Japanese Journal of Lactic Acid Bacteria

解　説

次世代シーケンサーデータの解析手法 第6回 ゲノムアセンブリ

谷澤 靖洋[1,2]、神沼 英里[2*]、中村 保一[2]、清水 謙多郎[3]、門田 幸二[3*]

[1] 東京大学大学院新領域創成科学研究科
[2] 国立遺伝学研究所生命情報研究センター
[3] 東京大学大学院農学生命科学研究科

ゲノムの *de novo* アセンブリ結果に影響を及ぼす主要なパラメータは、*k*-mer（任意の長さ *k* の連続塩基）の *k* 値である。第6回は、Bio-Linux にプレインストールされているゲノムアセンブリ用プログラム Velvet の基本的な利用法、make コマンドを用いたプログラムのインストール法、およびウェブツール DDBJ pipeline の利用法について述べる。複数の異なる *k*-mer で実行した乳酸菌ゲノム *de novo* アセンブリ結果の違い、用いたプログラム間の違い（Velvet vs. Platanus）などを述べる。また、ウェブサイト（R で）塩基配列解析（URL: http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html）中に本連載をまとめた項目（URL: http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#about_book_JSLAB）が存在する。ウェブ資料（以下、W）や関連ウェブサイトなどのリンク先を効率的に活用してほしい。

Key words：*de novo* assembly, quality control, install, DDBJ Pipeline

はじめに

2014年7月の連載第1回から一年以上経過し、各種プログラムもバージョンアップされている。これらに対応すべく、2015年11-12月にかけて第2回の仮想環境構築以降の情報を全般的に更新した。Bio-Linux 8[1)]のインストール手順は、2通り存在する［**W1-1**］。1つは拡張子が .iso となっているファイルからスタートするやり方である。煩雑だがインストールする PC の事情に合わせて一から構築できるため、拡張性が高いというメリットがある。もう1つは、拡張子が .ova となっているファイルからスタートするやり方である。既存の解析環境をそのまま導入することと同義であるため、手軽に同じ解析環境（ゲスト OS 環境）を別の PC に移植することができる。例えば、iso ファイルからのインストール手順に従って構築した「HDD 100GB、ユーザ名 iu、壁紙が白」などの PC 環境そのものを1つのファイルに保存したものが、拡張子 ova のファイルである［**W1-2**］。この程度なら別の PC 上で新規構築してもそれほど手間はかからないと思われるかもしれない。しかし、例えば「連載第5回までででインストールおよびダウンロードした各種プログラムおよびファイルが存在する PC 環境」を、別の PC で①一から再構築する労力と②その ova ファイルを導入する労力を比較するといいだろう。著者らの感覚では、後者以外の選択肢はありえない。

連載第3回終了時点以降の ova ファイルには、共有フォルダ設定情報も含まれている。読者自身が作成した ova ファイルを読者の別の PC に導入（インポート）するであれば、基本的に共有フォルダはそのまま利用できるだろう。しかし、例えば C ドライブの概念があるか（Windows）ないか（Macintosh）の OS の違いや、同一 OS でも PC 間で異なるユーザ名にしている場合には、ova ファイル導入後に共有フォルダ設定情報の一部を手動で変更する必要がある。これらは実害を被ってはじめて理解できる場合が多いが、「共有フォルダ設定情報を含む ova ファイルからのインストール手順」を概観し、記憶に留めておくといいだろう［**W1-3**］。

*To whom correspondence should be addressed.
Phone ： +81-3-5841-2395
Fax ： +81-3-5841-1136
E-mail ： ekaminum@nig.ac.jp (for DDBJ Pipeline)
E-mail ： kadota@bi.a.u-tokyo.ac.jp (for the others)

第6回は、Bio-Linuxにプレインストールされているゲノム *de novo* アセンブリ用プログラムVelvet[2]の基本的な利用法を述べる。次に、makeコマンドを用いたVelvetプログラムのインストールについて述べ、インストール時にオプションを追加することで、指定可能な*k*-merの*k*値の範囲を広げるテクニックを紹介する。また、ウェブツールDDBJ Pipeline[3]の利用法を紹介し、ユーザ登録からアセンブリの実行まで幅広く述べる。乳酸菌（*Lactobacillus hokkaidonensis* LOOC260^T）ゲノム配列決定論文[4]のデータを用い、*k*値やプログラム（Velvet vs. Platanus[5]）によるアセンブリ結果の違いについても述べる。Bio-Linux上での作業は主に「~/Documents」以下で行い、配列のクオリティチェック用プログラムFastQC[6]（ver. 0.11.3以上）、およびアダプター除去やクオリティフィルタリングを行うFaQCs[7]（ver. 1.34）が利用可能という前提で話を進める［**W1-4**］。

連載第5回終了時点のovaファイル（約6.4 GB）も提供しているため［**W1-3**］、WindowsユーザもMacintoshユーザも平等に第6回以降を独立して実習可能である。つまり、例えば第4回でFastQCまたはFaQCsのいずれかのインストールに失敗していた一部読者も、［**W1-3**］の手順通りにovaファイルを導入すれば、**インストール失敗がなかったことになる**。実習を諦めていたヒトも、ぜひ気持ちを新たに再チャレンジしてほしい。

NGSデータ取得とクオリティチェック

第5回[8]でも触れたが、第6回で用いる乳酸菌（*L. hokkaidonensis* LOOC260^T）ゲノム配列決定論文[4]の生データは、2種類のNGS機器由来データからなる。1つは平均4 kbpのPacBioデータ（DRR024500; 163,376リード; DRR024500は本稿執筆時に登録内容の誤りがあったことが判明し、DRR054113-054116に差し替えになっている）、そしてもう1つは250 bpのpaired-end Illumina MiSeqデータ（DRR024501; 2×2,971,310リード）である。原著論文[4]および公共データベース（以下、公共DB）のDDBJ SRA（以下、DRA）[9]やENA[10]を眺めることで、各種IDの対応関係、ファイルサイズ、そしてwgetコマンドを用いたダウンロード時に必要なファイルのURL情報を得ることができる［**W2**］。

本稿では、MiSeqデータ（DRR024501）の一部（最初の30万リード）を解析する。paired-endの場合は、forward側（DRR024501_1.fastq.bz2）とreverse側（DRR024501_2.fastq.bz2）の2つのファイルに分割されており、それぞれ約300万リード、合計約1GBになることがわかる［**W2-5**］。一旦全リードデータのダウンロードを行うやり方［**W3-1**］では、ネットワーク環境的に厳しい読者もいると思われる。ダウンロードの段階から最初の30万リードのみに限定することもできるので、各自の事情に合わせて行ってほしい［**W3-2**］。知らなければ想像もできないテクニックだと思われるが、こういうこともできるという経験を積むことは重要である。2015年12月に第3回ウェブ資料軽量版［**W22-2**］でも紹介しているように、オリジナルが約15GBにもなる乳酸菌RNA-seqデータでも、指定したリード数からなるサブセットのファイルサイズに準じた時間でダウンロード可能である。

30万リードからなるgzip圧縮FASTQファイル（DRR024501sub_1.fastq.gzとDRR024501sub_2.fastq.gz）の各々に対して、FastQC（ver. 0.11.4）を用いたクオリティチェックを行う［**W4-1**］。FastQC実行結果を眺めることで、原著論文中では配列長が250 bpと書いているが実際には251 bpになっていることの確証を得ることができる［**W2-3 と W4-2**］。話の本筋からそれるので深入りはしないが、気になる読者は、「MiSeq 250 251 実際」などでウェブ検索すればよい。第2回[11]でも紹介したNGSのQ & Aサイトである SEQanswers[12]にたどり着き、疑問が氷解するであろう。比較的配列長の長いMiSeqデータにおけるFastQC実行時の注意点は、--nogroupオプションの有無による結果の違いを適切に把握することであろう。第5回［**W19**］でも述べたが、FastQCをデフォルト（--nogroupオプション無）で実行すると、QCレポートファイルの一定の横幅に収まるように10番目以降の塩基が元の配列長に応じてグループ化される。このデータ（251 bp）の場合、例えばクオリティスコア分布は、5塩基分を平均化した結果となっている点に注意してほしい。理想的には、--nogroupオプションをつけた結果も出力して両者を眺めたほうがよい。

図1は、FastQC結果の中から特に着目してもらいたい2つの項目（Per base sequence qualityとAdapter Content）をまとめたものである。このデータは、典型的なMiSeqの特徴を持つ。1つは、リードの終端に近づくほどクオリティスコアが下がる点である。これは一般的なシーケンサーの特徴そのものではあるが、配列長が100 bp程度のIllumina HiSeqシリーズのNGS機器で得られた乳酸菌RNA-seqデータのクオリティスコア分布（第4回W8-3や第5回W15-4）と比較すると、全般的に悪い印象をもつであろう。しかしよく見ると、MiSeqデータも最初の150 bp付近までは、HiSeqと同程度のクオリティスコア（目安として30以上）分布になっていることがわかる。クオリティスコアが右肩下がりの傾向のままで250 bp付近まで読み進めるために、終端付近のスコアが悪く見えるのである。判断基準の目安としては、きれいなデータ例がforward側のスコア分布［**W4-2**］であり、そうでない例がreverse側の分布［**W4-3**］である。信号機の色をイメージしてもらえばよいが、FastQCレポート中の項目ごとに青黄赤で色分けして、立ち止まって眺めるべき項目を赤色で、注意を払うべきと思われる項目を黄色で示していることがわかる。

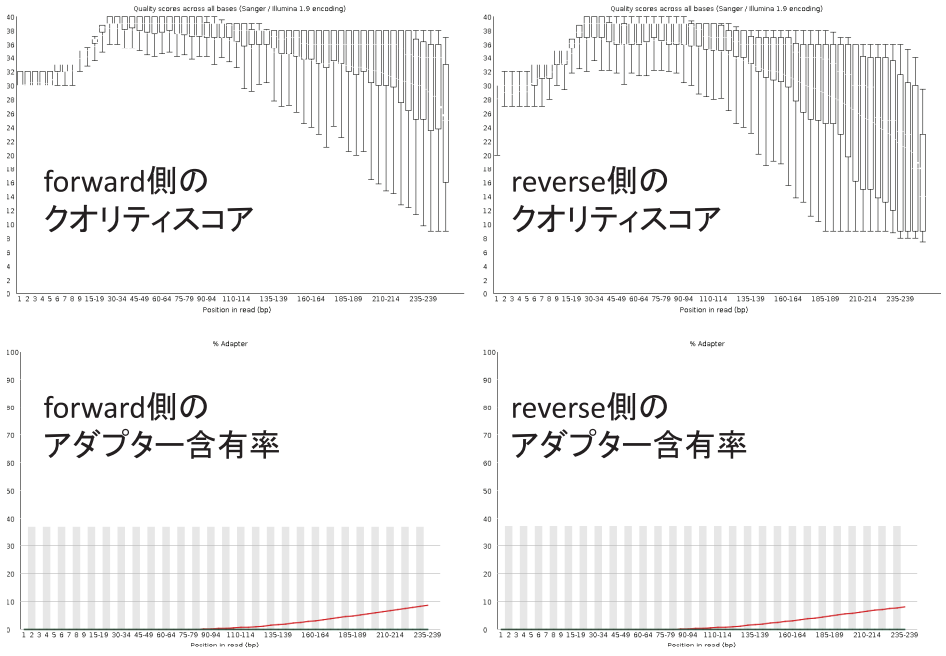

図 1．FaQCs 実行前の FastQC 実行結果。
クオリティスコアは reverse 側のほうが相対的に低く、アダプター含有率（Illumina Universal Adapter）は forward 側 reverse 側ともに同程度であることがわかる。

着目してもらいたい 2 つめの項目は、forward 側 reverse 側ともに黄色の Adapter Content（アダプター含有率）である（**図 1 下**）。これは、リードのポジションごとに既知のアダプター配列含有率をプロットしたものである。これまで HiSeq シリーズ以前のデータしか眺めてこなかった読者にとっては衝撃的かもしれないが、一般に読み進んでいくほど 3' 側の末端部分にアダプター配列が含まれる確率は上昇する。このデータの場合、Illumina Universal Adapter が 3' 側に多く含まれており、終端付近では全リードの 10% 弱になっていることがわかる。そこでリードの前処理としてアダプタートリミングの実行を検討する。自らシーケンスを行う場合や受託企業に依頼して行う場合には、MiSeq ではシーケンス実行時にアダプターをトリムするオプションを指定して実行することで、自分でアダプターを取り除く手間を省くこともできる。

アダプタートリミング

FaQCs [7]（ver. 1.34）を用いて、アダプター除去を行う［**W5-1**］。FaQCs の入力は計 60 万リードからなる 2 つの gzip 圧縮 FASTQ ファイル、主な出力は処理後の FASTQ ファイルおよび QC レポートファイルである［**W5-2**］。-adapter オプションをつけて実行する FaQCs プログラムは、まずファイルごとに独立してトリミングを実行し、その後 forward 側と reverse 側で共通して生き残ったリードを非圧縮 FASTQ ファイル（QC.1.trimmed.fastq および QC.2.trimmed.fastq）として出力する。プログラム実行によって、全部で 150,600,000 塩基（＝251 bp×60 万リード）のうち 4,868,725 塩基（3.23%）がトリムされ、ペアで生き残ったリード数は計 595,266 個（99.78%）であることがわかる［**W5-3**］。ファイル中の行数をカウントして得られた数値（1,190,532）を 4 で割った値（＝297,633）がリード数に相当する［**W5-2**］。著者らは、paired-end の 2 つのファイルでともに同じ値になっていること、およびこれらを足した値（297,633 + 297,633＝595,266）がレポートファイル（QC.stats.txt）中の値と一致していることの確認などを通じて、プログラムの挙動や結果の理解を深めていく。

図 2 は、FaQCs の出力ファイルを入力として FastQC を再度実行し、FaQCs のアダプター除去精度を調べた結果である［**W6**］。FaQCs 実行前（**図 1**）に比べると、クオリティスコア分布は全体としてわずかに上昇し、アダプター含有率は大幅に低下していることがわかる。このことから、FaQCs はアダプター除去がメインのプログラムであり、クオリティフィルタリングはほとんど機能していないのではないかという疑問が浮かぶ。そして「FaQCs.pl -h」で利用可能なオプションや、そのデフォルト（初期設定）がどうなっているのかを学ぶのである［**W6-3**］。

一般に、許容するクオリティスコアの閾値を厳しくする（上げる）ほど、生き残るリード数は減っていく。このトレードオフに関する明確なガイドラインはおそらくないが、最近ではクオリティを維持しつつ、できるだけ有効なリード数を残すための最適な閾値を自動で決めるプログラム UrQt なども提案されている [13]。本稿では「FaQCs → アセンブリ」の流れで行うが、アセンブリおよびその検証

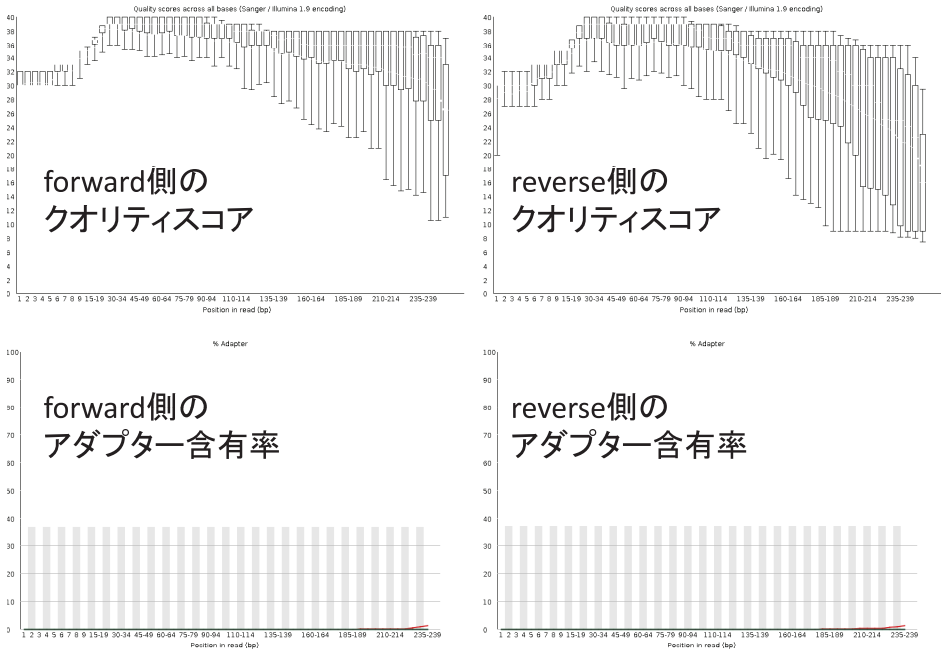

図 2. FaQCs 実行後の FastQC 実行結果。
クオリティスコアは全体的にわずかに上昇し、アダプター含有率（Adapter Contents）は大幅に低下していることがわかる。

結果次第では「FaQCs → UrQt → アセンブリ」を試してみてもいいかもしれない。

Velvet によるゲノムアセンブリ（Bio-Linux）

代表的なゲノムアセンブリプログラムの 1 つである Velvet [2] は、2 つのプログラム（velveth と velvetg）から構成されている。Velvet のウェブサイトから辿れるマニュアルを概観すれば、velveth → velvetg の順番で実行すればよいことがわかる。Bio-Linux には Velvet（ver. 1.2.09）がプレインストールされているため、（パスも通されているのでどのディレクトリ上からでも）これらのコマンドをすぐに実行できる［**W7-1**］。但し、velveth 実行時に指定するアセンブル時の主要なパラメータである *k* 値は、指定可能な最大値が 31 に制限されている［**W7-3**］。これは Bio-Linux に限った話ではなく、Velvet プログラムをデフォルトオプションでインストールするとそうなる。ここではまず、デフォルトの上限である *k* = 31 を用いて、Velvet の基本的な利用法を紹介する［**W7-7**］。合計約 60 万リードの 2 つの gzip 圧縮 FASTQ ファイル（QC.1.trimmed.fastq.gz と QC.2.trimmed.fastq.gz）を入力として与え、Velvet 実行結果を uge ディレクトリに保存する場合は、「velveth uge 31 -shortPaired -fmtAuto -separate QC.1.trimmed.fastq.gz QC.2.trimmed.fastq.gz」→「velvetg uge」の順番で実行すればよい。数分程度で終了し、uge ディレクトリ中に主なアセンブリ結果である contigs.fa という multi-FASTA 形式のファイルが作成される。

アセンブリ結果の概要を知るべく、まずは配列数を調べる。第 3 回の W14 でも述べたように、配列数は multi-FASTA ファイルの description 行の数と同じなので、grep コマンドを用いて “>” から始まる行の数をカウントすればよい［**W7-8**］。アセンブルされた総塩基数、配列の平均長、N50 などは、例えば第 5 回の W12-3 で行ったやり方を参考にして、R [14] で実行してもよい［**W8-1**］。ここでは、入力ファイル名が contigs.fa、出力ファイル名が result_hoge.txt として実行する R スクリプトファイル JSLAB6_1.R を予め用意し、ゲスト OS のターミナル画面上で R のバッチモードとして実行する例を紹介したが、ホスト OS の R GUI 版で行う通常のやり方（対話モード）で実行してもよい。後者の場合は、contigs.fa を共有フォルダ経由でホスト OS に移動させ、**（R で）塩基配列解析**の「イントロ | NGS | 読み込み | FASTA 形式 | 基本情報を取得」という項目を参考にすればよい［**W8-1**］。

著者らの経験上、推定ゲノムサイズ 2-3Mbp の一般的な乳酸菌ゲノムでは、配列数が数十個になれば上出来である。しかし、今回得られたような数万配列というのは明らかに悪い結果である。これはひとえに、指定した *k* 値（*k* = 31）が小さいことに起因する。最適な *k*-mer はゲノムサイズによっても異なるが、一般にデータ量（= リード数 × リード長 *L*）が多いほど大きくなる傾向にある。Velvet 開発時点では充分と考えられていた 31 という最大値も、その後のシークエンス技術の急速な進展を考えると明らかに不十分である。*L* = 250 bp の MiSeq データを十分に活かすのであれば、*k* = 101, 121, 141, 161, 181 あたり（場合

によってはもっと大きい値）を指定したいところである。しかし、例えばk＝141を指定しても、31以上の数値は全て31として実行される［**W8-3**］。解決策は、この画面の実行ログに記されている。それは、指定可能なk値の上限を変更するオプションをつけ、Velvetをrecompile（再インストール）することである。

Velvetの再インストールと実行（Bio-Linux）

2016年1月4日現在のVelvetの最新版は、ver. 1.2.10である［**W9-1**］。マニュアルの「2.2 Compiling instruction」に、インストールの基本形はmakeだということが書かれている。そして「2.3.3 MAXKMERLENGTH」に、指定可能なk値の上限を変更するやり方が示されている。Velvetプログラム（velvet_1.2.10.tgz）のダウンロード、-zxvfオプションつきのtarコマンドで.tgzファイルの解凍、そして解凍後に作成されるvelvet_1.2.10ディレクトリ上での「make 'MAXKMERLENGTH＝201'」によって、k値の上限を201まで指定可能な実行プログラム（velvethとvelvetg）を作成できる［**W9**］。例えば、k＝181でVelvet（ver. 1.2.10）を実行すると、配列数が198個、総塩基数が2,386,048 bp（約2.4MB）という結果が得られる［**W10-1**］。配列数29,502個、総塩基数4,077,679 bp（約4.1MB）のk＝31の結果と比べると、よくつながっている（配列数が減っている）ことがわかる。尚、総塩基数はゲノムサイズに相当する。「塩基（bp）」と「バイト（bytes）」を意図的に混在させているが、「1塩基＝1 byte」であることを利用すれば、ファイルサイズからも総塩基数を計算可能である［**W8-2**］。ヒトゲノムが約30億塩基対で、ファイルサイズが約3GB（約30億バイト）であったことを思い出せば納得できるであろう。尚、このデータの正解は、配列数が3（1 chromosome + 2 plasmids）、2,400,586 bp（約2.4MB）である[4]。k値の選択の重要性がよくわかる例といえよう。

通常、Velvetを実行する場合は複数の異なるk値を用いてアセンブルを行い、それらの結果を眺める［**W10**］。ここでは、計10個のk値（k＝31, 61, 91, 111, 121, 131, 151, 171, 181, 191）で実行した結果を眺め、主に配列数の観点から、k＝171周辺の結果が一番よさそうだと解釈する。もちろんこのデータの場合は、「真のゲノムサイズは約2.4MB」だという答えがわかった状態でアセンブリ結果の評価を行っていることになるが、実際には近縁種との比較により妥当と考えられるゲノムサイズを検討する。ここではそのような情報が得られなかったと仮定して「ゲノムサイズ推定」を行い、アセンブリ結果の評価を行う。

ゲノムサイズ推定

ゲノムサイズの推定は、フローサイトメトリー（flow cytometry）という手法を用いて実験的に求めるやり方と、k-mer解析によってNGSデータから計算で求める2種類のやり方が存在する。k-mer解析の理論はそれほど難しくはないが、現在はJerryfish[15]、KmerGenie[16]、KmerStream[17]など専用のプログラムがいくつか存在する。実用上は、このうちのどれかを用いて得られた推定値を信用すればよい（もちろんアセンブリ結果との著しい乖離があればその限りではない）。ここでは、KmerGenieのインストール、基本的な利用法、およびその結果が確かに真の値に近いことを示す。

KmerGenieは、NGSデータを入力としてゲノムアセンブル時に用いる最適なk値（Predicted best k）を主な出力として返すプログラムであるが、同時にゲノムサイズを推定した結果（Predicted assembly size）も返してくれる。ここでは、ver. 1.6982のインストールを行ったが、基本的な手順（tarで解凍、makeでインストール）はVelvetのときと同じである［**W11**］。但し、解凍後のkmergenie-1.6982ディレクトリ内にあるREADMEファイル（ウェブサイト上のREADMEでもよい）を眺めると、デフォルトではk値の探索範囲が120までに制限されていることがわかる。既にk＝191までのアセンブリ結果が手元にあるので、例えばk＝200まで探索可能な実行ファイルを作成したい場合は「make k＝200」とすればよい［**W11-5**］。

k＝31から191の探索範囲でKmerGenie（ver. 1.6982）を実行した結果を**図3**に示す。推定ゲノムサイズ（Predicted assembly size）は、forward側のsingle-endのみの結果が2,356,713 bp（**図3左；W11-9**）、paired-endの結果が2,367,453 bpであった（**図3右；W11-11**）。どちらの推定値も真の値（2,400,586 bp）に近いことがわかる。推奨のk値が大きく異なるのは、paired-end（k＝141）ではsingle-end（k＝87）に比べデータ量が単純に2倍になっているからである。実際にはpaired-endのデータを用いてアセンブリを行うので、paried-endでの推奨k値（＝141）の前後であるk＝131-191あたりを手厚く探索するとよい結果が得られそうだと判断する。

配列長によるフィルタリング

比較的マイナーな事柄ではあるが、通常下記に示す3つの理由から、アセンブリ結果から短い配列を除外する：

1. MiSeqを含むショートリードの*de novo*アセンブリでは、挿入配列（insertion sequence）やリボソームRNA遺伝子領域（rDNA）のような、ゲノム中に複数コピーが散在する反復領域（dispersed repeat）の再現は難しい。配列（コンティグ）がこれらの反復領域部分で分断されてしまうからである。アセンブリ結果に含まれる短いコンティグは、これらの反復領域の一部である場合が多く、その後の解析には大きな影響を及

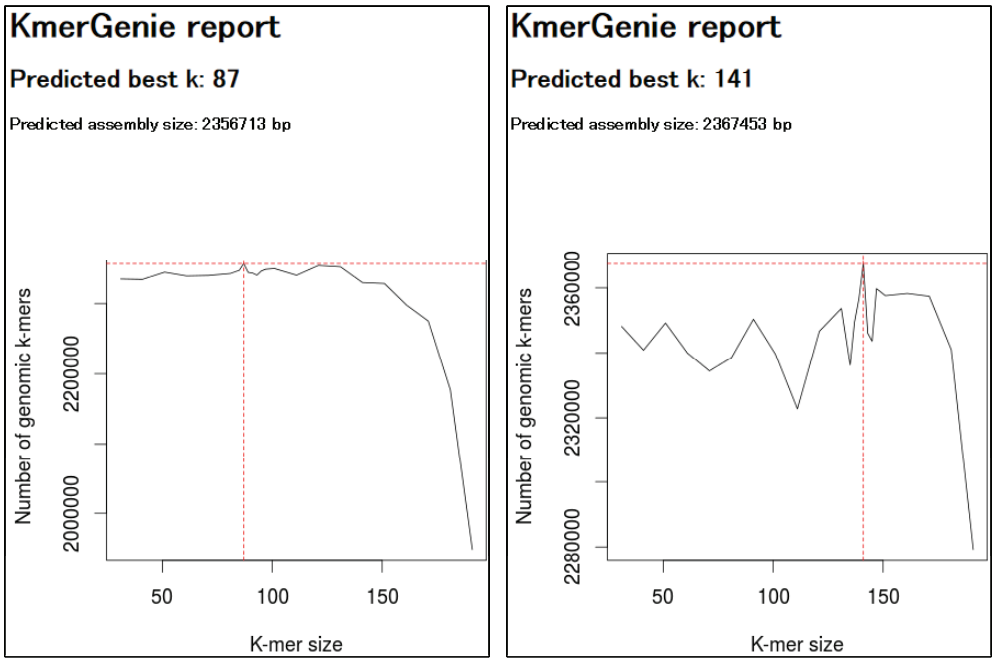

図 3．KmerGenie 実行結果の一部。
左が single-end での実行結果（forward 側のみ）、右側が paired-end での実行結果。推奨の *k* 値（Predicted best k）は異なるが、ゲノムサイズの推定値（Predicted assembly size）はほぼ同じであることがわかる。

ぼさない。

2. 国際塩基配列データベース（DDBJ/ENA/GenBank）に登録する場合、200 塩基未満の短い断片は除外することが推奨されている（http://www.ncbi.nlm.nih.gov/genbank/wgs）。
3. アセンブリ結果の検証の際に、得られた配列をリファレンス配列として用いるが、元のリードの長さよりも短い配列にはマップできないため検証が困難である。

第 5 回でも述べたが、この乳酸菌ゲノム配列決定論文[4]の Platanus プログラム[5]による MiSeq データアセンブリ結果（53 配列）は、300 bp 未満の配列をフィルタリングしたものである。「filter contig length」などでウェブ検索すれば手段は見つかるが、ここではプログラミング言語 Python で自作した fastaLengthFilter.py という名前のプログラムをダウンロードして利用するやり方を示す。Permission やパスなど第 4 回で述べた事柄以外にも、様々な既出の手段を駆使してプログラムの動作確認や解析結果の全貌の把握を行っている［**W12**］。

表 1 は、主にフィルタリング前後の Velvet 実行結果をまとめたものである。配列長（< 300 bp）によるフィルタリング前は、最大で約 5.7MB（*k* = 111 での Velvet アセンブリ結果）というゲノムサイズに達していたが、フィルタリング後は最大でも約 2.6MB（*k* = 151 での結果）となっていることがわかる。KmerGenie の推奨 *k* 値（= 141）以上のアセンブリ結果では、少なくとも調べた範囲（*k* = 151, 171, 191）では、300 bp 未満の配列は存在しないことがわかる。フィルタリング結果（特に配列数）は、設定する閾値によって大きく異なりうる。アセンブリ結果の解釈にも大きく影響する一方で、明確なガイドラインを示すのは現実には難しい。配列数が少なく、近縁種のゲノムサイズと近い値になっていたとしても、本来あるべき遺伝子がごっそり抜けていることもある。**表 1** で示すような様々な *k* 値や閾値の結果、近縁種のゲノムサイズとの比較などに加え、①アセンブリ結果をリファレンス配列としたマッピングや、②必須遺伝子の存在確認を通じたゲノムの質の評価も重要である。これらについては、第 7 回で詳述する予定である。

表 1．Velvet アセンブリ結果。
（a）Velvet の生の出力結果（フィルタリング前）、（b）300 bp 未満の配列を除去した後（フィルタリング後）。*k* = 171 のときに配列数が最少の 168 個、総塩基数（ゲノムサイズ）が真の値（約 2.4MB）に近いことがわかる。一番下の行は、後述する Platanus（ver. 1.2.2）実行結果。

	(a) フィルタリング前			(b) フィルタリング後		
k-mer	配列数	総塩基数	ウェブ資料	配列数	総塩基数	ウェブ資料
31	29502	4077679	W10-4			
61	15445	3886574	W10-4			
91	8583	3412266	W10-4			
111	23761	5718204	W10-5			
121	15776	4690144	W10-5	2942	1563384	W12-9
131	8398	3710829	W10-5	2449	2151400	W12-9
151	1306	2599377	W10-5	1306	2599377	W12-9
171	168	2381523	W10-5	168	2381523	W12-9
181	198	2386048	W10-1			
191	336	2405431	W10-5	336	2405431	W12-9
Platanus	117	2356019	W20-3	52	2346500	W20-7

DDBJ Pipeline（概要からアカウント作成まで）

これまで行ってきた解析結果は、全データの約 1/10 のリード数からなるサブセットについてのものである。乳酸菌を含むバクテリア程度であれば、オリジナルデータを用いた *de novo* アセンブリも実行可能かもしれない。どの程度のデータ量まで手元の PC で可能か？どの程度メモリを積んだノート PC であればこのデータ量の解析が可能か？といったデータ量やスペックに関するグレーゾーンの議論はここでは行わない。計 200 万リードからなる paired-end RNA-seq データの *de novo* transcriptome assembly が 2GB メモリでできた（第 5 回の W5-2）こと、本稿で示した合計約 60 万リードからなる paired-end データの Velvet アセンブリがマニュアル通りのやり方でできたことなど、実体験の積み重ねのほうが有意義であろう。

データ量が大きな配列解析を行う手段の 1 つは、国立遺伝学研究所（以下、遺伝研）が運用するスーパーコンピュータシステム（以下、スパコン）[18] の利用である。本連載で何度か紹介した DDBJ Read Annotation Pipeline（以下、DDBJ Pipeline）[3] は、遺伝研・大量遺伝情報研究室において開発・運用されている NGS 解析に特化した遺伝研スパコンを遠隔利用できるクラウドウェブサービスの 1 つである。DDBJ Pipeline は、基礎処理部と高次処理部からなる。基礎処理部ではマッピングや *de novo* アセンブリが利用可能であり、高次処理部では第 1 回でも紹介したデータ解析プラットフォーム Galaxy [19] が利用可能である。一口に Galaxy とはいっても、提供サイトごとにいくつか独自の解析プログラムが組み込まれている。理解しづらい概念かもしれないが、独特の GUI 画面（ウェブサイトの見栄え）は Galaxy という統一基準のものがあり、基本的な解析ツールはどの提供サイトでもほぼ同じだが、売り（独自機能）として解析ツールの組み合わせ部分（ワークフローと呼ばれる）が異なると解釈すればよい。例えば、TRAPLINE [20] は RNA-seq 解析に、第 5 回でも紹介した Orione [21] はバクテリア解析用に特化した Galaxy ベースのワークフローを提供している。DDBJ Pipeline の高次処理部（Galaxy）は、基礎処理部の結果をインポートして多型解析や発現量解析を行うための独自機能が追加されている。本稿では、新規アカウント作成から基礎処理部の *de novo* アセンブリまでの一連の手順を示す。

新規アカウント作成は、氏名・所属・利用目的などの必要事項を入力する［**W13-1**］。アカデミック・企業ユーザに関係なく誰でも利用申請可能で無料で利用できるが、研究利用に限られ受託解析などの商用利用はできない。

DDBJ Pipeline（クエリファイルの登録）

ログイン後に最初に行うことは、解析したい NGS データ（クエリファイル）の登録作業である［**W13-2**］。大まかには、DDBJ Pipeline に解析してもらいたいデータのインポートまたはアップロードで遺伝研スパコンに設置する作業と読み替えればよい。この作業には、3 通りのやり方が存在する：

① DRA/ERA/SRA から始まる ID の指定（DRA からのインポート）
② FTP 経由でアップロード
③ HTTP 経由でアップロード

公共 DB で公開されている NGS データを解析したい場合は、①で DRA ID を与えればよい。但し、本稿で取り扱っている DRR024501 という ID は、DRR ～であり DRA ～ではない。つまり DRR024501 を与えてもエラーとなるため、DRR024501 を頼りに公共 DB を眺めて指定可能な DRA ID（この場合は DRA002643）を探す必要がある［**W2; W13-3**］。手元のファイルを解析したい場合は、②FTP 経由か③ HTTP 経由でアップロードする。FTP 経由のアップロードは、FTP クライアントソフトウェア（以下、FTP ソフト）を利用する［**W13-6**］。WinSCP（Windows 用）や Cyberduck（Macintosh 用）がホスト OS 上にインストールされていれば、マニュアルの指示通りに設定情報を入力すればよい［**W14-2**］。FTP ソフトを介した DDBJ Pipeline への接続後は、指定された query フォルダ内に解析したいファイルをドラッグ & ドロップでアップロードするだけである［**W14-4**］。

注意点としては、アップロードが完了した query フォルダ内のファイルが消えたように見えることである。これは、DDBJ Pipeline の FAQ（Q. FTP でアップロードしたファイルが web に反映されません）にも記載されているが、アップロードが完了したファイルは、遺伝研スパコン内の所定の場所に自動的に移動するように設定されている。つまり、一定時間（～ 1 分）経過後に FTP ソフト内で query フォルダ（DDBJ Pipeline 側）をリロードするとファイルがなくなるのは正常である。

アップロード後は、再びウェブブラウザ上での作業になる。すぐにアセンブルの実行ボタンを押したいところではあるが、アップロードしたファイル群の中から、どのファイルとどのファイルが paired-end のペアなのかなどを指定する必要がある。paired-end であっても single-end として解析したい場合など、別々のものとして認識させたい局面もあるだろう。クエリファイルと呼ばれる DDBJ Pipeline 上に置いた NGS データのレイアウト情報（paired-end または single-end）や、データを取得した Instrument model（NGS 機器のこと；具体的には Illumina か PacBio かなど）の情報を、ファイル名と関連づけて認識させる作業が登録（registration）である［**W14-5**］。ここでは、Bio-Linux 上で Velvet アセンブリ実行時に用いたものと同じ、合計約 60 万リードの 2 つ

の gzip 圧縮 FASTQ ファイル（QC.1.trimmed.fastq.gz と QC.2.trimmed.fastq.gz）を FTP 経由でアップロードし、paired-end データとして登録する例を示した［**W14**］。ここで今一度、第 2 回[11] の「バイオインフォマティクス分野の常識・非常識」に目を通してもらいたい。ファイル名の途中にスペースやカッコなどの特殊文字を含めるのは非常識である。拡張子も、FASTQ 形式の場合は、.fq か .fastq にしておくのが無難である。

DDBJ Pipeline（基礎処理部 Velvet アセンブラの実行）

DDBJ Pipeline 基礎処理部で指定する事柄は、大まかに 5 つのステップからなる：①クエリファイルの選択、②解析したいツール（プログラム）の指定、③クエリセットの指定、④ツールオプションの指定、そして⑤実行。①のクエリファイルの指定は、二度手間だと思われるかもしれない。しかし、例えば予め登録しておいたクエリファイルが 3 個以上あり、そのうちの 2 個だけを実行したい場合などに必須である［**W15-1**］。②で選択可能なツールは、マッピングが 5 種類、そして *de novo* Assembly が 6 種類存在する（2016 年 1 月 16 日現在）［**W15-2**］。③のクエリセットは、①で複数のクエリファイルを指定した場合に効力を発揮する（単独のクエリファイルの場合は意味をなさない）。具体的には、複数のクエリファイルを連結して 1 つのクエリセットとして実行するか、あるいは別々のものとして並列に実行するかを指定する。例えば、部位ごとにサンプル取得した RNA-seq のデータがあり、これを部位に関係なく発現している転写産物全てに対して解析するなら前者を、部位ごとに独立して解析したいなら後者を選択するであろう［**W15-3**］。④のツールオプションは、マッピングの場合は許容するミスマッチ数や、複数個所にマップされるリードの取り扱いなどを指定する。アセンブリの場合は、*k*-mer の *k* 値や、scaffolding 時のインサートサイズなどを指定する。もちろんオプションの種類は、②で指定するツールによって異なる［**W15-4**］。

⑤は基本的に実行ボタンを押すだけであるが、ジョブの実行が完了すれば電子メールでお知らせしてくれる点に注目してもらいたい［**W15-5**］。自分だけで占有しているノート PC などとは違って、DDBJ Pipeline の登録ユーザ数は 1,000 名弱存在する（2016 年 1 月 16 日現在）。また、DDBJ Pipeline 以外のスパコンユーザも数多く存在する。それゆえ、通常は RUN ボタンを押してもすぐに自分のジョブが実行されるわけではない。実際に、著者らが DDBJ Pipeline 上で Velvet アセンブリ（$k=131$）を実行したときは、約 100 分かかった［**W16-1**］。もちろんこれは、スパコンにジョブとして登録された時間と終了した時間の差分であり、実際の計算時間（約 1 分）とは異なる。スパコン上でのジョブの待ち時間が律速になっているのが現状である。

DDBJ Pipeline 出力結果の確認

DDBJ Pipeline のアセンブリ結果は、同じプログラム（Velvet ver. 1.2.10）を同じオプションで実行している限り、基本的に Bio-Linux 上での実行結果［**W12-10**］と同じである。例えば $k=131$ で得られた Bio-Linux 上での配列数は、配列長フィルタリング前が 8,398 個［**W10-5**］、フィルタリング（300 bp **未満**の配列を除去）後が 2,449 個［**W12-9**］であった。DDBJ Pipeline 上でも同様のオプションを指定して実行した結果［**W15-4**］、フィルタリング前は全く同じ 8,398 個［**W16-2**］、フィルタリング後は 2,446 個であった［**W17-3**］。結論から述べると、この配列数 3 個の違いは、300 bp **未満**（less than 300）か**以下**（300 or less; not more than 300）かの違いである。DDBJ Pipeline のオプション指定のところでは、Set filtered length for contigs と書かれているだけであり、閾値（threshold）未満か以下かについては触れられていない。また、フィルタリング前のアセンブリ結果の概要（Assembly statistics）は表示されているものの、フィルタリング後の配列数が 2,446 個という情報は示されていない［**W16-2**］。ここでは、DDBJ Pipeline の出力結果をダウンロード（Bio-Linux 上にインポート）し、Bio-Linux 上で確認する手段を示す。

DDBJ Pipeline では、Velvet 実行結果の生データファイル（velvet.zip; **W18-1**）と、フィルタリング後の FASTA 形式に近いファイル（out_WGS.fasta.gz; ［**W16-3**］）の 2 種類をダウンロード可能である。著者らはまず、Bio-Linux 上で後者のファイルを概観し、基本的には FASTA 形式ではあるものの、各配列の間に「//」の行が挿入されていることを最初に認識した［**W17-2**］。次に配列数が 2,446 個であることを確認し、Bio-Linux 上での実行結果［**W12-10**］と微妙に異なっていることを見出した。そして、複数の異なる *k*-mer から得られたフィルタリング後の配列数の分布、および Bio-Linux 上で実行した 300 bp **未満**の配列を除く fastaLengthFilter.py 実行結果が 2,449 個であったという事実を総合的に勘案し［**W12-9**］、もし DDBJ Pipeline のフィルタリングが 300 bp **以下**だったとすれば 2,446 個という数値は妥当だろうと判断した。予め、DDBJ Pipeline から得られたフィルタリング前のファイル（contigs.fa; ［**W18-3**］）を入力として、閾値**未満**の配列を除く fastaLengthFilter.py 実行結果が 2,449 個という傍証を得た［**W18-4**］。このプログラムを閾値**以下**という条件判定に書き換えて実行し、2,446 個という数値が得られれば、**未満**か**以下**の違いだったといえる。

このプログラムは Python というプログラミング言語で記述されているが、たとえ Python のプログラミング経験がなくとも、通常どこで条件判定を行っているか程度は認識可能である［**W18-5**］。オリジナルの条件判定は「>＝threshold」として閾値以上の長さの配列を残し、それ**未満**のものは除くように指定されている。これを「>

threshold」と変更して閾値より長い配列を残し、それ**以下**のものを除くように指定し直した新たなプログラムfastaLengthFilter_orlonger.pyとして保存するのである。ここでは、実行結果として2,446個という配列数を得ることで予想を確信に変えることができた［**W18-8**］。これが、第2回でも述べた「目的に近いプログラムをコピーして、必要最小限の箇所を変更することで、新たな別のプログラムとしての機能を持たせる」の具体例である[11]。

DDBJ Pipeline（基礎処理部 Platanus アセンブラの実行）

乳酸菌（*L. hokkaidonensis* LOOC260^T）ゲノム配列決定論文[4]では、Illumina MiSeqデータ（DRR024501）の*de novo*アセンブリを、Platanus[5] ver. 1.2のデフォルト設定で実行している。原著論文中では明記されていないが、生のリードデータをそのまま入力として用い、300 bp未満の配列をフィルタリングした結果、53配列が得られた。ここでは、DDBJ Pipelineに登録済みのデータ（オリジナルの約1/10のリード数からなるサブセット）［**W14**］を入力として、DDBJ Pipelineで利用可能なPlatanus（ver. 1.2.2）を実行し、Velet実行結果との違いを議論する。

Platanusプログラムは、3つのステップから構成される：Step1）Assembly、Step2）Scaffold、Step3）Gap Close［**W19-4**］。Step1では、Veletと同様、リードを*k*-merに分割し、*k*-merの重なりから構築したde Bruijnグラフを利用したアルゴリズム[22]でコンティグを作成する。Step2では、paired-endのペアのリードを得られたコンティグにマップし、ペア間の情報を用いてコンティグ同士を連結（scaffold）する。連結されたコンティグ間は、未知塩基Nで埋められたギャップで表現される。この作業は、scaffoldingと呼ばれる。Step3では、リードを得られたscaffoldにマップし、ギャップ付近にマップされたリード情報を利用してギャップを埋める作業を行う（gap closingと呼ばれる）。

Platanusでは、Scaffoldingの作業（Step2に相当）を行う際、NGSデータ取得時の実験情報をオプションとして与えることもできる。例えばこのデータ（DRR024501）は、ゲノムDNAを520 bp±50の長さに断片化し、断片の両端から251 bpずつをMiSeqでシークエンスしたものである。この実験情報は、DRX022186から得られる［**W2-3**］。DRX022186を眺めると、LayoutがPAIRED、Nominal Lengthが520、Nominal Sdevが50.0、Spot Lengthが502になっている。このことから、paired-endデータ、ペアのリード間の塩基数（インサート長、インサートサイズともいう）の平均が520 bp、標準偏差が50、リードの長さが502 bp（paired-endなので片側251 bp）なのだと読み解き、原著論文の記述内容と合わせて理解を深める。PlatanusのStep2では、インサートサイズの実験情報を利用すべく「-a 520」というオプションを与えてもよい。このオプションを指定しなかった場合［**W19-4**］、配列にマップした結果から自動で見積もられるため、特に指定しなくても動作する。バクテリアの場合は自動見積もりで十分うまくいくという印象をもっているが、うまくいかない場合はこのような実験情報を援用してみるといいだろう。インサートサイズの指定は、おそらく（paired-endではなく）メイトペア法でシークエンスされたリードを用いたアセンブリでより効果を発揮すると思われる。

著者らがデフォルト設定でPlatanusを実行したときは、約16時間（実際の計算処理以外の待ち時間を含む）を要した［**W20-2**］。300 bp未満の配列をフィルタリングする前の状態で、配列数が117個、総塩基数（Total contig size）が2,356,019 bpという結果が得られた。注目すべき点は2つある。1つは、この117個という配列数は、計10通りの*k*値でVeletを実行して得られた最少配列数168個（$k=171$）よりも少ないという点である［**W20-3**］。そしてもう1つは、Platanus実行時に*k*値を指定していないという点である。Step1実行時のログファイルを眺めると、$k=32$から123まで調べていることがわかる。最初のうちは10おきに調べ（$k=32, 42, 52, \cdots$）、最後のほうでは1おきに調べている（$k=\cdots, 121, 122, 123$）ことがわかる。Platanusは、Veletのように単一の*k*-merを利用するアセンブラ（single-*k* genome assemblers）ではなく、複数の*k*-merを利用するアセンブラ（multi-*k* genome assemblers）のカテゴリーに属する。

小さい*k*値（短い*k*-mer；この場合$k=32$）を採用するのは、coverageが低い領域をアセンブリ結果に含めるためである。但し、例えば$k=32$の場合は32塩基の完全一致を頼りにつなげていくため、ゲノム中のリピート領域が32塩基よりも長い場合にはそこで分断されてしまう。大きい*k*値（長い*k*-mer；この場合$k=123$）まで調べるのは、指定した*k*値よりも短いリピート領域を乗り越えるためである。Platanusの場合は、$k=32$でまずアセンブリを行って得られたコンティグはそのまま採用した上で、$k=42$でつながるところをつなぎ、$k=52$でつながるところをつなぐ、という操作を繰り返している。（オリジナルの1/10のサブセットからなる）このデータの場合は$k=123$までしか探索していないが、この探索範囲はデータ量に応じて自動で決まるようである。ここまでの説明で、表1（a）で*k*値が小さいほど総塩基数が多くなる傾向にある理由（coverageが低い領域も含むため）がわかったであろう。また、本稿ではKmerGenieを主にゲノムサイズ推定目的で利用したが、本来の目的（推奨*k*値を返す）で考えた場合に、「KmerGenie predictions can be applied to single-k genome assemblers（e.g. Velvet, …」のような説明書きがある理由もわかったのではないだろうか［**W11-1**］。

Platanus（ver. 1.2.2）は、5つの.faと3つの.tsvというファイルを出力する。エンドユーザが欲しい最終結果ファイルは、Step3実行結果のout_gapClosed.faである。それ以外

の4つの.faファイルは、Step1およびStep2の実行結果ファイル、つまり中間ファイルである。grepコマンドで.faからなるファイル中の配列数を眺めることで［**W20-6**］、Step1で349個のコンティグが得られ（out_contig.fa）、Step2で349個のコンティグが117個のscaffold（配列）にまとめられたと判断する（out_scaffold.fa）。最後のStep3は、ギャップを埋める作業（gap closing）である（out_gapClosed.fa）。scaffold（配列）数が少なくなる要素はどこにもないため、out_scaffold.fa（Step2実行後）とout_gapClosed.fa（Step3実行後）間でscaffold（配列）数は不変である。おそらくコンティグ間のギャップ部分のNが一定数減っているのであろう。Platanusの最終結果ファイル（out_gapClosed.fa）に対して300 bp未満/以下の配列をフィルタリングした結果、52個という結果が得られた［**W20-7**］。オリジナルの約1/10のリード数からなるサブセットを入力としたにも関わらず、原著論文の結果（53配列）よりも、少なくとも配列数の点ではよい結果が得られている。これは、FaQCs[7]実行によるアダプター除去の効果かもしれないし、偶然かもしれない。

追記事項として、本稿ではオリジナルのVelvetを利用したが、Bio-Linuxにプレインストールされているロングリード用の派生版を使えば多少異なる結果になったかもしれない［**W20-8**］。厳密には、VelvetはPlatanusのStep2に相当する部分までしか行わない（gap close機能はない）。Scaffoldingに特化したプログラムもいくつか存在する。例えば、「scaffolder」のPubMed検索で見つかったプログラムの利用も一手ではある［**W20-9**］。しかし、gap closeまでall-in-oneで行ってくれるPlatanusをDDBJ Pipeline上で手軽に利用しない手はないだろう。また、本稿ではIllumina MiSeqデータ（DRR024501）のみを解析対象としたが、原著論文[4]ではロングリードのPacBioデータ（DRR054113-DRR054116）を併用することで完全なゲノム配列決定に至っている。先に述べたように、ショートリードを用いたアセンブリでは反復領域の再現には限界がある。一般的な比較解析が目的であれば、ショートリードのデータから十分に精度の高いドラフトゲノムを得ることができる。完全なゲノム配列の再現を目指すのであれば、ロングリードのPacBioデータを利用するのが昨今の主流になっている。このようなハイブリッドアセンブリを行いたい場合は、SSPACE-LongRead[23]などを試してみるといいかもしれない。

おわりに

第6回は、乳酸菌（*L. hokkaidonensis* LOOC260^{T}）ゲノム配列決定論文のデータを用い、任意のリード数からなるサブセットのダウンロードから、DDBJ Pipelineで利用可能なPlatanus（ver. 1.2.2）アセンブリ実行結果の再現までを行った。原著論文中では、Platanusアセンブリ結果の検証を行っている。反復領域などアセンブルできなかった部分を除き、ほぼ全ての遺伝子構造を再現できていることを確認済みである。アセンブリ結果のさらなる検証については、PacBioデータの解析手段も含めて次回以降で取り上げる予定である。

読者の多くは、必要最小限の大規模計算をスパコン（DDBJ Pipeline）上で行い、それ以外をパソコン（Bio-Linux）上で行う基本スキルが既に身についている。Platanusが内部的に行っている3つのステップを理解する上で役立つ情報は、各ステップ実行後に得られるFASTAファイル中の塩基ごとの出現回数であろう。本連載の枠組みでは今のところRを用いた塩基配列解析手段をそれほど示してはいないが、目的（A, C, G, T, Nの出現回数を調べる）に近い例題をテンプレートとして利用すればよい［**W21-1**］。Step2（scaffolding）実行結果ファイル中に含まれていた491個の未知塩基Nが、Step3（gap closing）後にものの見事に（この場合は）なくなっていることがわかる［**W21-2**］。「この説明通りだと結果がこうなっているはず」という予想とその確認作業が自在にできれば、これまでブラックボックスであったプログラムの中身の理解も容易になる。尚、主要なプログラムの多くは継続的にアップデートがなされている。Platanusもver. 1.2.4が2015年10月にリリースされている。本稿で紹介したVelvet［**W9**］やKmerGenie［**W11**］のインストール手順を参考にして、是非Bio-Linux上でのインストールおよび利用にチャレンジしてほしい。

謝　辞

本連載の一部は、国立研究開発法人科学技術振興機構バイオサイエンスデータベースセンター（NBDC）との共同研究の成果によるものです。東京工業大学・大学院生命理工学研究科の伊藤武彦先生には、Platanusのアルゴリズムおよび実行結果の解釈部分についてアドバイスをいただきました。国立遺伝学研究所・生命情報研究センターの有田正規先生には、原稿全般に目を通していただき、有意義なコメントをいただきました。農業・食品産業技術総合研究機構 畜産草地研究所の遠野雅徳先生には、本稿で用いた乳酸菌ゲノム配列解読論文の責任著者として、また編集委員会委員として本連載の円滑な執筆環境構築に尽力いただきました。

参 考 文 献

1) Field D, Tiwari B, Booth T, Houten S, Swan D, et al. (2006) Open software for biologists: from famine to feast. Nat Biotechnol **24**: 801–803.
2) Zerbino DR, Birney E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res **18**: 821–829.
3) Nagasaki H, Mochizuki T, Kodama Y, Saruhashi S, Morizaki S, et al. (2013) DDBJ read annotation pipeline: a cloud computing-based pipeline for high-throughput analysis of next-generation sequencing data. DNA Res **20**: 383–390.
4) Tanizawa Y, Tohno M, Kaminuma E, Nakamura Y, Arita M. (2015) Complete genome sequence and analysis of *Lactobacillus hokkaidonensis* LOOC260^{T}, a psychrotrophic lactic acid bacterium isolated from silage. BMC Genomics **16**: 240.
5) Kajitani R, Toshimoto K, Noguchi H, Toyoda A, Ogura Y, et al. (2014) Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads. Genome Res **24**: 1384–1395.
6) Andrews S. (2015) FastQC a quality control tool for high throughput sequence data, http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
7) Lo CC, Chain PS. (2014) Rapid evaluation and quality control of next generation sequencing data with FaQCs. BMC Bioinformatics **15**: 366.
8) 孫建強，清水謙多郎，門田幸二（2015）次世代シーケンサーデータの解析手法：第5回アセンブル、マッピング、そしてQC．日本乳酸菌学会誌 **26:** 193–201.
9) Kodama Y, Shumway M, Leinonen R, International Nucleotide Sequence Database Collaboration (2012) The Sequence Read Archive: explosive growth of sequencing data. Nucleic Acids Res **40**: D54–56.
10) Silvester N, Alako B, Amid C, Cerdeño-Tárraga A, Cleland I, et al. (2015) Content discovery and retrieval services at the European Nucleotide Archive. Nucleic Acids Res **43**: D23–29.
11) 孫建強，湯敏，西岡輔，清水謙多郎，門田幸二（2014）次世代シーケンサーデータの解析手法：第2回 GUI 環境からコマンドライン環境へ．日本乳酸菌学会誌 **25:** 166–174.
12) Li JW, Schmieder R, Ward RM, Delenick J, Olivares EC, et al. (2012) SEQanswers: an open access community for collaboratively decoding genomes. Bioinformatics **28**: 1272–1273.
13) Modolo L, Lerat E. (2015) UrQt: an efficient software for the Unsupervised Quality trimming of NGS data. BMC Bioinformatics **16**: 137.
14) R Core Team (2015) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.
15) Marçais G, Kingsford C. (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics **27**: 764–770.
16) Chikhi R, Medvedev P. (2014) Informed and automated k-mer size selection for genome assembly. Bioinformatics **30**: 31–37.
17) Melsted P, Halldórsson BV. (2014) KmerStream: streaming algorithms for k-mer abundance estimation. Bioinformatics **30**: 3541–3547.
18) Mashima J, Kodama Y, Kosuge T, Fujisawa T, Katayama T, et al. (2016) DNA data bank of Japan (DDBJ) progress report. Nucleic Acids Res **44**: D51–57.
19) Goecks J, Nekrutenko A, Taylor J; Galaxy Team (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol **11**: R86.
20) Wolfien M, Rimmbach C, Schmitz U, Jung JJ, Krebs S, et al. (2016) TRAPLINE: a standardized and automated pipeline for RNA sequencing data analysis, evaluation and annotation. BMC Bioinformatics **17**: 21.
21) Cuccuru G, Orsini M, Pinna A, Sbardellati A, Soranzo N, et al. (2014) Orione, a web-based framework for NGS analysis in microbiology. Bioinformatics **30**: 1928–1929.
22) Pevzner PA, Tang H, Waterman MS. (2001) An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci U S A. **98**: 9748–9753.
23) Boetzer M, Pirovano W. (2014) SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. BMC Bioinformatics **15**: 211.

Methods for analyzing next-generation sequencing data VI. genome assembly

Yasuhiro Tanizawa[1, 2], Eli Kaminuma[2], Yasukazu Nakamura[2], Kentaro Shimizu[3] and Koji Kadota[3]

[1]Graduate School of Frontier Sciences, The University of Tokyo.
[2]Center for Information Biology, National Institute of Genetics.
[3]Graduate School of Agricultural and Life Sciences, The University of Tokyo.

Abstract

Genome assembly is a major task of NGS analyses. For this purpose, many assemblers based on the de Bruijn graph have been developed. In the framework, each node represents a series of overlapping *k*-mers (*k* nucleotides) and contigs can be obtained as paths solved from the *k*-mer graph. The most significant parameter is therefore *k*. We overview conventional approaches for *de novo* assembly of bacterial genomes. We first describe a command-line based NGS analysis pipeline on Bio-Linux: (1) the characteristics of Illumina MiSeq data using the quality check program FastQC, (2) adapter trimming using FaQCs, (3) *de novo* assembly using Velvet with different *k*-mers, (4) estimation of genome size using KmerGenie, and (5) filtering short sequences using an in-house Python program. We next describe the web-based NGS analysis tool called "DDBJ Pipeline", taking two major assemblers (Velvet and Platanus) for instance. We discuss the effect of different *k*-mers on assembly results using Velvet and the difference between Velvet and Platanus.