KDB ELN

Matthew Ralston mrals89@gmail.com Analyst

Objectives

1. Is the best fit independent of k?
2. What parameters optimize the profile generation time?
3. How does the profile generation time scale with k?
4. Can the correlation distance recapitulate phylogenetic knowledge?
5. If so, what depth is asymptotically similar to

Abstract

Document Purpose

In contrast to the CHANGELOG.md, this file is a log (an ELN of sorts) of my thought process through debugging or improving specific calculations or models. It is ordered chronologically, as the questioned occurred, rather than reflecting specific timings of implementation or organized in a way that reflects interactions/separations in functions. It's a human and scientific document, not a integration testing document.

Introduction

The advent of Next-Generation Sequencing technology may be one of the most influencial technologies of the early 21st century.

Methodology

Results

Outstanding concerns

Many issues remain in the formulation of the probability calculation, addressed below. Additionally, the calculation of sequence profiles from streamed fastq files is taking a considerable amount of time. Each individual read might only contain a small number of kmers, but the object creation and streaming libraries used make the streaming of individual reads slow. Generating a profile from even 50k 150bp reads take longer than the correlation calculation does.

K-mer count distribution

We begin with the k-mer count distribution, a graphical analysis to help us understand if the 4^k k-mers' count distribution is in agreement with existing k-mer distributions in the literature. This ELN does not claim to address the question of "which distribution is most suitable to model k-mer counts" but instead offers the distribution to illustrate what background model might be appropriate for modeling efforts based on individual k-mer counts taken from k-mer spectra, like those generated by this software. Below is a simple histogram generated from retention of the intermediate SQLite3 database. The intermediate database is an exact replica of the k-mer database bgzf file (.kdb), and is used here because no bgzf parser exists for R. The k-mers table consists of all 4^k k-mer counts and is the basis for the histogram (Fig 1.).

```
library(DBI)

con <- dbConnect(RSQLite::SQLite(), "/home/matt/Projects/kdb2/test/data/tmp6viig_9k.sqlite3")
dbListTables(con)
```

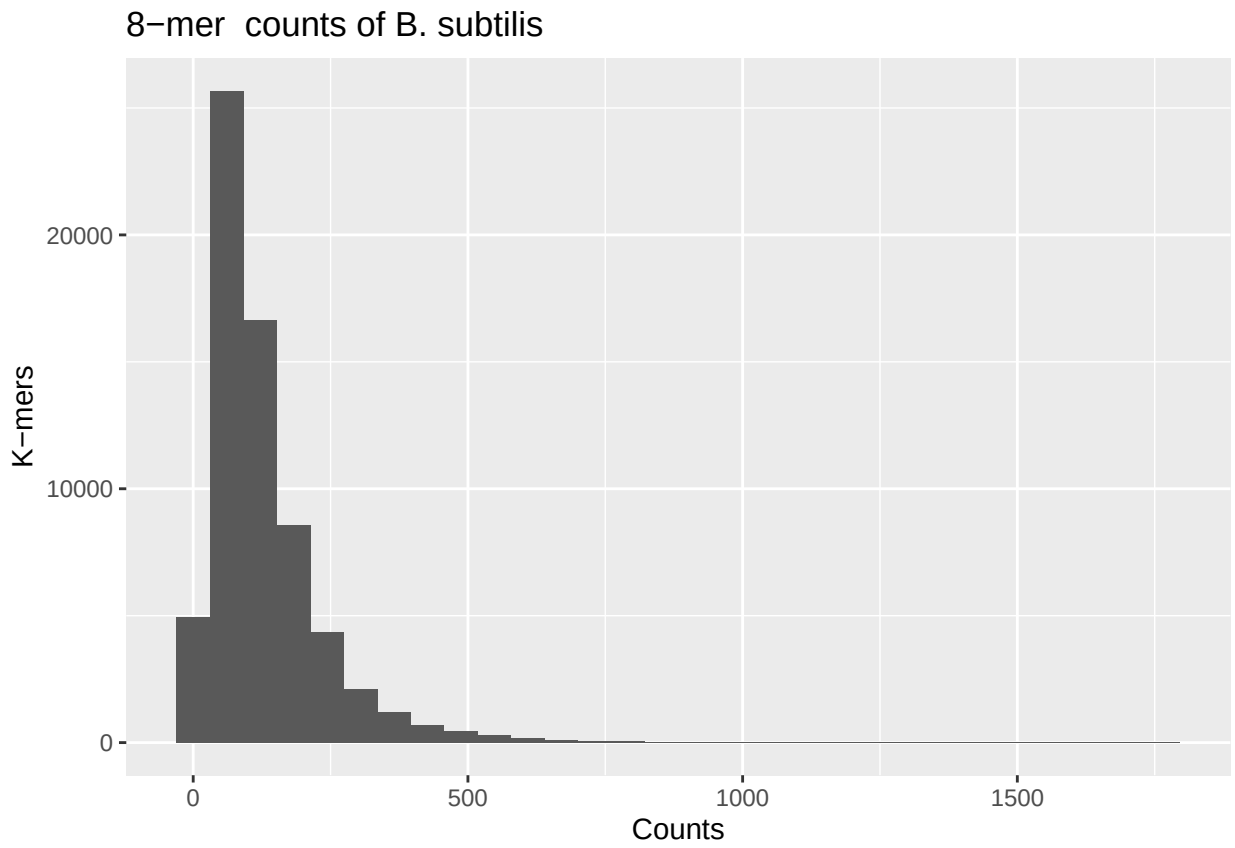

Figure 1: Primary histogram showing the discrete distribution of count data

```
## [1] "kmers" "reads"
```

```
kmers <- dbReadTable(con, "kmers")
dbDisconnect(con)

#summary(kmers$count) # Repetitive
ggplot(kmers) + geom_histogram(aes(x=count)) + ylab("K-mers") + xlab("Counts") + ggtitle("8-mer  counts
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

The k-mer profile is visualized here as a histogram to illustrate the distribution of counts that any k-mer may have. In addition to the summary statistics presented below, the mode is 44. Interestingly, 952 8-mers occur more than 500 times in the *C. acetobutylicum* genome. These sequences likely represent homopolymers, regions where misassemblies are likely to occur, and potentially repetitive regulatory motifs.

By generating the histogram and associated skewness-kurtosis analysis (Cullen, Frey, and Frey (1999) graph, Fig 2.), we can ask ourselves whether a Poisson model is appropriate, or if alternatives are more appropriate for modeling probabilities of counts of specific k-mer features associated with a genome. Though the negative-binomial seems more appropriate as suggested by the R package `fitdistrplus`, the kurtosis is still more extreme than would be required for an ideal fit in the NB model (Delignette-Muller, Dutang, and others (2015)).

```
descdist(kmers$count, discrete=T, boot=20)
```

```
## summary statistics
## ------
## min:  2   max:  1768
```

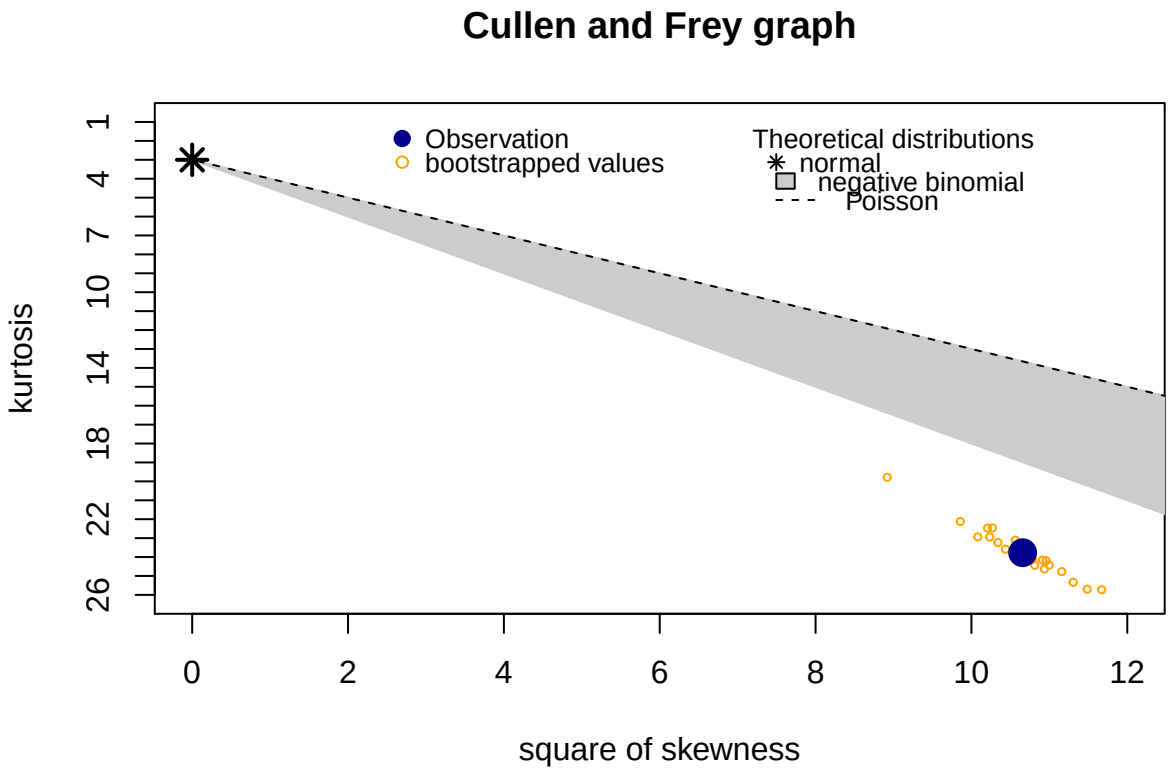

Figure 2: Cullen and Frey analysis of skewness and kurtosis suggesting best fit

```
## median:  98
## mean:  128.6499
## estimated sd:  113.5757
## estimated skewness:  3.264502
## estimated kurtosis:  23.77188
```

```
hist(kmers$count, prob=T, breaks=200, main="Histogram of k-mer counts", xlab="Kmer counts")
poisson_fit <- fitdist(kmers$count, 'pois')
poisson_fitD <- dpois(0:max(kmers$count), lambda=poisson_fit$estimate)
lines(poisson_fitD, lwd="3", col="blue")
nbinom_fit <- fitdist(kmers$count, 'nbinom')
nbinom_fitD <- dnbinom(0:max(kmers$count), size=nbinom_fit$estimate[1], mu=nbinom_fit$estimate[2])
lines(nbinom_fitD, lwd="3", col="red")
```

```
nbinom_fit$estimate
```

```
##       size         mu
##   1.860563 128.637010
```

To illustrate this further, Fig 3. shows us the two competing discrete distributions suggested by `fitdistrplus`. In blue, a Poisson model is shown to be a poor fit of the existing count data on the histogram from Fig 1. In contrast, a negative-binomial fit(red) with size = 1.8605634 and μ = 128.6370102 provides a reasonable fit for the dataset. The negative binomial model is a canonical discrete model often used in the RNA-Seq and k-mer literature to model count data like those obtained through second-generation sequencing experiments (Anders and Huber (2010), Daley and Smith (2013)).

In summary, the k-mer count distribution is best approximated by a negative-binomial model. The k-mer counts/frequencies and their distribution could be used to model sequence likelihoods via Markov probabilities.

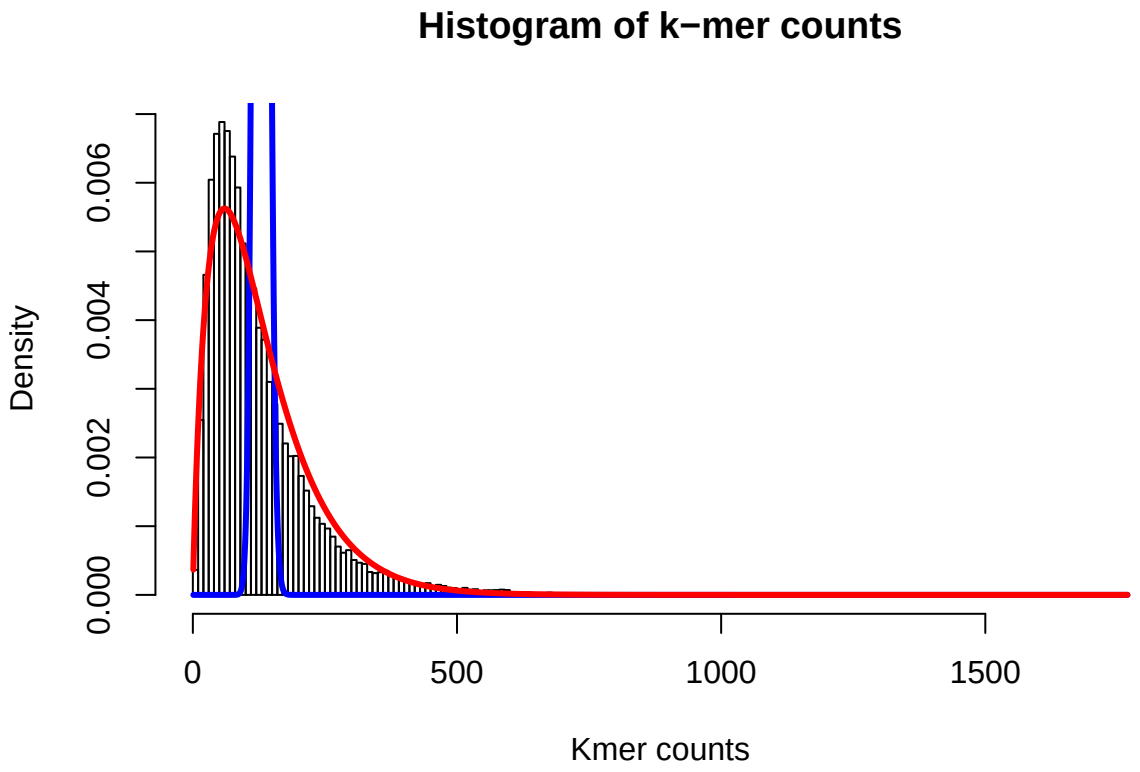

Figure 3: Alternate histogram showing Poisson model (blue) and Negative binomial fit (red)

Other applications of k-mer probabilities will be explored below.

Building k-mer profiles from fastq data sacrifices accuracy

A streaming inter-profile distance metric was implemented to assess similarities between profiles. The correlation coefficient was chosen as a self-normalizing metric to assess the differences between a profile generated from second-generation sequencing compared to its reference genome.

A dataset was generated from the *C. acetobutylicum* ATCC824 genome with `art_illumina` and sampled at various depths with `fastq-tools` to understand how well sequencing datasets could reflect the true k-mer profile that can be derived from its reference genome(Huang et al. (2012), Jones (2015)). As shown in Fig 4., increasing k tends to decrease similarity between the sequenced dataset and the reference genome, reflected by the correlation coefficient of the profiles.

It can be stated that whereever possible, k-mer profiles from reference genomes should be utilized for inferences. This could be an artifact introduced during the subsampling routine used when generating the simulated WGS dataset. However, the reference genome represents a condensed and unbiased estimate of the consensus assembly and should be used for the reference k-mer profile, when a reference is available. Additionally, in some cases sufficient sequencing depths are not available to accurately reflect the reference genome. In WGS and assembly applications, between 10-30x is advised to ensure even and minimum coverage across chromosomes. When such depths are not available, lower choices of k can be used to make basic, simple inferences about sequences with k-mer profiles.

An Amazon Linux instance was started in the US-East-1 availability zone.

Prepare Amazon Linux AMI for parameter sweep

```
sudo yum update
sudo yum install tmux git
sudo yum install python3
sudo yum groupinstall "Development Tools"
mkdir pckges && cd pckges
wget ftp://ftp.pcre.org/pub/pcre/pcre-8.44.zip
wget https://ftpmirror.gnu.org/parallel/parallel-20200222.tar.bz2
bzip2 -dc parallel-20200222.tar.bz2 | tar xvf -
git clone https://github.com/dcjones/fastq-tools
# unzip, ./configure, make, make install the above
cd
git clone https://github.com/MatthewRalston/kdb.git
cd kdb && sudo pip3 install -r requirements.txt
```

Parameter sweep

```
# Generate a representative and oversequenced sample with basic indel error profile
# Simulating HiSeq 2500x series reads, readlength of 150, 100x coverage
art_illumina -ss HS25 -i $FASTA -l 150 -f 10000 -o single_out

subsample(){
  s=$1
  k=$2
  #echo "Running in $(pwd)" >&2
  sample=$(mktemp tmp.XXXX)
  echo "Subsampling ${s} reads into ${k}-mer profile: ${sample}" >&2
  fastq-sample -n $s -s $RANDOM single_out.fq -o $sample
  /home/matt/Projects/kdb/bin/kdb profile -k $k $sample.fastq $sample.kdb
  corr=$(/home/matt/Projects/kdb/bin/kdb distance correlation $sample.kdb $FASTA.kdb)
  echo -e "${k}\t${s}\t${corr}" >> Cac_subsample_correlations.txt
  rm $sample $sample.fastq $sample.kdb
}
export -f subsample
parallel -j $CPU 'subsample {1} {2}' ::: $(seq 10000 40000 800000) ::: $(seq 8 12) ::: $(seq $CPU)
```

```
# R code to generate histogram from parameter sweep
corr_to_ref<-read.table("~/Projects/kdb/data/Cac_subsample_correlations.txt", header=F)
colnames(corr_to_ref) <- c("k", "Reads", "Correlation")

ggplot(corr_to_ref) + geom_point(aes(x=Reads, y=Correlation, colour=k))
```

Building k-mer profiles from fastq data takes more time

Streaming distance metric calculations

Given that some choices of k may be too extreme to load the vector into memory, a streaming implementation of each distance metric is necessary. Some metrics will necessarily require only one pass, such as the unnormalized Euclidean distance metric. Other metrics may require multiple passes over each file to generate arithmetic averages that may be used in indvidual components of the summation.

The first metric to implement was the correlation distance. This distance metric is described first to illustrate a single-pass metric with a streaming implementation, which is necessary to support large values of k. In early implementations I have written have the summation as essentially:

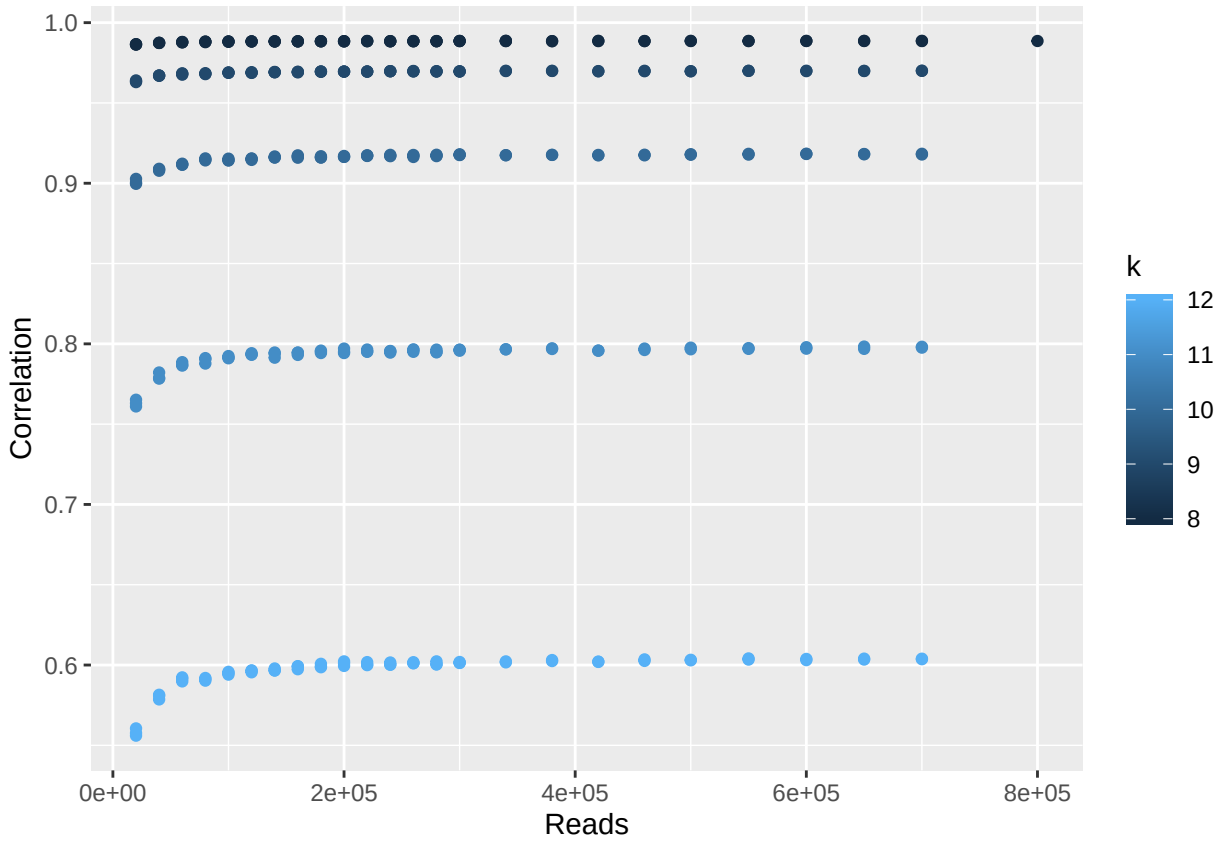

Figure 4: Correlation distance between subsampled profiles and reference genome

$r = ssxy/\sqrt{ssxx \cdot ssyy}$

$ssxy_i = (freq_x - mean_x)(freq_y - mean_y)$ $ssxx_i = (freq_x - mean_x)^2$ $ssyy_i = (freq_y - mean_y)^2$

For each element i of the 4^k k-mer profile, the residuals are calculated and added to a running sum, which is then used to calculate the final correlation coefficient as shown.

Supplemental

The following is incomplete

K-mer counting program runtime vs k

The most relevant question that needed answering early in the analytical process was how long the profile generation time could be expected to take between fastq datasets generated via `art_illumina` vs ideal k-mer profiles generated from fasta files of sequenced organisms.

From initial inspections, it seems like generating k-mer count profiles from BioPython SeqRecord objects streamed from fastq files requires considerable calculation time. This could be improved by using an alternative fastq parser library for Python that would read only the sequence information into memory. Additionally, a data fastq-fasta preprocessing step could lessen the amount of parsing and object creation and thus GC overhead experienced by the program, which may be a factor reducing efficiencies.

We explored how the run time varied with respect to the choice of k on a fixed number of 250k pairs of reads subsampled from the *C. acetobutylicum* RNA-Seq dataset SRR1774150. By increasing k, we were able to investigate the tradeoff between the spectrum's sensitivity to species specific k-mers (k) and the average runtime required for a specific sensitivity.

Additionally, the number of processing cores had a mild effect at reducing processing time.

Alternate histogram of 10-mer distributions

Bibliography

Anders, Simon, and Wolfgang Huber. 2010. "Differential Expression Analysis for Sequence Count Data." *Nature Precedings*. Nature Publishing Group, 1–1.

Cullen, Alison C, H Christopher Frey, and Christopher H Frey. 1999. *Probabilistic Techniques in Exposure Assessment: A Handbook for Dealing with Variability and Uncertainty in Models and Inputs*. Springer Science & Business Media.

Daley, Timothy, and Andrew D Smith. 2013. "Predicting the Molecular Complexity of Sequencing Libraries." *Nature Methods* 10 (4). Nature Publishing Group: 325.

Delignette-Muller, Marie Laure, Christophe Dutang, and others. 2015. "Fitdistrplus: An R Package for Fitting Distributions." *Journal of Statistical Software* 64 (4): 1–34.

Huang, Weichun, Leping Li, Jason R Myers, and Gabor T Marth. 2012. "ART: A Next-Generation Sequencing Read Simulator." *Bioinformatics* 28 (4). Oxford University Press: 593–94.

Jones, Daniel C. 2015. "Dcjones/Fastq-Tools." *GitHub*. https://github.com/dcjones/fastq-tools.