

# 宏基因组结构变异 (SV) 检测

原创 非常明 非常明 2021-12-28 16:41

收录于合集

#生物信息 7 #干货 7 #宏基因组 2



微信搜一搜



非常明

👉 长按上方图片关注，学习更多生信干货 👉

## 1. 概述

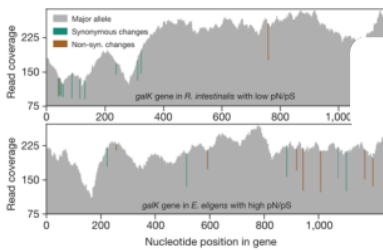
---

在高度可变的微生物基因组中，存在大量遗传变异，包括单核苷酸变异（Single-nucleotide variation, SNV）、拷贝数变异（Copy number variation, CNV）以及结构变异（Structural variation, SV）等等，这些遗传变异在微生物的表型塑造乃至宿主-微生物互作中起到了重要的作用。

## Microbial Genetic Variations

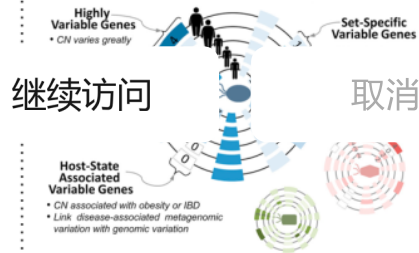
### ● SNV

Single nucleotide variation



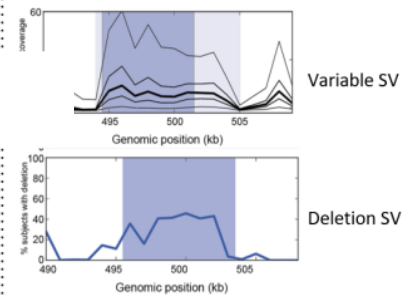
### ● CNV

Copy number variation



### ● SV

Genomic structural variation



#### ARTICLE

##### Genomic variation landscape of the human gut microbiome

Stephan Schirmer<sup>1\*</sup>, Marlene Schirmer<sup>1</sup>, Shihua Tang<sup>1</sup>, Mikaela Miron<sup>1</sup>, Yifan Tang<sup>1</sup>, Ana Zhai<sup>1</sup>, Oliver Müller<sup>1</sup>, Stefan A. Hahne<sup>1</sup>, Jeroen Raaijmakers<sup>1</sup>, John Storr<sup>1</sup>, Corinna Kueh<sup>1</sup>, Daniel R. Spector<sup>1</sup>, George M. Weinstock<sup>1</sup> & Peter Bork<sup>1</sup>

<https://doi.org/10.1038/nature11711>

#### Article

##### Extensive Strain-Level Copy-Number Variation across Human Gut Microbiome Species

Sharon Greenblum<sup>1</sup>, Roger Can<sup>1</sup> and Eshani Borenstein<sup>1,2,3\*</sup>

<http://dx.doi.org/10.1016/j.cell.2014.12.038>

#### ARTICLE

##### Structural variation in the gut microbiome associates with host health

David Zeevi<sup>1,2\*</sup>, Tal Korem<sup>1</sup>, Anastasia Godneva<sup>1</sup>, Noam Bar<sup>1</sup>, Alexander Kurilshikov<sup>1</sup>, Maya Lotan-Pompan<sup>1</sup>, Adina Weinberger<sup>1</sup>, Jingyuan Fu<sup>1</sup>, Cisca Wijmenga<sup>1</sup>, Alexandra Zhernakova<sup>1</sup> & Eran Segal<sup>1,2\*</sup>

<https://doi.org/10.1038/s41586-019-1065-y>

宏基因组测序是目前对人体及环境微生物群落进行探究的重要手段，但从宏基因组数据中对微生物的遗传变异进行检测及下游分析依然存在挑战，相关研究者也不断在开发相关的工具来提供合适的解决方案。SGVFinder是一个用于从宏基因组数据中检测微生物基因组SV的工具，于2019年发表在*Nature*上，在本文中我将主要对SGVFinder的使用方法进行介绍。

# nature

Explore content ▾

About the journal ▾



Publish with us ▾

Subscribe

[nature](#) > [articles](#) > [article](#)

Article | [Published: 27 March 2019](#)

## Structural variation in the gut microbiome associates with host health

[David Zeevi](#) , [Tal Korem](#), [Anastasia Godneva](#), [Noam Bar](#), [Alexander Kurilshikov](#), [Maya Lotan-Pompan](#), [Adina Weinberger](#), [Jingyuan Fu](#), [Cisca Wijmenga](#), [Alexandra Zhernakova](#) & [Eran Segal](#) 

[Nature](#) **568**, 43–48 (2019) | [Cite this article](#)

**36k** Accesses | **101** Citations | **422** Altmetric | [Metrics](#)

在正式介绍之前，想先吐槽一下这个工具.....小bug有点多并且一直没有维护更新，我修复了以后在Github上提交了解决方案也没有收到开发者响应，不知道下个版本什么时候出来。我来到目前课题组之后的第一个任务，就是搭建宏基因组SV检测的流程。在初步排除了一些问题后，我撰写了一份使用指南放在课题组的Github中 ([https://github.com/GRONINGEN-MICROBIOME-CENTRE/Groningen-Microbiome/tree/master/Scripts/SV\\_pipeline](https://github.com/GRONINGEN-MICROBIOME-CENTRE/Groningen-Microbiome/tree/master/Scripts/SV_pipeline))，以供同事及其他用户参考。后来我在多个不同队列数据中进行了大量实测，发现了一些新的问题。在这一过

程中，还有很多认识和不认识的同行用户看了我的指南后来咨询，也提出了一些我没有碰到过的问题。在这篇教程中，我会对这些问题及其解决方案进行汇总。文中的每个注意事项都很重要，别看说起来很自然而然，但最开始遇到报错排bug的过程真是费头发.....只希望别的用户不用再走弯路了。

继续访问

取消

## 2. 准备工作

---

### 2.1 前置软件

- **GEM Mapper**，注意不能使用GEM3，SGVFinder目前只兼容旧版本的GEM Mapper，例如版本2：

```
1 wget https://sourceforge.net/projects/gemlibrary/files/gem-library/Binary%20pr
2 bzip2 -dc GEM-binaries-Linux-x86_64-core_2-20121106-022124.tbz2 | tar -xvf -
```

将该软件的路径加入到环境变量中，这样就可以在系统中直接输入gem-mapper来调用GEM Mapper，将如下代码添加到~/.bashrc中：

```
1 export PATH="your_path/GEM-binaries-Linux-x86_64-core_2-20121106-022124:$PATH"
```

然后重新载入~/.bashrc文件来更新环境变量：

```
1 source ~/.bashrc
```

- **Python 2.7.8**以及相关的模块：
  - numpy (tested with 1.14.2)
  - biopython (tested with 1.68)
  - ujson (tested with 1.35)，**注意**一定要使用该版本！
  - pandas (tested with 0.23.4)
  - scipy (tested with 1.1.0)
  - bokeh (tested with 0.12.6)

强烈建议使用Anaconda，通过它可以非常方便地安装和管理Python模块、切换Python版本环境等：

```
1 conda create -n sgvfinder python=2.7 numpy biopython ujson pandas scipy bokeh
```

- Cpp11 [继续访问](#) [取消](#)
- Cython，也可以通过Anaconda来安装

## 2.2 下载和安装SGVFinder

下载软件和数据库，安装软件：

```
1 wget https://zenodo.org/record/3237975/files/DataFiles.tar.gz
2 tar DataFiles.tar.gz
3 git clone https://github.com/segalab/SGVFinder.git
4 mv DataFiles SGVFinder/
5 cd SGVFinder/src/cy_ext/
6 conda activate sgvfinder
7 python setup.py build_ext
```

## 2.3 Bug修复

或许是由于原始开发环境与用户使用环境的差别，该工具的原始脚本中有些小bug，需要手动修正，否则在运行中会出现奇怪的报错：

(1) SGVFinder.py

① Line 32，这里需要将pos1和average\_read\_length都强制转化为整型：

```
1 ind1 = int((pos1 + (average_read_length / 2)) / bin_size)
```

修改为：

```
1 ind1 = int((int(pos1) + (int(average_read_length) / 2)) / bin_size)
```

② Line 35，此处同上：

```
1 ind2 = int((pos2 + (average_read_length / 2)) / bin_size)
```

修改为：

```
1 ind2 = int((int(pos2 + (average_read_length / 2)) / bin_size)
```

(2) SGVF\_cmd.py

① Line 15, 此处严格来说并非错误, 但该处的默认参数与 SGVF\_PerFile.py 中的值不同, 如果直接以默认值运行的话则会导致错误:

```
1 parser.add_argument('--x_coverage', help = 'The desired coverage across the genome')
```

修改为:

```
1 parser.add_argument('--x_coverage', help = 'The desired coverage across the genome')
```

② Line 39, 这里是变量名引用错误:

```
1 if args.byother:
```

修改为:

```
1 if args.byorig:
```

### 3. 输入文件

---

SGVFinder对单端测序和双端测序数据都可以进行处理，下面会对两种测序数据都进行举例。

(1) 单端测序reads：

- input\_a.fastq
  - input\_b.fastq
  - input\_c.fastq
- 继续访问 取消

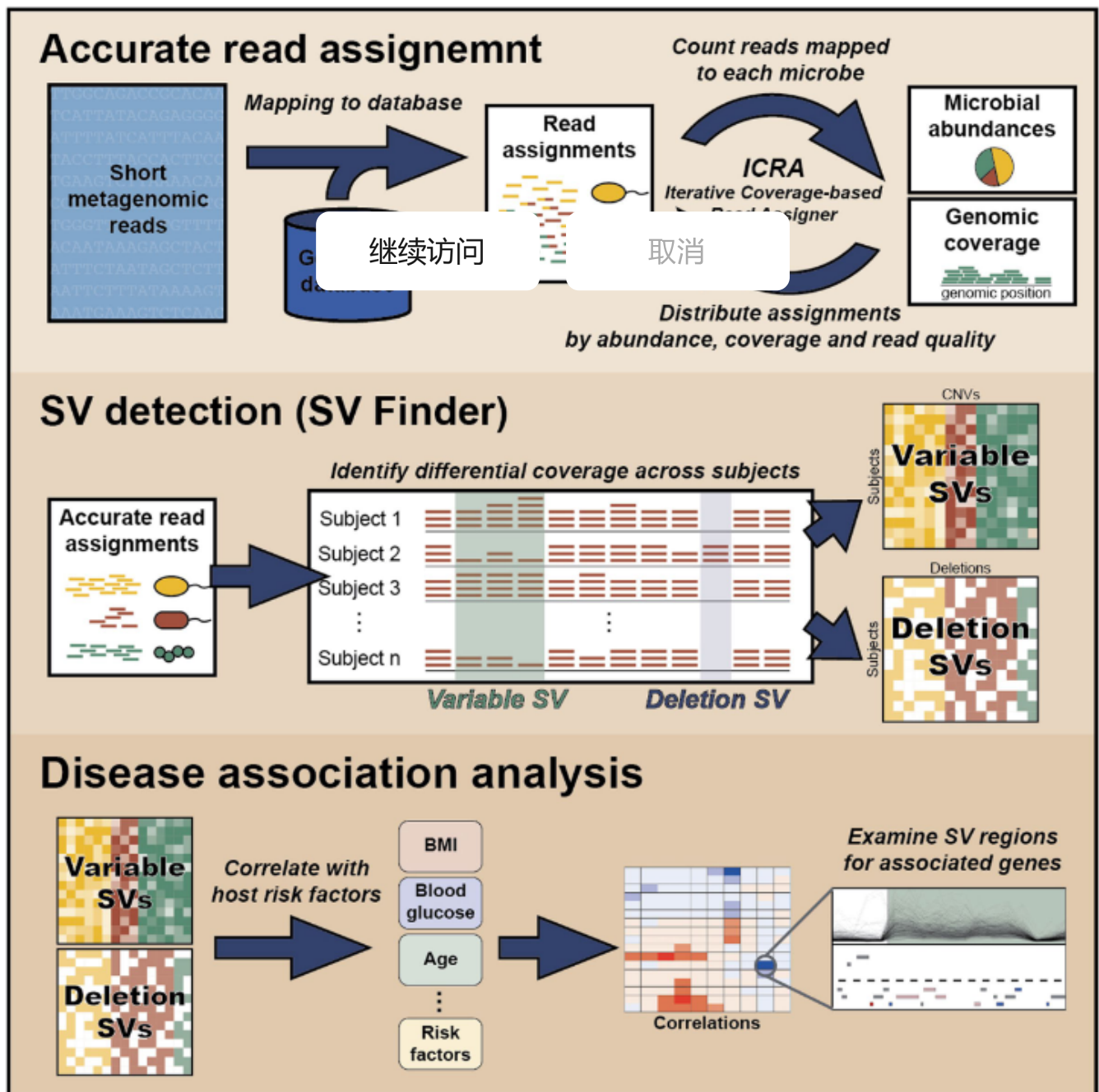
(2) 双端测序reads：

- input\_a\_1.fastq, input\_a\_2.fastq
- input\_b\_1.fastq, input\_b\_2.fastq
- input\_c\_1.fastq, input\_c\_2.fastq

## 4. 检测SV

---

如果使用的是人类肠道微生物宏基因组测序数据，那么在检测SV之前需要进行质控，去除宿主基因组来源的reads、接头序列和低质量reads，这里推荐使用kneaddata (<https://huttenhower.sph.harvard.edu/kneaddata/>)，不再详述。对于其他微生物宏基因组测序数据，假定以上输入文件均是质控后的clean files，下面便进入SV检测流程的主干部分。SV检测流程主要分为三大步骤：比对、覆盖度计算和群体变异检测。



来自原图 (Zeevi et al., 2019)

## 4.1 运行GEM mapper和ICRA

在这一步中，SGVFinder会使用GEM mapper将reads比对到参考基因组上，并通过ICRA (Iterative coverage-based read assignment) 算法对ambiguous reads进行分配。

对于单端测序reads：

```
1 mkdir -p tmp/icra
2 python ICRA_cmd.py tmp/icra input_a
```

对于双端测序reads:

```
1 mkdir -p tmp/icra
2 python ICRA_cmd.py tmp/icra input_a --pe
```

**Output:**

[继续访问](#)

[取消](#)

- tmp/icra/input\_a.jsdel
- tmp/icra/input\_b.jsdel
- tmp/icra/input\_c.jsdel

该步骤最高占用内存为22 Gb/样本, 运行时间为7小时/样本。

**注意:**

- 只接受未压缩的fastq文件;
- 'tmp/icra' 是一个目录而不是文件。

## 4.2 运行SGVF\_PerFile

产生每个样本在每个细菌参考基因组的每个片段 (bin) 上的覆盖度文件:

```
1 mkdir -p tmp/SGVF_PerFile
2 python SGVF_PerFile_cmd.py tmp/icra/input_a.jsdel tmp/SGVF_PerFile/input_a.jsdel
```

**Output:**

- tmp/SGVF\_PerFile/input\_a.jsdel
- tmp/SGVF\_PerFile/input\_b.jsdel
- tmp/SGVF\_PerFile/input\_c.jsdel

该步骤最高占用内存约为10 Gb/样本, 运行时间约为0.1小时/样本。

## 4.3 运行SGVF



在所有样本的fastq文件包括input\_a, input\_b, input\_c.....都被以上步骤处理后，便可以运行最后一步，也就是在群体水平上，检测标准化覆盖度高度可变的基因组片段，这一步需要在群体水平上跨样本进行统计学运算处理，因此SGVFinder无法只基于单个样本进行SV检测（除非使用的是基于reference的模式）。

```
1 mkdir -p results          继续访问          取消
2 mkdir -p results/html
3 python SGVF_cmd.py "tmp/SGVF_PerFile/*.jsdel" results/dsgv.csv results/vsgv.csv
```

该步骤最高占用内存为5 Gb/队列（10个样本），运行时间与队列大小有关：0.5小时/队列（10个样本），1小时/队列（1000个样本），5小时/队列（12000个样本）。

注意:

- 对于小群体, 推荐使用基于reference的检测模式，添加参数--byorig。这样产生的SV profile的区域就与开发团队在*Nature*上发表的以色列队列的SV区域一样；
- 本示例中，"tmp/SGVF\_PerFile/\*.jsdel" 叫做input\_glob\_string，是一个用引号括起来的、含文件名通配符的正则表达式例如："/my/path/\*.jsdel"；
- --min\_samp\_cutoff可以根据你的样本群体队列大小来调整，推荐使用10% \* sample size（小队列）或者默认值75（大队列）。

4.4 所需计算资源和运行时间

下面是我在小规模测试时用到的计算资源和运行时间，使用的样本来自Lifelines-DEEP队列（<https://www.lifelines.nl/researcher/data-and-biobank/additional-study/additional-data-samples-2>），可以大致参考下（请忽略测试时加入的subsampling步骤）：

	Raw data (1 sample)	Reads cleaning (1 sample)	Subsampling (1 sample)	ICRA (1 sample)	SGVF_PerFile (1 sample)	SGVF (10 samples)	Peak/Amount
Memory/Gb	-	30	50	22	10	5	50/-
Storage/Gb	8	40	3.4-12	30	0.001	0.02	40/100
Time/h	-	1	0.2	7	0.1	0.5	7/10

5. 最终结果

- results/dsgv.csv：deletion SVs profile

- results/vsgv.csv: variable SVs profile
- results/html/\*.html: 各个物种的SV区域的交互式可视化, 具体说明可以参见*Nature*文章的Extended Data Fig. 3。

## 6. 其他事项

继续访问

取消

### 6.1 格式转换

如果你没有使用--csv\_output这个参数, 你可以使用如下的python脚本进行格式转换, 把pandas data frame导出为CSV表格:

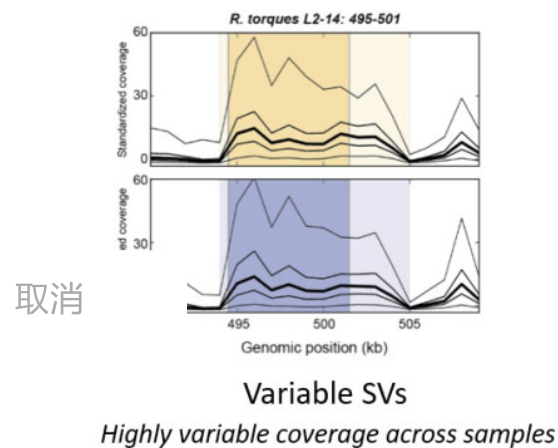
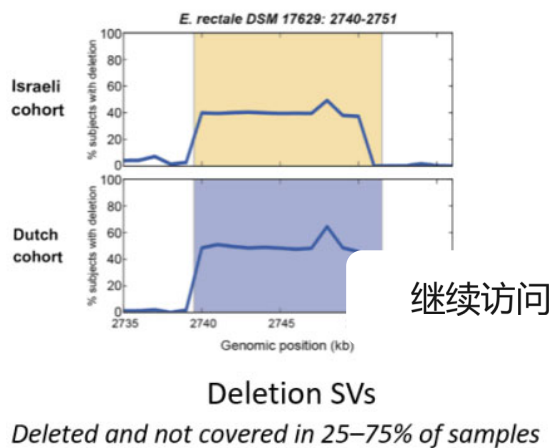
```
1 # python
2 import pandas as pd
3 dsgv_df = pd.read_pickle("results/dsgv.df")
4 dsgv_df.to_csv("results/dsgv.csv")
5 vsgv_df = pd.read_pickle("results/vsgv.df")
6 vsgv_df.to_csv("results/vsgv.csv")
```

### 6.2 结果内容

- results/dsgv.csv: 二分变量, 1/0值代表deletion SVs的存在/缺失状态;
- results/vsgv.csv: 连续变量, 代表variable SVs在标准化以后的覆盖度。

**注意:**

Deletion SV和variable SV只是根据数据特性进行的划分, 通过改变程序参数我们可以调整二者的判别标准 (SGVF\_cmd.py 的--dels\_detect\_thresh以及--real\_del\_thresh), 二者在生物学本质上没有区别。理论上我们可以得到所有高度可变片段的覆盖度, 也就是只有variable SV这一种形式。但是有些片段的缺失率太高, 导致数据过于稀疏, 因此处理成存在/缺失的值以后可以尽可能利用缺失该片段的样本的信息 (我猜想的开发者的思路)。



修改自原图 (Zeevi et al., 2019)

## 7 Troubleshooting

在这里，我将列举几个比较难解决的报错及解决方案，希望能对别的用户有所帮助。在这之前大家一定要把我前面提到过的bug修复了，这里就不再重复。

(1) `TypeError: 'double_precision' is an invalid keyword argument for this function`

这一步是由于ujson模块的`ujson.dumps()`函数的一个参数`double_precision`失效导致的。开发者使用的是ujson 1.35，而该模块在更新以后，`ujson.dumps()`函数的参数`double_precision`被移除了，因此调用该参数便产生了错误。如果前面按照我的说明严格安装了版本1.35，就不会出现这个问题了。严格来说这个应该也不能怨SGVFinder的开发者.....教训就是以后别的工具也尽量按照跟开发者一样的环境来配置。

修复办法当然就是安装ujson 1.35：

```
1 conda install ujson=1.35
```

(2) `OverflowError`

这一报错前的traceback如果是追溯到“`return 10** (gqa.calc_quality(srquals[0], ga1) + gqa.calc_quality(srquals[1], ga2)), off1, off2`”的话，那应该是与fastq文件的Phred碱基质量值的编码方式有关。尽管GEM mapper也支持Phred64，但SGVFinder应该只能直接处理Phred33的fastq文件。所以可以检查下碱基质量值编码方式：<https://www.jianshu.com/p/248308513e2e>。如果是Phred64的，可以找下工具转换成Phred33再进行尝试。

(3) `AssertionError`

我找不到之前的报错记录了，如果你在处理双端测序数据时遇到与GEMMapping.py的176行有关的AssertionError，可能与pair-end的两个mate的文件中，存在顺序不一样的成对mate。比如有个test\_1.fq中的reads abc001/1位于第100行，而它对应的在test\_2.fq中的abc001/2位于第200行，那么就可能会引起这个报错。

解决方案就是提前检查数据对的。我使用的工具是fastq-pair (<https://github.com/lin>)。继续访问 取消 添加这一步进行成对处理，因为这个报错跟具体样本有关，很可能部分样本报错了，而另一部分没有报，统一提前处理就可以避免重跑报错样本。

如果觉得文章对你有帮助的话，欢迎转发分享，帮忙点个“赞”和“在看”吧~

欢迎长按识别下方二维码持续关注~



微信搜一搜



非常明

## 8. 参考资料

- Zeevi D, Korem T, Godneva A, Bar N, Kurilshikov A, Lotan-Pompan M, Weinberger A, Fu J, Wijmenga C, Zhernakova A & Segal E (2019) Structural variation in the gut microbiome associates with host health. Nature 568, 43–48.
- ICRA and SGVFinder: <https://github.com/segalab/SGVFinder/>
- Exmple in python script: [https://github.com/segalab/SGVFinder/blob/master/src/linear\\_example.py](https://github.com/segalab/SGVFinder/blob/master/src/linear_example.py)
- Structural variation calling for metagenomic data: [https://github.com/GRONINGEN-MICROBIOME-CENTRE/Groningen-Microbiome/tree/master/Scripts/SV\\_pipeline](https://github.com/GRONINGEN-MICROBIOME-CENTRE/Groningen-Microbiome/tree/master/Scripts/SV_pipeline)



非常明

继续访问



取消

1 人喜欢



收录于合集 #生物信息 7

上一篇  
人类基因组SNP基因型数据集质控

下一篇  
可视化展示盘阵存储情况

阅读原文 阅读 355



非常明 关注

分享 收藏 3 9

写下你的留言

精选留言



高杰 广东 置顶  
太棒了！很细心

1



非常明(作者)  
哈哈多谢支持~



Cliff  
学习了

1



张强  
太牛逼了



非常明(作者)

没有没有，拾人牙慧

继续访问

取消