

期末文献阅读报告

《SpeedSeq: ultra-fast personal genome analysis and interpretation》

自 24 宋绍铭 2012011467

自 25 王猛 2012011492

1. 论文概述及选取背景

我们此次进行文献阅读的文献是来自 2015 年 10 月《Nature Methods》上的一篇名为《SpeedSeq: ultra-fast personal genome analysis and interpretation》的论文，这篇论文通过构建了一个名为 speedseq 的程序完整地实现了对基因进行测序获得原始文件之后，对于测序片段进行整合、校验并最终和参考数据库进行比对来找到其中的 SNV（单核苷酸多态性）以及 indels（insertion and deletion，碱基缺失或插入）部分，并最终形成 VCF 文件供给其他工具使用。

我们之所以选取这篇文章，是因为我们目前都在生物信息学部江瑞老师的实验室中进行课题研究，最近一段时间我们在尝试着利用多个工具来跑从原始数据到最后的成熟的 VCF 注释以及报告的生成这一流程，我们的大体流程如下：

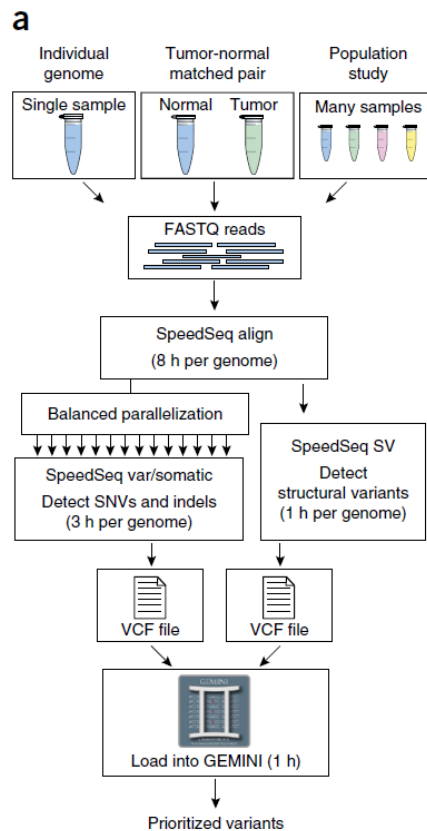
order		Flow	Program	Output Data Format	
1		Raw data		fastq	
2		Reads trimming	Trimmomatic	fastq	
3		Quality control	FastQC or Qualimap	fastq	
4		Reads mapping	BWA	sam	Speedseq Gemini
	Mapping Quality Control	convert to bam?	Samtools	bam	
7		Coverage	Qualimap	some report	
5		Sort	Samtools	bam	
6		Index	Samtools	bam	
8	Variant Calling	Indel Realignment	GATK	?	
9		Base recalibration	GATK	?	
10		many steps	GATK	VCF	
11		VCF Quality Control	GeneTalk	some report	
12		Variant annotation	AnnoVar	VCF	

可以看出，其中在 Reads Mapping、Mapping Quality Control 及 Variant Calling 部分，我们先后使用了 BWA，samtools、qualimap 以及 GATK 四种软件，除了要调整不同软件间的输入输出格式标准之外，所消耗的计算资源和时间十分巨大。而 Speedseq 结合 Gemini 可以在很短的时间内实现这一点。除此之外，speedseq 还能检查测序结果中存在的 structural variant (SV)，这是我们自己的工作流中所不能实现的。

因此在综述中，我们将首先介绍 speedseq 的原理及表现，并结合实际的操作流程对 speedseq 的使用进行简述。

2. Speedseq 原理

整个 Speedseq 的原理流程图如下：



关于这一流程的文字描述可以参见上面的那个表格，在这个流程过程中，speedseq 主要进行了一下几个工作：

1. FASTQ alignment and BAM processing
2. SNV and indel detection 或 Structural variation detection and genotyping

第 2 步当中，SNV/indel 和 SV 是两个不同的路线，因为前者多为几十 bp 长度以下的突变，而后者多为 1kb 到 3Mb 之间的大规模缺失或增加。

下面我将分别介绍其原理。

1. FASTQ alignment and BAM processing

这一步是将很多个 fastq 片段 mapping 到一起并且进行一些其他的处理。在这里，speedseq 首先使用了 BWA-MEM 软件（与我们流程中的软件相同）将一个 fastq 文件与 GRCh37 人类基因组数据库进行比对找到适配的位置，之后将匹配了的片段直接导入 SAMBLASTER 当中，这个过程发生在每一次 BWA 比对匹配的间隙。SAMBLASTER 可以对有 overlap 的部分进行检验，由目前的 30x 以上的测序深度可以找到该位置最有可能正确的数据。最后，利用 Sambamba 对 bam 文件进行快速排序并且压缩。

在这个流程中，由于 SAMBLASTER 和 BWA 交替调用，在充分利用计算资源的前提下也降低了存储量上的消耗。

2. SNV and indel detection

Speedseq 在进行 variant calling 的时候使用了一款名为 FreeBayes 的工具，该工具需要同时读取左右两个染色体相同位点的数据，以支持检测等位基因位置上的突变。对于体细

胞的变异，因为体细胞中有很多发生频率很低的变异，为了增加这些变异在识别中的敏感度，我们需要对参数进行调整，同时，对于每一个变异，我们都对其打分以评估其可靠性。

这个分数叫做 **somatic score (SSC)**，是两个参数:LODT 和 LODN 的和，这两个参数分别代表癌细胞和正常细胞的对数增长率。这里可以这样理解：对于不同的个体，这两种细胞的增长率都是不同的，但一点共性就是细胞的增长率越高，发生突变的概率越大，此处有突变的可信度就更高。之所以采用对数增长率是因为癌细胞和普通细胞具有天然的增长率差别，这样能够统一结果。具体的公式如下：

$$LOD_T = \log \frac{P_T(alternative)}{P_T(reference)}$$

$$LOD_N = \log \frac{P_N(alternative)}{P_N(reference)}$$

$$SSC = LOC_T + LOC_N$$

虽然它使用的是现成的工具，但是在加速方面，**speedseq** 进行了以下操作：首先经过测试，它将 **bam** 文件分为平均值为 **84kb** 一段的很多段，这是他们认为效率最高的分段长度。同时，他们还用 **GNU Parallel** 对这些片段进行并行计算。除此之外，通过计算 **CEPH 1463** 数据库中的平均测序深度作为阈值，对于一些测序深度过高的片段随机排除掉测序深度过高的部分以加速 **freebayes** 的计算速度。结果十分显著，与单线程相比，他们的速度是其 **13 倍**；与以染色体为单位的分段进行的多线程相比，他们的速度也要快上 **34.9%**。

3. Structural variation detection and genotyping

Speedseq 使用了名为 **LUMPY** 的工具进行 SV 的检测，对于末端配对 (**paired-end**) 以及间隔读取 (**split-read**) 设置的权重都是 **1**。它使用 **CNVnator** 工具按照基因所在的染色体的不同进行并行计算并且将输出的拷贝数分割按照 **100bp** 的长度进行展示。

在判断基因型的方面，因为对于等位基因而言，如果一个位置有 SV 的存在，在 SV 的断点处有两种情况，一种是不一致的片段 (**split-reads**) 另一种是一致的片段 (**read-pairs**)，通过观察基因上这两种片段的比率和数量我们可以判断这个基因的基因型是 **reference** (参考)，**heterozygous** (杂合) 还是 **homozygous** (纯合)。

算法如下，我们使用名为 **SVTyper** 的算法来判断 SV，我们定义函数 **S(g)** 为：

$$S(g) = \begin{cases} 0.1 & \text{if } g = \text{homozygous reference} \\ 0.4 & \text{if } g = \text{heterozygous} \\ 0.8 & \text{if } g = \text{homozygous alternate} \end{cases}$$

对每一个位点我们做一次判断，获得其 **S(g)** 对应的数值，**A** 为 **alternative** 的数量，**R** 为 **reference** 的数量 (二者根据 **mapping** 的质量进行过归一化)，由全概率公式和贝叶斯公式我们有：

$$P(A, R | g) = \binom{A + R}{A} \cdot S(g)^A \cdot (1 - S(g))^R$$

$$P(g|A,R) = \frac{P(A,R|g) \cdot P(g)}{P(A,R)} = \frac{P(A,R|g) \cdot P(g)}{\sum_{g \in G} P(A,R|g) \cdot P(g)}$$

$$\hat{g} = \arg \max_{g \in G} P(g|A,R)$$

这就是一个基本的贝叶斯分类器，我们可以判断出整个等位基因具体是杂合的还是纯合的。

3. Speedseq 的表现

1. 消耗时间

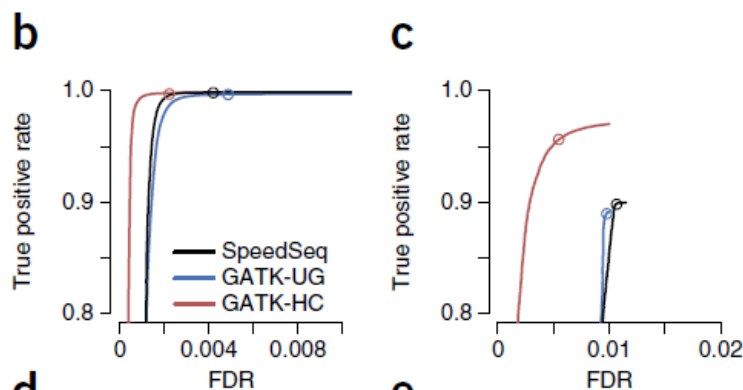
根据论文中的描述，他们使用的是 NA12878 genome from the Illumina Platinum Genomes data set，这是一组测序平均深度为 50x 的 WGS data (whole genome sequencing)，共有 17 组数据，采用第三代测序方法获得。所使用的硬件是双至强 E5-2670+128GB RAM。Speedseq 能够在低存储资源消耗的情况下，以 13h 的时间完成整套流程，与之相比较的则是一套 BWA-GATK-SAMTOOLS 的流程，比我们现在正在做的流程仅仅少一个 qualimap 部分。这套传统流程需要 60-70 小时的时间。经过比较，speedseq 在计算速度上的优势比较明显。

2. 准确率

关于准确率的评判，论文中给出了以下几种评判的方法：

2.1 SNV & indel calling

在检查 SNV 和 indel 的准确率的时候，采用 Genome in a Bottle Consortium (GIAB) 数据库中给到的 NA12878 数据库的学习结果作为正确的结果，SNV 的准确率达到 99.9%，indel 的准确率达到 89.9%。其 FDR (false discovery rates) 为 0.4% 和 1.1%。作为对比，使用 GATK-UG 的正确率分别为 99.7%，89.0%，两类错误率为 0.5% 和 1.0%，GATK-HC 的正确率为 99.8%，95.7%，两类错误率为 0.2% 和 0.6%。speedseq 的 ROC 曲线如下：



其中左侧为 SNV，右侧为 indels。

可以看出，speedseq 在运算速度大量增强的情况下，准确率上和目前主流的方法基本相同。

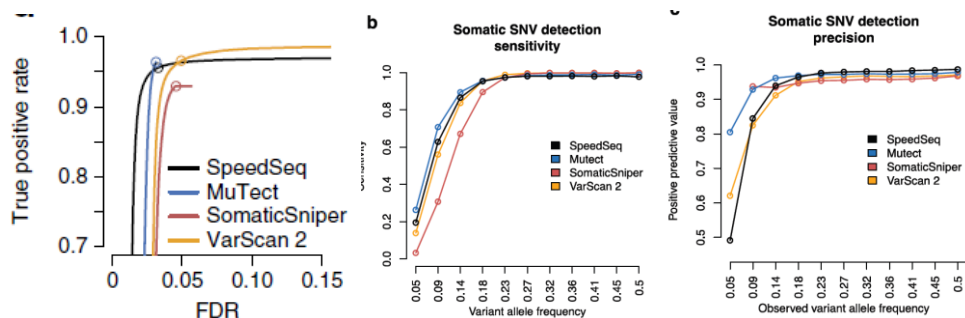
之后他们又对整个序列中不同功能的区域进行了分割校验，得到如下的结果：

	SNVs			Indels			SVs	
Feature	TP N=2,798,941	FP N=12,070	FN N=3,974	TP N=327,165	FP N=3,529	FN N=38,527	TP N=3,388	FP N=3,308
Long interspersed nuclear elements (LINE)	22.5%	18.7%	21.3%	23.6%	18.4%	14.5%	23.3%	20.6%
Short interspersed nuclear elements (SINE)	16.6%	19.6%	40.9%	21.4%	21.8%	34.3%	41.8%	24.5%
Long terminal repeat elements (LTR)	10.5%	16.8%	6.4%	7.1%	8.6%	5.5%	9.7%	11.5%
DNA repeat elements (DNA)	3.6%	3.2%	1.1%	3.7%	2.6%	2.1%	2.9%	3.2%
Satellite repeats	0.2%	5.8%	0.1%	0.1%	0.9%	0.1%	0.8%	1.5%
Simple repeats (micro-satellites)	-	-	-	-	-	-	10.6%	29.4%
Segmental duplications	-	-	-	-	-	-	4.5%	23.7%
Within non-unique 100-mer	7.5%	13.5%	43.7%	4.6%	4.8%	7.9%	50.0%	40.4%
Within 1 Mb of centromere	1.1%	1.7%	3.0%	1.0%	1.1%	1.4%	2.8%	5.4%
Within 1 Mb of telomere	0.5%	5.7%	0.3%	0.4%	1.5%	0.3%	1.1%	4.3%
Within 10 kb of assembly gap	0.1%	1.6%	0.3%	0.1%	0.3%	0.1%	0.1%	1.4%

可以看出在某些特定的区域 speedseq 的真阳性率、假阳性率和假阴性率都很高，这个也是将来需要主要改进的地方。

2.2 Cancer Genome Analysis(FAKE)

癌症细胞基因的检测也是基因检测中很重要的分支，因为这一系列的检测对于时间的敏感度很高。在这个测试中，我们定义 NA12878 中的 13 个样本是正常的，并且取 11 个小孩的样本作为假的癌症样本（因为儿童的细胞分裂速度比成年人要快），并对 NA12878 母系样本中存在而父系样本 NA12877 中不存在的 SNV 记作体细胞变异，非等位基因的频率（VAF）记为 0.05 至 0.5，并利用这个模型进行校验，和 MuTect, SomaticSniper, VarScan 2 这三个体细胞变异检测工具的结果进行比对，得到下面的图表：



可以看出，speedseq 的准确率和两类错误率都优于 SomaticSniper，和剩下两种工具比起来也不遑多让。

2.3 Cancer Genome Analysis-REAL)

这个测试中我们找到了 5 对已经确认了体细胞变异位置的正常-患癌序列数据（其中癌症序列测序深度为 50x，正常序列测序深度为 30x。变异位点是 The Cancer Genome Atlas (TCGA)提供的，得到如下的结果：

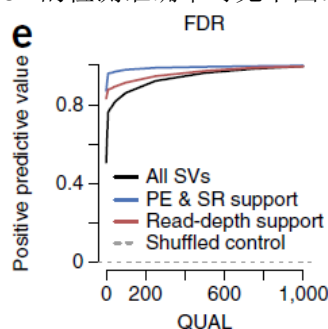
Sample	Type	Detected	Known	Sensitivity	COSMIC variants detected	COSMIC variants known	COSMIC variant sensitivity
TCGA-B6-A0I6	Breast	74	79	93.7%	2	2	100.0%
TCGA-A6-6141	Colorectal	485	510	95.1%	14	14	100.0%
TCGA-CA-6718	Colorectal	1,280	1,307	97.9%	44	44	100.0%
TCGA-D5-6540	Colorectal	779	819	95.1%	19	20	95.0%
TCGA-13-0751	Ovarian	30	31	96.8%	3	3	100.0%
Overall		2,648	2,746	96.4%	82	83	98.8%

可以看到效果很理想。

2.4 Ascertainment of structural variants

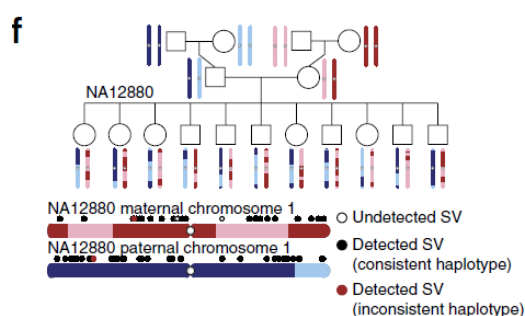
SV 的检测一直以来有两个难点：其一，SV 本身很难被可靠的检测到；其二，由于 SV 本身的长度很长，加之 SV 的断点位置往往定位并不严格精确，很难对于 SV 做出功能解释。这也是为什么很多像 speedseq 这样的流水线处理工具很少对 SV 进行检测（包括我们正在做的这个）。

因为上面的因素，目前学界中还没有足够可靠的 SV 检测结果的数据库，因此我们假设 1000 Genomes Project (1KGP) 中对于 NA12878 的 SV 结果就是正确的并进行对比。结果是阳性预测和删除检测的正确率都达到了 60% 以上，这也是目前各种工具中能够达到的最好的效果。如果我们将条件放宽一些，认为 PacBio and Illumina Molecule platforms 和 1KGP 中的结果都是正确的，我们可以得到 86% 的准确率。对于不同 quality 的 SV 的检测准确率可见下图：



2.5 Genotype

为了对基因型的判断做出检测，我们利用 SNV 的检查结果对于 NA12878 的样本进行了谱系的构建，得到这样的结果：



经过筛选，我们在 11 个孙辈的样本中找到了 8397 个可信度较高的 SV 位点以及这些位点上这些样本的基因型。对于这些 SV，speedseq 能够达到 90.2% 的准确率，并且其中仅存在于 1 人身上的独有的 SV，speedseq 也能检测到 1722 个中的 1660 个，具体的表格如下：

	Type	Correctly genotyped	Percent correctly genotyped	Detected	Detection sensitivity	Informative occurrences	Unique variants
All SVs	All	7,203	95.1%	7,578	90.2%	8,397	1,722
	Deletion	5,768	95.5%	6,042	90.8%	6,651	1,342
	Duplication	860	93.8%	917	89.5%	1,025	217
	Inversion	555	92.7%	599	85.9%	697	152
	Distant	20	100.0%	20	83.3%	24	11
Heterozygous SVs	All	6,845	96.6%	7,083	89.9%	7,883	1,505
	Deletion	5,421	96.1%	5,641	90.4%	6,240	1,173
	Duplication	853	97.9%	871	89.2%	976	193
	Inversion	551	100.0%	551	85.7%	643	128
	Distant	20	100.0%	20	83.3%	24	11
Homozygous SVs	All	358	72.3%	495	96.3%	514	217
	Deletion	347	86.5%	401	97.6%	411	169
	Duplication	7	15.2%	46	93.9%	49	24
	Inversion	4	8.3%	48	88.9%	54	24
	Distant	0	-	0	-	0	0

在纯合和杂合的判断方面，我们上面提到的 SVTypers 算法能够检测到 96.6%的杂合子和 72.3%的纯合子。Speedseq 中检测到的 SV 按照不同的类别分类如下：

	Number of SVs	Number of deletions	Number of deletions validated	Percent of deletions validated
≥ 7 support	8,456	5,540	4,369	78.9%
Monomorphic	2,525	1,356	1,151	84.9%
Polymorphic	5,931	4,184	3,218	76.9%
Mendelian transmission	5,509	3,853	3,047	79.1%
Mendelian violation	422	331	171	51.7%
Traceable from grandparent	1,722	1,342	1,059	78.9%

因为市面上的工具很少有能够做到这些分析的，因此 speedseq 得到的结果也就必然是目前我们能得到的最好的结果。

4. Speedseq 的使用

1. 安装

1.1 首先服务器上需要安装下面这些软件：

- g++ and the standard C and C++ development libraries (<https://gcc.gnu.org/>)
- CMake (<http://www.cmake.org/>)
- GNU awk and core utils
- Python 2.7 (<https://www.python.org/>)
 - numpy
 - pysam 0.8.0+
 - scipy
- ROOT (<https://root.cern.ch/>) (required if running CNVnator)
- Variant Effect Predictor (<http://www.ensembl.org/info/docs/tools/vep/index.html>)

(required if annotating VCF files)

1.2 然后运行下面三行代码即可快速完成安装和编译:

```
git clone --recursive https://github.com/hall-lab/speedseq
cd speedseq
make
```

如果编译不通过, 可能是 python 版本不正确或者某个软件没有安装。

2. 运行

Speedseq 分为下面 align, var, somatic, sv, realign 这五个彼此独立的模块, 下面我将分别介绍其功能、步骤和运行结果。

2.1 Speedseq align

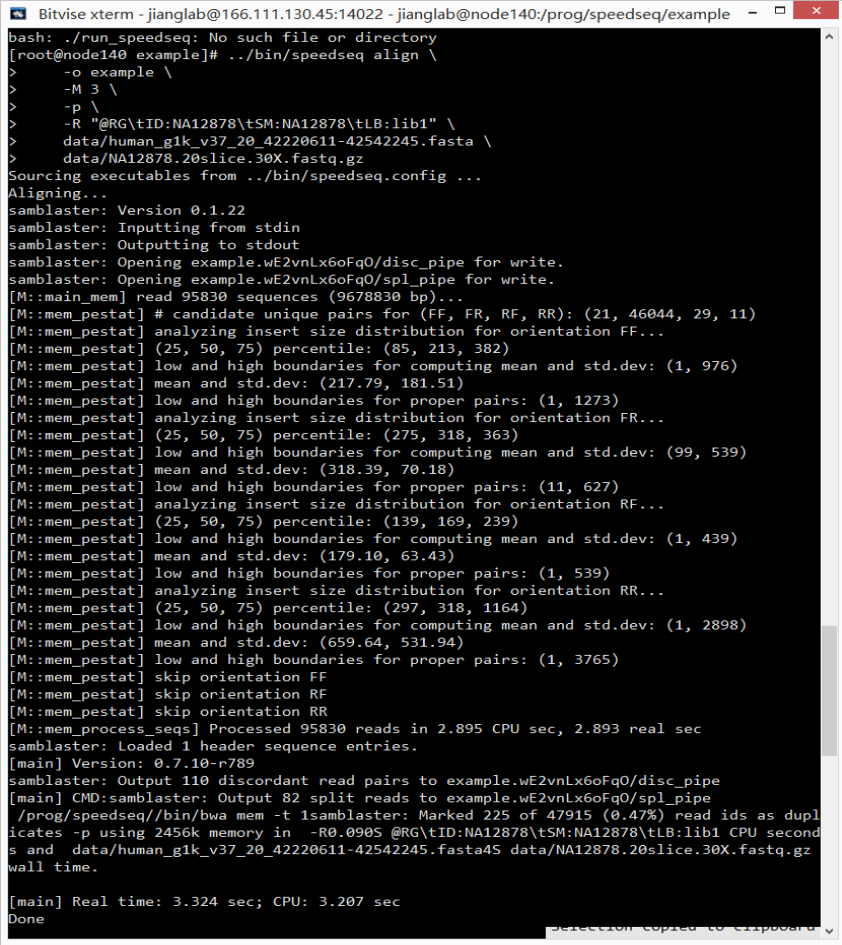
功能: 把 FASTQ 文件处理成 BAM 文件;

用法: `speedseq align [options] <reference.fa> <in1.fq> [in2.fq]`

主要参数: reference.fa 是对照文件, 这里采用的是 human_g1k_v37_20_42220611-42542245.fasta; in1.fq 和 in2.fq 是成对的 fastq 文件, 如果 in1.fq 是 interleaved 的, 那么使用参数 -p 可以只使用 in1.fq 一个输入文件。

输出: 三个 BAM 文件, outprefix.bam 是双重标记排序后的 BAM 文件, 可以做后面模块的输入文件; outprefix.splitters.bam 包含用 BWA-MEM call 的 reads, 默认排除了复制的 reads, 可以用作 speedseq sv 的输入文件; outprefix.discordants.bam 包含用 BWA-MEM call 的不协调的 reads, 可以做 speedseq sv 的输入文件。

运行结果截图:



```
Bitwise xterm - jianglab@166.111.130.45:14022 - jianglab@node140:/prog/speedseq/example
bash: ./run_speedseq: No such file or directory
[root@node140 example]# ../bin/speedseq align \
> -o example \
> -M 3 \
> -p \
> -R "@RG\tID:NA12878\tSM:NA12878\tLB:lib1" \
> data/human_g1k_v37_20_42220611-42542245.fasta \
> data/NA12878.20slice.30X.fastq.gz
Sourcing executables from ../bin/speedseq.config ...
Aligning...
samblaster: Version 0.1.22
samblaster: Inputting from stdin
samblaster: Outputting to stdout
samblaster: Opening example.wE2vnLx6oFq0/disc_pipe for write.
samblaster: Opening example.wE2vnLx6oFq0/spl_pipe for write.
[M::main_mem] read 95830 sequences (9678830 bp)...
[M::mem_pestat] # candidate unique pairs for (FF, FR, RF, RR): (21, 46044, 29, 11)
[M::mem_pestat] analyzing insert size distribution for orientation FF...
[M::mem_pestat] (25, 50, 75) percentile: (85, 213, 382)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 976)
[M::mem_pestat] mean and std.dev: (217.79, 181.51)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 1273)
[M::mem_pestat] analyzing insert size distribution for orientation FR...
[M::mem_pestat] (25, 50, 75) percentile: (275, 318, 363)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (99, 539)
[M::mem_pestat] mean and std.dev: (318.39, 70.18)
[M::mem_pestat] low and high boundaries for proper pairs: (11, 627)
[M::mem_pestat] analyzing insert size distribution for orientation RF...
[M::mem_pestat] (25, 50, 75) percentile: (139, 169, 239)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 439)
[M::mem_pestat] mean and std.dev: (179.10, 63.43)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 539)
[M::mem_pestat] analyzing insert size distribution for orientation RR...
[M::mem_pestat] (25, 50, 75) percentile: (297, 318, 1164)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 2898)
[M::mem_pestat] mean and std.dev: (659.64, 531.94)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 3765)
[M::mem_pestat] skip orientation FF
[M::mem_pestat] skip orientation RF
[M::mem_pestat] skip orientation RR
[M::mem_process_seqs] Processed 95830 reads in 2.895 CPU sec, 2.893 real sec
samblaster: Loaded 1 header sequence entries.
[main] Version: 0.7.10-r789
samblaster: Output 110 discordant read pairs to example.wE2vnLx6oFq0/disc_pipe
[main] CMD:samblaster: Output 82 split reads to example.wE2vnLx6oFq0/spl_pipe
/prog/speedseq/bin/bwa mem -t 1samblaster: Marked 225 of 47915 (0.47%) read ids as dupl
icates -p using 2456k memory in -R0.0905 @RG\tID:NA12878\tSM:NA12878\tLB:lib1 CPU second
s and data/human_g1k_v37_20_42220611-42542245.fasta45 data/NA12878.20slice.30X.fastq.gz
wall time.
[main] Real time: 3.324 sec; CPU: 3.207 sec
Done
```


2.2 Speedseq var

功能: call SNVs 和 indels

用法: `speedseq var [options] <reference.fa> <input1.bam> [input2.bam [...]]`

主要参数: reference.fa 是对照文件, 这里采用的是 human_g1k_v37_20_42220611-42542245.fasta; input.bam 是输入的 BAM 文件, 要求包含 readgroup 的信息而且 SM 的 readgroup 的标签必须是 VCF 列的标题。

输出: 用 VEP 标记的 VCF 文件, outprefix.vcf.gz

运行结果截图:

```
[root@node140 example]# # 2. Detect SNVs and indels
[root@node140 example]# ../bin/speedseq var \
> -o example \
> data/human_g1k_v37_20_42220611-42542245.fasta \
> example.bam
Sourcing executables from ../bin/speedseq.config ...
Calling variants...
Done
```

2.3 Speedseq somatic

功能: 在肿瘤和正常的 BAM 文件上运行 FreeBayes

用法: `speedseq somatic [options] <reference.fa> <normal.bam> <tumor.bam>`

主要参数: reference.fa 是对照文件, 这里采用的是 human_g1k_v37_20_42220611-42542245.fasta; normal.bam 是正常的 BAM 文件, 要求包含 readgroup 的信息而且 SM 的 readgroup 的标签必须是 VCF 列的标题; tumor.bam 是肿瘤的 BAM 文件, 要求和 normal.bam 相同。

输出: 用 VEP 标记的 VCF 文件, outprefix.vcf.gz

2.4 Speedseq sv






功能: 运行 LUMPY 去 call 结构性变异 (sv)

用法: `speedseq sv [options] <reference.fa> <full.bam> <split.bam><discordant>`

主要参数: reference.fa 是对照文件, 这里采用的是 human_g1k_v37_20_42220611-42542245.fasta; full.bam 是完整的 BAM 文件; split.bam 是分离的 reads 的 BAM 文件; discordant.bam 是不协调的 reads 的 BAM 文件。

输出: VCF 文件

2.5 全部的输出文件

 example.bam	2016/1/16 22:25	BAM 文件
 example.discordants.bam	2016/1/16 22:25	BAM 文件
 example.splitters.bam	2016/1/16 22:25	BAM 文件
 example.sv.vcf.gz	2016/1/16 22:25	WinRAR 压缩文件
 example.vcf.gz	2016/1/16 22:25	WinRAR 压缩文件

5. 总结和评价

总而言之，speedseq 是一款非常具有创新性的序列 mapping、分析、SNV&indel calling 以及 SV calling 的工具。它之所以能够发在《Nature Methods》这样的期刊上，在我看来主要有三个原因：

1. 在整合了一些已有工具的同时，它们通过算法、输入参数的优化已经硬件方面的优化，极大地降低了整个序列处理流程的时间的同时保持了可观的准确率，这个是它最大的亮点。
2. 和其他类似的 pipeline 工具相比，speedseq 能够对 SV 进行检测并可以对等位基因的杂合纯合做出比较准确的判断。这是它所特有的功能，虽然在具体一些部分的准确率上有很大的提升空间，但无论如何它做出来的结果都是 state-of-the-art 的。
3. 在验证准确率的过程中，尤其是检验 SV 的准确率的过程中，他们使用了很多具有原创性的交叉验证、分类验证的方法，包括最后基因型判断中的建立谱系等一系列方法经过标准化之后都具有成为标准测试流程的潜力。

以上就是我们关于这篇文章的一些理解，谢谢阅读。

6. Reference

1. Chiang C, Layer R M, Faust G G, et al. SpeedSeq: Ultra-fast personal genome analysis and interpretation[J]. bioRxiv, 2014: 012179.
2. <https://github.com/hall-lab/speedseq>.