Aberystwyth University

Computer Science and Statistics (GG34)

CS396: Minor Project

Application of Machine Learning Techniques to Next Generation Sequencing Quality Control

Author:
Sam Nicholls msn

Supervisor:
Dr. Amanda Clare afc

Draft

April 2, 2014

Declaration

I certify that except where indicated, all material in this thesis is the result of my own investigation and references used in preparation of the text have been cited. The work has not previously been submitted as part of any other assessed module, or submitted for any other degree or diploma.

Sam Nicholls
2014

Abstract

Over the past few years advances in genetic sequencing hardware have introduced the concept of massively parallel DNA sequencing; allowing potentially billions of chemical reactions to occur simultaneously, reducing both time and cost required to perform genetic analysis[3]. However, these "next-generation" processes are complex and open to error[2], thus quality control is an essential step to assure confidence in any downstream analyses performed.

During sample sequencing a large number of quality control metrics are generated to determine the quality of the reads from the sequencing hardware itself. At the Wellcome Trust Sanger Institute, the automated QC system currently relies on hard thresholds to make such quality control decisions with individual hard-coded values on particular metrics determining whether a lane has reached a level that requires a warning, or has exceeded the threshold and failed entirely. Whilst this does catch most of the very poor quality lanes, a large number of lanes are flagged for manual inspection at the warning level; a time consuming task which invites inefficiency and error.

In practise most of these manual decisions are based on inspecting a range of diagnostic plots which suggests that a machine learning classifier could potentially be trained on the combinations of quality control statistics available to make these conclusions without the need for much human intervention.

Contents

Contents iii

1 Introduction 1

1.1 Project Aims 1

1.1.1 Analysis of Current System 1

1.1.2 Identification of Properties that affect Downstream Analysis 2

2 Analysis of Current System 3

2.1 Introduction 3

2.1.1 Data Collection and Format 3

Appendix A samtools stats example output 5

Chapter 1

Introduction

Over the past few years advances in genetic sequencing hardware have introduced the concept of massively parallel DNA sequencing; allowing potentially billions of chemical reactions to occur simultaneously, reducing both time and cost required to perform genetic analysis[3]. However, these "next-generation" processes are complex and open to error[2], thus quality control is an essential step to assure confidence in any downstream analyses performed.

1.1 Project Aims

The project consists of two sub-projects;

- Analysis of a current quality control system in place
- Identification of quantifiable sample properties that affect downstream analysis

1.1.1 Analysis of Current System

With the support of the Wellcome Trust Sanger Institute in Cambridge, this project works with the Human Genetics Informatics team to investigate **auto_qc**, the institute's current automated quality control tool.

During genetic sequencing a large number of metrics are generated to determine the quality of the data read from the sequencing hardware itself. As part of the current vertebrate sequencing pipeline[1] at the institute, **auto_qc** is responsible for applying quality control to samples within the pipeline by comparing a modest subset of these metrics to simple hard-coded hard thresholds; determining whether a particular sample has reached a level that requires a warning, or has exceeded the threshold and failed entirely. Whilst this does catch most of the very poor quality outputs, a large number of samples are flagged for manual inspection at the warning level; a time consuming task which invites both inefficiency and error.

In practise most of these manual decisions are based on inspecting a range of diagnostic plots which suggests that a machine learning classifier could potentially be trained on the combinations of quality control statistics available to make these conclusions without the need for much human intervention.

The first part of the project aims to apply machine learning techniques to replicate the current **auto_qc** rule set by training a decision tree classifier on a large set of these quality metrics. The idea is to investigate whether these simple threshold based rules can be recovered from such data, or whether a new classifier would produce different rules entirely. During this analysis it is hoped the classifier may be able to identify currently unused quality metrics that improve labelling accuracy. An investigation on the possibility of aggregating or otherwise reducing the dimensions of some of the more detailed quality statistics to create new parameters will also be conducted.

The goal is to improve efficiency of quality control classification, whether by improving accuracy of pass and fail predictions over the current system or merely being able to provide additional information to a lab technician inspecting samples labelled with a warning to reduce arbitrary decisions.

1.1.2 Identification of Properties that affect Downstream Analysis

The other half of this project is motivated by the question "What *is* good and bad in terms of quality?"

To be able to classify samples as a pass or a fail with understanding, we need an idea of what actually constitutes a good or bad quality sample and must look at the effects quality has on analysis performed downstream from sequencing. An example of such is **variant calling** — the process of identifying differences between a DNA sample (such as your own) and a known reference sequence.

Given two high quality data sources where DNA sequences from individuals were identified in two different ways (one of which being next-generation sequencing) it would be possible to measure the difference between each corresponding pair. Using this, we could investigate the effect of leaving out part of the next-generation sample during the variant calling process. If we were to leave a part of a sample out of the variant calling pipeline would the variants found be more (or less) accurate than if it had been included? Would they agree more (or less) with the variants called after using the non next-generation sequencing method?

Having identified such sub-samples, can quality control metrics from the previous part be found in common? If so, such parameters would identify "good" or "bad" samples straight out of the machine! Samples that exhibit these quality variables will go on to improve or detriment analysis.

Chapter 2

Analysis of Current System

2.1 Introduction

2.1.1 Data Collection and Format

As part of the project I've been granted access to significant data sets at the Sanger Institute, unlocking quality control data for two of the largest studies currently undergoing analysis. A wide array of quality metrics are available for each and every lanelet that forms part of either of the two studies; totalling 13,455 files.

The files are created by **samtools stats** — part of a collection of widely used open-source utilities for post processing and manipulation of large alignments such as those produced by next-generation sequencers that are released under the umbrella name of "SAMtools" (Sequence Alignment and Map Tools). **samtools stats** collects statistics from sequence data files and produces key-value summary numbers as well as more complex tab delimited dataframes tabulating several metrics over time.

The output of **samtools stats** is then parsed by an in-house tool called **bamcheckr**, named so as **samtools stats** was once known as **bamcheck** and the tool is written in R. **bamcheckr** supplements the summary numbers section of the **samtools stats** output with additional metrics that are later used by **auto_qc** for classification. This process does not change the file other than adding a few additional key-value pairs in the summary numbers section. An truncated example of a "bamcheckr'd" file can be found in Appendix A.

References

[1] vr-pipe, a generic pipeline system [Github]. [Online]. Available: https://github.com/wtsi-hgi/vr-pipe/

[2] M. Kircher, U. Stenzel and J. Kelso, "Improved base calling for the Illumina Genome Analyzer using machine learning strategies," *Genome Biology*, vol. 10, no. 8, p. R83, 2009.
Useful introduction to relevant Illumina hardware and the errors that can occur during sequencing.

[3] T. Strachan and A. Read, *Human Molecular Genetics*, 4th ed. Garland Science, 2011, pp. 214–254.
A concise introduction to the processes involved in massively parallel DNA sequencing.

Appendix A

samtools stats example output

```
# Summary Numbers. Use ‘grep ^SN | cut -f 2-‘ to extract this part.
SN	raw total sequences:	41400090
SN	filtered sequences:	0
SN	sequences:	41400090
SN	is paired:	1
SN	is sorted:	1
SN	1st fragments:	20700045
SN	last fragments:	20700045
SN	reads mapped:	41291484
SN	reads unmapped:	108606
SN	reads unpaired:	60000
SN	reads paired:	41231484
SN	reads duplicated:	5756822
SN	reads MQ0:	1038644
SN	reads QC failed:	0
SN	non-primary alignments:	0
SN	total length:	3105006750
SN	bases mapped:	3096861300
SN	bases mapped (cigar):	3090885143
SN	bases trimmed:	0
SN	bases duplicated:	431761650
SN	mismatches:	9107833
SN	error rate:	0.002946675
SN	average length:	75
SN	maximum length:	75
SN	average quality:	36
SN	insert size average:	178.7
SN	insert size standard deviation:	44.1
SN	inward oriented pairs:	20577242
SN	outward oriented pairs:	3140
SN	pairs with other orientation:	3711
SN	pairs on different chromosomes:	31535
SN	fwd percent insertions above baseline:	1.43135383851191
SN	fwd percent insertions below baseline:	0.686265539012562
SN	fwd percent deletions above baseline:	1.38326380878871
SN	fwd percent deletions below baseline:	0.44923551909251
SN	rev percent insertions above baseline:	1.08264446659241
SN	rev percent insertions below baseline:	0.457290262062496
SN	rev percent deletions above baseline:	1.15931214598243
SN	rev percent deletions below baseline:	0.413119424753248
SN	contiguous cycle dropoff count:	36
SN	fwd.percent.insertions.above.baseline:	1.43135383851191
```

```
SN	fwd.percent.insertions.below.baseline:	0.686265539012562
SN	fwd.percent.deletions.above.baseline:	1.38326380878871
SN	fwd.percent.deletions.below.baseline:	0.44923551909251
SN	rev.percent.insertions.above.baseline:	1.08264446659241
SN	rev.percent.insertions.below.baseline:	0.457290262062496
SN	rev.percent.deletions.above.baseline:	1.15931214598243
SN	rev.percent.deletions.below.baseline:	0.413119424753248
SN	quality.dropoff.fwd.high.iqr.start.read.cycle:	0
SN	quality.dropoff.fwd.high.iqr.end.read.cycle:	0
SN	quality.dropoff.fwd.high.iqr.max.contiguous.read.cycles:	0
SN	quality.dropoff.fwd.mean.runmed.decline.start.read.cycle:	20
SN	quality.dropoff.fwd.mean.runmed.decline.end.read.cycle:	51
SN	quality.dropoff.fwd.mean.runmed.decline.max.contiguous.read.cycles:	32
SN	quality.dropoff.fwd.mean.runmed.decline.high.value:	36.9775883578997
SN	quality.dropoff.fwd.mean.runmed.decline.low.value:	36.301749247405
SN	quality.dropoff.rev.high.iqr.start.read.cycle:	0
SN	quality.dropoff.rev.high.iqr.end.read.cycle:	0
SN	quality.dropoff.rev.high.iqr.max.contiguous.read.cycles:	0
SN	quality.dropoff.rev.mean.runmed.decline.start.read.cycle:	18
SN	quality.dropoff.rev.mean.runmed.decline.end.read.cycle:	56
SN	quality.dropoff.rev.mean.runmed.decline.max.contiguous.read.cycles:	39
SN	quality.dropoff.rev.mean.runmed.decline.high.value:	36.1517621338504
SN	quality.dropoff.rev.mean.runmed.decline.low.value:	35.3152133727245
SN	quality.dropoff.high.iqr.threshold:	10
SN	quality.dropoff.runmed.k:	25
SN	quality.dropoff.ignore.edge.cycles:	3
SN	A.percent.mean.above.baseline:	0.0991164444444441
SN	C.percent.mean.above.baseline:	0.127379555555556
SN	G.percent.mean.above.baseline:	0.0603679999999997
SN	T.percent.mean.above.baseline:	0.0868000000000005
SN	A.percent.mean.below.baseline:	0.0991164444444451
SN	C.percent.mean.below.baseline:	0.127379555555555
SN	G.percent.mean.below.baseline:	0.0603680000000002
SN	T.percent.mean.below.baseline:	0.0867999999999993
SN	A.percent.max.above.baseline:	0.601733333333332
SN	C.percent.max.above.baseline:	0.394266666666667
SN	G.percent.max.above.baseline:	0.2956
SN	T.percent.max.above.baseline:	0.768000000000001
SN	A.percent.max.below.baseline:	0.318266666666666
SN	C.percent.max.below.baseline:	0.825733333333332
SN	G.percent.max.below.baseline:	0.554400000000001
SN	T.percent.max.below.baseline:	0.251999999999999
SN	A.percent.max.baseline.deviation:	0.601733333333332
SN	C.percent.max.baseline.deviation:	0.825733333333332
SN	G.percent.max.baseline.deviation:	0.554400000000001
SN	T.percent.max.baseline.deviation:	0.768000000000001
SN	A.percent.total.mean.baseline.deviation:	0.198232888888889
SN	C.percent.total.mean.baseline.deviation:	0.254759111111111
SN	G.percent.total.mean.baseline.deviation:	0.120736
SN	T.percent.total.mean.baseline.deviation:	0.1736
# First Fragment Qualitites. Use ‘grep ^FFQ | cut -f 2-‘ to extract this part.
# Columns correspond to qualities and rows to cycles. First column is the cycle number.
FFQ	1	8968	3619	9863	747	5094	0	6642	1609	4673	4208	2
FFQ	2	21676	0	0	0	0	0	0	0	43	1885	0	0	0
FFQ	3	7	0	177	0	0	0	0	0	0	0	0	0	0
FFQ	4	0	0	0	0	65	0	0	0	272	0	0	0	14277
FFQ	5	1917	173	1249	0	1890	0	0	0	0	10874	0	0
[...]
FFQ	72	4098	0	0	4806	0	0	0	65507	0	0	0	0
FFQ	73	3894	2	0	0	0	0	4931	53483	0	0	0	0
FFQ	74	3697	39	0	919	4933	0	0	56866	1524	0	0	2
FFQ	75	4542	0	0	0	0	0	4634	77822	0	0	0	0
FFQ	76	0	0	0	0	0	0	0	0	0	0	0	0	0
```

```
# Last Fragment Qualitites. Use ‘grep ^LFQ | cut -f 2-‘ to extract this part.
# Columns correspond to qualities and rows to cycles. First column is the cycle number.
LFQ	1	8869	0	0	0	0	0	63	0	0	1156	616	173
LFQ	2	3300	0	0	0	0	0	0	0	0	0	389	0	0
LFQ	3	6816	0	0	0	573	0	83	0	7011	1171	107134	0
LFQ	4	5492	0	0	13	0	66	730	708	8134	2422	84052
LFQ	5	3512	0	0	0	1023	185	0	8653	1995	0	115559
[...]
LFQ	72	5135	166	0	0	2872	13643	0	59649	4249	11351	346
LFQ	73	6025	229	0	86	1042	13417	0	66093	3741	8151	354
LFQ	74	5980	3	91	0	0	0	1340	9696	72939	4924	304090
LFQ	75	4314	0	0	168	0	848	8591	0	70358	3827	352180
LFQ	76	0	0	0	0	0	0	0	0	0	0	0	0	0
# Mismatches per cycle and quality. Use ‘grep ^MPC | cut -f 2-‘ to extract this part.
# Columns correspond to qualities, rows to cycles. First column is the cycle number, second
# is the number of N’s and the rest is the number of mismatches
MPC	1	14078	0	2594	6777	416	1919	0	2222	352	987	537
MPC	2	21407	0	0	0	0	0	0	0	0	5	223	19	0
MPC	3	3205	0	0	37	0	43	0	12	0	691	71	6984
MPC	4	1774	0	0	0	2	29	4	65	73	863	192	6749
MPC	5	1913	0	94	885	0	969	23	0	959	213	1203	84
[...]
MPC	72	361	0	13	0	573	276	934	0	16376	426	1066
MPC	73	1005	0	11	0	4	79	777	539	15025	363	699
MPC	74	779	0	3	0	131	440	0	93	7485	6589	387	24
MPC	75	136	0	0	0	3	0	47	704	9302	5886	260	2650
MPC	76	0	0	0	0	0	0	0	0	0	0	0	0	0
# GC Content of first fragments. Use ‘grep ^GCF | cut -f 2-‘ to extract this part.
GCF	0.5	56
GCF	1.76	60
GCF	3.02	126
GCF	4.27	212
GCF	5.78	347
[...]
GCF	93.72	378
GCF	95.23	186
GCF	96.48	87
GCF	97.74	55
GCF	99.25	17
# GC Content of last fragments. Use ‘grep ^GCL | cut -f 2-‘ to extract this part.
GCL	0.5	118
GCL	1.76	175
GCL	3.02	230
GCL	4.27	354
GCL	5.78	525
[...]
GCL	93.72	613
GCL	95.23	430
GCL	96.48	274
GCL	97.74	185
GCL	99.25	110
# ACGT content per cycle. Use ‘grep ^GCC | cut -f 2-‘ to extract this part. The columns are: cycle, and A,C,G,T counts [%]
GCC	1	26.93	23.09	22.77	27.2
GCC	2	26.78	23.24	22.97	27.02
GCC	3	26.46	23.59	23.3	26.66
GCC	4	26.29	23.79	23.45	26.46
GCC	5	26.47	23.61	23.3	26.62
[...]
GCC	70	26.09	24.26	23.45	26.2
GCC	71	26.07	24.25	23.46	26.22
GCC	72	26.04	24.27	23.49	26.2
GCC	73	26.07	24.25	23.47	26.22
GCC	74	26.08	24.24	23.45	26.23
```

```
GCC     75      26.01       24.31       23.51       26.18
# Insert sizes. Use ‘grep ^IS | cut -f 2-‘ to extract this part. The columns are: pairs total, inward oriented pairs, outward oriented pa
IS      0       10      0       1       9
IS      1       3       0       3       0
IS      2       4       0       4       0
IS      3       5       0       5       0
IS      4       2       0       2       0
IS      5       3       0       3       0
[...]
IS      110     33952       33952       0       0
IS      111     38433       38433       0       0
IS      112     43373       43370       0       3
IS      113     48160       48159       0       1
IS      114     53175       53171       0       4
IS      115     59504       59502       0       2
IS      116     64668       64668       0       0
IS      117     71107       71105       0       2
IS      118     77157       77156       0       1
IS      119     84044       84044       0       0
IS      120     90116       90110       3       3
[...]
IS      327     6546        6546        0       0
IS      328     6483        6483        0       0
IS      329     6201        6201        0       0
IS      330     6228        6228        0       0
IS      331     5852        5852        0       0
# Read lengths. Use ‘grep ^RL | cut -f 2-‘ to extract this part. The columns are: read length, count
RL      75      41400090
# Indel distribution. Use ‘grep ^ID | cut -f 2-‘ to extract this part. The columns are: length, number of insertions, number of deletions
ID      1       128650      183418
ID      2       26409       39770
ID      3       10213       16046
ID      4       7756        11444
ID      5       1746        3455
[...]
ID      35      0       8
ID      36      0       1
ID      37      0       1
ID      38      0       1
ID      40      0       2
# Indels per cycle. Use ‘grep ^IC | cut -f 2-‘ to extract this part. The columns are: cycle, number of insertions (fwd), .. (rev) , numbe
IC      1       0       0       105     97
IC      2       24      15      150     179
IC      3       129     138     441     509
IC      4       253     310     623     829
IC      5       557     724     786     1164
[...]
IC      70      571     710     638     761
IC      71      350     428     309     434
IC      72      154     150     38      45
IC      73      60      61      15      23
IC      74      20      19      11      12
# Coverage distribution. Use ‘grep ^COV | cut -f 2-‘ to extract this part.
COV     [1-1]       1       332980694
COV     [2-2]       2       105004580
COV     [3-3]       3       29112182
COV     [4-4]       4       13415014
COV     [5-5]       5       6716815
[...]
COV     [996-996]       996     2
COV     [997-997]       997     2
COV     [998-998]       998     2
COV     [1000-1000]     1000        4
COV     [1000<]     1000     116
# GC-depth. Use ‘grep ^GCD | cut -f 2-‘ to extract this part. The columns are: GC%, unique sequence percentiles, 10th, 25th, 50th, 75th a
GCD     0       0.001     0       0       0       0       0
GCD     0.4      0.002     0.101     0.101     0.101     0.101     0.101
GCD     19      0.003     0.049     0.049     0.049     0.049     0.049
GCD     20      0.004     0.06      0.06      0.06      0.06      0.06
GCD     21      0.004     0.045     0.045     0.045     0.045     0.045
[...]
GCD     66      99.99     0.244     2.693     6.746     11.794     15.885
GCD     67      99.994     1.279     1.279     4.305     9.667     11.483
GCD     68      99.997     4.148     4.148     4.463     5.741     7.354
GCD     69      99.999     0.499     0.499     0.499     1.935     1.935
GCD     72      100     0.476     0.476     0.476     1.219     1.219
```