

Using savR

R. Brent Calder

May 2, 2019

```
> library(savR)

> fc <- savR(system.file("extdata", "MiSeq", package="savR"))

> fc
savProject instance with 1 lanes, 82 total cycles, and 2 sequence reads ( 1 sequencing and 1 index)
Default naming convention.
With InterOp data for: savCorrectedIntensityFormat (correctedIntensities)
savQualityFormat (qualityMetrics)
savTileFormat (tileMetrics)
savExtractionFormat (extractionMetrics)

> pfBoxplot(fc)
```

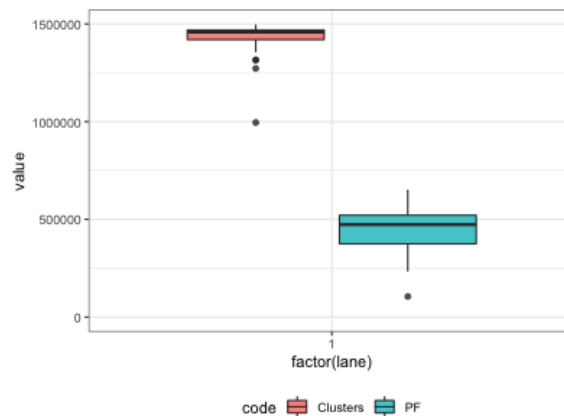


Figure 1: Boxplot of total vs. PF clusters

Introduction

The Illumina Sequence Analysis Viewer (SAV) is a Windows application provided by Illumina that presents graphs made in real time from data collected over the course of basecalling. This data was previously also made available in HTML format for inspection after the run; however, it is now preserved in binary format and not simply parsed by users who wish to perform automated quality assessment. Here is presented *savR*, an R package to parse the binary output, generate QC assessment plots and make the data available to users of Illumina sequencing instruments. For more information about Illumina SAV, please consult the Illumina iCom website and the Sequencing Analysis Viewer User's Guide, available *online*.

Description

The `savR` function is passed a path to an Illumina HiSeq or MiSeq run, and returns a `savProject` object, containing the parsed data. Accessor methods are available for information in the `RunInfo.xml` file as well as the parsed SAV Metrics files. These include corrected intensities, quality metrics, tile metrics, and extraction metrics. The *savR* package comes with an example MiSeq data set which can be loaded thusly:

```
> fc <- savR(system.file("extdata", "MiSeq", package="savR"))
```

RunInfo.xml

The `RunInfo.xml` file is parsed and stored in the slots of the `savProject` object. There are accessor methods for the project's location, reads, number of "ends" or directions, the run ID, the number of cycles, and a description of the flowcellLayout.

```
> directions(fc)
```

```
[1] 1
```

```
> reads(fc)
```

```
[[1]]
```

```
An object of class "illuminaRead"
```

```
Slot "number":
```

```
[1] 1
```

```
Slot "cycles":
```

```
[1] 76
```

```
Slot "index":
```

```
[1] FALSE
```

```
[[2]]
```

```
An object of class "illuminaRead"
```

```
Slot "number":
```

```
[1] 2
```

```
Slot "cycles":
```

```
[1] 6
```

```
Slot "index":
```

```
[1] TRUE
```

```
> cycles(fc)
```

```
[1] 82
```

```
> flowcellLayout(fc)
```

```
An object of class "illuminaFlowCellLayout"
```

```
Slot "lanecount":
```

```
[1] 1
```

```
Slot "surfacecount":
```

```
[1] 2
```

```
Slot "swathcount":
```

```
[1] 1
```

```
Slot "tilecount":
```

```
[1] 19
```

```
Slot "sectionperlane":
```

```
[1] NA
```

```
Slot "lanepersection":
```

```
[1] NA
```

```
Slot "tilenaminingconvention":
```

```
[1] ""
```

Corrected intensities

Corrected intensity metrics (obtained from `CorrectedIntMetricsOut.bin`) can be inspected by the `correctedIntensities` accessor method:

```
> head(correctedIntensities(fc), n=1)
```

| | lane | tile | cycle | avg_intensity | avg_cor_A | avg_cor_C | avg_cor_G | avg_cor_T | |
|---|-------|-------|--------|------------------|------------------|------------------|------------------|-----------|--|
| 2 | 1 | 1101 | 1 | 80 | 72 | 17 | 116 | 101 | |
| | | | | avg_cor_called_A | avg_cor_called_C | avg_cor_called_G | avg_cor_called_T | num_none | |
| 2 | | | 212 | 266 | 283 | 277 | 339 | | |
| | num_A | num_C | num_G | num_T | sig_noise | | | | |
| 2 | 97572 | 17051 | 136607 | 127150 | 6.704378 | | | | |

This is a `data.frame` of intensity metrics; one line for each set of lane, tile and cycle measurements. Reported statistics include average intensity, corrected intensity (for cross-talk between bases and phasing/pre-phasing), called corrected intensities, number of called bases and signal to noise ratio. There are methods which act upon `savProject` objects to produce QC plots, for example `plotIntensity` to assess signal intensity for each channel as in figure 2.

```
> plotIntensity(fc)
```

Quality Metrics

The quality metrics (`QMetricsOut.bin`) file contains per-lane/tile/cycle metrics for the number of clusters with quality at each PHRED value from 1-50.

```
> head(qualityMetrics(fc), n=1)
```

| | lane | tile | cycle | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 | Q11 | Q12 | Q13 | Q14 | Q15 | Q16 | Q17 |
|---|--------|------|-------|------|-----|------|------|-----|-----|-------|-----|-----|-----|-------|-------|-------|-----|-----|-----|-----|
| 1 | 1 | 1101 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17087 | 0 | 0 | 0 | 0 | 0 |
| | | | | Q18 | Q19 | Q20 | Q21 | Q22 | Q23 | Q24 | Q25 | Q26 | Q27 | Q28 | Q29 | Q30 | Q31 | Q32 | Q33 | |
| 1 | 0 | 0 | 0 | 6562 | 0 | 5323 | 5901 | 0 | 0 | 13362 | 0 | 0 | 250 | 22276 | 13477 | 12757 | | | | |
| | | | | Q34 | Q35 | Q36 | Q37 | Q38 | Q39 | Q40 | Q41 | Q42 | Q43 | Q44 | Q45 | Q46 | Q47 | Q48 | Q49 | Q50 |
| 1 | 281724 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
> qualityHeatmap(fc,1,1)
```

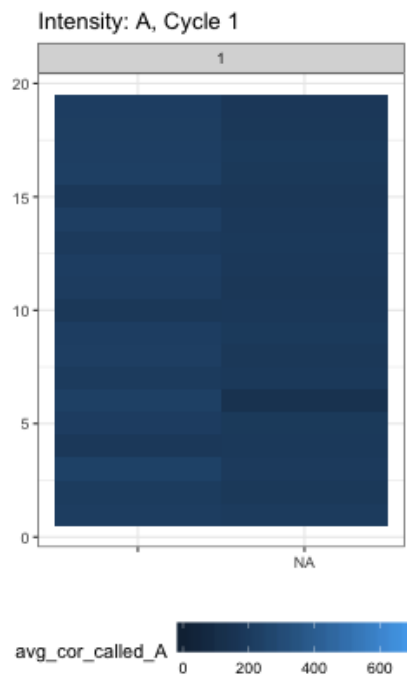


Figure 2: Corrected intensity plot: cycle 1, base “A”.

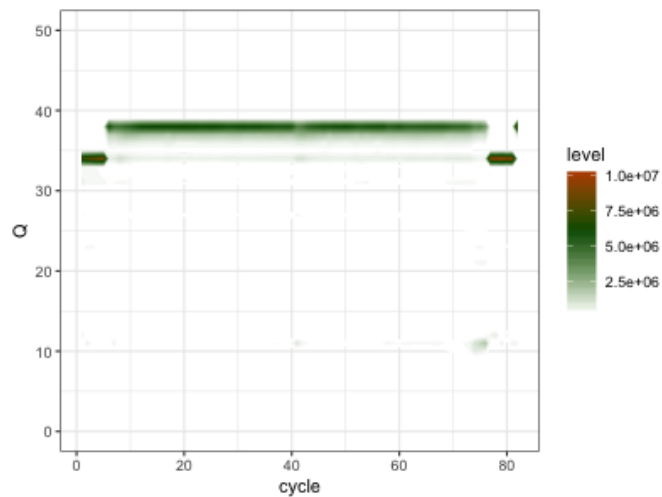


Figure 3: Quality heatmap: lane 1, read 1.

Tile Metrics

The tile metrics (`TileMetricsOut.bin`) file contains coded information about per-lane/cycle/tile cluster density, pass-filter clusters, phasing and pre-phasing data. Consult the `tileMetrics` help page for more information.

```
> head(tileMetrics(fc), n=4)
  lane tile code  value
39    1 1101  100 1271891.8
41    1 1102  100 1455525.0
```

```
43    1 1103  100 995404.6
45    1 1104  100 1463691.8
```

Extraction Metrics

The extraction metrics (`ExtractionMetricsOut.bin`) file contains per-lane/cycle/tile information about per-base FWHM (full width pixel size of clusters at half maximum) and 90th %-ile intensity of signal intensity.

```
> head(extractionMetrics(fc), n=1)
  lane tile cycle  FWHM_A  FWHM_C FWHM_G  FWHM_T int_A int_C int_G int_T
2    1 1101     1 2.235387 2.308783 1.86132 2.174398  180  326  357  400
```

Coda

There is a convenience function (`buildReports`), which partially reconstructs the Illumina reports folder that was previously generated by the Illumina instrument software and which was superseded by SAV and InterOp files.