kmer頻度に基づく新しいポリッシュプログラムの開発

Developing a new method to polish the long read assembled contig based on *k*mer frequency

◯宮澤秀幸、野口英樹 （ゲノムデータ解析支援センター）

Introduction

DNAシーケンシング技術の向上により、様々なアプローチで生物学的研究が進められるようになった。これまでMiSeq等の比較的短い(<500bp)が正確性が高く、大量のreadを出力するシーケンサーが広く用いられてきた。しかし近年、Pacific Bioscience (PacBio) やOxford Nanopore Technology (ONT) で開発された長い(>10kbp) readを合成する技術により、様々な生物種において全ゲノム解読(Whole genome sequencing)は行われるようになった。しかしその出力されるreadには多くのエラーが含まれ、これらをアセンブルして得られるcontigレベルでもエラーが多く残されている。私たちの酵母ゲノムのデータ(SRA ID: ERR1655119; Materials & Methods)を用いた予備的実験でも、ゲノム全体、エキソンレベル共に多くのエラー観察されている（表1）。そのため、short readから得られたデータを用いてこうしたエラーを修正するための処理(polishing)が広く行われている。

Pilon (Walker et al., 2014)、ntEdit (Warren et al., 2020)、NextPolish (Hu et al., 2020)、JASPER (Guo et al., 2023)など、これまでにいくつかのpolishingプログラムが開発されている。しかし、計算処理に非常に時間がかかる、正確性に問題がある等、多くの課題が残されている。

そこで私たちは、short read data由来の*k*mer頻度を基いてcontig上のエラーの探索と修正を行う新しいpolishingプログラム ***Kpolish*** を開発している。ここではその開発状況について報告する。

Polishing前のcontigの統計情報

配列数 ： 38
総塩基長 ： 10,383,955
N50 ： 318,232

	塩基置換	挿入	欠失
genome	5,689	45,652	13,114
exon	4,094	32,932	8,876

表1) Polishing前のcontigの状況

Materials & Methods

Sequence Data

*)データ量が多すぎたためにデータ量を半分にしている

生物種	Long read	Short read
酵母 *S. cerevisiae*	SRA ID: ERR1655119* (PacBio CLR, 840.7M bases) depth=70.1	SRA ID: ERR1938683 (MiSeq, 995.5M bases) depth=83.0

Assemblyとlong readによるpolishing

Assembly: canu (Koren et al., 2017) > Polishing using long reads: arrow

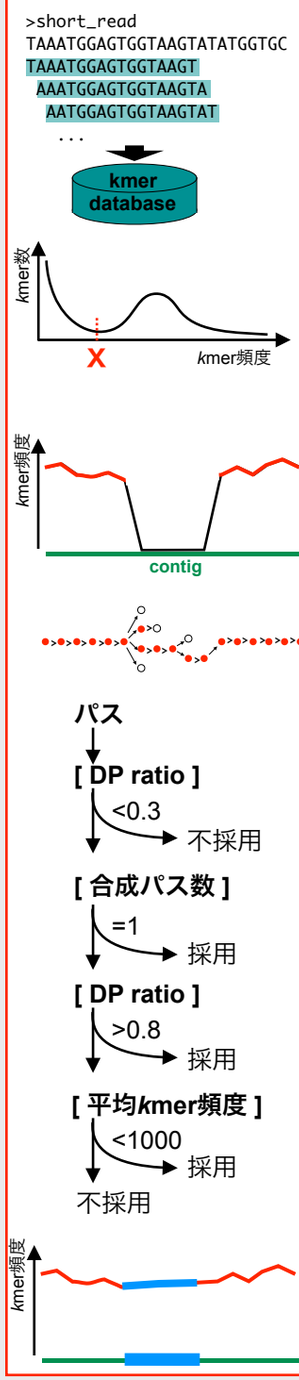
比較polishingプログラム

Alignment base: Pilon v1.23 (Walker et al., 2014) 4 times, NextPolish v1.1.0 (Hu et al., 2020) 2 times,
*K*mer base: ntEdit v1.3.2 (Warren et al., 2019), JASPER v1.0.2 (Gu et al., 2023).

評価方法

Reference (S288C: GCF_000146045.2) にpolishing前後の配列をアライメント
> 培養系統間でのSNP等の差異をGATKで検出し削除
> 正しく修正されたエラー(corrected)、未修正エラー(uncorrected)、polishingによって誤って生じたエラー(falsely corrected)をそれぞれカウント

Algorithm of *K*polish



1) *K*mer カウント

short readのデータを用いて*k*mer (現行K=31)のカウントを行う。その頻度分布は下図のようになる。***x***より頻度が大きい*k*merのほとんどは実際にゲノム上に存在することが期待できる。

2) *K*mer低頻度領域の検出とパス探索

*k*mer頻度が低い領域はshort readから支持されておらず、エラーを含んでいる可能性が高い。その前後の高頻度*k*merを繋ぐパスを探索する。

3) パスの選択

3-1) 元配列との比較

Dynamic Programing score の配列長比(DP ratio)を求め、これが著しく低いパスは用いない。

3-2) 複数パスからの選択

あまりに高頻度な*k*merは反復配列由来である可能性がある。DP ratioが高い、もしくは平均kmer頻度が高くないパスを選択する。

4) 配列の修正

配列を選択したパスと置換する。

Results

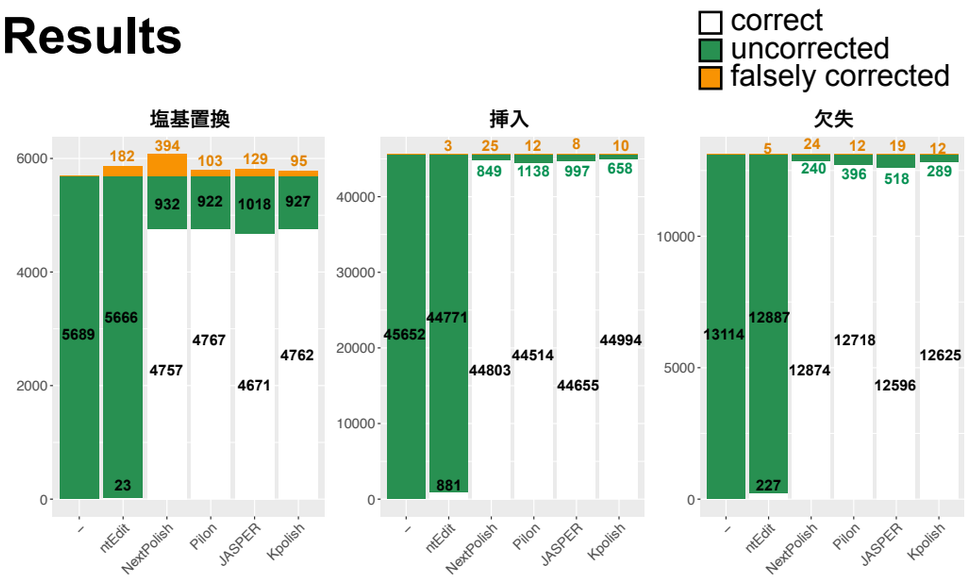


Fig.1) genome全体でのエラー数

酵母ゲノム全体で、*K*polishはPilon、NextPolish、JASPERと同等の修正能力を示した(Fig.1)。塩基置換と欠失の修正数で*K*polishはそれぞれPilon、NextPolishに劣ったものの、挿入の修正数は最も多かった。さらにこれら4つのプログラムではもっとも塩基置換と欠失の誤修正数が少なかった。

exon領域において、*K*polishは塩基置換、挿入、欠失いずれにおいても最も修正数が多かった(Fig.2)。また誤修正数も比較的小さかった。

*K*polishの解析処理にかかる時間は比較的短かいものの、メモリ消費量が大きいことが明らかになった(Fig. 3)。アライメントに時間を多く費やすNextPolishとPilonに対し、*k*mer頻度を用いる*K*polishとJASPERが短時間で解析を終えることができたと考えられる。

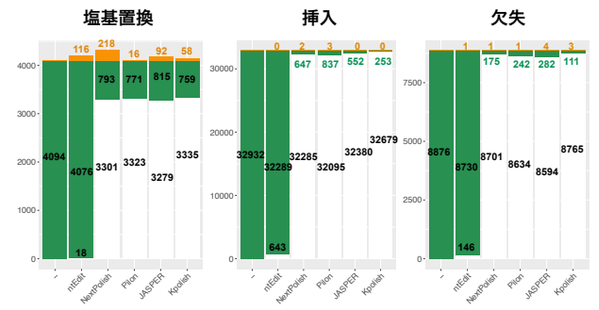


Fig.2) exon領域でのエラー数

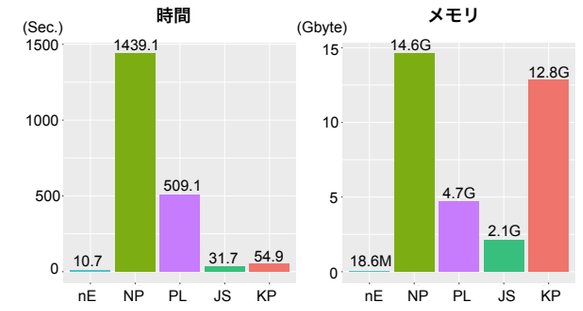


Fig.3) 消費時間とメモリ

Discussion

*K*polishは既存のpolishingプログラムと同等の修正能力を持ち、かつその計算処理にかかる時間が比較的短い、一方で消費メモリが大きいことが判明した。今後は高い修正能力を維持しつつ消費メモリを抑える方法を開発する。さらに酵母以外の様々な生物種においてもテストし、既存のプログラムとの比較を進める。また、現在その普及が急速に進みつつあるPacBio HiFi シーケンシングへの対応も検討している。これにより合成されるreadに含まれるエラーはこれまでのものより遥かに少なく、polishingによりエラーが増える懸念がある。誤修正を最小限に抑える方法についても開発を進める。