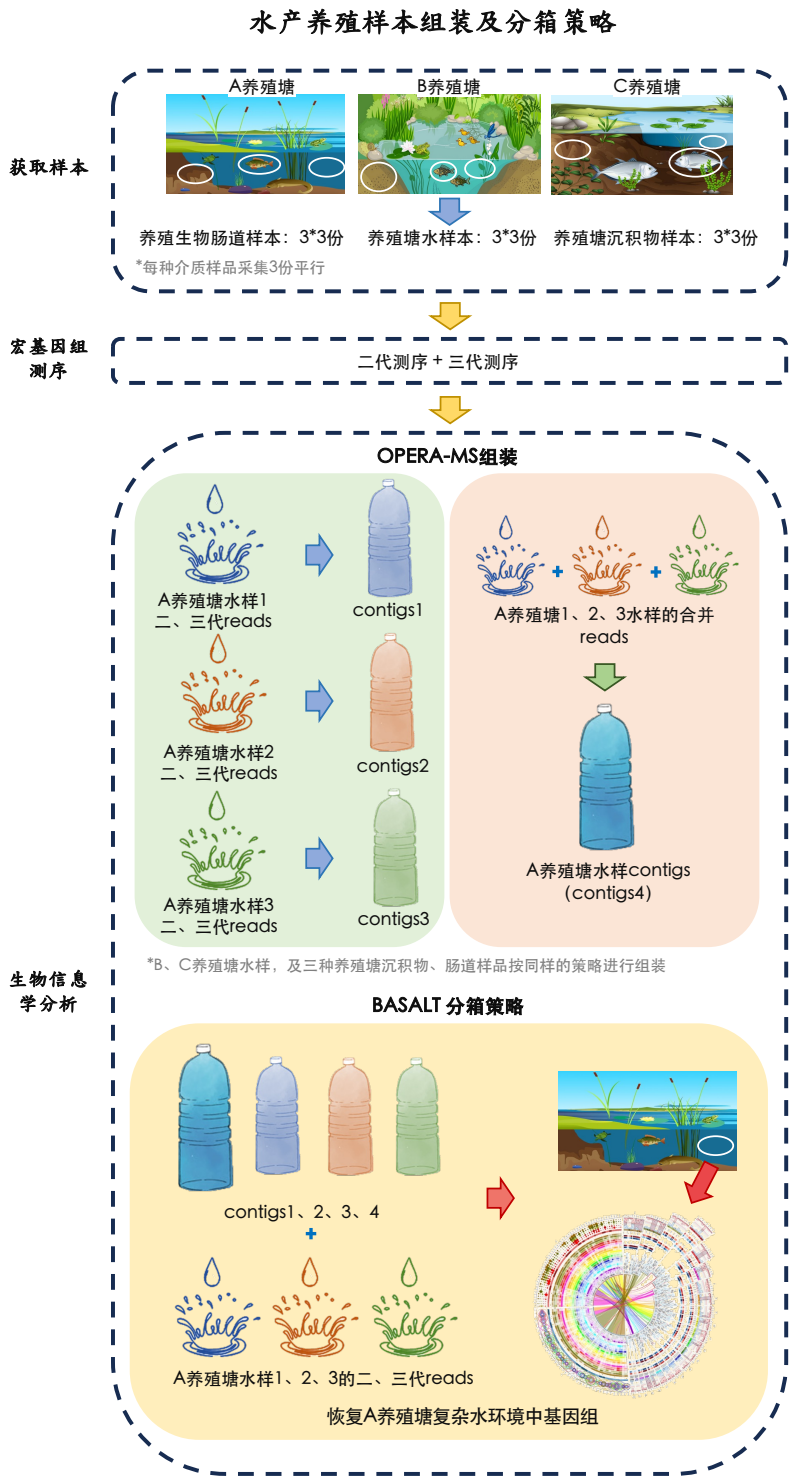


图 1 水产养殖样本组装及分箱策略

b) **项目二**：不同生长周期鸟类肠道样本组装及分箱策略

项目背景：在鸟类生长不同时期采集若干种不同鸟类粪便样本，进行宏基因组二代测序。

组装策略：首先，通过 metaspades 软件对二代测序数据进行单个样本的组装；同时，对所获取的同种鸟类的同一生长周期样品平行样的双端数据数据合并后进行多个样本的组装，以获取该生长周期样本更多维度信息，便于后续分箱处理；在获取单个样本及多个平行样本混合的组装文件后，进行分箱处理。

示例——以 A 鸟样本为例：

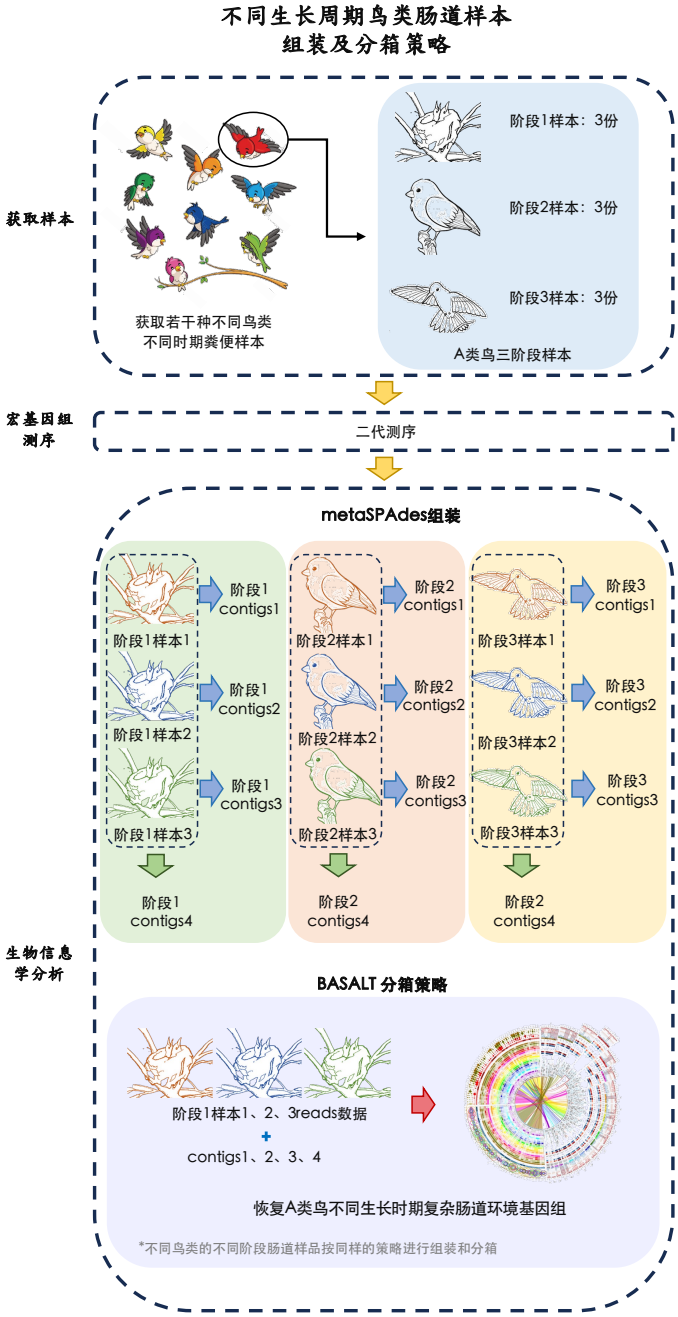


图 2 不同生长周期鸟类肠道样本组装及分箱策略

➢ 样本数据：

(A 类鸟阶段 1 样本)

sampleA_s1_1_R1.fq, sampleA_s1_1_R2.fq,

sampleA_s1_2_R1.fq, sampleA_s1_2_R2.fq,

sampleA_s1_3_R1.fq, sampleA_s1_3_R2.fq

(A 类鸟阶段 2 样本)

sampleA_s2_1_R1.fq, sampleA_s2_1_R2.fq,

sampleA_s2_2_R1.fq, sampleA_s2_2_R2.fq,

sampleA_s2_3_R1.fq, sampleA_s2_3_R2.fq

(A 类鸟阶段 3 样本)

sampleA_s3_1_R1.fq, sampleA_s3_1_R2.fq,

sampleA_s3_2_R1.fq, sampleA_s3_2_R2.fq,

sampleA_s3_3_R1.fq, sampleA_s3_3_R2.fq

- 组装结果：
  - 阶段 1 单样品组装结果

    sampleA_s1_1_assembly.fa,sampleA_s1_2_assembly.fa,sampleA_s1_3_assembly.fa
  - 阶段 2 单样品组装结果

    sampleA_s2_1_assembly.fa,sampleA_s2_2_assembly.fa,sampleA_s2_3_assembly.fa
  - 阶段 3 单样品组装结果

    sampleA_s3_1_assembly.fa,sampleA_s3_2_assembly.fa,sampleA_s3_3_assembly.fa
  - 阶段 1 多样品合并组装结果

    sampleA_s1_assembly.fa
  - 阶段 2 多样品合并组装结果

    sampleA_s2_assembly.fa
  - 阶段 3 多样品合并组装结果

    sampleA_s3_assembly.fa
- 分箱命令：

  (一阶段样品分箱命令)

  BASALT -a sampleA_s1_assembly.fa,

  sampleA_s1_1_assembly.fa,sampleA_s1_2_assembly.fa,sampleA_s1_3_assembly.fa -s

  sampleA_s1_1_R1.fq,sampleA_s1_1_R2.fq/sampleA_s1_2_R1.fq,sampleA_s1_2_R2.fq/

  sampleA_s1_3_R1.fq,sampleA_s1_3_R2.fq -t 60 -m 250

*注：二、三阶段样品分箱命令和一阶段相同，仅需替换对应文件即可

(七) 安装和运行中常见问题

1）按照官网流程安装后在运行 BASALT 的过程中遇到报错：

samtools: error while loading shared libraries: libcrypto.so1.0.0: cannot open shared object file: No such file or directory

解决方案：a) 首先查看安装 BASALT 环境的路径下是否存在 libcrypto.so1.0.0：

- *~/anaconda3/<path>/<to>/BASALT/lib$* ls libcry*

b）BASALT 环境路径下如果存在 libcrypto.so.1.1 但是不存在 libcrypto.so.1.0.0 则建立软链接将 libcrypto.so.1.1 链接到 libcrypto.so.1.0.0：

- *~/anaconda3/<path>/<to>/BASALT/lib$* ln -s ~/anaconda3/<path>/<to>/BASALT/lib/libcrypto.so.1.1 ~/anaconda3/<path>/<to>/BASALT/lib/libcrypto.so.1.0.0

c）在 BASALT 环境下测试 samtools 能否使用：

- samtools –help

2）运行 BASALT 的过程中遇到报错：

Traceback (most recent call last):
File "/users/raw937/.conda/envs/BASALT/bin/BASALT", line 57, in
datasets[str(n)].append(pr[1].strip())
IndexError: list index out of range

解决方案：该问题是由于 BASALT 不支持使用路径输入序列文件导致，可以通过建立一个单独的文件夹用于存放需要处理的文件来解决。运行命令可以参考：

- BASALT -a a1.fa,a2.fa -s seq1_r1.fq,seq1_r2.fq/seq2_r1.fq,seq2_r2.fq -l ont_seq.fq -t 60 -m 250

3）运行 BASALT 的过程中遇到报错：

Traceback (most recent call last):
File "/dss/dsshome1/lxc04/ge24yaf2/.conda/envs/BASALT/bin/BASALT", line 53, in
datasets_list=sr_datasets.split('/')

解决方案：该问题是由于 BASALT 不支持仅输入三代测序文件导致（目前版本仅支持 PacBio-HiFi 的纯三代数据输入），可以通过输入二代+三代测序文件来解决。

4）运行 BASALT 的过程中遇到报错：

INFO: Running CheckM2 version 1.0.1

[03/13/2024 12:56:34 PM] INFO: Running quality prediction workflow with 30 threads.

[03/13/2024 12:56:34 PM] ERROR: DIAMOND database not found. Please download database using <checkm2 database --download>

解决方案：该问题是由于 CheckM2 数据库无法在 conda 安装过程中自动安装导致，可以通过按照 CheckM2 安装指南手动下载来解决。CheckM2 安装指南：https://github.com/chklovski/CheckM2

5）运行 BASALT 的过程中遇到报错：

BASALT: command not found

解决方案：将 BASALT_script.zip 放入 BASALT 环境下（一般路径为/home/user/anaconda2/envs）。

- unzip BASALT_script.zip
- chmod -R 777 BASALT_scrip
- mv BASALT_scrip /* your_conda/envs/BASALT/bin

6）运行 BASALT 的过程中遇到报错：

Traceback (most recent call last):

File "/home/pthierin/miniconda3/envs/BASALT/bin/BASALT", line 137, in <module>

BASALT_main_d(assembly_list, datasets, num_threads, lr_list, hifi_list, hic_list, eb_list, ram, continue_mode, functional_module, autobining_parameters, refinement_paramter, max_ctn, min_cpn, pwd, QC_software)

File "/home/pthierin/miniconda3/envs/BASALT/bin/BASALT_main_d.py", line 494, in BASALT_main_d

Contig_recruiter_main(best_binset_from_multi_assemblies, outlier_remover_folder, num_threads, continue_mode, min_cpn, max_ctn, assembly_mo_list, connections_list, lr_connection_list, coverage_matrix_list, refinement_paramter, pwd)

File "/home/pthierin/miniconda3/envs/BASALT/bin/S6_retrieve_contigs_from_PE_contigs_10302023.py", line 1819, in Contig_recruiter_main

parse_bin_in_bestbinset(assemblies_list, binset+'_filtrated', outlier_remover_folder, PE_connections_list, lr_connection_list, num_threads, last_step, coverage_matrix_list, refinement_mode)

File "/home/pthierin/miniconda3/envs/BASALT/bin/S6_retrieve_contigs_from_PE_contigs_1030 2023.py", line 1695, in parse_bin_in_bestbinset

bin_comparison(str(binset), bins_checkm, str(binset)+'_retrieved', refinement_mode, num_threads)

File "/home/pthierin/miniconda3/envs/BASALT/bin/S6_retrieve_contigs_from_PE_contigs_1030 2023.py", line 731, in bin_comparison

for line in open('quality_report.tsv','r'):

FileNotFoundError: [Errno 2] No such file or directory: 'quality_report.tsv'

解决方案：该问题是由于数据 coverage 较低导致 binning 文件夹中的 bins 数量过少，从而导致 quality_report.tsv 无法生成。

7）运行 BASALT 的过程中我需要设定输出路径吗？

解决方案：不需要。BASALT 在输出时候自动将结果及重要过程文件输出至当前工作路径下，建议用户将需要分析的文件单独建立文件夹存放和运行 BASALT，以免重复输出覆盖原有结果。我们正在考虑在后续版本中增加路径的设置。

(八) 常见其他问题（FAQ）

1）Q: 在单独组装与合并组装文件同时输入的情况下，某些 contigs 被重复利用，会影响分箱结果吗？

A: 在单独组装与合并组装文件同时输入的情况下，确实会产生冗余的基因组，如：bin1 从 sampleA1_assembly.fa 中获得，bin2 从 sampleA1A2A3_assembly.fa 中获得，而 bin1 与 bin2 实际上为同一个基因组。考虑到这种情况的产生，BASALT 在 bin selection 中可以识别并去除冗余的基因组，因此最终产出的基因组为非冗余基因组。

2）Q: 与 metaWRAP 相比，BASALT 运行速度通常会更慢。是否有加快运行的方法？

A: 在使用单个组装文件运行的过程中，BASALT 的完整流程所耗时间确实会比 metaWRAP 要多出一倍左右，其耗时比例还会随着样品的复杂度增加而进一步增加。但是，在文章中也提到，即使不运行 gap filling 模块，BASALT 所得到的基因组质量与数量均比 metaWRAP 要好。实际上，**在多个组装文件输入模式下，BASALT 比 metaWRAP 会更省时**，因为 BASALT 只需运行一次，且其**单独组装+合并组装+去冗余的模式可以大大增加非冗余基因组的产出**。如用户对基因组的深度挖掘要求不高，且需要加快整个分箱进程，建议以下两种策略：

(1) 使用 MetaBAT2 + Semibin2 作为初始 Binners，可以大大加快 auto-binning 的进程；

(2) 仅运行 BASALT 中的 auto-binning + bin selection + refinement 模块，不使用 gap filling 模块可以减少运行时间；

(3) 在多个样品同时输入的情况下，仅使用单一的合并组装文件（如：sampleA1A2A3_assembly.fa），可以减少 auto-binning 的运算时间，但也会相应减少基因组的产生。

* 注：BASALT 的三种运行模式仍在测试之中， 预计在 BASALT-1.0.2 版本中会得到更新（约 2024 年 5 月更新），请耐心等待。

3）Q: 我已经有之前分箱得到的基因组，想要直接进行 refinement，该如何操作？

A: 在已有基因组的情况下，可以利用 data feeding 模块进行数据的输入。

* 注：BASALT 的 data feeding 运行模式仍在测试之中， 预计在 BASALT-1.0.2 版本中会得到更新（约 2024 年 5 月更新），请耐心等待。

4）Q: 运行 BASALT 的过程中我需要设定输出路径吗？

A: 不需要。BASALT 在输出时候自动将结果及重要过程文件输出至当前工作路径下，建议用户将需要分析的文件单独建立文件夹存放和运行 BASALT，以免重复输出覆盖原有结果。

(九) 更新日志

BASALT v1.1.0:

- 更改了 module mode 的默认模式为‘sensitive’；
- 更改了 refinement 的默认模式为‘quick’；
- 将 quality check 的参数名称‘-qc’改为‘-q’；
- 增加了‘-o’参数，对最终输出的 final binset 文件夹进行重命名；
- 增加了 data feeding 的模式：-d, -b, 和-r。现在用户可以直接输入利用其他软件获得的 bins 作为输入文件，对这些 bins 进行去冗余和进一步修正。
- 修正了一些错误。

(十)软件引用

Z Qiu, L Yuan, C Lian, B Lin, J Chen, R Mu, X Qiao, L Zhang, Z Xu, L Fan, Y Zhang, S Wang, J Li, H Cao, B Li, B Chen, C Song, Y Liu, L Shi, Y Tian, J Ni, T Zhang, J Zhou, W Zhuang, K Yu. BASALT refines binning from metagenomic data and increases resolution of genome-resolved metagenomic analysis. ***Nature Communications*** 2024, 15, 2179. https://doi.org/10.1038/s41467-024-46539-7

```
@article{qiu2024basalt,
  title={BASALT refines binning from metagenomic data and increases resolution of genome-resolved metagenomic analysis},
  author={Qiu, Zhiguang and Yuan, Li and Lian, Chun-Ang and Lin, Bin and Chen, Jie and Mu, Rong and Qiao, Xuejiao and Zhang, Liyu and Xu, Zheng and Fan, Lu and others},
  journal={Nature communications},
  volume={15},
  number={1},
  pages={2179},
  year={2024},
  publisher={Nature Publishing Group UK London}
}
```