

NGS Quality Control

Projektmanagement im Softwarebereich - SeqAn 2013

Daniel Kersting Antje Oldenburg

Berlin, 24. April, 2013

Background NGS Quality Control

Vorbilder: FastQc FastX, PrinSeq

Functional Overview

Input fastq file

Data collection

Output tsc file

Formatting R for rendering graphs, html to display results

Statistical Information we plan to show

and how we collect the data for it

We grouped our quality metrics into groups, depending on whether it describes data

1. about the whole file
2. about all reads
3. all positions in all reads

This reflects where we collect the data:

- ▶ open `seqan::SequenceStream`
 - ▶ read `Record`
 - ▶ read nucleotide and quality and update data object

Simple statistics about all positions in all reads

and how we collect the data for it

Our data object holds:

- ▶ a 2-dimensional counter for score per position
- ▶ a 2-dimensional counter for Dna5 element per position

This will enable us to collect the following data:

- ▶ basic quality distribution data: median, mean, quantiles (10,25,75,90)
- ▶ distribution of [A,C,G,T]
- ▶ GC percent content
- ▶ N Content

Statistics about about all reads

and how we collect the data for it

Our data object holds:

- ▶ a 2-dimensional counter for score per position
- ▶ counter for encountered sequence lengths

This will enable us to collect the following data:

- ▶ mean qualities distribution
- ▶ sequence length distribution

Statistics about the whole file

and how we collect the data for it

Our data object holds:

- ▶ all program arguments, explicitly set and not
- ▶ a 2-dimensional counter for score per position
- ▶ a 2-dimensional counter for Dna5 element per position
- ▶ a counter for the number of records encountered

This will enable us to collect the following data:

- ▶ input filename and format
- ▶ which scoring system was used
- ▶ total number of sequences
- ▶ overall quality score average of all bases in all sequences
- ▶ overall GC percent
- ▶ overall N percent

K-mer content statistics

k-mer distribution

sequence duplication

and how we collect the data for it

Goal How can we achieve a

Problem memory and time constraints

Strategy for n reads contained in dataset

1. collect the first k reads and create a suffix array
2. starting from k , increment a counter if exact matches occur

Implementation

- ▶ `<StringSet<Dna5> as haystack`
- ▶ `appendValue` to `haystack` k times
- ▶ `Index<StringSet<Dna5>, IndexEsa<> >`
- ▶ `clear` and `find on`
`Finder<Index<StringSet<Dna5>, IndexEsa<>`
`> >`

Implications if used for 200.000 reads of length 100, needed memory for this would be ??? space complexity for holding SA is $O(n)$. Dna5 Element size in memory is 1Byte, also 20000 MBytes

Data output and formatting

Steps to create a visual summary

1. output tcv file with tabular data from our data
2. secondary app: script that does
 - 2.1 creates a static R script
 - 2.2 calls R script on tcv data to create png's
 - 2.3 creates a static html file that displays results

NICE-TO-HAVEs

what we want to add if time allows

- ▶ read for data formats: fastq compressed, bam, sam,
- ▶ sequence complexity statistics
- ▶ KNIME integration
- ▶ Galaxy integration

Milestones

- 1st week testing and implementation of a functionally minimal version that works through all steps
- 2nd week testing an implementation of all basic statistics (A) and k-mer content (D)
- 3rd week testing and implementation of sequence duplication (A) and output refinement (D)
- 4th week buffer for surprises, testing and implementation of NICE-TO-HAVE features (A+D)