Xenograft NGS データの解析

keywords: Xenograft WES RNA-seq Mutect2

はじめに

ヒトのがんを免疫不全マウスに移植(xenograft)して抗がん剤の薬効などを調べる実験が行われることがあります。この組織を抽出して次世代シーケンサー(NGS)を使って whole exome sequencing (WES)や whole transcriptome sequencing (RNA-seq)を行ったデータを解析する場合の手順をまとめました。

参照論文

Computational approach to discriminate human and mouse sequences in patient-derived tumour xenografts の記載内容をまとめると、

1. マウス由来 DNA read (WES)は 7%くらい bwa でヒトゲノムにマップされるが、ヒト+マウスゲノムにマップするとヒトにマップされるのは 0.1%くらいとなる。
2. 体細胞変異検出ではリファレンスゲノムにマウスゲノムを加えておかないと検出される変異数が激増する。
3. マウスの RNA-seq データは 5%くらい STAR でヒトゲノムにマップされるが、ヒト+マウスゲノムにマップするとヒトにマップされるのは 1%くらいとなる。
4. 発現量解析ではマウスゲノムを加えることにより 1500 遺伝子(ヒト)くらいは発現量が減少する。

となると思います。

使用するツール

本 NGS データ解析で使用するツールは下記になります。必要に応じてご用意ください。

* bwa
* GATK
* picard
* samtools
* bcftools
* tabix
* Manta
* STAR
* StringTie
* STAR-Fusion
* Arriba

WES データ解析

統合ゲノムデータの作成

がんの WES データの解析の主な目的の一つは体細胞変異を同定することです。ここでは GATK best practices の Somatic short variant discovery (SNVs + Indels)に準拠した解析を行います。

NGS データ解析におけるヒトゲノムデータは GRCh38 analysis set が用意されており、これをダウンロードして使うことも多いと思います。ただし今回は対象ががんであり、またがん部と非がん部のペアで call を行うので TCGA の各種ウイルスゲノムを含めて alternative haplotypes 等を除いたゲノム GRCh38.d1.vd1.fa.tar.gz を用います。ダウンロード後アーカイブを

```
tar xvfz GRCh38.d1.vd1.fa.tar.gz
```

と展開しておきます。 マウスゲノムは Ensembl の Mouse GRCm39 から Download DNA sequence をクリックして Mus_musculus.GRCm39.dna_sm.primary_assembly.fa.gz をダウンロードします(GENCODE から PRI をダウンロードしても OK です)。 ここで「dna_sm」はリピート領域が小文字で表現されているゲノム配列を表現しています。「dna_rm」はリピート領域が N でマスクされてしまっていますので、NGS の read をマッピングすると多数のミスマッピングが生じて偽陽性 call が発生するので使用を避けます。

GRCh38 と GRCm39 を cat コマンドで統合するだけなのですが、read マッピング後にマウス由来ゲノムであることが分かるように染色体番号の前に MusChr の文字列を追加しておきます。

```
gzip -dc Mus_musculus.GRCm39.dna_sm.primary_assembly.fa.gz | sed -e 's/>/>MusChr/' -e
's/ .*//' > mm39.fa
```

ゲノム配列を統合して各種インデックスファイルを作成しておきます(最後の 2 つは数時間程度かかることがあります)。

```
cat GRCh38.d1.vd1.fa mm39.fa > hsaMg.fa
samtools faidx hsaMg.fa
gatk CreateSequenceDictionary -R hsaMg.fa -O hsaMg.dict
gatk BwaMemIndexImageCreator -I hsaMg.fa -O hsaMg.img
mkdir BWA; cd BWA; ln -s ../hsaMg.fa .; bwa index -a bwtsw hsaMg.fa
```

Mutect2 実行時に参照するファイルをダウンロードします。 下記の somatic-hg38 のリソースをダウンロードします。

* af-only-gnomad.hg38.vcf.gz
* small_exac_common_3.hg38.vcf.gz
* 1000g_pon.hg38.vcf.gz

ただし他人のゲノムとのコンタミネーションの推定に使う SNPs リスト small_exac_common_3.hg38.vcf.gz は東北メディカル・メガバンク機構の ToMMo 60KJPN-SNV/INDEL Allele Frequency Panel データから日本人で頻度の高いリストを作成しても良いと思います(弊社 csDAI では日本人用の VCF ファイルを作成して使っています)。

Mutect2 の実行

これで Mutect2 を実行する準備が整いました。後は mutect2_wdl の手続きに従い、前述のヒトとマウスのゲノムを統合した参照配列にマッピングを行い somatic mutation call を行います。 まず、がん部の FASTQ ファイルを hsaMg.fa にマッピングし、BAM ファイルを作成します。

```
mkdir tmp
bwa mem -K 100000000 -t 12 -Y -R "@RG\tID:SampleT-WES:FlowCellIDL001\tSM:SampleT-
WES\tLB:SampleT-WES\tPL:Illumina" BWA/hsaMg.fa SampleT-WES_Index_L001_R1_001.fastq.gz
SampleT-WES_Index_L001_R2_001.fastq.gz 2> SampleT-WES.mapping.log | samtools view -b - >
SampleT-WES.bam
java -Djava.io.tmpdir=tmp -jar picard.jar MarkDuplicates INPUT=SampleT-WES.bam
OUTPUT=SampleT-WES.marked.bam METRICS_FILE=SampleT-WES.matrics VALIDATION_STRINGENCY=SILENT
ASSUME_SORT_ORDER=queryname CLEAR_DT=false ADD_PG_TAG_TO_READS=false
OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500 > SampleT-WES.MarkDuplicates.log 2>&1
java -Djava.io.tmpdir=tmp -jar picard.jar SortSam INPUT=SampleT-WES.marked.bam
OUTPUT=SampleT-WES.marked.sort.bam SORT_ORDER=coordinate CREATE_INDEX=true
MAX_RECORDS_IN_RAM=5000000 > SampleT-WES.SortSam.log 2>&1
```

FASTQ ファイル名 (上記では SampleT-WES_Index_L001_R1_001.fastq.gz と SampleT-WES_Index_L001_R2_001.fastq.gz)やサンプル ID (上記では SampleT-WES)、スレッド数 (上記では 12)は適宜環境に合わせて変更してください。 非がん部の FASTQ ファイル(これは Xenograft 組織由来ではないと思います)も同様に hsaMg.fa マップして BAM ファイルを作成しておきます。

```
bwa mem -K 100000000 -t 12 -Y -R "@RG\tID:SampleN-WES:FlowCellIDL001\tSM:SampleN-
WES\tLB:SampleN-WES\tPL:Illumina" BWA/hsaMg.fa SampleN-WES_Index_L001_R1_001.fastq.gz
SampleN-WES_Index_L001_R2_001.fastq.gz 2> SampleN-WES.mapping.log | samtools view -b - >
SampleN-WES.bam
java -Djava.io.tmpdir=tmp -jar picard.jar MarkDuplicates INPUT=SampleN-WES.bam
OUTPUT=SampleN-WES.marked.bam METRICS_FILE=SampleN-WES.matrics VALIDATION_STRINGENCY=SILENT
ASSUME_SORT_ORDER=queryname CLEAR_DT=false ADD_PG_TAG_TO_READS=false
OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500 > SampleN-WES.MarkDuplicates.log 2>&1
java -Djava.io.tmpdir=tmp -jar picard.jar SortSam INPUT=SampleN-WES.marked.bam
OUTPUT=SampleN-WES.marked.sort.bam SORT_ORDER=coordinate CREATE_INDEX=true
MAX_RECORDS_IN_RAM=5000000 > SampleN-WES.SortSam.log 2>&1
```

続いて Mutect2 で call し、各種フィルターを適用します。メモリー容量の指定やベイトファイル (bait.bed.gz: bgzip で圧縮し tabix で index を作成)は適宜変更してください。

```
gatk --java-options "-Xmx32g -Xms32g" Mutect2 --tmp-dir tmp -R hsaMg.fa -I SampleT-
WES.marked.sort.bam -I SampleN-WES.marked.sort.bam -normal SampleN-WES -pon
1000g_pon.hg38.vcf.gz --germline-resource af-only-gnomad.hg38.vcf.gz --af-of-alleles-not-
in-resource 0.00003125 -O SampleT-WES.raw.vcf.gz -L bait.bed.gz -bamout SampleT-
WES.mutect2.bam --f1r2-tar-gz SampleT-WES.f1r2.tar.gz > SampleT-WES.Mutect2.log 2>&1
gatk --java-options "-Xmx32g -Xms32g" GetPileupSummaries --tmp-dir tmp -R hsaMg.fa -I
SampleT-WES.marked.sort.bam -V small_exac_common_3.hg38.vcf.gz -O SampleT-
WES.tumor.pileups.tsv --interval-set-rule INTERSECTION -L bait.bed.gz -L
small_exac_common_3.hg38.vcf.gz > SampleT-WES.GetPileupSummaries.log 2>&1
gatk --java-options "-Xmx32g -Xms32g" GetPileupSummaries --tmp-dir tmp -R hsaMg.fa -I
SampleN-WES.marked.sort.bam -V small_exac_common_3.hg38.vcf.gz -O SampleN-
WES.tumor.pileups.tsv --interval-set-rule INTERSECTION -L bait.bed.gz -L
small_exac_common_3.hg38.vcf.gz > SampleN-WES.GetPileupSummaries.log 2>&1
gatk --java-options "-Xmx32g -Xms32g" LearnReadOrientationModel --tmp-dir tmp -I SampleT-
WES.f1r2.tar.gz -O SampleT-WES.artifact-priors.tar.gz > SampleT-
```

```
WES.LearnReadOrientationModel.log 2>&1
gatk --java-options "-Xmx32g -Xms32g" CalculateContamination --tmp-dir tmp -I SampleT-
WES.tumor.pileups.tsv -O SampleT-WES.contamination.table --tumor-segmentation SampleT-
WES.segments.table -matched SampleN-WES.tumor.pileups.tsv > SampleT-
WES.CalculateContamination.log 2>&1
gatk --java-options "-Xmx32g -Xms32g" FilterMutectCalls --tmp-dir tmp -R hsaMg.fa -V
SampleT-WES.raw.vcf.gz -O SampleT-WES.filter1st.vcf.gz --contamination-table SampleT-
WES.contamination.table --tumor-segmentation SampleT-WES.segments.table --ob-priors
SampleT-WES.artifact-priors.tar.gz -stats SampleT-WES.raw.vcf.gz.stats --filtering-stats
SampleT-WES.filter1st.stats > SampleT-WES.FilterMutectCalls.log 2>&1
gatk --java-options "-Xmx32g -Xms32g" FilterAlignmentArtifacts --tmp-dir tmp -R hsaMg.fa -V
SampleT-WES.filter1st.vcf.gz -I SampleT-WES.marked.sort.bam --bwa-mem-index-image hsaMg.img
-O SampleT-WES.mutect2.vcf.gz > SampleT-WES.FilterAlignmentArtifacts.log 2>&1
```

最終的な出力結果は SampleT-WES.mutect2.vcf.gz になります。 実際の解析パイプラインでは領域分割を行って実行結果をマージすることで並列化することができ、より高速に実行できます。

Manta の実行

Manta もリファレンスファイルに統合したゲノムファイルを指定すれば実行できます。

```
configManta.py --tumorBam SampleT-WES.marked.sort.bam --normalBam SampleN-
WES.marked.sort.bam --referenceFasta hsaMg.fa --callRegions bait.bed.gz --exome --runDir
Manta > SampleT-WES.configManta.log 2>&1
Manta/runWorkflow.py -j 8 > SampleT-WES.runWorkflow.log 2>&1
```

統合ゲノムデータの効果

実際の xenograft サンプルを解析したときの call 数は下記の様になりました。

Mutect2 の call 数

Sample	GRCh38	GRCh38+GRCm39
Sample1	13,155	170
Sample2	13,845	157

Manta の 150 bp 以上の indel call 数(indel)と structural variant call 数 (SV)

	indel	indel	SV	SV
Sample	GRCh38	GRCh38+GRCm39	GRCh38	GRCh38+GRCm39
Sample1	2,096	0	226	2
Sample2	2,133	2	214	0

Somatic mutation call においてはマウスゲノムを加える効果は絶大でした。

(参考)Base quality score recalibration (BQSR)

BQSR を実行する場合には GATK resource bundle の vcf ファイルのヘッダーを統合したゲノムのものに差し替えておく必要が有ります。これらのファイルのダウンロード元はいくつかありますが、下記では Google API の URL をリンクしておきました。

* Homo_sapiens_assembly38.dbsnp138.vcf
* 1000G_omni2.5.hg38.vcf.gz
* Mills_and_1000G_gold_standard.indels.hg38.vcf.gz
* Homo_sapiens_assembly38.known_indels.vcf.gz
* Axiom_Exome_Plus.genotypes.all_populations.poly.hg38.vcf.gz
* hapmap3.3.hg38.vcf.gz

ダウンロードしたら別のディレクトリー(ここでは Download)に置いておきます。続いて VCF ファイルのヘッダーを前述で作成した統合ゲノムデータのものに差し替えます。

```
for i in Download/*.vcf*
do
	echo $i
	OUT_FILE=`basename $i .gz`
	bcftools reheader --fai hsaMg.fa.fai -o $OUT_FILE $i
	tabix -p vcf $OUT_FILE
done
```

BQSR を実行する際にはヘッダーを差し替えた上記の VCF ファイルをご使用ください。

RNA-seq データ解析

STAR index の準備

STAR ではマニュアルにリファレンスゲノムは primary assembly を使うように記載があります。Analysis set には alternative haplotypes 等が含まれているため、これらを削る(decoy は残す)か primary assembly のみのゲノム配列をダウンロードします。ここでは Xenograft のため引き続き TCGA の GRCh38.d1.vd1.fa を使いますが、GENCODE の GRCh38.primary_assembly.genome.fa.gz を使用することもできます。 遺伝子のアノテーション情報はここでは TCGA のゲノムに合わせて gencode.v36.annotation.gtf.gz を下記では使いますが、より新しいバージョン gencode.v48.primary_assembly.annotation.gtf.gz 等を用いることもできます。

マウスの遺伝子アノテーション情報は Ensembl の Mouse GRCm39 から Download GTF or GFF3 をクリックして Mus_musculus.GRCm39.109.gtf.gz をダウンロードします(GENCODE のゲノムを使う場合は GENCODE の GTF ファイルを使用してください)。 ゲノムと同様に染色体番号の前に MusChr の文字列を追加しておきます。

```
gzip -dc gencode.v36.annotation.gtf.gz | grep -v ^# > gencode.v36.annotation.gtf
gzip -dc Mus_musculus.GRCm39.109.gtf.gz | grep -v ^# | sed -e "s/^/MusChr/" >
mm39.gencode.v109.gtf
cat gencode.v36.annotation.gtf mm39.gencode.v109.gtf > gencode.v36mm39.v109.gtf
```

続いて STAR のインデックスを作成します。sjdbOverhang 長は FASTQ ファイルの read 長から 1 bp 短くした長さを指定することが推奨されていますが、指定しなくても大体うまくいくそうです (default は 100)。ここでは read 長 100 bp 用のインデックスを作成します。STAR のインデックス作成は default ではメモリー32GB に最適化されているらしいのですが、ヒト+マウスゲノムではメモリー64GB のマシンでも落ちました。そのためメモリーが 64 GB 以下の計算機ではメモリー使用量を減らすパラメーターを指定してインデックスを作成します(この処理には数時間程度かかると思います)。

```
STAR --runMode genomeGenerate --genomeDir STAR.hsaMg --genomeFastaFiles hsaMg.fa --
sjdbGTFfile gencode.v36mm39.v109.gtf --sjdbOverhang 99 --genomeSAsparseD 3 --
genomeSAindexNbases 12 --runThreadN 8 > STAR.hsaMg.log 2>&1
```

このパラメーター指定での最大使用メモリー量は 37 GB 程度でした。

STAR の実行

上記で作成したインデックスを指定して STAR を実行し、BAM ファイルを作成します。

```
STAR --runThreadN 4 --genomeDir STAR.hsaMg --twopassMode Basic --outFileNamePrefix SampleT-
WTS. --outSAMattributes NH HI AS nM NM MD jM jI MC XS --outSAMattrIHstart 0 --
outSAMstrandField intronMotif --outMultimapperOrder Random --runRNGseed 777 --quantMode
TranscriptomeSAM GeneCounts --readFilesCommand gzip -dc --readFilesIn SampleT-
WTS_Index_L001_R1_001.fastq.gz SampleT-WTS_Index_L001_R2_001.fastq.gz > SampleT-
WTS.star2pass.log 2>&1
mkdir -p tmp
gatk --java-options -Djava.io.tmpdir=tmp AddOrReplaceReadGroups -I SampleT-
WTS.Aligned.out.sam -O SampleT-WTS.bam -SO coordinate -RGID FlowCellIDL005 -RGLB SampleT-
WTS -RGPL Illumina -RGPU NextSeq -RGSM SampleT-WTS -CREATE_INDEX true > SampleT-
WTS.AddOrReplaceReadGroups.log 2>&1
gatk --java-options -Djava.io.tmpdir=tmp MarkDuplicates -I SampleT-WTS.bam -O SampleT-
WTS.marked.bam --METRICS_FILE SampleT-WTS.metrics --VALIDATION_STRINGENCY SILENT --
ASSUME_SORTED true --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP 1024 --CREATE_INDEX true --CLEAR_DT
false --OPTICAL_DUPLICATE_PIXEL_DISTANCE 2500 > SampleT-WTS.MarkDuplicates.log 2>&1
```

STAR は--twopassMode を指定してもメモリー使用量は 30 GB 程度でした。

StringTie の実行

STAR でマッピングした BAM ファイルから StringTie で発現量推定を行うことができます。TPM 値の推定ではマウス RNA を含まない GTF ファイルを指定して実行することもできます。

```
stringtie -e -B -p 8 -G gencode.v36.annotation.gtf -o SampleT-WTS.gencode.v36.gtf -A
SampleT-WTS.gencode.v36.gene.tab SampleT-WTS.marked.bam
```

もちろん gencode.v36mm39.v109.gtf を指定することも出来ます。 規格化をヒト転写物のみで行うかマウス転写物も含めるかは目的によって使い分けることになると思います。

Fusion 解析

Fusion 解析では参照データもツールと共に提供されていること多く、この参照データをヒト+マウス統合ゲノムデータで作り直すことはやや手間が大きいと思います。そこでここではヒト+マウス統合ゲノムにマップした BAM ファイルからマウスゲノムにマップされた read を除いた FASTQ ファイルを作成して、STAR-Fusion と Arriba を実行してみます。 Paired-end reads の内、片方でもマウスゲノムにマップされた read を下記に様に除きます。

```
samtools view SampleT-WTS.bam | grep MusChr | cut -f 1 | sort | uniq > SampleT-
WTS.MusChr.reads
samtools view -h SampleT-WTS.marked.bam | grep -v -f SampleT-WTS.MusChr.reads | gatk --
java-options -Djava.io.tmpdir=tmp SamToFastq -I /dev/stdin -F SampleT-
WTS.withoutMusChr_R1_001.fq.gz -F2 SampleT-WTS.withoutMusChr_R2_001.fq.gz > SampleT-
WTS.SamToFastq.log 2>&1
```

ヒトの参照データは STAR-Fusion では GRCh38_gencode_v37_CTAT_lib_Mar012021.plug-n-play.tar.gz (もしくはより新しいバージョンの GRCh38_gencode_v44_CTAT_lib_Oct292023.plug-n-play.tar.gz)をダウンロードして

```
wget
https://data.broadinstitute.org/Trinity/CTAT_RESOURCE_LIB/GRCh38_gencode_v37_CTAT_lib_Mar01
2021.plug-n-play.tar.gz
tar xvfz GRCh38_gencode_v37_CTAT_lib_Mar012021.plug-n-play.tar.gz
```

と展開しておきます。一方 Arriba はインストールしたディレクトリーに移動して付属の download_references.sh スクリプトを使って、必要なファイルのダウンロードおよび STAR インデックス作成を

```
./download_references.sh hg38viral+GENCODE38
```

と作成します。これらのヒトの参照データに対して fusion 解析を実行します。

```
STAR-Fusion --genome_lib_dir GRCh38_gencode_v37_CTAT_lib_Mar012021.plug-n-
play/ctat_genome_lib_build_dir --left_fq SampleT-WTS.withoutMusChr_R1_001.fq.gz --right_fq
SampleT-WTS.withoutMusChr_R2_001.fq.gz --output_dir STAR-Fusion --CPU 8 --tmpdir tmp --
FusionInspector validate --examine_coding_effect --denovo_reconstruct > SampleT-WTS.STAR-
Fusion.log 2>&1
```

Arriba の database は Arriba のパッケージに含まれていますので適切にパスを指定して実行してください。

```
run_arriba.sh STAR_index_hg38viral_GENCODE38 GENCODE38.gtf.gz hg38viral.fa
database/blacklist_hg38_GRCh38_v2.3.0.tsv.gz
database/known_fusions_hg38_GRCh38_v2.3.0.tsv.gz
database/protein_domains_hg38_GRCh38_v2.3.0.gff3 8 SampleT-WTS.withoutMusChr_R1_001.fq.gz
SampleT-WTS.withoutMusChr_R2_001.fq.gz > SampleT-WTS.Arriba.log 2>&1
draw_fusions.R --fusions=fusions.tsv --alignments=Aligned.sortedByCoord.out.bam --
output=SampleT-WTS.Arriba.pdf --annotation=GENCODE38.gtf.gz --
cytobands=database/cytobands_hg38_GRCh38_v2.3.0.tsv --
proteinDomains=database/protein_domains_hg38_GRCh38_v2.3.0.gff3 > SampleT-
WTS.draw_fusions.log 2>&1
```

マウスゲノムにマップされた read の削除前後での融合遺伝子検出件数を下記に示しました。

Sample	STAR-Fusion-1.11.0	Arriba-v2.3.0
Sample1 手術検体	7	17
Sample1 Xenograft	19	59
Sample2 Xenograft	13	39

Sample1 手術検体 マウスゲノムマップ read 削除後	6	11
Sample1 Xenograft マウスゲノムマップ read 削除後	18	32
Sample2 Xenograft マウスゲノムマップ read 削除後	16	10

Sample1 はマウス移植前の手術検体があり、これにはマウス RNA の混入はありません。それでもマウスゲノムを含めたゲノムに STAR でマップしてマウスゲノムにマッピングされた read を除くと検出数が減っているので、この手続きでは偽陰性が生じる可能性があるのかもしれません。一方で Xenograft RNA において Arriba ではマウスゲノムにマッピングされた read を除くと検出される fusion 数がかなり抑えられており、偽陽性を削減する効果はありそうです。

まとめ

Xenograft 由来 NGS データの解析ではリファレンスゲノムにマウスゲノムを加えることで体細胞変異については劇的に偽陽性の数を減らすことができました。 RNA-seq データについては WES データ解析に比べると効果は限定的ですが、一定の効果はあると考えています。