# Avida DNA ターゲットシーケンス解析 テクニカルガイド

オープンソースソフトウェアパッケージを使用した Avida DNA シーケンスデータの解析

テクニカルガイド［2024 年 9 月版 和文］

Version B0 対応

For Research Use Only. Not for use in Diagnostic Procedure

## 通知



### 保証

本書に含まれる資料は「現状有姿」で提供され、将来の改版に際しては、予告なしに変更される可能性があります。 さらに、適用法で認められる最大限の範囲で、Agilent は、本書および本書に含まれる情報に関して、明示または黙示を問わず、商品性および特定目的への適合性の黙示の保証を含むがこれらに限定されない、全ての保証を否認します。Agilent は、本書または本書に含まれる情報の提供、使用またはパフォーマンスに関連するエラーまたは偶発的もしくは間接的な損害について責任を負わないものとします。Agilent とユーザーが、本書の内容と矛盾する保証条件を別個の契約書として結んでいる場合は、別個の契約書の保証条件が優先されます。

### 本プロトコルについて

プロトコルは予告なく変更になることがあります。プロトコルを日本語化するにあたり、作業時間が発生するため、日本語プロトコルは英語の最新バージョンに比べて、遅れが生じます。製品ご購入の際は、必ず英語版プロトコルの Version をお確かめの上、日本語版が古い場合は、使用プロトコルについて、弊社までお問い合わせいただきますようお願い申し上げます。

本日本語プロトコルは、英語版の
Avida DNA Targeted Sequencing Analysis Analysis of Avida DNA sequencing data using open-source software packages Version B0, September 2024 (G9409-90001) に対応しています。

本プロトコルに関するご質問やご意見などございましたら、下記のメールアドレスにご連絡ください。

email_japan@agilent.com

### 安全上の注意

**CAUTION**

CAUTION 表示は危険性を示します。正しく実行または遵守されなかった場合に、製品の損傷や重要なデータの損失につながる可能性のある操作手順や方法などを示しています。CAUTION 表示の個所は、その条件を完全に理解し満たすまで、その先に進まないでください。

**WARNING**

WARNING 表示は危険性を示します。操作手順への注意を喚起するもので、この表示を無視して誤った取扱いをすると、人が死亡または重傷を負う可能性が想定される内容を示しています。
WARNING 表示の個所は、条件を完全に理解し満たすまで、その先に進まないでください。

本資料では、Avida DNA 試薬で調製したライブラリから得られるイルミナ NGS シーケンスファイルの解析パイプラインについて説明します。

## 1. Avida 製品の概要

この章では、Avida 製品の概要と、Avida ターゲットシーケンス解析テクニカルガイドの目的について説明します。

## 2. Avida 解析パイプラインの手順

この章では、Avida DNA ターゲットシーケンシングデータの解析パイプラインのステップについて説明します。

## 3. シェルスクリプト例

この章では、第 2 章「2. Avida 解析パイプラインのステップ」に記載されている各ステップのシェルスクリプト例が含まれています。

# 1. Avida 製品の概要



この章では、Avida 製品の概要と、Avida ターゲットシーケンス解析テクニカルガイドの目的について説明します。

# Avida 試薬キットとパネルの概要

セルフリー DNA (cfDNA) サンプル中の腫瘍特異的な変異検出は、サンプル量の制限や腫瘍由来の DNA の存在量が相対的に少ないため、困難である場合があります。Avida ライブラリ調製およびターゲットエンリッチメントシステムは、これらの課題に対処するために設計されており、cfDNA (変異検出 : 1～100 ng、メチル化変化検出 : 3～100 ng) または 10～100 ng の断片化 DNA (gDNA) を用いて、体細胞変異およびメチル化された DNA を高感度に検出します。

Avida プラットフォームは、いくつかの技術革新によって実現されています :

- cfDNA 分子および断片化 gDNA をイルミナシーケンサーでシーケンス可能なインサートに効率的に変換する、高度に最適化されたライブラリ調製法
- トップ鎖とボトム鎖にタグを付け、エラー修正を可能にするデュアルユニーク分子バーコード（UMI: unique molecule identifiers）
- バイアスおよびアレル脱落のリスクを低減する PCR フリーのライブラリ調製法
- プローブの相乗的な結合によって DNA 分子の両鎖の 70～80%をキャプチャーするターゲットエンリッチメント

これらの特徴によって、0.1% の頻度までの低い変異や、低頻度の構造的変異を検出することができます（表 2 から表 4）。

Avida の製品ポートフォリオは、DNA 解析（Avida DNA 試薬キット）、メチル化解析（Avida Methyl 試薬キット）、およびひとつのサンプルからの DNA＋メチル化解析（Avida Duo Methyl 試薬キット）をサポートする 3 つのライブラリ調製キットオプションで構成されています。ライブラリ調製キットとともに、3 種類の DNA と 1 種類のメチル化特異的カタログパネルが用意されています。Avida DNA および Avida Duo ワークフローと互換性のある DNA パネルは、さまざまなアプリケーション向けに設計され、異なる遺伝子セットをカバーしています。26 kb の Avida DNA Focused Cancer Panel は、14 のがん遺伝子およびがん抑制遺伝子のホットスポット領域をカバーし、低アレル頻度の検出を低コストで行うのに適しています。より大規模な 345 kb の Avida DNA Expanded Cancer Panel には、100 を超える遺伝子が含まれ、そのうち 80 遺伝子はエクソンを完全にカバーしているため、変異とコピー数の両方を検出することができます。また、Expanded パネルには、転座を検出するための主要なイントロン領域も含まれています。2.7 Mb の Avida DNA Discovery Cancer Panel はゲノム・プロファイリング用途として 682 遺伝子の全エクソンおよび転座ホットスポットのイントロンをカバーしています。

# Avida ターゲットシーケンス解析テクニカルガイドについて

本テクニカルガイドでは、オープンソースソフトウェアパッケージを使用した Avida DNA ターゲットシーケンスデータの解析手順について説明します。Avida DNA および Avida Duo Methyl ワークフローのライブラリ調製およびターゲットキャプチャステップの詳細なプロトコルは、アジレントの Web サイトでご覧いただけます。
https://www.agilent.com/cs/library/usermanuals/public/G9409-90000.pdf
https://www.agilent.com/cs/library/usermanuals/public/G9439-90000.pdf

Avida DNA ワークフローの解析パイプラインには、イルミナシーケンサーによって得られる FASTQ 形式のペアエンドシーケンスデータファイルを用います。パイプラインは、UMI 解析、配列アライメント、データ QC メトリクスの作成、変異コールで構成され、1 本鎖または 2 本鎖（デュプレックス）コンセンサスリードの作成や、ポジションベースの重複除去を実行するオプションがあります。

このテクニカルガイドでは、これらの解析パイプラインの各ステップについて、詳細なステップバイステップのコマンドライン例を示しながら紹介します。これらの情報は、FASTQ 入力ファイルを使用して、ゲノムの変化、特に一塩基変異（SNV）、挿入および欠失（indel）、コピー数多型（CNV）、転座（TL）を検索する際の目安とすることを目的としたものです。

本書で説明するすべてのステップは、一般に公開されているオープンソースツールを利用しています。表 1 に、"3. シェルスクリプト例" のスクリプト例”で使用したオープンソースソフトウェアパッケージを示します。

# Avida 解析のためのオープンソースツール

本テクニカルガイドでは、オープンソースソフトウェアパッケージを使用した Avida DNA ターゲットシーケンスデータの解析手順について説明します。Avida DNA および Avida Duo Methyl ワークフローのライブラリ調製およびターゲットキャプチャステップの詳細なプロトコルは、アジレントの Web サイトでご覧いただけます。
https://www.agilent.com/cs/library/usermanuals/public/G9409-90000.pdf
https://www.agilent.com/cs/library/usermanuals/public/G9439-90000.pdf

Avida DNA ワークフローの解析パイプラインは、イルミナシーケンサーによって作成された FASTQ 形式のペアエンドシーケンスデータファイルから始まります。パイプラインは、UMI 解析、配列アライメント、データ QC メトリクスの作成、変異コールで構成され、1 本鎖または 2 本鎖（デュプレックス）コンセンサスリードの作成や、ポジションベースの重複除去を実行するオプションがあります。

このテクニカルガイドでは、これらの解析パイプラインの各ステップについて、詳細なステップバイステップのコマンドライン例を示しながら説明します。これらの情報は、FASTQ ファイルを使用して、ゲノムの変化、特に一塩基変異（SNV）、挿入および欠失（indel）、コピー数多型（CNV）、転座（TL）を検索する際の目安となることを目的としています。

本書で説明するすべてのステップは、一般に公開されているオープンソースツールを利用しています。表 1 に、"3. シェルスクリプト例"のスクリプト例で使用したオープンソースソフトウェアパッケージを示します。

表 1 オープンソースソフトウェアパッケージ*

| パッケージ名 | 本ガイドで使用したバージョン |
|---|---|
| BEDTools | 2.25.0 |
| BWA | 0.7.12-r1039 |
| fgbio | 1.5.2 |
| Java | 1.8.0_322 |
| Miniconda | 3 3.9.5 |
| Picard | 2.20.1 |
| SAMtools | 1.9 |
| SnpEff/SnpSift | 4.2 |
| VarDict | 1.5.0 |
| R | 3.5.1 |
| PureCN | 2.8.0 |
| bcftools 1.11 | bcftools 1.11 |
| GATK | 4.2.0.0 |
| GeneFuse | 0.6.0 |

*本ソフトウェアは、様々なオープンソースおよびサードパーティソフトウェアライセンスの下で利用可能なサードパーティソフトウェア（以下「サードパーティコンポーネント」と言います）を利用する場合があります。サードパーティコンポーネントに関連する条件は、ExDViewer Program Files ディレクトリ内の comprehensive-license-disclosure.txt に記載されています。サードパーティコンポーネントに関連する条件に含まれる保証の免責に加え、Agilent は、サードパーティコンポーネントに関して、Agilent 自身および著作権所有者、提供者、使用許諾者を代表して、以下の免責を行います ： 適用される法律により認められる最大限の範囲において、サードパーティコンポーネントは、著作権者、提供者、使用許諾者、および Agilent により「現状有姿のまま」提供され、

Agilent は、口頭または書面を問わず、いかなる種類の保証または責任も負いません。Agilent は、明示的、黙示的、または制定法、慣習、取引過程、もしくは取引慣行によって生じたものであるかを問わず、権利、商品性、特定目的への適合性、および非侵害の黙示的保証を含むものの、これには限定されない、いかなる種類の保証または表明も行いません。いかなる場合においても、著作権者、提供者、使用許諾者、または Agilent は、サードパーティコンポーネントの使用から何らかの形で生じる、直接的、間接的、偶発的、付随的、罰則的、または派生的な損害（代替品またはサービスの調達、使用、データ、または利益の損失、または事業の中断を含むが、これらに限定されない）について、契約、厳格責任、または不法行為（過失またはその他を含む）であるか否かにかかわらず、いかなる原因、いかなる法的根拠においても責任を負いません。

# 変異解析結果の例

表 2 から表 4 のデータは、本テクニカルガイドに記載されているパイプラインを用いて得られた変異解析結果をまとめたものです。データ例に使用したシーケンスライブラリは、SeraCare ctDNA Reference Material および Avida DNA カタログパネルを用いて調製しました。SeraCare ctDNA Reference Material V4 ; 0.5% アレル頻度 (AF) を SeraCare ctDNA Reference Material (LGC Clinical Diagnostics からアジレントに提供されたベータ製品) ; 0% AF で滴定し、さまざまな変異アレル頻度 (variant allele frequency : VAF) のサンプルを調製しました。これらのテストでは、DNA インプット量として 20 ng および 40 ng、キャプチャパネルとして Avida DNA Expanded Cancer Panel (「Expanded パネル」) および Avida DNA Focused Cancer Panel (「Focused パネル」) をそれぞれ用いました。解析指標は、用いたパネルがカバーする変異に基づいて算出されています。

表 2 SeraCare ctDNA Reference Material を用いた複数の VAF レベルおよび複数のインプット量における SNV/indel 解析性能

| Avida DNA Panel | Sequencing depth | Specificity | Detection rate | | | | |
|---|---|---|---|---|---|---|---|
| | | | 0.1% VAF 20 ng input | 0.1% VAF 40 ng input | 0.2% VAF 20 ng input | 0.2% VAF 40 ng input | 0.5% VAF 20 ng input |
| Focused panel | 10 M/20 M PE | 96% | 85% | 100% | 100% | 100% | 100% |
| | 5 M PE | 100% | 72% | 93% | 100% | 100% | 100% |
| Expanded panel | 80 M/160 M PE | 93% | NA | 92% | 94% | 98% | 100% |
| | 80 M PE | 100% | NA | 86% | 92% | 96% | 100% |

表 3 SeraCare ctDNA Reference Material を用いた 0.5% VAF および 20 ng インプットでの CNV 解析性能 : 遺伝子レベルの CNV は PureCN で検出

| Avida DNA Panel | Sequencing depth | Detection rate | Detection rate Positive Predictive Value (PPV) |
|---|---|---|---|
| Expanded panel | 40 M/80 M PE | 100% | 100% |

表 4 SeraCare ctDNA Reference Material を用いた複数の VAF レベルおよび複数のインプット量における転座検出性能

| Avida DNA Panel | Sequencing depth | Detection rate | | | |
|---|---|---|---|---|---|
| | | 0.1% VAF 20 ng input | 0.2% VAF 20 ng input | 0.2% VAF 40 ng input | 0.5% VAF 20 ng input |
| Expanded panel | 80 M/160 M PE | 71% | 71% | 86% | 93% |

# 2. Avida 解析パイプラインの手順



この章では、Avida DNA ターゲットシーケンスデータの解析パイプラインの手順について説明します。

# Avida DNA ターゲットシーケンス解析パイプラインの概要

Avida DNA 解析パイプラインの一連の手順を図 1 に示します。このテクニカルガイドで使用されている手順構成は、コンセプトとロジックがシンプルになるように設計されています。実際のパイプラインでは、ディスク I/O と処理時間を最小化するためにプロセスを最適化することが望ましい場合、これらのステップのいくつかをシェルコマンドラインのパイプ（"|"）で接続することができます。それらの具体例は fgbio ガイドにあります：
github.com/fulcrumgenomics/fgbio/blob/main/docs/best-practice-consensus-pipeline.md.

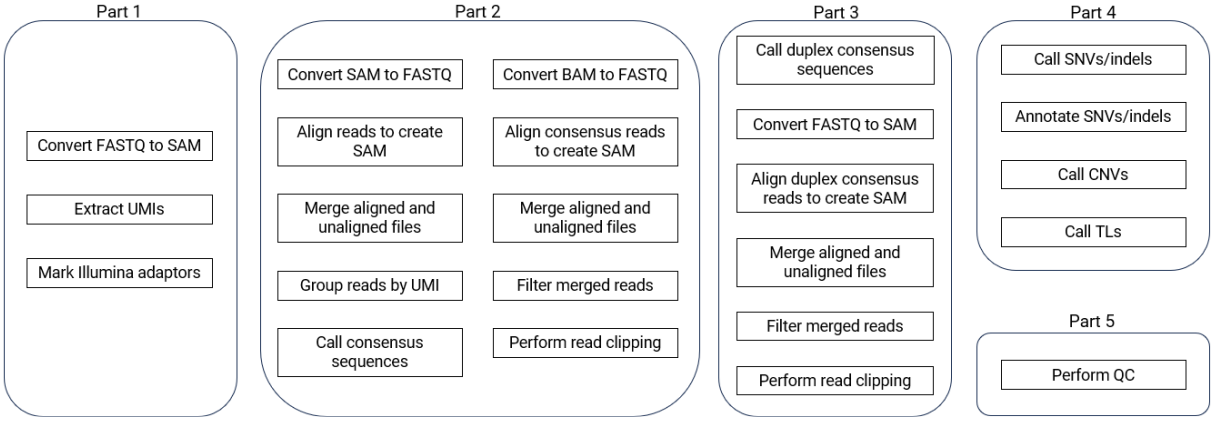


図 1 Avida DNA 解析パイプライン

図 1 に示した手順について、次項で詳しく説明します。手順をブロック（「パート」）にグループ分けすることで、概念的な明確化を図っています。

# 解析パイプラインの手順

## Part 1. リードの前処理（ステップ1~3）

パート 1 での解析パイプラインのインプットファイルは、ひとつの DNA ライブラリサンプルのペアエンド FASTQ ファイル（R1.fq と R2.fq）です。最初に、Picard ツールを使用して FASTQ ファイルを SAM ファイル形式に変換します。次に、 SAM ファイルをインプットファイルとし、fgbio コンセンサスパイプラインを用いて、リードの両端（R1 と R2 の両方の 5'末端）からインライン UMI 配列を抽出してトリミングし、SAM ファイルの RX タグに含めます。最後に、Picard を使用して、標準的なイルミナアダプターをマークします。

1. FASTQ ファイルを BAM ファイルに変換します（Picard）。
2. BAM ファイルから UMI 配列を抽出します（fgbio）。
3. BAM ファイル内のイルミナアダプターをマークします（Picard）。

## Part 2. 1本鎖 UMI の処理とアライメント（ステップ4~13）

パート 2 では、パート 1 で作成した SAM ファイルから、トリミングした R1 および R2 リードを含むひとつの FASTQ ファイルに変換します。次に、BWA でリードをアライメントして BAM ファイルを作成し、その BAM ファイルを fgbio で 1 本鎖 UMI コンセンサスマッチングに使用します。次に、UMI コンセンサスリードを含む BAM ファイルを FASTQ ファイルに変換し、その FASTQ ファイルから、SAM ファイルを作成します。

得られた SAM ファイルを用いて、fgbio で指定された品質メトリクスに基づき UMI 重複除去リードをフィルタリングします、次いで R1 および R2 リードの間でオーバーラップする塩基をクリッピングして、ペアのうち片方のリードの塩基コールのみ残します。

4. BAM ファイルを FASTQ ファイルに変換します（Picard）。
5. FASTQ ファイルのリードをアライメントして、SAM ファイルを作成します（BWA）。
6. 未アラインメント BAM ファイルとアライメント BAM ファイルのデータをマージします（Picard）。
7. BAM ファイル内のリードを UMI でグループ化します（fgbio）。
8. UMI でグループ化されたリードの各セットに対して、コンセンサス配列をコールします（fgbio）。
9. BAM ファイルを FASTQ ファイルに変換します（Picard）。
10. FASTQ ファイル内の UMI コンセンサスリードをアライメントして SAM ファイルを作成します（BWA）。
11. 未アライメント BAM ファイルとアライメント BAM ファイルのデータをマージします（Picard）。
12. BAM ファイル内のマージされたリードを、品質メトリクスに基づいてフィルタリングします（fgbio）。
13. BAM ファイルでリードクリッピングを行い、R1 および R2 リードの重複塩基を除去します（fgbio）。

## Part 3. 2本鎖UMI（デュプレックス）の処理とアライメント（ステップ14~19）

パイプラインのパート 3 はオプションです。これは、2 本鎖 UMI ベース（"デュプレックス"）で重複除去を行いたいユーザーを対象としています。1 本鎖 UMI ベースの重複除去のみを必要とするユーザーは、パート 4 に進んでください。パート 3 の手順は、パート 2 のステップ 8 から 13 と似ています。

**14.** 2 本鎖コンセンサス配列をコールします（fgbio）。

**15.** BAM ファイルを FASTQ ファイルに変換します（Picard）。

**16.** UMI コンセンサスリードをアラインメントして SAM ファイルを作成します（BWA）。

**17.** 未アライメントおよびアライメント BAM ファイルのデータをマージします（Picard）。

**18.** BAM ファイル内のマージされたリードを、品質メトリクスに基づいてフィルタリングします（fgbio）。

**19.** BAM ファイルでリードクリッピングを行い、R1 および R2 リードの重複塩基を除去します（fgbio）。

## Part 4. 変異解析（ステップ20~23）

パート 4 は、サンプル中の変異を検出するためのものです。このパートでは、DNA パネルやアッセイに適したステップを選択します。変異コールは、以下の出力ファイル 1) パート 2 で作成した 1 本鎖 UMI ベースの重複除去を行った BAM ファイル、2) パート 3 で作成した 2 本鎖 UMI ベースの重複除去を行った BAM ファイル、3) ポジショナル重複除去を用いて作成した BAM ファイル のいずれかの重複除去したリードに対して実行できます。ポジショナル重複除去はこのテクニカルガイドでは扱いませんが、実行するためのリソースは容易に入手可能です。必要に応じ、これらの異なる重複除去方法による変異コール結果を組み合わせて、ハイブリッドモードの変異解析を行うことが可能です。なお、次章で示すシェルスクリプトの例では、1 本鎖 UMI ベースの重複除去で作成された出力ファイルを使用しています。

**20.** SNV および indel 変異をコールします（VarDict）。

**21.** SNV および indel のコールに対し、アノテーション付けします（snpSift、snpEff）。

**22.** CNV コールします（PureCN）。

**23.** TL コールします（GeneFuse）。

## Part 5. アライメントメトリクスとQC（ステップ24）

ユーザーで定義した pass/fail 基準と比較するために、アライメントとサンプル品質に関する QC メトリクスを計算します。QC 解析では、パート 1～4 の出力ファイルを使用します。

**24.** アライメントとカバレッジの QC を実行します（BEDTools、Picard、SAMtools）。

# 3. シェルスクリプト例



この章では、第 2 章「2. Avida 解析パイプラインの手順」に記載されている各ステップのシェルスクリプト例が含まれています。

各パートおよびステップのコマンドライン例が記載されています。例は以下のフォルダ構造に基づいています。ご使用のコンピューティングノードが異なるフォルダ構造の場合は、それに応じて掲載例を修正してください。

- "/localdata" - 参照ゲノム、ソフトウェアパッケージ、入力 FASTQ ファイル、出力ファイルが保存されているローカルストレージマウント
- "/localdata/references" - FASTQ フォーマットでインデックス化されたリファレンスゲノムが格納されているディレクトリ
- /localdata/tools" – SAMtools や BEDTools などのソフトウェアパッケージがインストールされているディレクトリ
- sample_dn" - サンプルの入力 FASTQ ファイルと出力結果が格納されるディレクトリ
  例えば sample_dn="/localdata/analyses/run1/sample1"

# Part 1. リードの前処理（ステップ1~3）

**1.** FASTQ ファイルを BAM ファイルに変換します（Picard）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar picard.jar FastqToSam \
  FASTQ=$sample_dn/R1.fq FASTQ2=$sample_dn/R2.fq \
  OUTPUT=$sample_dn/raw_unmapped.bam \
  READ_GROUP_NAME=$sid SAMPLE_NAME=$sid \
  LIBRARY_NAME=pe PLATFORM=illumina PLATFORM_UNIT=HiSeq \
  MAX_RECORDS_IN_RAM=1000000
```

**2.** BAM ファイルから UMI 配列を抽出します（fgbio）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar ExtractUmisFromBam \
  --input=$sample_dn/raw_unmapped.bam \
  --output=$sample_dn/raw_unmapped_umi.bam \
  --read-structure=3M2S146T 3M2S146T --molecular-index-tags=ZA ZB --single-tag=RX
```

Note：ステップ 2 のスクリプトは 151 bp のシーケンス長用に作成されていますが、より短いシーケンス長でも使うことができます。

**3.** BAM ファイル内のイルミナアダプターをマークします（Picard）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar picard.jar MarkIlluminaAdapters \
  INPUT=$sample_dn/raw_unmapped_umi.bam \
  OUTPUT=$sample_dn/raw_unmapped_markedAdpt.bam \
  METRICS=$sample_dn/readstats/adapter_metrics.txt MAX_RECORDS_IN_RAM=1000000
```

# Part 2. 1本鎖 UMI の処理とアライメント（ステップ4~13）

**4.** BAM ファイルを FASTQ ファイルに変換します（Picard）。

```
java -Xmx16G -jar picard.jar SamToFastq \
  INPUT=$sample_dn/raw_unmapped_markedAdpt.bam \
  FASTQ=$sample_dn/raw_unmapped_markedAdpt.fq \
  MAX_RECORDS_IN_RAM=1000000 CLIPPING_MIN_LENGTH=36 \
  INTERLEAVE=true INCLUDE_NON_PF_READS=true \
  CLIPPING_ATTRIBUTE=XT CLIPPING_ACTION=X
```

**5.** FASTQ ファイルのリードをアライメントして、SAM ファイルを作成します（BWA）。

```
bwa mem -p -t 10 \
  /localdata/references/hg19/Sequence/bwaIndex/genome.fa \
  $sample_dn/raw_unmapped_markedAdpt.fq > $sample_dn/raw_aligned.sam
```

**6.** 未アラインメント BAM ファイルとアライメント BAM ファイルのデータをマージします（Picard）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar picard.jar MergeBamAlignment \
  UNMAPPED=$sample_dn/raw_unmapped_markedAdpt.bam \
  ALIGNED=$sample_dn/raw_aligned.sam \
  OUTPUT=$sample_dn/picard_fixed.bam \
  REFERENCE_SEQUENCE=/localdata/references/hg19/Sequence/genome.fa \
  CLIP_ADAPTERS=false VALIDATION_STRINGENCY=SILENT \
  CREATE_INDEX=true EXPECTED_ORIENTATIONS=FR MAX_GAPS=-1 \
  SORT_ORDER=coordinate ALIGNER_PROPER_PAIR_FLAGS=false \
  MAX_RECORDS_IN_RAM=1000000
```

**7.** BAM ファイル内のリードを UMI でグループ化します（fgbio）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar GroupReadsByUmi \
  --input=$sample_dn/picard_fixed.bam \
  --output=$sample_dn/grouped_umi.bam \
  --strategy=paired --raw-tag=RX --assign-tag=MI --min-map-q=10 --edits=1
```

**8.** UMI でグループ化されたリードの各セットに対して、コンセンサス配列をコールします（fgbio）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar CallMolecularConsensusReads \
  --input=$sample_dn/grouped_umi.bam \
  --output=$sample_dn/ss_consensus_unmapped.bam \
  --rejects=$sample_dn/ss_rejects.bam \
  --error-rate-pre-umi=45 --error-rate-post-umi=40 \
  --min-input-base-quality=10 --min-reads=1
```

**9.** BAM ファイルを FASTQ ファイルに変換します（Picard）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar picard.jar SamToFastq \
 INPUT=$sample_dn/ss_consensus_unmapped.bam \
 FASTQ=$sample_dn/ss_consensus_unmapped.fastq \
 INTERLEAVE=true INCLUDE_NON_PF_READS=true MAX_RECORDS_IN_RAM=1000000
```

**10.** FASTQ ファイル内の UMI コンセンサスリードをアライメントして SAM ファイルを作成します（BWA）。

```
/localdata/tools/bwa.kit/bwa mem -p -t 10 \
 /localdata/references/hg19/Sequence/bwaIndex/genome.fa \
 $sample_dn/ss_consensus_unmapped.fastq \
 > $sample_dn/ss_consensus_mapped.sam
```

**11.** 未アライメント BAM ファイルとアライメント BAM ファイルのデータをマージします（Picard）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar picard.jar MergeBamAlignment \
 UNMAPPED=$sample_dn/ss_consensus_unmapped.bam \
 ALIGNED=$sample_dn/ss_consensus_mapped.sam \
 OUTPUT=$sample_dn/ss_consensus_mapped.bam \
 CLIP_ADAPTERS=false VALIDATION_STRINGENCY=SILENT \
 CREATE_INDEX=true ORIENTATIONS=FR MAX_GAPS=-1 \
 SORT_ORDER=coordinate ALIGNER_PROPER_PAIR_FLAGS=false \
 ATTRIBUTES_TO_RETAIN=X ATTRIBUTES_TO_RETAIN=ZS \
 ATTRIBUTES_TO_RETAIN=ZI ATTRIBUTES_TO_RETAIN=ZM \
 ATTRIBUTES_TO_RETAIN=ZC ATTRIBUTES_TO_RETAIN=ZN \
 ATTRIBUTES_TO_REVERSE=cd ATTRIBUTES_TO_REVERSE=ce \
 REFERENCE_SEQUENCE=/localdata/references/hg19/Sequence/genome.fa \
 MAX_RECORDS_IN_RAM=1000000
```

**12.** BAM ファイル内のマージされたリードを、品質メトリクスに基づいてフィルタリングします（fgbio）。

```
 java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar FilterConsensusReads \
 --input=$sample_dn/ss_consensus_mapped.bam \
 --output=$sample_dn/ss_consensus_filtered.bam \
 --ref=/localdata/references/hg19/Sequence/genome.fa \
 --max-read-error-rate=0.05 --max-base-error-rate=0.1 \
 --min-base-quality=10 --max-no-call-fraction=0.2 --min-reads=3
```

**13.** BAM ファイルでリードクリッピングを行い、R1 および R2 リードの重複塩基を除去します（fgbio）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar ClipBam \
 --input=$sample_dn/ss_consensus_filtered.bam \
 --output=$sample_dn/ss_consensus_filtered_clipped.bam \
 --ref=/localdata/references/hg19/Sequence/genome.fa \
 --soft-clip=false --clip-overlapping-reads=true
```

# Part 3. 2本鎖 UMI （デュプレックス）の処理とアライメント（ステップ14~19）

**14.** 2 本鎖コンセンサス配列をコールします（fgbio）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar CallDuplexConsensusReads \
  --input=$sample_dn/grouped_umi.bam \
  --output=$sample_dn/ds_consensus_unmapped.bam \
  --error-rate-pre-umi=45 --error-rate-post-umi=40 --min-input-base-quality=10
```

**15.** BAM ファイルを FASTQ ファイルに変換します（Picard）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar picard.jar SamToFastq \
  INPUT=$sample_dn/ds_consensus_unmapped.bam \
  FASTQ=$sample_dn/ds_consensus_unmapped.fastq \
  INTERLEAVE=true INCLUDE_NON_PF_READS=true \
  MAX_RECORDS_IN_RAM=1000000
```

**16.** UMI コンセンサスリードをアラインメントして SAM ファイルを作成します（BWA）。

```
bwa mem -p -t 10 /localdata/references/hg19/Sequence/bwaIndex/genome.fa \
  $sample_dn/ds_consensus_unmapped.fastq > $sample_dn/ds_consensus_mapped.sam
```

**17.** 未アライメントおよびアライメント BAM ファイルのデータをマージします（Picard）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar picard.jar MergeBamAlignment \
  UNMAPPED=$sample_dn/ds_consensus_unmapped.bam \
  ALIGNED=$sample_dn/ds_consensus_mapped.sam \
  OUTPUT=$sample_dn/ds_consensus_mapped.bam \
  CLIP_ADAPTERS=false VALIDATION_STRINGENCY=SILENT \
  CREATE_INDEX=true ORIENTATIONS=FR MAX_GAPS=-1 \
  SORT_ORDER=coordinate ALIGNER_PROPER_PAIR_FLAGS=false \
  ATTRIBUTES_TO_RETAIN=X0 ATTRIBUTES_TO_RETAIN=ZS \
  ATTRIBUTES_TO_RETAIN=ZI ATTRIBUTES_TO_RETAIN=ZM \
  ATTRIBUTES_TO_RETAIN=ZC ATTRIBUTES_TO_RETAIN=ZN \
  ATTRIBUTES_TO_REVERSE=cd ATTRIBUTES_TO_REVERSE=ce \
  REFERENCE_SEQUENCE=/localdata/references/hg19/Sequence/genome.fa \
  MAX_RECORDS_IN_RAM=1000000
```

**18.** BAM ファイル内のマージされたリードを、品質メトリクスに基づいてフィルタリングします（fgbio）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar FilterConsensusReads \
  --input=$sample_dn/ds_consensus_mapped.bam \
  --output=$sample_dn/ds_consensus_filtered.bam \
  --ref=/localdata/references/hg19/Sequence/genome.fa \
  --max-read-error-rate=0.2 --max-base-error-rate=0.4 \
  --min-base-quality=30 --max-no-call-fraction=0.4 --min-reads=2 1 1
```

**19.** BAM ファイルでリードクリッピングを行い、R1 および R2 リードの重複塩基を除去します（fgbio）。

```
java -Djava.io.tmpdir=$sample_dn -Xmx16G -jar fgbio.jar ClipBam \
  --input=$sample_dn/ds_consensus_filtered.bam \
  --output=$sample_dn/ds_consensus_filtered_clipped.bam \
  --ref=/localdata/references/hg19/Sequence/genome.fa \
  --soft-clip=false --clip-overlapping-reads=true
```

# Part 4. 変異解析（ステップ20~23）

以下の出力ファイルは、変異解析に適したファイルです： 1) パート 2 で作成した BAM ファイル（1 本鎖 UMI ベースの重複除去を実施）、2) パート 3 で作成された BAM ファイル（2 本鎖 UMI ベースの重複除去を実施）、3) ポジショナル重複除去を実施して作成した BAM ファイル（本テクニカルガイドでは対象外）。これらの異なる重複除去方法による変異コール結果を組み合わせて、ハイブリッドモードの変異解析を行うことができます。以下に示す例は、1) の方法で作成された出力ファイルを使用しています。SNV および indel コールにアノテーション付けを行い、遺伝子記号、アミノ酸変化、transcript ID、cDNA 変化、および機能的変化（タンパク質コード、イントロンなど）を追加し、さらに解析することができます。アノテーション付けした変異は、アノテーションに基づいてさらにフィルタリングしたり、dbSNP、ExAC、COSMIC などのデータベースと比較したりすることができます。ここでは、アノテーション付けされた変異のフィルタリングのためのスクリプト例は含まれていません。

**20.** SNV および indel 変異をコールします（VarDict）。

```
VarDict -G /localdata/references/hg19/Sequence/genome.fa \
 -N $sid -b $sample_dn/ss_consensus_filtered_clipped.bam \
 -f 0.001 -c 1 -S 2 -E 3 -g 4 -r 3 -F 0x700 -th 12 \
 $sample_dn/devel401_designed_regions.bed | \
 /localdata/tools/VarDict/teststrandbias.R | \
 /localdata/tools/VarDict/var2vcf_valid.pl -N $sid -E -f 0.001 \
 > $sample_dn/vd.vcf
```

**21.** SNV および indel のコールに対し、アノテーション付けします（snpSift、snpEff）。

```
java -Xmx16G -jar /localdata/tools/snpEff/SnpSift.jar annotate \
 -dbsnp $sample_dn/vd_roi.vcf > $sample_dn/snpsift.vd.vcf

java -Xmx16G -jar /localdata/tools/snpEff/snpEff.jar \
 -config /localdata/tools/snpEff/snpEff.config \
 -noStats -noLog -nodownload hg19 \
 $sample_dn/snpsift.vd.vcf > $sample_dn/snpeff.vd.vcf
```

**22.** CNV コールします（PureCN）。

一本鎖 UMI ベースの重複除去（パート 2）で作成された BAM ファイルと、Agilent SureDesign から入手できる covered regions file (Covered.bed) または target regions file (Regions.bed) を使用します。ロバストな CNV コールのためには、リファレンスのレプリケートが必要です。以下のスクリプト例では、Regions.bed ファイルを使用しています。また、以下の例は 1 つのサンプルと 1 つのリファレンス（Sample1.bam、Normal1.bam、Normal2.bam）に基づいています。スクリプト例を含むベストプラクティスのガイドラインは、bioconductor.org で入手できます。アジレントは、IntervalFile.R. に以下の最小限の変更を加えて、これらのガイドラインをテストしました。

- off-target オプションは、ノイズを最小限にするために省略しています。
- average-target-width オプションと -min-target-width オプションは、それぞれ 100 と 50 に設定しています。

```
# Prepare reference files. Could be done once per design and set of normal samples
Rscript /localdata/tools/IntervalFile.R \
  --in-file /localdata/Regions.bed \
  --fasta /localdata/hg19.fa \
  --out-file /localdata/baits_hg19_intervals.txt \
  --genome hg19 \
  --mappability /localdata/wgEncodeCrgMapabilityAlign100mer.bigWig \
  --average-target-width 100 \
  --min-target-width 50

# Prepare a file "normals.list" to include Normal1.bam and Normal2.bam
echo "/localdata/Normal1.bam" > /localdata/normals.list
echo "/localdata/Normal2.bam" >> /localdata/normals.list

# Calculate GC-normalized coverage
mkdir /localdata/normals

Rscript /localdata/tools/Coverage.R \
  --bam /localdata/normals.list \
  --out-dir /localdata/normals \
  --intervals /localdata/baits_hg19_intervals.txt \
  --cores 4
```

ステップ 22 のスクリプトは次のページに続きます。

```
# Do variant calling for normal files
gatk Mutect2 \
  -R /localdata/hg19.fa \
  -I /localdata/Normal1.bam \
  -O /localdata/Normal1_mutect2.vcf.gz \
  --intervals /localdata/Regions.bed \
  --genotype-germline-sites true \
  --genotype-pon-sites true \
  --germline-resource /localdata/af-only-gnomad.vcf.gz \
  --interval-padding 200 \
  --native-pair-hmm-threads 8

gatk Mutect2 \
  -R /localdata/hg19.fa \
  -I /localdata/Normal2.bam \
  -O /localdata/Normal2_mutect2.vcf.gz \
  --intervals /localdata/Regions.bed \
  --genotype-germline-sites true \
  --genotype-pon-sites true \
  --germline-resource /localdata/af-only-gnomad.vcf.gz \
  --interval-padding 200 \
  --native-pair-hmm-threads 8

gatk FilterMutectCalls \
  -V /localdata/Normal1_mutect2.vcf.gz \
  -R /localdata/hg19.fa \
  -stats /localdata/Normal1_mutect2.vcf.gz.stats \
  -filtering-stats /localdata/Normal1_mutect2.filtered.stats \
  -O /localdata/Normal1_mutect2.filtered.vcf.gz \
  -f-score-beta 0.1

gatk FilterMutectCalls \
  -V /localdata/Normal2_mutect2.vcf.gz \
  -R /localdata/hg19.fa \
  -stats /localdata/Normal2_mutect2.vcf.gz.stats \
  -filtering-stats /localdata/Normal2_mutect2.filtered.stats \
  -O /localdata/Normal2_mutect2.filtered.vcf.gz \
  -f-score-beta 0.1
```

ステップ 22 のスクリプトは次のページに続きます。

### 3. シェルスクリプト例

```
# Merge normal samples VCF files
ls Normal*.filtered.vcf.gz > normal_vcfs.list

bcftools merge \
  -l normal_vcfs.list \
  -Oz \
  -o /localdata/panel_of_normals.vcf.gz \
  --force-samples

bcftools index \
  --tbi \
  /localdata/panel_of_normals.vcf.gz

# Build a normal database
ls -a /localdata/normals/*_loess.txt.gz | cat > /localdata/normal_coverages.list

Rscript /localdata/tools/NormalDB.R \
  --out-dir /localdata/ \
  --coverage-files normal_coverages.list \
  --normal-panel /localdata/panel_of_normals.vcf.gz \
  --genome hg19 \
  --assay avida_expanded

# Process sample
$SAMPLEID="Sample1"
mkdir /localdata/$SAMPLEID

# Calculate GC-normalized coverage
Rscript /localdata/tools/Coverage.R \
  --out-dir /localdata/${SAMPLEID} \
  --bam /localdata/${SAMPLEID}.bam \
  --intervals /localdata/baits_hg19_intervals.txt

# Generate a VCF for each sample using Mutect2
gatk Mutect2 \
  -R /localdata/hg19.fa \
  -I /localdata/${SAMPLEID}.bam \
  -O /localdata/${SAMPLEID}_mutect2.vcf.gz \
  --intervals /localdata/Regions.bed \
  --genotype-germline-sites true \
  --genotype-pon-sites true \
  --germline-resource /localdata/af-only-gnomad.vcf.gz \
  --interval-padding 200 \
  --native-pair-hmm-threads 8
```

ステップ 22 のスクリプトは次のページに続きます。

```
gatk FilterMutectCalls \
  -V /localdata/${SAMPLEID}_mutect2.vcf.gz \
  -R /localdata/hg19.fa \
  -stats /localdata/${SAMPLEID}_mutect2.vcf.gz.stats \
  -filtering-stats /localdata/${SAMPLEID}_mutect2.filtered.stats \
  -O /localdata/${SAMPLEID}_mutect2.filtered.vcf.gz \
  -f-score-beta 0.1

# Run PureCN to normalize, segment and determine purity and ploidy
Rscript /localdata/tools/PureCN.R \
  --out /localdata/${SAMPLEID} \
  --tumor $OUT/${SAMPLEID}/${SAMPLEID}_coverage_loess.txt.gz \
  --sampleid ${SAMPLEID} \
  --vcf /localdata/${SAMPLEID}_mutect2.filtered.vcf.gz \
  --normaldb /localdata/normalDB_avida_expanded_hg19.rds \
  --mapping-bias-file /localdata/mapping_bias_avida_expanded_hg19.rds \
  --intervals /localdata/baits_hg19_intervals.txt \
  --genome hg19 \
  --interval-padding 200 \
  --post-optimize
```

**23.** TL コールします（GeneFuse）。

```
# cut adapter
  cutadapt -m 30 -q 15 \
  -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC \
  -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
  -o $sample_dn/r1_cutadapt.fq -p $sample_dn/r2_cutadapt.fq \
  $sample_dn/R1.fq $sample_dn/R2.fq > $sample_dn/readstats/cutadapt.log

# call translocation
  genefuse -u 1 -t 30 \
  -r /localdata/references/hg19/Sequence/genome.fa \
  -f /localdata/tools/genefuse-0.6.1/genes/cancer.hg19.csv \
  -1 $sample_dn/r1_cutadapt.fq. -2 $sample_dn/r2_cutadapt.fq \
  -h $sample_dn/genefuse_report.html
```

# Part 5. アライメントメトリクスと QC （ステップ24）

アライメントとサンプル QC のメトリクスを作成・集計し、事前に設定したサンプル QC の pass/fail 基準と比較します。

**24.** アライメントとカバレッジの QC を実行します（BEDTools、Picard、SAMtools）。

**a.** 染色体長ファイルを作成します。

```
# chrom lengths
samtools view -H $sample_dn/picard_fixed.bam \
 | grep @SQ | sed 's/@SQ\tSN:\|LN://g' | cut -f1,2 > $sample_dn/genome_length.tsv
```

**b.** パネル内の全塩基のユニークカバレッジを集計します。未加工の BAM ファイルやデュプレックスで重複除去した BAM ファイルでも同様に行うことができます。

```
bedtools coverage -sorted -a [panel_target_regions].bed \
 -b $sample_dn/ss_consensus_mapped.bam \
 -d -g $sample_dn/genome_length.tsv > $sample_dn/ss_base.cover.tsv
```

**c.** SAMtools flagstat を実行します。

```
samtools flagstat $sample_dn/picard_fixed.bam > $sample_dn/flagstats.txt
```

**d.** インサート長を集計します。

```
java -Xmx16G -jar picard.jar CollectInsertSizeMetrics \
 INPUT=$sample_dn/ss_consensus_mapped.bam \
 OUTPUT=$sample_dn/insert_metrics.tsv \
 HISTOGRAM_FILE=$sample_dn/insert_histo.pdf \
```

G240680

ゲノミクス関連製品に関するお問い合わせ

Tel: 0120 - 477 - 111

Mail: email_japan@agilent.com

電話・メール受付時間　土、日、祝祭日、5/1 を除く

9：00～12：00、13：00～17：00

※プロトコル名とともに、テクニカルな質問と明示してください。

※価格、納期等のご質問は担当営業にご連絡ください。