目錄

项目简介

关于 GBC 1.1

下载与安装 1.2

文件格式 1.3

API 文档 1.4

使用手册

命令行程序 2.1

- 压缩基因型 2.1.1
  - 程序参数 2.1.1.1
  - 程序实例 2.1.1.2
  - 算法介绍 2.1.1.3
- 提取基因型 2.1.2
  - 程序参数 2.1.2.1
  - 程序实例 2.1.2.2
- 显示 GTB 文件摘要信息 2.1.3
  - 程序参数 2.1.3.1
  - 程序实例 2.1.3.2
- 按照坐标对 GTB 文件排序 2.1.4
  - 程序参数 2.1.4.1
  - 程序实例 2.1.4.2
- 合并多个 GTB 文件 2.1.5
  - 串联多个 GTB 文件 2.1.5.1
    - 程序参数 2.1.5.1.1
    - 程序实例 2.1.5.1.2
  - 合并多个 GTB 文件 2.1.5.2
    - 程序参数 2.1.5.2.1
    - 程序实例 2.1.5.2.2
- 重设样本名 2.1.6
  - 程序参数 2.1.6.1
  - 程序实例 2.1.6.2
- GTB 文件剪枝 2.1.7
  - 程序参数 2.1.7.1

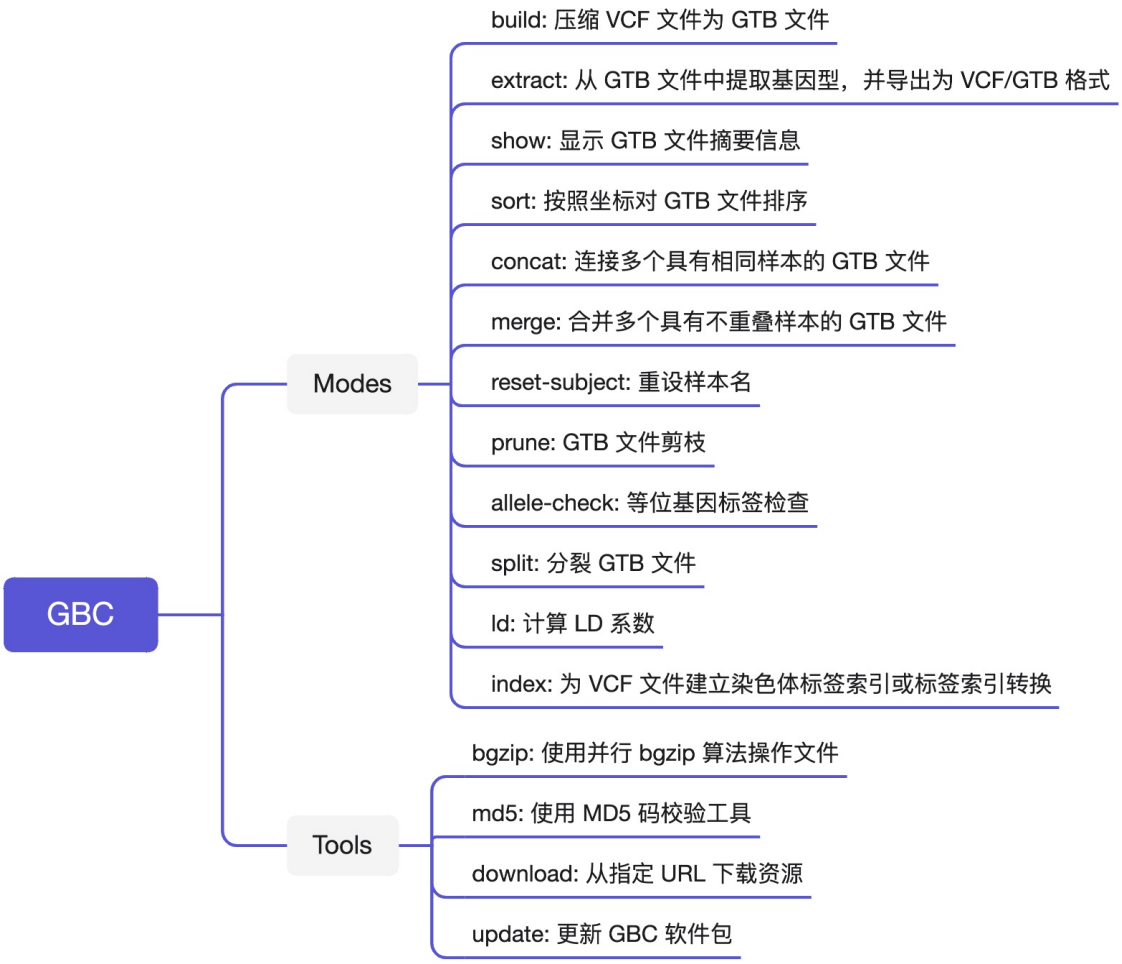
程序实例 2.1.7.2
等位基因标签检查 2.1.8
程序参数 2.1.8.1
程序实例 2.1.8.2
分裂 GTB 文件 2.1.9
程序参数 2.1.9.1
程序实例 2.1.9.2
计算 LD 系数 2.1.10
程序参数 2.1.10.1
程序实例 2.1.10.2
设置染色体标签 2.1.11
Contig 文件格式 2.1.11.1
为 VCF 文件构建 Contig 文件 2.1.11.2
重建染色体标签索引 2.1.11.3

关于 **GBC**

GenoType Blocking Compressor (简称 GBC) 是一个基因型数据分块压缩算法，它旨在为 VCF 文件中的基因型数据创建一个统一、灵活的结构 GenoType Block (简称 GTB)。使用 GTB 格式替代传统的 gz 格式，用户可以实现更少的硬盘空间占用、更快速的数据访问与提取、更方便的群体文件管理以及更高效的数据分析功能。GBC 包含的功能如下：

- 高效压缩**:** 单线程压缩时内存占用 < 4 GB、速度高达 78516269 genotypes/s、具有最高压缩比;
- 质量控制**:** 位点水平、基因型水平、群体等位基因频率水平质量控制, 并保留扩展接口;
- 快速查询**:** 查询连续/随机位点, 按照等位基因频数/频率过滤位点, 提取子集样本等;
- 文件管理**:** 合并、连接、拆分、子集样本选择、排序、等位基因标签检查等;
- 复杂计算**:** 快速 LD 计算;
- 广泛单倍体/二倍体生物的基因型编码表示

GBC 是一个免费使用的独立工具包，可以用于改进大规模基因型数据的储存与文件管理。同时，GBC 也是基础性的 API 开发工具库，可以很方便地整合到现有的工具流中，加速基于基因型数据的分析和计算流程。



> [!COMMENT|label:联系开发者]

> 张柳彬, suranyi.sysu@gmail.com

下载与安装

GBC 是基于 Oracle JDK 8 开发的应用程序，任何兼容支持或兼容 Oracle JDK 8 的计算机设备都可以使用我们的软件。用户需要先下载安装 Oracle JDK 或 Open JDK。Apple Silicon 设备可以使用 zulu JDK 作为替代。此外，我们也提供 Dockerfile 构建 GBC 的运行环境镜像。

资源类型	路径
软件包	http://pmglab.top/gbc/download/gbc.jar
源代码	https://github.com/Zhangliubin/gbc
说明文档	http://pmglab.top/gbc/
API 文档	http://pmglab.top/gbc/api-docs/
示例数据	http://pmglab.top/gbc/download/example.zip http://pmglab.top/genotypes/#/

通过 **wget** 下载软件包

```
# 下载 GBC 软件包
wget http://pmglab.top/gbc/download/gbc.jar -O gbc.jar

# 运行 GBC 软件包
java -jar gbc.jar
```

通过 **Docker** 运行 **GBC** 软件包

```
# 下载 GBC 的 Dockerfile 文件
wget http://pmglab.top/gbc/download/Dockerfile -O Dockerfile

# 从 Dockerfile 文件构建镜像
docker build -t gbc .

# 运行 GBC 软件包
docker run -it --rm gbc
```

从 **Github** 下载 **GBC** 源代码

```
# 下载 GBC 源代码
git clone https://github.com/Zhangliubin/gbc gbc-source

# 进入文件夹
cd gbc-source

# 运行 GBC 软件包
java -jar gbc.jar
```

更新软件包

GBC 目前已经迭代到稳定版本，通常情况下我们只会在现有的软件包基础上增加新的功能，或为增强稳定性和提升性能进行的少量代码架构更新。检查新版本可用性，请使用：

```
java -jar gbc.jar update
```

运行要求

GBC 进行了严格的内存需求控制，对于小规模的基因组数据，通常可以以默认内存分配量运行。对于大规模的基因组数据，GBC 单线程的内存使用量最多不超过 4 GB。因此，我们建议在始终不小于 4GB 的堆内存中运行 GBC 程序。用户通过以下指令分配 GBC 的运行时堆内存：

```
java -Xms4g -Xmx4g -jar gbc.jar
```

使用 Docker 运行 GBC 时，我们推荐使用以下模版语句：

```
# Macos 或 Linux
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc [options]

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -e -m 4g gbc [options]
```

在该语句中，`-v` 表示将指定的主机路径映射到容器路径 (`宿主机路径:容器路径`)，`-w` 表示设置当前工作路径 (相当于运行 `cd` 指令)，`-it` 表示在交互式终端中运行，`-m 4g` 表示设置 JVM 最大堆大小为 4GB。

更新日志

> [!UPDATE|label:2022/06/20]
>
> - 发布 GBC 的第二个版本，版本号 1.2。
>   - Github 仓库地址：https://github.com/Zhangliubin/gbc
>   - 文档地址：http://pmglab.top/gbc/
> - 请注意，GBC-1.1 与 GBC-1.2 是完全兼容的版本。GBC-1.2 专注于提升开发者 API 的使用体验，并移除了图形界面启动功能 (该功能仍然在 GBC-1.1 中提供)。

> [!UPDATE|label:2022/4/2]
>
> - 发布 GBC 的第一个版本，版本号 1.1。
>   - Github 仓库地址：https://github.com/Zhangliubin/Genotype-Blocking-Compressor
>   - 文档地址：http://pmglab.top/gbc/

输入与输出格式

GBC 实现了 VCF 格式、BGZIP 压缩的 VCF 格式、GTB 格式之间的相互转换。通常情况下，我们建议使用 GTB 作为输入与输出，它将获得完全的多线程支持。

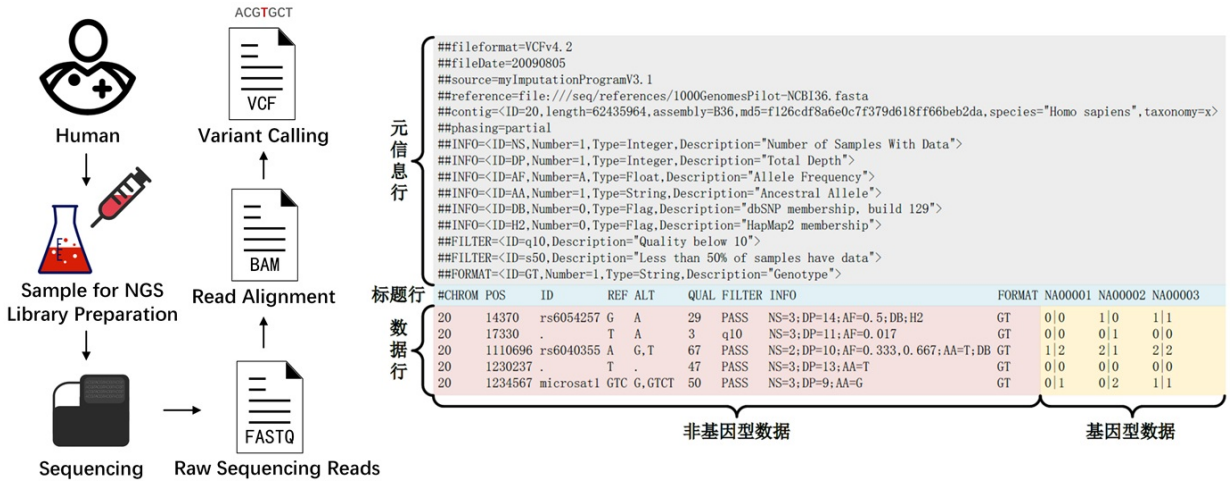
$$\text{Other Formats} \Longleftrightarrow \text{VCF / BGZIP 压缩的 VCF} \Longleftrightarrow \text{GTB}$$

输入格式	输出格式	支持工具
其他格式	VCF 或 VCF.GZ	其他工具
VCF	BGZIP 压缩的 VCF	GBC - bgzip - -c
BGZIP 压缩的 VCF	VCF	GBC - bgzip - -d
GZIP 压缩的 VCF	VCF / BGZIP 压缩的 VCF	GBC - bgzip - -d
VCF / BGZIP 压缩的 VCF	GTB	GBC - build
GTB	VCF / BGZIP 压缩的 VCF	GBC - extract

VCF 格式

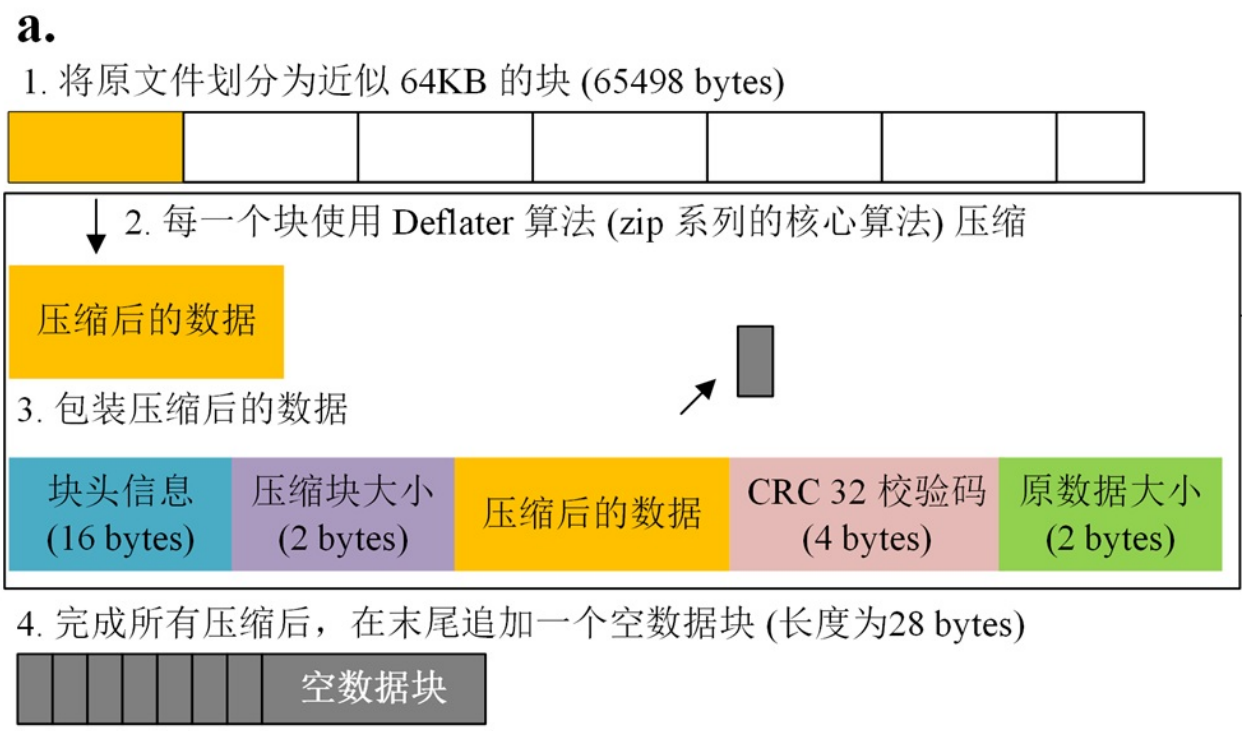
VCF（Variant Call Format）是存储变异位点的标准格式，它是专门用于记录和描述 SNP、InDel、SV 和 CNV 等变异信息的文本文件。VCF文件包含注释信息行、标题行、数据行。其中，注释信息行的行首为“##”，内容是键值对的形式，通常包含 VCF 文件版本信息、参考基因组信息、软件执行信息、相关字段含义等。标题行的行首为“#”，前9个字段都是固定的，分别为CHROM(染色体编号)、POS(变异位点在参考基因组中的位置)、ID(变异位点编号)、REF(参考碱基)、ALT(相对参考序列突变的碱基)、QUAL(变异位点的Phred-scaled质量分数)、FILTER(变异位点过滤状态)、INFO(附加信息)、FORMAT(样本数据格式)，之后若干列为样本名称；数据行中每一个变异位点占用一行，变异位点信息按照标题行顺序进行填写，使用制表符分割数据，缺失信息使用“.”占位。

> [!NOTE|label:VCF 文件格式: https://samtools.github.io/hts-specs/VCFv4.2.pdf]



使用 BGZIP 压缩的 VCF 格式

VCF 文件的文本格式通常占用庞大的磁盘空间，因此，使用 BGZIP 压缩 VCF 是常见的方法 (使用 BGZIP 压缩产生的文件也被称之为 BGZF)。BGZIP 是在标准 GZIP 文件格式之上实现的块压缩方法，它的目的是在提供良好的压缩比的同时，允许对数据进行随机访问 (即提取少量数据时，避免全文件解压)。



并行 BGZIP 压缩算法的 Java 实现：http://pmglab.top/commandParser/zh/example/BGZToolkit.html

> [!NOTE|label:BGZIP 文档: http://www.htslib.org/doc/bgzip.html]

GTB 格式

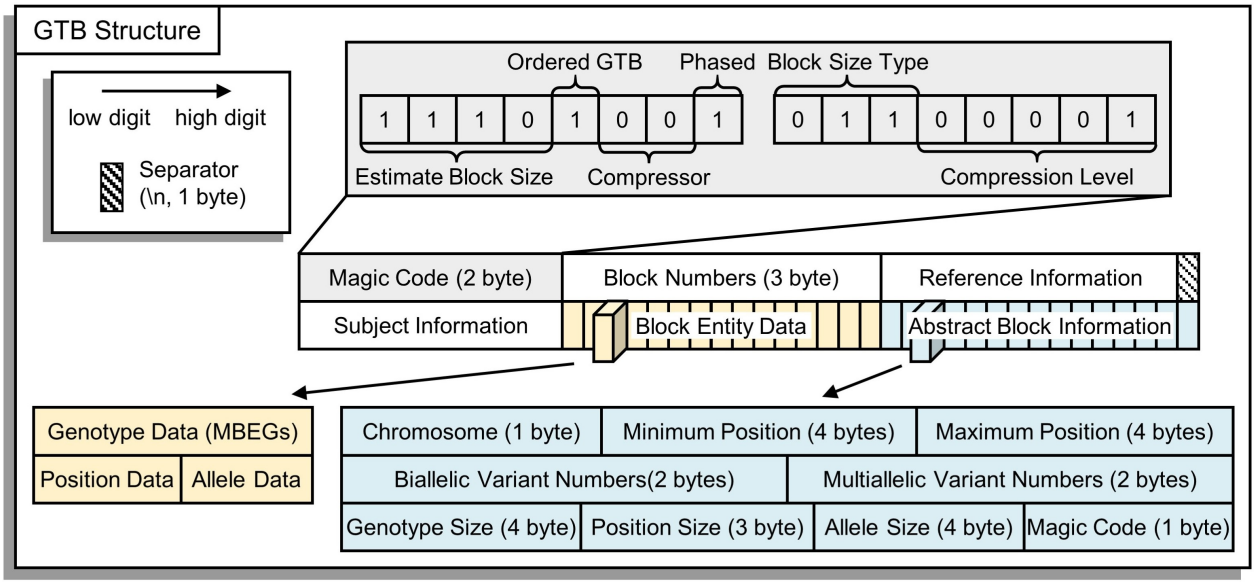
GTB 是用于压缩和存储具有不同等位基因数、染色体数、单倍体和二倍体物种、Phased 和 Unphased 的、任意数量基因型数据的文件格式。它具有统一、灵活、可扩展的良好设计：

- 统一**:** 与其他文件格式 (如 GTShark、GTC、BGT、PBWT) 对比，GTB 格式仅有单独的文件作为输出，同时它支持在一个文件内组织多个染色体的基因型数据 (与 PBWT 对比)。这有利于文件的储存和传输；
- 解耦**:** GTB 使用解耦合设计，使得文件的修改尽可能只涉及到极少数的数据。此外，独立的结构特性也使得并行化变得非常容易——按位点、块或染色体进行并行化都可以很轻易地实现；
- 可扩展**:** GTB 本质上是双索引键 (染色体、位置) 的 MDRT 格式，它为基因型的快速随机访问设计了许多良好易用的 API 方法。MDRT 允许具有可变的多索引键，并采用与 GTB 类似的结构，它也被用于储存 VCF 文件中的非基因型数据。

GTB 的文件格式如下图所示。各部分的含义如下：

- **Magic Code:** 文件的前 2 个字节是文件标记码 (Magic Code)，它表示了当前文件构建的基本细节，例如是否为有序的 GTB 文件、使用的基压缩器编码、基因型是否分向、GTB 块的最大位点容量 (也称块大小)、压缩级别等；
- **Block Numbers:** 文件的第 3～5 字节是文件中包含的 GTB 块的个数，它被以 3 字节整数的形式保存到文件中；

- **Reference Information:** 文件的参考序列地址，通常为指向公共资源或本地资源的 URL 地址；
- **Subject Information:** 样本名称列表；
- **Block Entity Data:** 块的压缩数据，基因型数据、位置数据、等位基因数据被分别压缩并拼接；
- **Abstract Block Information:** 块摘要信息，描述了每个块对应的数据节点的基本信息 (如染色体编号、最小最大位置值、压缩数据段长度等)，用于创建 GTB 树表，实现快速访问。



使用 GBC

在命令行程序 (终端) 中使用以下语句查看 GBC 文档：

```
# 不传入任何参数时，默认打印文档
java -Xms4g -Xmx4g -jar gbc.jar

# 传入 -h, -help, --help 时，打印文档
java -Xms4g -Xmx4g -jar gbc.jar -h
```

此时，终端打印程序文档：

```
Usage: java -jar gbc.jar [mode/tool] [options]
Version: GBC-1.2 (last edited on 2022.06.20, http://pmglab.top/gbc)
Mode:
  build          Compress and build *.gtb for vcf/vcf.gz files.
                 format: build <input(s)> -o <output> [options]
  extract        Retrieve variants from *.gtb file, and export them to
                 (compressed) VCF format or GTB format.
                 format: extract <input> -o <output> [options]
  show           Display summary of the GTB File.
                 format: show <input> [options]
  sort           Sort variants in GTB by coordinates (chromosome and position).
                 format: sort <input> -o <output> [options]
  concat         Concatenate multiple VCF files. All source files must have the
                 same subjects columns appearing in the same order with
                 entirely different sites, and all files must have to be the
                 same in parameters of the status.
                 format: concat <input(s)> -o <output> [options]
  merge          Merge multiple GTB files (with non-overlapping subject sets)
                 into a single GTB file.
                 format: merge <input(s)> -o <output> [options]
  reset-subject  Reset subject names (request that same subject number and no
                 duplicated names) for gtb file directly. Subject names can be
                 stored in a file with ',' delimited form, and pass in via
                 '--reset-subject @file'.
                 format: reset-subject <input> -o <output> [options]
  prune          Prune GTB files by node-level or chromosome-level.
                 format: prune <input> -o <output> [options]
  allele-check   Correct for potential complementary strand errors based on
                 allele labels (A and C, T and G; only biallelic variants are
                 supported).
                 format: allele-check <template_input> <input> -o <output>
                 [options]
  split          Split a single GTB file into multiple subfiles (e.g. split by
                 chromosome).
                 format: split <input> -o <output> [options]
  ld             Calculate pairwise the linkage disequilibrium or genotypic
                 correlation.
                 format: ld <input> -o <output> [options]
  index          Index contig file for specified VCF file or reset contig file
                 for specified GTB file.
```

```
                format: index <input (VCF or GTB)> -o <output> [options]
Tool:
  bgzip     Use parallel bgzip to compress a single file.
            format: bgzip <string> <string> ...
  md5       Calculate a message-digest fingerprint (checksum) for file.
            format: md5 <string> <string> ...
  download  Download resources from an URL address.
            format: download <string> <string> ...
  update    Update GBC software packages.
            default: ./gbc.jar
            format: update <gbc.jar>
```

在交互模式下使用 **GBC**

当多次使用 GBC 运行程序指令时，我们建议在交互模式下进行 (如同 ipython)，它对程序的输入做出反馈，并减少启动 JVM 所需的时间 (尤其是在 Docker 中运行时，使用交互模式可以避免频繁创建、销毁容器)。进入交互模式请运行：

```
java -Xmx4g -Xms4g -jar gbc.jar -i
```

当然，用户也可以选择在运行完一次指令后进入交互模式，语句为：

```
java -Xmx4g -Xms4g -jar gbc.jar [mode/tool] [options] -i
```

在交互模式下运行指令时，不再需要指定 `java -Xmx4g -Xms4g -jar gbc.jar` 。除了兼容普通命令行参数外，还有以下四种额外的参数：

参数	描述
exit, q, quit	退出程序
clear	清空屏幕 (实际上打印出多行空白行)
reset	清空缓冲区
以“#”开头的指令	注释，不执行任何操作

压缩基因型

使用如下指令对基因组 VCF 文件进行压缩：

```
build <input(s)> -o <output> [options]
```

- GBC 可以为符合 VCF 文件规范 的文件进行压缩，GBC 的所有操作都是假定文件格式是符合此规范的；
- 输入文件可以是单个或多个 .vcf 文件或 .vcf.gz 文件，也可以是包含这些文件的文件夹路径。当路径为文件夹路径时，GBC 会筛选出该文件夹 (及其子文件夹) 中所有的 vcf 文件或 .vcf.gz 文件进行压缩。请注意，GBC 仅根据文件的扩展名判断文件类型，因此正确的文件扩展名才能够进行压缩；
- 为了便于统一全局的符号，GBC 当前仅支持对于人类基因组的压缩，CHROM当前只支持 1-22, X, Y, MT 以及带有 chr 前缀的 1-22, X, Y, MT；对于其他物种，需要先构建染色体标签文件；
- GBC 对多个文件进行合并压缩时，要求这些文件具有相同的样本序列 (样本顺序可以不一致)。若一个文件的样本序列是其他文件的子序列，它也可以被正确压缩，缺失的样本基因型将被替换为 .|.；
- 当输入的文件在坐标上无序时，GBC 也能正确地压缩，并产生标记为 `unordered` 的 GTB 文件。通常我们建议用户进一步使用 `sort <input> -o <output> [options]` 指令对该文件进行排序，否则它将在某些功能上不可用 (例如: 计算 LD 系数) 或影响其他操作的性能 (例如: 合并 VCF 文件)。

程序参数

```
语法: build <input(s)> -o <output> [options]
参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --threads,-t   设置并行压缩线程数.
                 默认值: 4
                 格式: --threads <int> (>= 1)
  --yes,-y       覆盖输出文件.
GTB 存档参数:
  --phased,-p           设置基因型数据为 phased.
  --biallelic           将多个多等位基因位点分裂为多个二等位基因位点.
  --simply              删除 ALT 中等位基因计数值为 0 的标签.
  --blockSizeType,-bs   设置每个压缩块的最大位点数, 根据 2^(7+x) 换算得到真实的块大小值.
                        默认值: -1 (即根据样本量自动设置)
                        格式: --blockSizeType <int> (-1 ~ 7)
  --no-reordering,-nr   禁用 Approximate Minimum Discrepancy Ordering (AMDO) 算法.
  --windowSize,-ws      设置 AMDO 算法的采样窗口大小.
                        默认值: 24
                        格式: --windowSize <int> (1 ~ 131072)
  --compressor,-c       设置基压缩器.
                        默认值: ZSTD
                        格式: --compressor <string> ([ZSTD/LZMA/GZIP] or
                        [0/1/2] (ignoreCase))
  --level,-l            基压缩器的压缩级别. (ZSTD: 0~22, 默认为 3; LZMA: 0~9,
```

```
                        默认为 3; GZIP: 0~9, 默认为 5)
                        默认值: -1
                        格式: --level <int> (-1 ~ 31)
  --readyParas,-rp      从外部 GTB 文件中导入模版参数 (-p, -bs, -c, -l).
                        格式: --readyParas <file> (Exists,File)
  --seq-ac              移除等位基因计数不在 [minAc, maxAc] 范围点的位点.
                        格式: --seq-ac <int>-<int> (>= 0)
  --seq-af              移除等位基因频率不在 [minAf, maxAf] 范围点的位点.
                        格式: --seq-af <double>-<double> (0.0 ~ 1.0)
  --seq-an              移除有效等位基因个数不在 [minAn, maxAn] 范围点的位点.
                        格式: --seq-an <int>-<int> (>= 0)
  --max-allele          移除等位基因种类超过指定值的位点.
                        默认值: 15
                        格式: --max-allele <int> (2 ~ 15)
质量控制参数:
  --no-qc       禁用所有的质量控制方法.
  --gty-gq      样本基因型 GQ 分数值小于指定值时, 该基因型将被替换为 './.'.
                默认值: 20
                格式: --gty-gq <int> (>= 0)
  --gty-dp      样本基因型 DP 分数值小于指定值时, 该基因型将被替换为 './.'.
                默认值: 4
                格式: --gty-dp <int> (>= 0)
  --seq-qual    移除 Phred Quality 分数值小于指定值的位点.
                默认值: 30
                格式: --seq-qual <int> (>= 0)
  --seq-dp      移除 DP 分数值小于指定值的位点.
                默认值: 0
                格式: --seq-dp <int> (>= 0)
  --seq-mq      移除 MQ 分数值小于指定值的位点.
                默认值: 20
                格式: --seq-mq <int> (>= 0)
```

程序实例

使用 GBC 压缩示例文件 `./example/assoc.hg19.vcf.gz`，并设置以下参数：

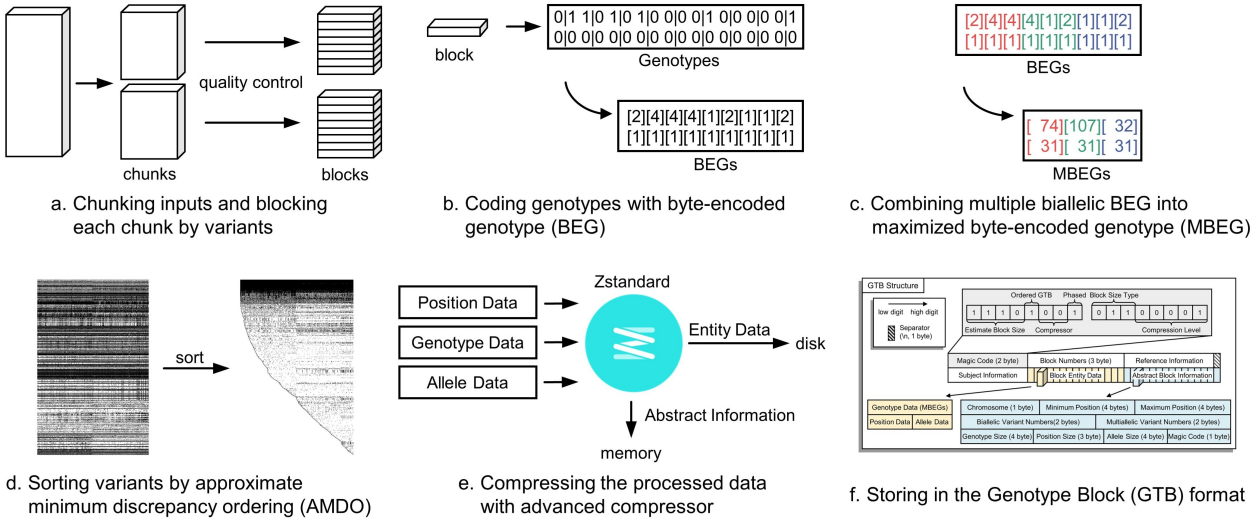
- 基因型设置为 phased；
- 压缩器压缩级别设置为 16；
- 将多等位基因位点分裂为二等位基因位点；
- 保留 MAF (次级等位基因频率) > 0.01 的位点；
- 若输出文件已存在，则覆盖该文件。

完成该任务的指令如下：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
build ./example/assoc.hg19.vcf.gz -o ./example/assoc.hg19.gtb -p -l 16 --biallelic --se
q-af 0.01-0.99 -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc build ./example/assoc.hg19.vcf.gz -o
 ./example/assoc.hg19.gtb -p -l 16 --biallelic --seq-af 0.01-0.99 -y
```

算法介绍

GBC 压缩算法的流程分为分块、字节编码、最大化字节编码、位点的 AMDO 排序、使用基压缩器压缩数据流、储存到 GTB 文件。该算法对于样本和位点的数量具有线性时间复杂度，内存使用量小 (小于4GB)，而且它提供了迄今为止几乎最快的压缩速度和最具竞争力的压缩比。



文件输入

多个文件输入时，GBC 将具有最多样本的文件作为母文件，其他的文件 (子文件) 的样本顺序则与母文件对齐。具体对齐策略如下：

- 当子文件的样本不包含在母文件中时，程序报错；
- 当子文件的样本顺序与母文件的样本顺序不一致时，子文件样本的基因型顺序将按照母文件的样本顺序进行重排；
- 当母文件的样本不包含在子文件中时，缺失的样本将以 `.|.` 或 `./.` 代替。

> [!NOTE|label:适用情形]
>
> 不少人类基因组项目 (如 1000GP3、SG10K) 都会将基因型按照不同染色体编号进行存放 (即 chr1.gz, chr2.gz, ...)。此外，在 Y 染色体文件中通常不包含雌性个体，造成 Y 染色体文件的样本数通常小于常染色体或 X 染色体。
>
> 这种策略可以将基因组的多个散布的文件合并压缩，也可以将人类基因组中的性染色体文件与常染色体文件合并压缩。

初始化

为控制每个线程的最大内存用量，当总样本量 M 被确定时，每个 GTB 块的最大位点数 N 也就被确定。通常情况下，样本量范围对应的块大小参数 (或块位点个数) 关系表如下：

参数	N	M	参数	N	M
-bs 7	16384	≤ 65536	-bs 3	1024	≤ 1048576
-bs 6	8192	≤ 131072	-bs 2	512	≤ 2097152
-bs 5	4096	≤ 262144	-bs 1	256	≤ 4194303

-bs 4	2048	≤ 524288	-bs 0	64	≤ 16777215

其中，$M \times N \leq 2^{30}$。

文件分块以并行化

输入的文件依次进行压缩。在每个文件压缩时，按照并行线程数 k 被近似地均等分成 k 块，每个线程处理 1 个块。当输入文件为 BGZIP 压缩的 VCF 格式时，GBC 会检查块边界处的信息、调整指针，确保块中的位点都是完整的。

位点和基因型水平质控

线程在处理单个块时，按照行读取位点，并解析非基因型字段。通过非基因型字段预先进行初步的位点水平质量控制 (例如: 根据 Phred Quality Score、MQ 进行质控)，当位点不满足质控要求时，线程继续读取下一个位点。

满足位点水平质控的位点则对其基因型进行质控。当位点 `FORMAT` 的值为 `GT` 时，不进行基因型水平的质控；若该值还包含了其他的关键字段，则对于每一个读取的基因型将进行相应的质量控制 (例如: 根据 DP、GQ 进行质控)。当基因型不满足质控要求时，该基因型被 `.|.` 替代。

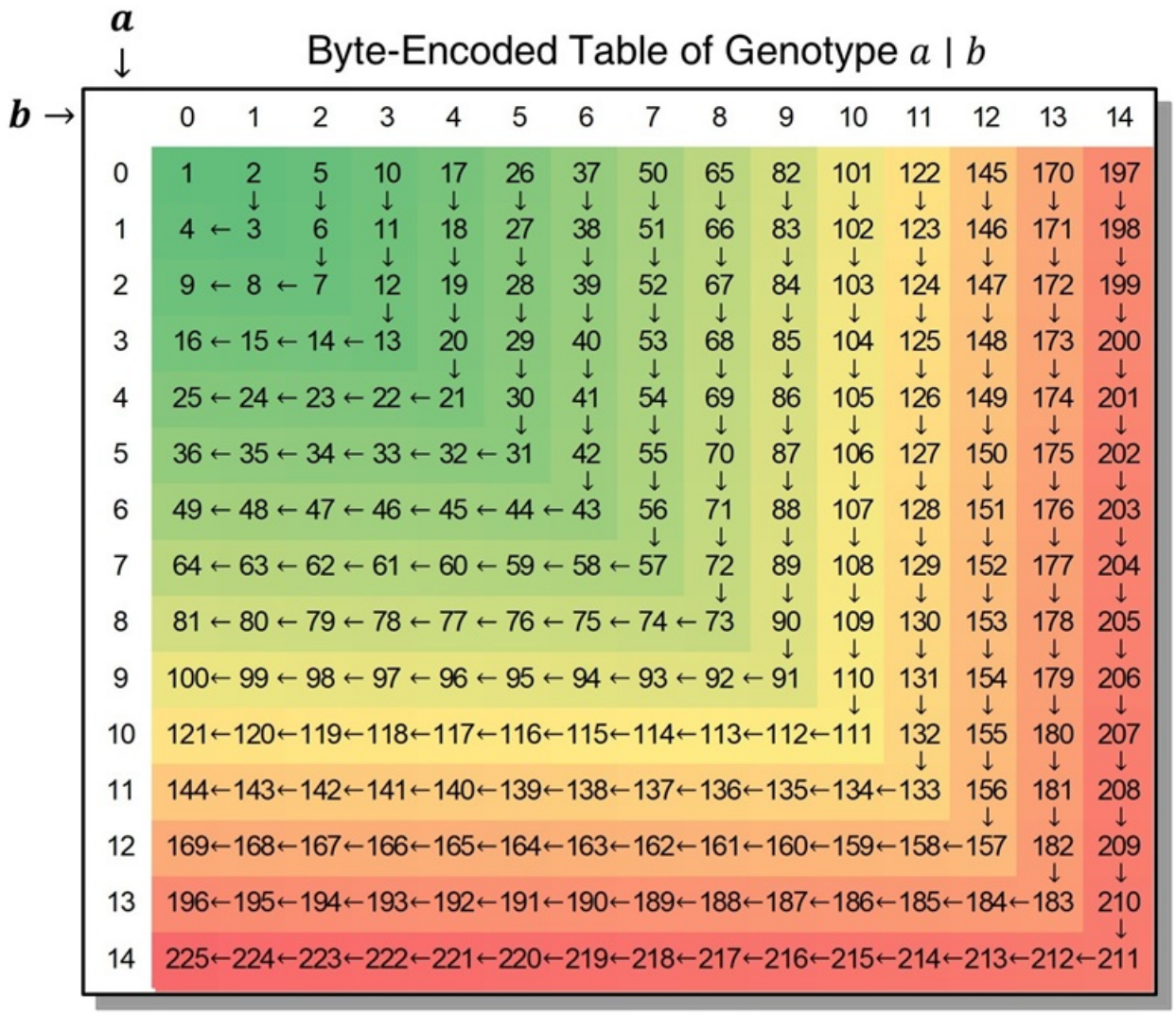
编码基因型为字节码 **(BEG)**

满足基因型水平质控的基因型进行字节编码。对于给定的位点 v, 非缺失基因型 $a \mid b$ 编码为：

$$a \mid b \to \begin{cases} (a+1)^2 - b & , a \geq b \\ b^2 + a + 1 & , a < b \end{cases}$$

即：$a \mid b \to (\max\{a,b\})^2 + \max\{a-b, 0\} + a + 1$。对于特殊情形，约定如下:

- 缺失基因型 . | . 编码为 0；
- 非分相 (unphased) 基因型 a/b 被转换为 $\min\{a,b\} \mid \max\{a,b\}$ 后按照上式编码；
- 单倍型 a 被转换为 $a \mid a$ 后按照上式编码。



将位点写入 **GTBWriter** 缓冲区

当位点的基因型都被编码完成时，该位点被传输到 GTBWriter 缓冲区，以进入后续的压缩环节。在这一步，位点被执行更多的过滤或转换操作：

- 当启动了 “--simply” 时：多等位基因位点将移除等位基因计数为 0 的 ALT 标签，基因型则根据新的等位基因标签重新编码为字节值；
- 当启动了 “--biallelic” 时：多等位基因位点被拆分为多个二等位基因位点；
- 当启动了位点水平质控时：该位点进行位点水平质量控制，当位点不满足质控要求时，该位点将被跳过；

当 GTBWriter 缓冲区大小没有达到 GTB 块的最大位点数 N 或该分块文件没有读取结束时，重复进行以上的位点读取、质控、编码、写入缓冲区的操作；否则，进行以下的压缩环节。

基因型阵列进行特征重排列 **(AMDO)**

缓冲区中的位点按照如下方式进行重排列，以提升压缩比：

Step 1: 样本 n 在位点 m 上的基因型的 "0" 等位基因个数, 记为 c_{mn}；

Step 2: 生成降采样序列 $C_m^{(l)} = \left[C_{m,0}^{(l)}, C_{m,1}^{(l)}, \cdots, C_{m,s-1}^{(l)}\right]$, 其中:

$$C_{m,i}^{(l)} = \sum_{j=i\cdot l}^{\min\{N-1,(i+1)l-1\}} \sum_{k=i\cdot l}^{j} c_{mk} = \begin{cases} \sum_{j=i\cdot l}^{(i+1)l-1} ((i+1)l-j)c_{mj} & , i < s-1 \\ \sum_{j=i\cdot l}^{N-1} (N-j)c_{mj} & , i = s-1 \end{cases}$$

式中 N 为总样本数, $l = \lceil N/s \rceil$ 为连续采样基因型的个数;

Step 3: 根据降采样序列, 二等位基因 (biallelic) 位点进行正向字典序排序。多等位基因 (multiallelic) 位点进行反向字典序排序。位点 v_i 和位点 v_j 的顺序被定义为 $C_i^{(l)}$ 和 $C_j^{(l)}$ 的字典序，即：

- 如果 $\exists k_0 \in \left[0, \lceil \frac{N}{s} \rceil - 1\right], \forall k \in [0, k_0 - 1]$, 使得 $C_{i,k_0}^{(s)} < C_{j,k_0}^{(s)}, C_{i,k}^{(s)} = C_{j,k}^{(s)}$, 则 $v_i > v_j$;
- 如果 $\exists k_0 \in \left[0, \lceil \frac{N}{s} \rceil - 1\right], \forall k \in [0, k_0 - 1]$, 使得 $C_{i,k_0}^{(s)} > C_{j,k_0}^{(s)}, C_{i,k}^{(s)} = C_{j,k}^{(s)}$, 则 $v_i < v_j$;
- 如果 $\forall k \in \left[0, \lceil \frac{N}{s} \rceil - 1\right]$, 使得 $C_{i,k}^{(s)} = C_{j,k}^{(s)}$, 则 $v_i = v_j$。

最终，获得位点的新索引 $I = [m_0, m_1, \cdots, m_{M-1}]$。

将基因型字节码 **(BEG)** 转为最大化字节码 **(MBEG)**

对于二等位基因 (biallelic) 位点, 连续的 3~4 个基因型按照以下方法组合为单个字节:

$$\begin{aligned} &\text{phased} : [\text{BEG}_0, \text{BEG}_1, \text{BEG}_2] \to 5^2 \cdot \text{BEG}_0 + 5 \cdot \text{BEG}_1 + \text{BEG}_2 \\ &\text{unphased} : [\text{BEG}_0, \text{BEG}_1, \text{BEG}_2, \text{BEG}_3] \to 4^3 \cdot \text{BEG}_0 + 4^2 \cdot \text{BEG}_1 + 4 \cdot \text{BEG}_2 + \text{BEG}_3 \end{aligned}$$

当位点基因型的个数不构成 3 或 4 的倍数时, 末尾处的"缺失基因型"被设置为上一个非缺失值.

合并数据流并压缩

每个区块中的所有排序的 MBEG 和 BEG 代码都由先进的压缩器进一步压缩。流行的压缩算法（如 ZSTD、LZMA、GZIP 等）在基因型上可以达到 100 倍以上的压缩率。我们默认选择了ZSTD，因为在广泛使用的压缩算法中，它提供了最快的压解速度。

具体而言，每个位点的基因型编码都直接串联成一个字节数组 B_1。接下来，每个位点的位置被转换为 4 个字节，然后块内所有位点的位置被串联到一个字节数组 B_2。最后，所有变体的等位基因被串联到另一个带有 `/` 分隔符的字节数组B_3。然后，这些串联的数据 B_1、B_2 和 B_3 被基压缩器压缩，分别产生 $\hat{B}_1$、$\hat{B}_2$ 和 $\hat{B}_3$，这三个部分组成了块的压缩数据。块数据会即时写入线程相应的输出文件。

此外，根据压缩情况，每个块对应产生一个摘要信息：染色体编号（1个字节）、最小和最大位置（分别为4个字节）、二等位基因位点的数量（2个字节）、多等位基因位点的数量（2个字节）、$\hat{B}_1$ 的长度（4个字节）、$\hat{B}_2$ 的长度（3个字节）、$\hat{B}_3$ 的长度（4个字节）和块的校验编码（1个字节）。摘要信息被写入线程

的内存队列中。

GBC 可以与不同的压缩算法相结合。ZSTD、LZMA、GZIP 算法已被嵌入，我们还保留了 1 种类型的压缩器，供开发者在未来扩展。

完成压缩

当一个线程完成压缩时，块摘要信息被拼接到线程相应的输出文件，文件指针挪动到文件头部，并修改块头信息。

当所有的线程都完成压缩时，GBC 主程序调用 `Concat` 方法将每个线程产生的 GTB 文件拼接成一个整体的 GTB 文件。

提取基因型

使用如下指令对基因型文件 GTB 进行信息提取：

```
extract <input> -o <output> [options]
```

- 当 `[options]` 中包含了 `--o-gtb` 或输出文件名以 `.gtb` 结尾时，识别为导出 GTB 文件格式；
- 当 `[options]` 中包含了 `--o-bgz` 或输出文件名以 `.gz` 结尾时，识别为导出 BGZIP 压缩的 VCF 文件格式；
- 当 `[options]` 中包含了 `--o-vcf` 或输出文件名不以 `.gz` 或 `.gtb` 结尾时，识别为导出 VCF 文件格式。

通常生物信息学工具都兼容 BGZIP 压缩的 VCF 文件格式，我们建议用户以 `--o-bgz` 或 `--o-gtb` 格式作为输出，以增强程序的并行输出性能。

程序参数

```
语法: extract <input> -o <output> [options]
输出参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --o-text       以 VCF 文本格式输出位点信息.
  --o-bgz        以 BGZIP 压缩的 VCF 格式输出位点信息.
  --o-gtb        以 GTB 格式输出位点信息.
  --level,-l     基压缩器的压缩级别. (ZSTD: 0~22, 默认为 3; LZMA: 0~9, 默认为 3;
                 GZIP: 0~9, 默认为 5)
                 默认值: -1
                 格式: --level <int> (-1 ~ 31)
  --no-clm       不使用循环锁定算法 (CLM) 并行输出. 在这个参数下, 并行解压意味着先输出到临
                 时文件, 在完成压缩时拼接这些临时文件.
  --threads,-t   设置并行压缩线程数.
                 默认值: 4
                 格式: --threads <int> (>= 1)
  --phased,-p    设置输出基因型的向型. (默认与输入的 GTB 文件向型一致)
                 格式: --phased [true/false]
  --hideGT,-hg   不输出样本基因型数据.
  --yes,-y       覆盖输出文件.
GTB 存档参数:
  --biallelic           将多个多等位基因位点分裂为多个二等位基因位点.
  --simply              删除 ALT 中等位基因计数值为 0 的标签.
  --blockSizeType,-bs   设置每个压缩块的最大位点数, 根据 2^(7+x) 换算得到真实的块大小值.
                        默认值: -1 (即根据样本量自动设置)
                        格式: --blockSizeType <int> (-1 ~ 7)
  --no-reordering,-nr   禁用 Approximate Minimum Discrepancy Ordering (AMDO) 算法.
  --windowSize,-ws      设置 AMDO 算法的采样窗口大小.
                        默认值: 24
                        格式: --windowSize <int> (1 ~ 131072)
```

```
  --compressor,-c       设置基压缩器.
                        默认值: ZSTD
                        格式: --compressor <string> ([ZSTD/LZMA/GZIP] or
                        [0/1/2] (ignoreCase))
  --readyParas,-rp      从外部 GTB 文件中导入模版参数 (-p, -bs, -c, -l).
                        格式: --readyParas <file> (Exists,File)
子集选择参数:
  --subject,-s    提取指定样本的基因型信息.
                  格式: --subject <string>,<string>,... 或 --subject @file
  --range,-r      按照指定的坐标范围提取基因型信息.
                  格式: --range <chrom>:<minPos>-<maxPos> <chrom>:<minPos>-<maxPos> ...
  --random        按照指定的坐标提取基因型信息. (坐标信息保存在文件中, 每一行的格式为
                  'chrom,position' 或 'chrom position', 一行代表一个位点)
                  格式: --random <file>
  --retain-node   按照指定的 GTB 节点索引提取基因型信息.
                  格式: --retain-node <string>:<int>-<int> <string>:<int>-<int> ...
  --seq-ac        移除等位基因计数不在 [minAc, maxAc] 范围点的位点.
                  格式: --seq-ac <int>-<int> (>= 0)
  --seq-af        移除等位基因频率不在 [minAf, maxAf] 范围点的位点.
                  格式: --seq-af <double>-<double> (0.0 ~ 1.0)
  --seq-an        移除有效等位基因个数不在 [minAn, maxAn] 范围点的位点.
                  格式: --seq-an <int>-<int> (>= 0)
  --max-allele    移除等位基因种类超过指定值的位点.
                  默认值: 15
                  格式: --max-allele <int> (2 ~ 15)
```

程序实例

使用 GBC 解压示例文件 `./example/assoc.hg19.gtb`，并设置以下参数：

- 基因型设置为 unphased；
- 提取坐标范围大于 1000000 的位点；
- 提取等位基因频率在 [0.4, 0.6] 范围内的位点；
- 提取样本名为 NA18963,NA18977,HG02401,HG02353,HG02064 的基因型。

完成该任务的指令如下：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
extract ./example/assoc.hg19.gtb -o ./example/assoc.hg19.extract.vcf \
-p true -r 1:1000000- --seq-af 0.4-0.6 -s NA18963,NA18977,HG02401,HG02353,HG02064 -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc extract ./example/assoc.hg19.gtb -o
./example/assoc.hg19.extract.vcf -p true -r 1:1000000- --seq-af 0.4-0.6 -s NA18963,NA18
977,HG02401,HG02353,HG02064 -y
```

显示 **GTB** 文件摘要信息

使用如下指令显示基因型文件 GTB 的摘要信息：

```
show <input> [options]
```

该指令用于快速访问 GTB 文件的信息，并将其输出到终端。Show 模式实际上包含了三种输出功能：

- 输出 GTB 文件的摘要信息 (如样本、位点个数，文件压缩参数，GTB 节点情况等)；
- 列出 GTB 文件的样本名；
- 列出满足子集筛选条件的位点摘要信息（如坐标、等位基因标签、等位基因频率等）。

详细的基因型数据及多样化的输出格式控制需要使用 Extract 模式。

程序参数

```
语法: show <input> [options]
参数:
  --contig  指定染色体标签文件.
            默认值: /contig/human/hg38.p13
            格式: --contig <file> (Exists,File,Inner)
摘要信息访问参数:
  --add-md5       打印文件的 MD5 码.
  --add-subject   打印 GTB 文件的样本序列.
  --add-tree      打印 GTB 文件的节点树表 (染色体级别).
  --add-node      打印 GTB 文件的节点树表 (节点级别).
  --full,-f       打印所有的摘要信息 (MD5码 除外).
GTB 文件访问参数:
  --list-subject-only   仅打印 GTB 文件的样本信息.
  --list-position-only  仅打印 GTB 文件的坐标信息.
  --list-site-only      仅打印位点的摘要信息 (坐标, 等位基因, INFO 信息).
子集选择参数 (仅在 --list-position-only 和 --list-site-only 下可用):
  --subject,-s    提取指定样本的基因型信息.
                  格式: --subject <string>,<string>,... 或 --subject @file
  --range,-r      按照指定的坐标范围提取基因型信息.
                  格式: --range <chrom>:<minPos>-<maxPos> <chrom>:<minPos>-<maxPos> ...
  --random        按照指定的坐标提取基因型信息. (坐标信息保存在文件中, 每一行的格式为
                  'chrom,position' 或 'chrom position', 一行代表一个位点)
                  格式: --random <file>
  --retain-node   按照指定的 GTB 节点索引提取基因型信息.
                  格式: --retain-node <string>:<int>-<int> <string>:<int>-<int> ...
  --seq-ac        移除等位基因计数不在 [minAc, maxAc] 范围点的位点.
                  格式: --seq-ac <int>-<int> (>= 0)
  --seq-af        移除等位基因频率不在 [minAf, maxAf] 范围点的位点.
                  格式: --seq-af <double>-<double> (0.0 ~ 1.0)
  --seq-an        移除有效等位基因个数不在 [minAn, maxAn] 范围点的位点.
                  格式: --seq-an <int>-<int> (>= 0)
  --max-allele    移除等位基因种类超过指定值的位点.
                  默认值: 15
                  格式: --max-allele <int> (2 ~ 15)
```

程序实例

使用 GBC 列出文件 `./example/assoc.hg19.gtb` 的摘要信息，并设置以下参数：

- 列出所有的 GTB 节点信息；
- 列出文件的 MD5 码。

完成该任务的指令如下：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
show ./example/assoc.hg19.gtb \
--full --add-md5

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc extract ./example/assoc.hg19.gtb s
how ./example/assoc.hg19.gtb --full --add-md5
```

此时，终端打印如下信息：

```
Summary of GTB File:
  GTB File Name: /Users/suranyi/Documents/project/GBC/GBC-1.1/example/assoc.hg19.gtb
  GTB File Size: 938.092 KB
  Genome Reference: ftp://ftp.1000genomes.ebi.ac.uk//vol1/ftp/technical/reference/phase
2_reference_assembly_sequence/hs37d5.fa.gz
  MD5 Code: dcb53e6a9844d413e05ea2a95cae7289
  Suggest To BGZF: false
  Phased: true
  Ordered GTB: true
  BlockSize: 16384 (-bs 7)
  Compression Level: 16 (ZSTD)
  Dimension of Genotypes: 1 chromosome, 18339 variants and 983 subjects
  Subject Sequence: HG01583 HG01586 HG01589 HG01593 HG02490 HG02491 HG02493 HG02494 HG0
2597 HG02600
                    HG02601 HG02603 HG02604 HG02648 HG02649 HG02651 HG02652 HG02654 HG02
655 HG02657
                    HG02658 HG02660 HG02661 HG02681 HG02682 HG02684 HG02685 HG02687 HG02
688 HG02690
                    HG02691 HG02694 HG02696 HG02697 HG02699 HG02700 HG02724 HG02725 HG02
727 HG02728
                    HG02731 HG02733 HG02734 HG02736 HG02737 HG02774 HG02775 HG02778 HG02
780 HG02783
                    HG02784 HG02786 HG02787 HG02789 HG02790 HG02792 HG02793 HG03006 HG03
007 HG03009
                    HG03012 HG03015 HG03016 HG03018 HG03019 HG03021 HG03022 HG03228 HG03
229 HG03234
                    HG03235 HG03237 HG03238 HG03488 HG03490 HG03491 HG03585 HG03589 HG03
593 HG03594
                    HG03595 HG03598 HG03600 HG03603 HG03604 HG03607 HG03611 HG03615 HG03
616 HG03619
                    HG03624 HG03625 HG03629 HG03631 HG03634 HG03636 HG03640 HG03642 HG03
643 HG03644
                    HG03645 HG03646 HG03649 HG03652 HG03653 HG03660 HG03663 HG03667 HG03
668 HG03672
```

```
                    HG03673 HG03679 HG03680 HG03681 HG03684 HG03685 HG03686 HG03687 HG03
689 HG03690
                    HG03691 HG03692 HG03693 HG03694 HG03695 HG03696 HG03697 HG03698 HG03
702 HG03703
                    HG03705 HG03706 HG03708 HG03709 HG03711 HG03713 HG03714 HG03716 HG03
717 HG03718
                    HG03720 HG03722 HG03727 HG03729 HG03730 HG03731 HG03733 HG03736 HG03
738 HG03740
                    HG03741 HG03742 HG03743 HG03744 HG03745 HG03746 HG03750 HG03752 HG03
753 HG03754
                    HG03755 HG03756 HG03757 HG03760 HG03762 HG03765 HG03767 HG03770 HG03
771 HG03772
                    HG03773 HG03774 HG03775 HG03777 HG03778 HG03779 HG03780 HG03781 HG03
782 HG03784
                    HG03785 HG03786 HG03787 HG03788 HG03789 HG03790 HG03792 HG03793 HG03
796 HG03800
                    HG03802 HG03803 HG03805 HG03808 HG03809 HG03812 HG03814 HG03815 HG03
817 HG03821
                    HG03823 HG03824 HG03826 HG03829 HG03830 HG03832 HG03833 HG03836 HG03
837 HG03838
                    HG03844 HG03846 HG03848 HG03849 HG03850 HG03851 HG03854 HG03856 HG03
857 HG03858
                    HG03861 HG03862 HG03863 HG03864 HG03866 HG03867 HG03868 HG03869 HG03
870 HG03871
                    HG03872 HG03873 HG03874 HG03875 HG03882 HG03884 HG03885 HG03886 HG03
887 HG03888
                    HG03890 HG03894 HG03895 HG03896 HG03897 HG03898 HG03899 HG03900 HG03
902 HG03905
                    HG03907 HG03908 HG03910 HG03911 HG03913 HG03914 HG03916 HG03917 HG03
919 HG03920
                    HG03922 HG03925 HG03926 HG03928 HG03931 HG03934 HG03937 HG03940 HG03
941 HG03943
                    HG03945 HG03947 HG03949 HG03950 HG03951 HG03953 HG03955 HG03960 HG03
963 HG03965
                    HG03967 HG03968 HG03969 HG03971 HG03973 HG03974 HG03976 HG03977 HG03
978 HG03985
                    HG03986 HG03989 HG03990 HG03991 HG03995 HG03998 HG03999 HG04001 HG04
002 HG04003
                    HG04006 HG04014 HG04015 HG04017 HG04018 HG04019 HG04020 HG04022 HG04
023 HG04025
                    HG04026 HG04029 HG04033 HG04035 HG04038 HG04039 HG04042 HG04047 HG04
054 HG04056
                    HG04059 HG04060 HG04061 HG04062 HG04063 HG04070 HG04075 HG04076 HG04
080 HG04090
                    HG04093 HG04094 HG04096 HG04098 HG04099 HG04100 HG04106 HG04107 HG04
118 HG04131
                    HG04134 HG04140 HG04141 HG04144 HG04146 HG04152 HG04153 HG04155 HG04
156 HG04158
                    HG04159 HG04161 HG04162 HG04164 HG04171 HG04173 HG04176 HG04177 HG04
180 HG04182
                    HG04183 HG04185 HG04186 HG04188 HG04189 HG04194 HG04195 HG04198 HG04
200 HG04202
                    HG04206 HG04209 HG04210 HG04211 HG04212 HG04214 HG04216 HG04219 HG04
222 HG04225
                    HG04227 HG04229 HG04235 HG04238 HG04239 NA20845 NA20846 NA20847 NA20
849 NA20850
```

```
                    NA20851 NA20852 NA20853 NA20854 NA20856 NA20858 NA20859 NA20861 NA20
862 NA20863
                    NA20864 NA20866 NA20867 NA20868 NA20869 NA20870 NA20872 NA20875 NA20
876 NA20877
                    NA20878 NA20881 NA20882 NA20884 NA20885 NA20886 NA20887 NA20888 NA20
889 NA20890
                    NA20891 NA20892 NA20894 NA20895 NA20896 NA20897 NA20899 NA20900 NA20
901 NA20902
                    NA20903 NA20904 NA20905 NA20906 NA20908 NA20911 NA21086 NA21087 NA21
088 NA21089
                    NA21090 NA21091 NA21092 NA21093 NA21094 NA21095 NA21097 NA21098 NA21
099 NA21100
                    NA21101 NA21102 NA21103 NA21104 NA21105 NA21106 NA21107 NA21108 NA21
109 NA21110
                    NA21111 NA21112 NA21113 NA21114 NA21115 NA21116 NA21117 NA21118 NA21
119 NA21120
                    NA21122 NA21123 NA21124 NA21125 NA21126 NA21127 NA21128 NA21129 NA21
130 NA21133
                    NA21135 NA21137 NA21141 NA21142 NA21143 NA21144 HG00403 HG00404 HG00
406 HG00407
                    HG00409 HG00410 HG00419 HG00421 HG00422 HG00428 HG00436 HG00437 HG00
442 HG00443
                    HG00445 HG00446 HG00448 HG00449 HG00451 HG00452 HG00457 HG00458 HG00
463 HG00464
                    HG00472 HG00473 HG00475 HG00476 HG00478 HG00479 HG00500 HG00513 HG00
525 HG00530
                    HG00531 HG00533 HG00534 HG00536 HG00537 HG00542 HG00543 HG00556 HG00
557 HG00559
                    HG00560 HG00565 HG00566 HG00580 HG00583 HG00589 HG00590 HG00592 HG00
593 HG00595
                    HG00596 HG00598 HG00599 HG00607 HG00608 HG00610 HG00611 HG00613 HG00
614 HG00619
                    HG00620 HG00622 HG00623 HG00625 HG00626 HG00628 HG00629 HG00631 HG00
632 HG00634
                    HG00650 HG00651 HG00653 HG00654 HG00656 HG00657 HG00662 HG00663 HG00
671 HG00672
                    HG00674 HG00675 HG00683 HG00684 HG00689 HG00690 HG00692 HG00693 HG00
698 HG00699
                    HG00701 HG00704 HG00705 HG00707 HG00708 HG00717 HG00728 HG00729 HG00
759 HG00766
                    HG00844 HG00851 HG00864 HG00879 HG00881 HG00956 HG00982 HG01028 HG01
029 HG01031
                    HG01046 HG01595 HG01596 HG01597 HG01598 HG01599 HG01600 HG01794 HG01
795 HG01796
                    HG01797 HG01798 HG01799 HG01800 HG01801 HG01802 HG01804 HG01805 HG01
806 HG01807
                    HG01808 HG01809 HG01810 HG01811 HG01812 HG01813 HG01815 HG01816 HG01
817 HG01840
                    HG01841 HG01842 HG01843 HG01844 HG01845 HG01846 HG01847 HG01848 HG01
849 HG01850
                    HG01851 HG01852 HG01853 HG01855 HG01857 HG01858 HG01859 HG01860 HG01
861 HG01862
                    HG01863 HG01864 HG01865 HG01866 HG01867 HG01868 HG01869 HG01870 HG01
871 HG01872
                    HG01873 HG01874 HG01878 HG02016 HG02017 HG02019 HG02020 HG02023 HG02
025 HG02026
```

```
                    HG02028 HG02029 HG02031 HG02032 HG02035 HG02040 HG02047 HG02048 HG02
049 HG02050
                    HG02057 HG02058 HG02060 HG02061 HG02064 HG02069 HG02070 HG02072 HG02
073 HG02075
                    HG02076 HG02078 HG02079 HG02081 HG02082 HG02084 HG02085 HG02086 HG02
087 HG02088
                    HG02113 HG02116 HG02121 HG02122 HG02127 HG02128 HG02130 HG02131 HG02
133 HG02134
                    HG02136 HG02137 HG02138 HG02139 HG02140 HG02141 HG02142 HG02151 HG02
152 HG02153
                    HG02154 HG02155 HG02156 HG02164 HG02165 HG02166 HG02178 HG02179 HG02
180 HG02181
                    HG02182 HG02184 HG02185 HG02186 HG02187 HG02188 HG02190 HG02250 HG02
351 HG02353
                    HG02355 HG02356 HG02360 HG02364 HG02367 HG02371 HG02374 HG02375 HG02
379 HG02380
                    HG02382 HG02383 HG02384 HG02385 HG02386 HG02389 HG02390 HG02391 HG02
392 HG02394
                    HG02395 HG02396 HG02397 HG02398 HG02399 HG02401 HG02402 HG02406 HG02
407 HG02408
                    HG02409 HG02410 HG02512 HG02513 HG02521 HG02522 NA18525 NA18526 NA18
528 NA18530
                    NA18531 NA18532 NA18533 NA18534 NA18535 NA18536 NA18537 NA18538 NA18
539 NA18541
                    NA18542 NA18543 NA18544 NA18545 NA18546 NA18547 NA18548 NA18549 NA18
550 NA18552
                    NA18553 NA18555 NA18557 NA18558 NA18559 NA18560 NA18561 NA18562 NA18
563 NA18564
                    NA18565 NA18566 NA18567 NA18570 NA18571 NA18572 NA18573 NA18574 NA18
577 NA18579
                    NA18582 NA18591 NA18592 NA18593 NA18595 NA18596 NA18597 NA18599 NA18
602 NA18603
                    NA18605 NA18606 NA18608 NA18609 NA18610 NA18611 NA18612 NA18613 NA18
614 NA18615
                    NA18616 NA18617 NA18618 NA18619 NA18620 NA18621 NA18622 NA18623 NA18
624 NA18625
                    NA18626 NA18627 NA18628 NA18629 NA18630 NA18631 NA18632 NA18633 NA18
634 NA18635
                    NA18636 NA18637 NA18638 NA18639 NA18640 NA18641 NA18642 NA18643 NA18
644 NA18645
                    NA18646 NA18647 NA18648 NA18740 NA18745 NA18747 NA18748 NA18749 NA18
757 NA18939
                    NA18940 NA18941 NA18942 NA18943 NA18944 NA18945 NA18946 NA18947 NA18
948 NA18949
                    NA18950 NA18951 NA18952 NA18953 NA18954 NA18956 NA18957 NA18959 NA18
960 NA18961
                    NA18962 NA18963 NA18964 NA18965 NA18966 NA18967 NA18968 NA18969 NA18
970 NA18971
                    NA18972 NA18973 NA18974 NA18975 NA18976 NA18977 NA18978 NA18979 NA18
980 NA18981
                    NA18982 NA18983 NA18984 NA18985 NA18986 NA18987 NA18988 NA18989 NA18
990 NA18991
                    NA18992 NA18993 NA18994 NA18995 NA18997 NA18998 NA18999 NA19000 NA19
001 NA19002
                    NA19003 NA19004 NA19005 NA19006 NA19007 NA19009 NA19010 NA19011 NA19
012 NA19054
```

```
                    NA19055 NA19056 NA19057 NA19058 NA19059 NA19060 NA19062 NA19063 NA19
064 NA19065
                    NA19066 NA19067 NA19068 NA19070 NA19072 NA19074 NA19075 NA19076 NA19
077 NA19078
                    NA19079 NA19080 NA19081 NA19082 NA19083 NA19084 NA19085 NA19086 NA19
087 NA19088
                    NA19089 NA19090 NA19091

Summary of GTB Nodes:
└─ Chromosome 1: posRange=[10177, 4999852], numOfNodes=4, numOfVariants=18339
    ├─ Node 1: posRange=[10177, 1787378], seek=1157, blockSize=241089, variantNum=4430 (
4430 + 0)
    ├─ Node 2: posRange=[1788685, 2914618], seek=242246, blockSize=228815, variantNum=3
998 (3998 + 0)
    ├─ Node 3: posRange=[2915041, 4063039], seek=471061, blockSize=249262, variantNum=4
818 (4818 + 0)
    └─ Node 4: posRange=[4063447, 4999852], seek=720323, blockSize=240183, variantNum=5
093 (5093 + 0)
```

使用 GBC 列出文件 `./example/assoc.hg19.gtb` 中满足以下条件的位点：

- 提取坐标范围在 0-100000 的位点；
- 提取等位基因频率在 [0.45, 0.55] 范围内的位点；
- 提取样本名为 NA18963,NA18977,HG02401,HG02353,HG02064 的基因型。

完成该任务的指令如下：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
show ./example/assoc.hg19.gtb \
-r 1:0-100000 --seq-af 0.45-0.55 -s NA18963,NA18977,HG02401,HG02353,HG02064 --list-site
-only

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc show ./example/assoc.hg19.gtb -r 1
:0-100000 --seq-af 0.45-0.55 -s NA18963,NA18977,HG02401,HG02353,HG02064 --list-site-onl
y
```

此时，终端输出以下信息：

```
1    15211    T    G        AC=5;AF=0.50000000;AN=10
1    54712    T    TTTTC    AC=5;AF=0.50000000;AN=10
```

按照坐标对 **GTB** 文件排序

使用如下指令对基因型文件 GTB 进行排序：

```
sort <input> -o <output> [options]
```

通常 VCF 文件都是有序的，该功能可以忽略。当您使用参考本版本转换 (例如从 hg19 切换到 hg38)时，位点的坐标可能会发生变化，导致文件变成无序。

程序参数

```
语法: sort <input> -o <output> [options]
参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --threads,-t   设置并行压缩线程数.
                 默认值: 4
                 格式: --threads <int> (>= 1)
  --subject,-s   提取指定样本的基因型信息.
                 格式: --subject <string>,<string>,... 或 --subject @file
  --yes,-y       覆盖输出文件.
GTB 存档参数:
  --phased,-p           设置输出基因型的向型. (默认与输入的 GTB 文件向型一致)
                        格式: --phased [true/false]
  --biallelic           将多个多等位基因位点分裂为多个二等位基因位点.
  --simply              删除 ALT 中等位基因计数值为 0 的标签.
  --blockSizeType,-bs   设置每个压缩块的最大位点数，根据 2^(7+x) 换算得到真实的块大小值.
                        默认值: -1 (即根据样本量自动设置)
                        格式: --blockSizeType <int> (-1 ~ 7)
  --no-reordering,-nr   禁用 Approximate Minimum Discrepancy Ordering (AMDO) 算法.
  --windowSize,-ws      设置 AMDO 算法的采样窗口大小.
                        默认值: 24
                        格式: --windowSize <int> (1 ~ 131072)
  --compressor,-c       设置基压缩器.
                        默认值: ZSTD
                        格式: --compressor <string> ([ZSTD/LZMA/GZIP] or
                        [0/1/2] (ignoreCase))
  --level,-l            基压缩器的压缩级别. (ZSTD: 0~22, 默认为 3; LZMA: 0~9,
                        默认为 3; GZIP: 0~9, 默认为 5)
                        默认值: -1
                        格式: --level <int> (-1 ~ 31)
  --readyParas,-rp      从外部 GTB 文件中导入模版参数 (-p, -bs, -c, -l).
                        格式: --readyParas <file> (Exists,File)
  --seq-ac              移除等位基因计数不在 [minAc, maxAc] 范围点的位点.
                        格式: --seq-ac <int>-<int> (>= 0)
  --seq-af              移除等位基因频率不在 [minAf, maxAf] 范围点的位点.
                        格式: --seq-af <double>-<double> (0.0 ~ 1.0)
  --seq-an              移除有效等位基因个数不在 [minAn, maxAn] 范围点的位点.
```

```
                        格式: --seq-an <int>-<int> (>= 0)
  --max-allele          移除等位基因种类超过指定值的位点.
                        默认值: 15
                        格式: --max-allele <int> (2 ~ 15)
```

程序实例

使用 GBC 压缩无序的 VCF 文件 `./example/randomsimu100000V_100S.chr1.vcf.gz`：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
build ./example/randomsimu100000V_100S.chr1.vcf.gz -o ./example/randomsimu100000V_100S.
chr1.gtb -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc build ./example/randomsimu100000V_
100S.chr1.vcf.gz -o ./example/randomsimu100000V_100S.chr1.gtb -y
```

查看 `./example/randomsimu100000V_100S.chr1.gtb` 文件摘要信息

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
show ./example/randomsimu100000V_100S.chr1.gtb

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc show ./example/randomsimu100000V_1
00S.chr1.gtb
```

此时，终端打印如下信息，提示该文件是 `unordered` 的文件：

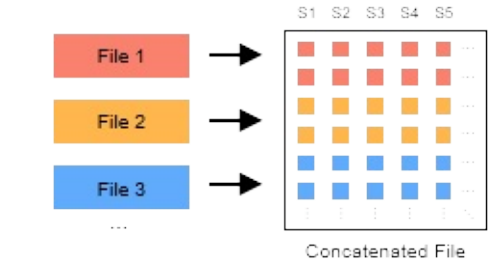
```
Summary of GTB File:
  GTB File Name: /Users/suranyi/Documents/project/GBC/GBC-1.1/example/randomsimu100000V
_100S.chr1.gtb
  GTB File Size: 2.764 MB
  Suggest To BGZF: false
  Phased: false
  Ordered GTB: false
  BlockSize: 16384 (-bs 7)
  Compression Level: 3 (ZSTD)
  Dimension of Genotypes: 1 chromosome, 100000 variants and 100 subjects
```

使用 `sort` 指令为该文件排序：

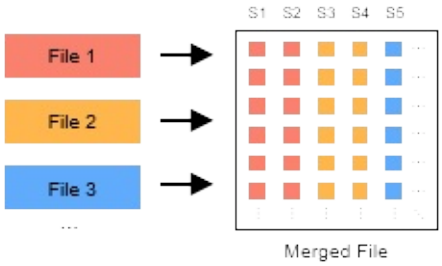
```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
sort ./example/randomsimu100000V_100S.chr1.gtb -o ./example/randomsimu100000V_100S.chr1
.order.gtb

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc sort ./example/randomsimu100000V_1
00S.chr1.gtb -o ./example/randomsimu100000V_100S.chr1.order.gtb
```

合并多个 **GTB** 文件

多个 GTB 文件的合并有两种主要类型：串联 (concat, 左图) 和合并 (merge, 右图)。前者通常用于连接多个单染色体的 GTB 文件 (如: chr1.gtb, chr2.gtb, ...)，这些 GTB 文件的位点之间不重叠。后者通常用于合并多个不同测序项目的基因型数据，以增大研究的样本量，这些 GTB 文件的样本之间不重叠。



Concatenate genotypes with identical subjects



Merge genotypes with non-overlapping samples

串联多个 **GTB** 文件

使用如下指令连接多个具有相同样本的基因型 GTB 文件 (左图)：

```
concat <input(s)> -o <output> [options]
```

- 当输入的多个文件具有相同的基因型向型、压缩器索引、样本信息时，文件连接将在几秒内完成。不满足条件时，GBC 需要更多的时间进行文件的预转换。
- 输入文件可以是单个或多个 .gtb 文件，也可以是包含这些文件的文件夹路径。当路径为文件夹路径时，GBC 会筛选出该文件夹 (及其子文件夹) 中所有的 .gtb 文件进行连接。请注意，GBC 仅根据文件的扩展名判断文件类型，因此正确的文件扩展名才能够进行连接。

程序参数

```
语法: concat <input(s)> -o <output> [options]
参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --yes,-y       覆盖输出文件.
```

程序实例

使用 GBC 压缩包含多个染色体位点的示例文件 `./example/simu100.coding.vcf.gz`：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
build ./example/simu100.coding.vcf.gz -o ./example/simu100.coding.gtb -y
```

```
# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc build ./example/simu100.coding.vcf
.gz -o ./example/simu100.coding.gtb -y
```

按照染色体编号拆分 `./example/simu100.coding.gtb` 文件：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
split ./example/simu100.coding.gtb -o ./example/simu100.coding --by chromosome

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc split ./example/simu100.coding.gtb
 -o ./example/simu100.coding --by chromosome
```

连接 `./example/simu100.coding` 文件夹中的所有 GTB 文件：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
concat ./example/simu100.coding -o ./example/simu100.coding.concat.gtb

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc concat ./example/simu100.coding -o
 ./example/simu100.coding.concat.gtb
```

合并多个 **GTB** 文件

使用如下指令合并多个具有不重叠样本的基因型 GTB 文件 (右图)：

```
merge <input(s)> -o <output> [options]
```

- 输入文件可以是单个或多个 .gtb 文件，也可以是包含这些文件的文件夹路径。当路径为文件夹路径时，GBC 会筛选出该文件夹 (及其子文件夹) 中所有的 .gtb 文件进行连接。请注意，GBC 仅根据文件的扩展名判断文件类型，因此正确的文件扩展名才能够进行连接；
- 当文件的样本名存在重叠时，需要使用 `reset-subject` 方法重设样本名；
- 未设置 GTB 存档格式时，输出的 GTB 存档格式默认与第一个传入的文件一致。

程序参数

```
语法: merge <input(s)> -o <output> [options]
参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --threads,-t   设置并行压缩线程数.
                 默认值: 4
                 格式: --threads <int> (>= 1)
  --union        处理不同文件中的位点的策略 (默认为取交集, 传入 `--union` 时将取并集).
```

```
  --yes,-y       覆盖输出文件.
GTB 存档参数:
  --phased,-p           设置输出基因型的向型. (默认与输入的 GTB 文件向型一致)
                        格式: --phased [true/false]
  --biallelic           将多个多等位基因位点分裂为多个二等位基因位点.
  --simply              删除 ALT 中等位基因计数值为 0 的标签.
  --blockSizeType,-bs   设置每个压缩块的最大位点数根据 2^(7+x) 换算得到真实的块大小值.
                        默认值: -1 (即根据样本量自动设置)
                        格式: --blockSizeType <int> (-1 ~ 7)
  --no-reordering,-nr   禁用 Approximate Minimum Discrepancy Ordering (AMDO) 算法.
  --windowSize,-ws      设置 AMDO 算法的采样窗口大小.
                        默认值: 24
                        格式: --windowSize <int> (1 ~ 131072)
  --compressor,-c       设置基压缩器.
                        默认值: ZSTD
                        格式: --compressor <string> ([ZSTD/LZMA/GZIP] or
                        [0/1/2] (ignoreCase))
  --level,-l            基压缩器的压缩级别. (ZSTD: 0~22, 默认为 3; LZMA: 0~9,
                        默认为 3; GZIP: 0~9, 默认为 5)
                        默认值: -1
                        格式: --level <int> (-1 ~ 31)
  --readyParas,-rp      从外部 GTB 文件中导入模版参数 (-p, -bs, -c, -l).
                        格式: --readyParas <file> (Exists,File)
  --seq-ac              移除等位基因计数不在 [minAc, maxAc] 范围点的位点.
                        格式: --seq-ac <int>-<int> (>= 0)
  --seq-af              移除等位基因频率不在 [minAf, maxAf] 范围点的位点.
                        格式: --seq-af <double>-<double> (0.0 ~ 1.0)
  --seq-an              移除有效等位基因个数不在 [minAn, maxAn] 范围点的位点.
                        格式: --seq-an <int>-<int> (>= 0)
  --max-allele          移除等位基因种类超过指定值的位点.
                        默认值: 15
                        格式: --max-allele <int> (2 ~ 15)
```

程序实例

从 `http://pmglab.top/genotypes` 下载 1000GP3 数据集，按照不同的群体分别存放在 `./example/1000GP3` 文件夹下，路径结构如下：

```
- 1000GP3
  - AFR
  - AMR
  - EAS
  - EUR
  - SAS
- randomsimu100000V_100S.chr1.vcf.gz
- rare.disease.hg19.vcf.gz
- query.txt
- query_1000GP3.txt
- simu100.coding.vcf.gz
- assoc.hg19.vcf.gz
```

使用 GBC 压缩每个群体的基因型数据：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
build ./example/1000GP3/AFR -o ./example/1000GP3/AFR.gtb -l 16 -p -y
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
build ./example/1000GP3/AMR -o ./example/1000GP3/AMR.gtb -l 16 -p -y
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
build ./example/1000GP3/EAS -o ./example/1000GP3/EAS.gtb -l 16 -p -y
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
build ./example/1000GP3/EUR -o ./example/1000GP3/EUR.gtb -l 16 -p -y
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
build ./example/1000GP3/SAS -o ./example/1000GP3/SAS.gtb -l 16 -p -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc build ./example/1000GP3/AFR -o ./exa
mple/1000GP3/AFR.gtb  -l 16 -p -y
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc build ./example/1000GP3/AMR -o ./exa
mple/1000GP3/AMR.gtb  -l 16 -p -y
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc build ./example/1000GP3/EAS -o ./exa
mple/1000GP3/EAS.gtb  -l 16 -p -y
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc build ./example/1000GP3/EUR -o ./exa
mple/1000GP3/EUR.gtb  -l 16 -p -y
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc build ./example/1000GP3/SAS -o ./exa
mple/1000GP3/SAS.gtb  -l 16 -p -y
```

合并 `1000GP3` 文件夹下的 5 个群体的 GTB 文件:

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
merge ./example/1000GP3/AFR.gtb ./example/1000GP3/AMR.gtb ./example/1000GP3/EAS.gtb ./e
xample/1000GP3/EUR.gtb ./example/1000GP3/SAS.gtb -o ./example/1000GP3.gtb -l 16 -p true
 -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc merge ./example/1000GP3/AFR.gtb ./ex
ample/1000GP3/AMR.gtb ./example/1000GP3/EAS.gtb ./example/1000GP3/EUR.gtb ./example/100
0GP3/SAS.gtb -o ./example/1000GP3.gtb -l 16 -p true -y
```

重设样本名

使用如下指令重设基因型文件 GTB 的样本名：

```
reset-subject <input> -o <output> [options]
```

程序参数

```
语法: reset-subject <input> -o <output> [options]
参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --yes,-y       覆盖输出文件.
  --subject,-s   提取指定样本的基因型信息.
                 格式: --subject <string>,<string>,... 或 --subject @file
  --prefix       使用格式 `[prefix][number][suffix]` 重设样本名. `prefix` 用于设置前缀.
                 默认值: S_
                 格式: --prefix <string>
  --suffix       使用格式 `[prefix][number][suffix]` 重设样本名. `suffix` 用于设置后缀.
                 格式: --suffix <string>
  --begin        使用格式 `[prefix][number][suffix]` 重设样本名. `begin` 用于设置起始编码.
                 默认值: 1
                 格式: --begin <int>
```

程序实例

使用 GBC 压缩示例文件 `rare.disease.hg19.vcf.gz`：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
build ./example/rare.disease.hg19.vcf.gz -o ./example/rare.disease.hg19.gtb -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc build ./example/rare.disease.hg19.vc
f.gz -o ./example/rare.disease.hg19.gtb -y
```

重设样本名为 `CASE_6_1`，`CASE_7_1` 和 `CASE_8_1`：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
reset-subject ./example/rare.disease.hg19.gtb -o ./example/out.gtb --prefix CASE_ --beg
in 6 --suffix _1 -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc reset-subject ./example/rare.disease
```

```
.hg19.gtb -o ./example/out.gtb --prefix CASE_ --begin 6 --suffix _1 -y
```

或：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
reset-subject ./example/rare.disease.hg19.gtb -o ./example/out.gtb --subject CASE_6_1,C
ASE_7_1,CASE_8_1 -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc reset-subject ./example/rare.disease
.hg19.gtb -o ./example/out.gtb --subject CASE_6_1,CASE_7_1,CASE_8_1 -y
```

GTB 文件剪枝

使用如下指令对基因型文件 GTB 进行节点修剪，实现快速提取、删除操作：

```
prune <input> -o <output> [options]
```

与 `extract` 相比， `prune` 不需要解压任何数据实现节点的提取或删除，具有更快的速度和更低的内存负担，所有操作均能在数秒内完成。

程序参数

```
语法: prune <input> -o <output> [options]
参数:
  --contig         指定染色体标签文件.
                   默认值: /contig/human/hg38.p13
                   格式: --contig <file> (Exists,File,Inner)
  *--output,-o     设置输出文件名.
                   格式: --output <file>
  --yes,-y         覆盖输出文件.
  --delete-node    删除指定的 GTB 节点.
                   格式: --delete-node <string>:<int>,<int>,... <string>:<int>,<int>,...
...
  --retain-node    保留指定的 GTB 节点.
                   格式: --retain-node <string>:<int>,<int>,... <string>:<int>,<int>,...
...
  --delete-chrom   删除指定的染色体.
                   格式: --delete-chrom <string>,<string>,...
  --retain-chrom   保留指定的染色体.
                   格式: --retain-chrom <string>,<string>,...
```

程序实例

使用 GBC 提取 `1000GP3.gtb` 的性染色体 (chrX 和 chrY)：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
prune ./example/1000GP3.gtb -o ./example/1000GP3.chrXY.gtb \
--retain-chrom X,Y \
-y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc prune ./example/1000GP3.gtb -o ./exa
mple/1000GP3.chrXY.gtb --retain-chrom X,Y -y
```

查看提取的 GTB 文件信息：

```
# Linux 或 MacOS
```

```
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
show ./example/1000GP3.chrXY.gtb --add-tree

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc show ./example/1000GP3.chrXY.gtb --a
dd-tree
```

此时，终端输出以下信息：

```
Summary of GTB File:
  GTB File Name: /Users/suranyi/Documents/project/GBC/GBC-1.1/example/1000GP3.chrXY.gtb
  GTB File Size: 66.759 MB
  Genome Reference: ftp://ftp.1000genomes.ebi.ac.uk//vol1/ftp/technical/reference/phase
2_reference_assembly_sequence/hs37d5.fa.gz
  Suggest To BGZF: false
  Phased: true
  Ordered GTB: true
  BlockSize: 16384 (-bs 7)
  Compression Level: 16 (ZSTD)
  Dimension of Genotypes: 2 chromosomes, 3530137 variants and 2504 subjects

Summary of GTB Nodes:
├─ Chromosome X: posRange=[60020, 155260478], numOfNodes=212, numOfVariants=3468095
└─ Chromosome Y: posRange=[2655180, 28770931], numOfNodes=4, numOfVariants=62042
```

等位基因标签检查

在合并不同批次的基因型时，相同坐标位点上的不同等位基因标签可能带来混淆（例如: 批次 1 使用正链 DNA，批次 2 使用反链 DNA）。GBC 提供了基于等位基因频率、计数、上下游强关联位点的 LD 模式来校正潜在的正反链错配。使用如下指令进行等位基因的标签检查：

```
allele-check <template_input> <input> -o <output> [options]
```

在坐标相同、碱基互补的情况下（例如: [A, T] 和 [C, G]），GBC 实现了三种检查方法：

- 等位基因频率绝对差值：校正满足 $|\mathrm{AF}_1 - \mathrm{AF}_2| <$ –freq-gap 的等位基因标签，这种方法只适用于次级等位基因频率远小于 0.5 的情形。
- 等位基因计数的卡方检验：由来自 2 个文件的互补配对碱基的等位基因计数值构造 2×2 的列联表进行卡方检验，零假设为 `这两个位点的互补等位基因计数没有显著差异` （即可以看作标签是相同的）。如果没有拒绝零假设，则互补的碱基会被纠正。
- 上下游窗口的 LD 模式: 先筛选出上下游（窗口通过 `--window-bp` 控制）中公有的与当前位点的基因型相关性（LD 系数）的绝对值在两批中分别超过阈值（例如：0.8，通过 `--min-r` 控制）的位点，然后统计这些公有位点相关系数的正负号。如果正负符号的数量在两批中有较大差距（例如：超过 80% 位点对的符号是相反的，通过 `--flip-scan-threshold` 控制），等位基因标签应该被翻转；否则，等位基因标签不被翻转。这个功能可以用于检查等位基因频率接近 0.5 的常见变异。

程序参数

```
语法: allele-check <template_input> <input> -o <output> [options]
参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --threads,-t   设置并行压缩线程数.
                 默认值: 4
                 格式: --threads <int> (>= 1)
  --union        处理不同文件中的位点的策略 (默认为取交集, 传入 `--union` 时将取并集).
  --yes,-y       覆盖输出文件.
对齐坐标参数:
  --p-value              翻转次级等位基因频率 <= --maf 的位点中卡方检验 p 值 >=
                         --p-value 的等位基因标签.
                         默认值: 0.05
                         格式: --p-value <double> (1.0E-6 ~ 0.5)
  --freq-gap             翻转次级等位基因频率 <= --maf 的位点中等位基因频率差值 <=
                         --freq-gap 的等位基因标签.
                         格式: --freq-gap <double> (1.0E-6 ~ 0.5)
  --no-ld                默认情况下, 翻转次级等位基因频率 > --maf 的位点中上下游强关联位
                         点的 LD 模式超过反转比例阈值的等位基因标签. 使用该参数禁用该检查.
  --min-r                在 LD 模式检查中, 排除基因型相关性小于 --min-r 的位点.
                         默认值: 0.8
                         格式: --min-r <double> (0.5 ~ 1.0)
```

```
  --flip-scan-threshold  在 LD 模式检查中, 超过反转比例阈值.
                         默认值: 0.8
                         格式: --flip-scan-threshold <double> (0.5 ~ 1.0)
  --maf                  设置用于区分常见变异和罕见变异的次级等位基因频率阈值.
                         默认值: 0.05
                         格式: --maf <double> (0.0 ~ 1.0)
  --window-bp,-bp        设置计算上下游 LD 系数的最大物理距离 (单位: bp).
                         默认值: 10000
                         格式: --window-bp <int> (>= 1)
GTB 存档参数:
  --phased,-p            设置输出基因型的向型. (默认与输入的 GTB 文件向型一致)
                         格式: --phased [true/false]
  --biallelic            将多个多等位基因位点分裂为多个二等位基因位点.
  --simply               删除 ALT 中等位基因计数值为 0 的标签.
  --blockSizeType,-bs    设置每个压缩块的最大位点数, 根据 2^(7+x) 换算得到真实的块大小值.
                         默认值: -1 (即根据样本量自动设置)
                         格式: --blockSizeType <int> (-1 ~ 7)
  --no-reordering,-nr    禁用 Approximate Minimum Discrepancy Ordering (AMDO) 算法.
  --windowSize,-ws       设置 AMDO 算法的采样窗口大小.
                         默认值: 24
                         格式: --windowSize <int> (1 ~ 131072)
  --compressor,-c        设置基压缩器.
                         默认值: ZSTD
                         格式: --compressor <string> ([ZSTD/LZMA/GZIP] or
                         [0/1/2] (ignoreCase))
  --level,-l             基压缩器的压缩级别. (ZSTD: 0~22, 默认为 3; LZMA: 0~9,
                         默认为 3; GZIP: 0~9, 默认为 5)
                         默认值: -1
                         格式: --level <int> (-1 ~ 31)
  --readyParas,-rp       从外部 GTB 文件中导入模版参数 (-p, -bs, -c, -l).
                         格式: --readyParas <file> (Exists,File)
  --seq-ac               移除等位基因计数不在 [minAc, maxAc] 范围点的位点.
                         格式: --seq-ac <int>-<int> (>= 0)
  --seq-af               移除等位基因频率不在 [minAf, maxAf] 范围点的位点.
                         格式: --seq-af <double>-<double> (0.0 ~ 1.0)
  --seq-an               移除有效等位基因个数不在 [minAn, maxAn] 范围点的位点.
                         格式: --seq-an <int>-<int> (>= 0)
  --max-allele           移除等位基因种类超过指定值的位点.
                         默认值: 15
                         格式: --max-allele <int> (2 ~ 15)
```

程序实例

从 `http://pmglab.top/genotypes` 下载 `EAS hg38` 数据集，使用该数据集作为模版文件，检查本地外显子组测序数据 `SNP.gtb` 的等位基因标签：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
allele-check ./example/EAS.gtb ./example/SNP.gtb -o ./example/SNP.checked.gtb --seq-af
0.000001-0.999999 -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc allele-check ./example/EAS.gtb ./exa
mple/SNP.gtb -o ./example/SNP.checked.gtb --seq-af 0.000001-0.999999 -y
```

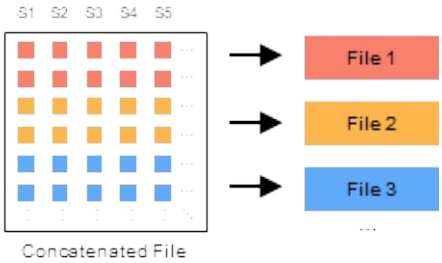
程序运行时，终端中输出检查日志：

```
2022-06-27 01:16:02 INFO  [ThreadPool-thread-1] AlleleCheck chr11:244961 TempREF=G Temp
ALT=C TempAF=0.8035714285714286 REF=G ALT=C AF=0.6944444444444444 -> REF=C ALT=G
2022-06-27 01:16:02 INFO  [ThreadPool-thread-1] AlleleCheck chr11:251057 TempREF=C Temp
ALT=G TempAF=0.08630952380952381 REF=C ALT=G AF=0.22093023255813954 -> REF=G ALT=C
2022-06-27 01:16:04 INFO  [ThreadPool-thread-1] AlleleCheck chr17:42969194 TempREF=C Te
mpALT=G TempAF=0.9742063492063492 REF=C ALT=G AF=0.02608695652173913 -> REF=G ALT=C
2022-06-27 01:16:06 INFO  [ThreadPool-thread-2] AlleleCheck chr1:1041823 TempREF=G Temp
ALT=C TempAF=0.9990079365079365 REF=G ALT=C AF=0.0045045045045045045 -> REF=C ALT=G
```

分裂 **GTB** 文件

使用如下指令分裂 GTB 文件为多个独立的子文件：

```
split <input> -o <output> [options]
```



程序参数

```
语法: split <input> -o <output> [options]
参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --by           按照染色体水平/节点水平分裂 GTB 文件为多个子文件.
                 默认值: chromosome
                 格式: --by <string> ([chromosome/node] or [0/1])
  --yes,-y       覆盖输出文件.
```

程序实例

按照染色体编号拆分 `./example/1000GP3.gtb` 文件：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 500m gbc \
split ./example/1000GP3.gtb -o ./example/1000GP3-chr -y

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 500m gbc split ./example/1000GP3.gtb -o ./e
xample/1000GP3-chr -y
```

运行该指令后，文件夹 `1000GP30-chr` 中产生 24 个 GTB 子文件。

计算 **LD** 系数

GBC 集成了 LD 系数的计算方法。使用如下指令计算基因型 GTB 文件的 LD 系数：

```
ld <input> -o <output> [options]
```

程序参数

```
语法: ld <input> -o <output> [options]
输出参数:
  --contig       指定染色体标签文件.
                 默认值: /contig/human/hg38.p13
                 格式: --contig <file> (Exists,File,Inner)
  *--output,-o   设置输出文件名.
                 格式: --output <file>
  --o-text       以文本格式输出位点的 LD 系数信息.
  --o-bgz        以 BGZIP 压缩的格式输出位点的 LD 系数信息.
  --level,-l     设置 BGZIP 压缩器的压缩级别.
  --threads,-t   设置并行压缩线程数.
                 默认值: 4
                 格式: --threads <int> (>= 1)
  --yes,-y       覆盖输出文件.
LD 计算参数:
  --hap-ld,--hap-r2     计算位点对的单倍型连锁不平衡系数.
  --geno-ld,--gene-r2   计算位点对的基因型连锁不平衡系数 (Pearson 相关系数).
  --window-bp,-bp       设置计算上下游 LD 系数的最大物理距离 (单位: bp).
                        默认值: 10000
                        格式: --window-bp <int> (>= 1)
  --min-r2              在 LD 模式检查中, 排除 R^2 小于 --min-r2 的位点.
                        默认值: 0.2
                        格式: --min-r2 <double> (0.0 ~ 1.0)
  --maf                 移除次级等位基因频率 < --maf 的位点.
                        默认值: 0.05
                        format: --maf <double> (0.0 ~ 0.5)
  --subject,-s          计算指定样本的 LD 系数.
                        格式: --subject <string>,<string>,...
  --range,-r            计算指定位点范围的 LD 系数.
                        格式: --range <chrom>:<minPos>-<maxPos>
                        <chrom>:<minPos>-<maxPos> ...
```

程序实例

计算 `EAS hg38` 数据集的连锁不平衡系数：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
ld ./example/EAS.gtb -o ./example/EAS.ld.gz -t 8 -y

# Windows
```

```
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc ld ./example/EAS.gtb -o ./example/EAS.ld.gz -t 8 -y
```

Contig 文件

GBC 程序内部并不显式地定义染色体标签的索引信息，而是通过 Contig 文件进行声明，位于 Contig 文件第一行的染色体索引为 0，第二行染色体的索引为 1... 因此，你可以轻易地扩展或者修改 Contig 文件，以实现基因型倍型、染色体标签的修改。

GBC 内置了人类基因组的染色体标签文件（见下表）。对于非人类的基因组，GBC 可能需要一个不同的 contig 文件来声明指定染色体的标签（如chrX, chrY, chrMT）和它的倍性，使得 GBC 也可以对其他单倍体和二倍体物种的基因型进行处理。

Contig 文件以 "#chromosome,ploidy,length "作为标题行，然后每行代表一个染色体。由于在GTB格式中只保留 1 个字节用于存储染色体数目，我们要求输入的contig文件中的染色体数目不超过 256。

```
##reference=https://www.ncbi.nlm.nih.gov/grc/human/data?asm=GRCh38.p13
#chromosome,ploidy,length
1,2,248956422
2,2,242193529
3,2,198295559
4,2,190214555
5,2,181538259
6,2,170805979
7,2,159345973
8,2,145138636
9,2,138394717
10,2,133797422
11,2,135086622
12,2,133275309
13,2,114364328
14,2,107043718
15,2,101991189
16,2,90338345
17,2,83257441
18,2,80373285
19,2,58617616
20,2,64444167
21,2,46709983
22,2,50818468
X,2,156040895
Y,2,57227415
MT,2,4485509
```

为 VCF 文件构建 Contig 文件

对于非人类基因型文件，使用如下指令为原始的 VCF 文件构建 Contig 文件：

```
index <input> -o <output> [options]
```

该方法将扫描 VCF 文件的注释信息，提取其中的 `##contig=` 字段并构建 Contig 文件。如果 VCF 文件为非标准文件，用户可以通过添加参数 `--deep-scan` 扫描整个 VCF 文件的 `CHROM` 字段，并构建相应的 Contig 文件。

重建染色体标签索引

使用如下指令重建 GTB 文件的染色体标签索引：

```
index <input> -o <output> --from-contig <from_contig> --to-contig <to_contig> [options]
```

例如，将 1000GP3 的所有染色体标签修改为 `Unknown`，需要建立如下文件：

```
#chromosome,ploidy,length
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
Unknown,2,0
```

然后运行：

```
# Linux 或 MacOS
docker run -v `pwd`:`pwd` -w `pwd` --rm -it -m 4g gbc \
index ./example/1000GP3.gtb -o ./example/1000GP3.unknown.gtb --to-contig ./example/1000
GP3.contig

# Windows
docker run -v %cd%:%cd% -w %cd% --rm -it -m 4g gbc index ./example/1000GP3.gtb -o ./exa
mple/1000GP3.unknown.gtb --to-contig ./example/1000GP3.contig
```