ChronQC Documentation

Release 1.0.2

POLARIS-BioIT@GIS

Jan 02, 2018

ChronQC

- Free software: MIT license
- Documentation: http://chronqc.readthedocs.io/en/latest/.

ChronQC is a quality control (QC) tracking system for clinical implementation of next-generation sequencing (NGS). ChronQC generates time series plots for various QC metrics, which allows comparison of the current run to historical runs. ChronQC has multiple features for tracking QC data including Westgard rules for clinical validity, laboratory-defined thresholds, and historical observations within a specified period. Users can record their notes and corrective actions directly onto the plots for long-term recordkeeping.

CHAPTER 1

Features

- Suited for different assays in a clinical laboratory
- Generates interactive time series plots for various metrics
- Records users' notes and corrective actions onto the graphs to facilitate long-term recordkeeping
- Provides high level of customization: works with local databases and generates different chart types
- Leverages existing standard tools such as MultiQC

CHAPTER 2

Example live ChronQC report

https://nilesh-tawari.github.io/chronqc

2.1 Introduction

ChronQC is a quality control (QC) tracking system for clinical implementation of next-generation sequencing (NGS). ChronQC generates time series plots for various QC metrics, which allows comparison of the current run to historical runs. ChronQC has multiple features for tracking QC data including Westgard rules for clinical validity, laboratory-defined thresholds, and historical observations within a specified period. Users can record their notes and corrective actions directly onto the plots for long-term recordkeeping.

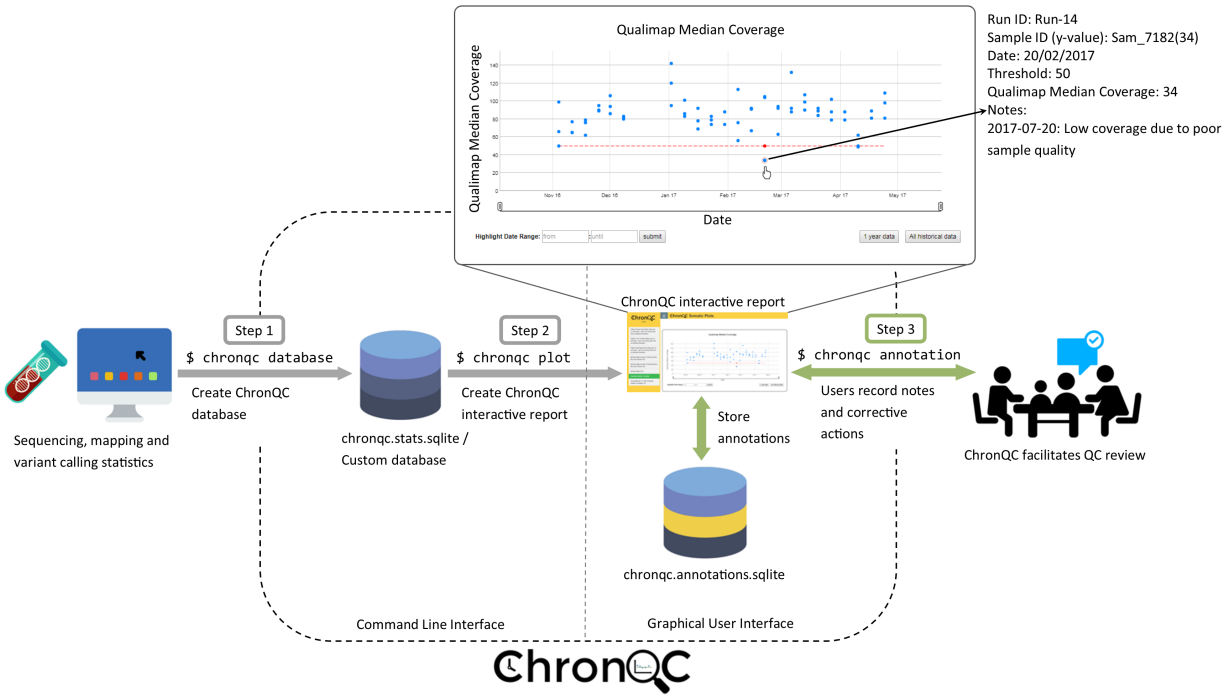
2.1.1 Features

- Suited for different assays in a clinical laboratory
- Generates interactive time series plots for various metrics
- Records users' notes and corrective actions onto the graphs to facilitate long-term recordkeeping
- Provides high level of customization: works with local databases and generates different chart types
- Leverages existing standard tools such as MultiQC

2.1.2 ChronQC Workflow

ChronQC has two components: a command line interface compatible with NGS sequencing machines and a graphical user interface compatible with the clinical environment. HTML plots display metrics for each run or sample. Annotations are displayed on the right side of

the plot and are stored in the chronqc.annotations.sqlite database for long-term recordkeeping.



2.1.3 Example live ChronQC report

https://nilesh-tawari.github.io/chronqc

2.2 Install ChronQC

2.2.1 Requirement

ChronQC is implemented in Python (tested with v2.7 / v3.5 / v3.6) and runs on all common operating systems (Windows, Linux and Mac OS X).

2.2.2 Installation

You can install ChronQC from PyPI using pip as follows:

```
pip install chronqc
```

Alternatively, you can install from GitHub:

```
git clone https://github.com/nilesh-tawari/ChronQC_dev.git
cd ChronQC
pip install -r requirements.txt
pip install --editable .
```

If you would like the development version instead, the command is:

```
pip install --upgrade --force-reinstall git+https://github.com/nilesh-tawari/ChronQC_
↪dev.git
```

Alternatively, ChronQC can also be used as Docker.

```
#Pull docker image
docker pull nileshtawari/chronqc:chronqc_1.0.4

#Run docker
docker run -it --rm -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix nileshtawari/
↪chronqc:chronqc_1.0.4

#Test ChronQC using example data
cd /home/ChronQC
```

Then follow instructions given in getting started.

```
#(optional)Run docker with mounted folder
docker run -it --rm -e DISPLAY=$DISPLAY -u $(id -u) -v /tmp/.X11-unix:/tmp/.X11-unix -
↪v /your_local_home_directory/your_data_directory:/data2 nileshtawari/
↪chronqc:chronqc_1.0.4
```

2.3 Getting started

Contents

2.4 Generating ChronQC plots

ChronQC plots can be generated,

2.4.1 Using a custom SQLite database

This example demonstrates generating ChronQC plots from a custom database,

```
cd examples/custom_db_example
```

Run following command to generate interactive plots in HTML,

```
chronqc plot chronqc_custom_db.sqlite Somatic config.json
```

The types of created plots and their properties are specified in "config.json" file. For details on creating the config . file visit documentation. Interactive html report is created under `chronqc_output` directory

2.4.2 Using Non-NGS custom SQLite database

This example demonstrates application of ChronQC time-series monitoring in non-NGS assay. This example generates plots for ENTROGEN PCR-based assay data. The example below utilizes 2 types of plots:

- A stacked plot that shows the total number of mutations present in samples ran with the PCR-based assay very month. Each bar indicates the total occurences of variants of mutants (KRAS, BRAF, WT) per month.
- Ct values of positive control are monitored using the absolute threshold plot which visualizes FAM / VIC Ct values with validated upper and lower thresholds. Stable Ct values in positive controls indicates that each run is valid and the reagents and protocol are stable.

```
cd examples/custom_db_example
```

Run following command to generate interactive plots in HTML,

```
chronqc plot chronqc_custom_PCR_db.sqlite ENTROGEN PCR_config_demo.json
```

The types of created plots and their properties are specified in "PCR_config_demo.json" file. For details on creating the config . file visit documentation. Interactive html report is created under `chronqc_output` directory.

2.4.3 Using the output of MultiQC

1. Creating ChronQC database and plots

For creating ChronQC database and plots, see the example below. This example demonstrates generating a ChronQC database and creating ChronQC graphs using the database.

```
cd examples/multiqc_example_1
```

Step 1: Create a ChronQC database

```
chronqc database --create --run-date-info run_date_info.csv -o . multiqc_data/multiqc_
↪general_stats.txt SOMATIC
```

A sqlite database `chronqc.stats.sqlite`, `chronqc.default.json` and `chronqc.stats.cols.txt` file are created in `chronqc_db` folder under the `.` (current) directory.

Step 2: Create ChronQC plots

The types of created plots and their properties are specified in JSON file. Plots can be generated using the default `chronqc.default.json` file,

```
chronqc plot -o . chronqc_db/chronqc.stats.sqlite SOMATIC chronqc_db/chronqc.default.
↪json
```

Interactive html report is created in `chronqc_output` under the `.` (current) directory.

Customizing the JSON

It is strongly adviced to create custom JSON file, based on the laboratories assay tracking strategy. A sample of custom JSON is included in the folder. *chronqc.stats.cols.txt* file contains column names present in the database, and can be used to create the config file. For details on creating the config . file visit documentation.

```
chronqc plot -o . chronqc_db/chronqc.stats.sqlite SOMATIC sample.json
```

Interactive html report is created in `chronqc_output` under the `.` (current) directory.

2. Creating and updating ChronQC database and creating plots

For creating, updating ChronQC database and plots, see the example below. This example demonstrates generating a ChronQC database, updating the generated database with new data and creating ChronQC graphs using the database.

```
cd examples/multiqc_example_2
```

Step 1: Create a ChronQC database

```
chronqc database --create --run-date-info year_2016/run_date_info.csv -o . year_2016/
↪multiqc_data/multiqc_general_stats.txt Germline
```

A sqlite database `chronqc.stats.sqlite`, `chronqc.default.json` and *chronqc.stats.cols.txt* file are created in `chronqc_db` folder under the `.` (current) directory. A default JSON file named `chronqc.default.json` is also created in chronqc_db under the `.` (current) directory.

Step 2: Update existing ChronQC database

```
chronqc database --update --db chronqc_db/chronqc.stats.sqlite --run-date-info year_
↪2017/run_date_info.csv year_2017/multiqc_data/multiqc_general_stats.txt Germline
```

Step 3: Create ChronQC plots

The types of created plots and their properties are specified in JSON file. Plots can be generated using the default `chronqc.default.json` file,

```
chronqc plot -o . chronqc_db/chronqc.stats.sqlite Germline chronqc_db/chronqc.default.
↪json
```

Interactive html report is created in `chronqc_output` under the `.` (current) directory.

Customizing the JSON

It is strongly adviced to create custom JSON file, based on the laboratories assay tracking strategy. A sample of custom JSON is included in the folder. *chronqc.stats.cols.txt* file contains column names present in the database, and can be used to create the config file. For details on creating the config . file visit documentation.

```
chronqc plot -o . chronqc_db/chronqc.stats.sqlite Germline sample.json
```

Interactive html report is created in `chronqc_output` under the `.` (current) directory.

2.5 Using ChronQC plots

ChronQC is designed to be interactive. ChronQC plots can be adjusted to a time period and are zoomable. Mousing over a point displays its associated data such as run ID, sample IDs, and corresponding values. To use the annotation feature of ChronQC plots, start the annotation database connectivity by using *chronqc annotation* command. Default port used for annotation server is 8000, this can be customized by using –port option. Then open the ChronQC output html in a recent browser (tested on: Google Chrome and Mozilla Firefox). Users can record notes and corrective

actions on the plots by clicking on a point or selecting a date. User notes and corrective actions are stored for long-term recordkeeping in the SQLite ChronQC annotations database. The plots are interlinked so that when an individual point or date is annotated in one graph, the same annotation appears on other graphs. By using the ChronQC report with the ChronQC annotations data-base (by starting the annotation server with "chronqc annotation" command), users can see the notes that have been recorded previously.

2.6 ChronQC config files

- `chronqc.stats.cols.txt` file generated during the ChronQC stats database creation can be used to get column names present in the database.
- Using the statistics database and a configuration file (JSON), ChronQC generates time series plots for various metrics to create an interactive, self-contained HTML file.
- Plots should be mentioned simultaneously in JSON, if are generated from same SQLite table. This ensures proper grouping in sidebar of HTML report.
- If special characters are present in config file, users must encode it in UTF-8.

2.7 Default config file

- chronqc database command in create mode (-c) generates default JSON file named `chronqc.default.json`
- Name of tools along with corresponding QC metrics and chart types implemented in default config file (JSON file) generated by ChronQC (database -c) are listed in the table below.
- This can be changed by modifing the chronqc.conf file in config folder under chronqc installation directory.

Tool name	QC metrics	Chart type implemented in default JSON (config file)
FastQC	FastQC_percent_gc	time_series_with_mean_and_stdev
	FastQC_total_sequences	time_series_with_mean_and_stdev
	FastQC_percent_duplicates	time_series_with_mean_and_stdev
	FastQC_percent_fails	time_series_with_mean_and_stdev
	FastQC_avg_sequence_length	time_series_with_mean_and_stdev
QualiMap	QualiMap_30_x_pc	time_series_with_mean_and_stdev
	QualiMap_percentage_aligned	time_series_with_mean_and_stdev
	QualiMap_avg_gc	time_series_with_mean_and_stdev (if FastQC_percent_gc is present this plot is omitted to avoid duplication)
	QualiMap_mapped_reads	time_series_with_mean_and_stdev
	QualiMap_median_coverage	time_series_with_mean_and_stdev
	QualiMap_total_reads	time_series_with_mean_and_stdev (if FastQC_total_sequences is present this plot is omitted to avoid duplication)
Bamtools	Bamtools_mapped_reads_pct	time_series_with_mean_and_stdev (if QualiMap_mapped_reads is present this plot is omitted to avoid duplication)
Samtools	SamtoolsFlagstat_mapped_passed	time_series_with_mean_and_stdev (if QualiMap_percentage_aligned is present this plot is omitted to avoid duplication)
Bcftools	BcftoolsStats_number_of_MNPs	time_series_with_box_whisker_plot
	BcftoolsStats_number_of_SNPs	time_series_with_box_whisker_plot
	Bcftools_Stats_number_of_indels	time_series_with_box_whisker_plot
	BcftoolsStats_number_of_records	time_series_with_box_whisker_plot
	BcftoolsStats_tstv	time_series_with_mean_and_stdev
Peddy	Peddy_error	time_series_with_percentage_category (Default category: True)
Any other tool	Columns with numeric data	time_series_with_mean_and_stdev

Below is an example of default ChronQC config file generated in examples/multiqc_example_2:

```
[
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_absolute_threshold",
        "chart_properties": {
            "y_value": "QualiMap_median_coverage",
            "lower_threshold": 30
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_percentage_of_samples_above_threshold",
        "chart_properties": {
            "y_value": "QualiMap_median_coverage",
            "threshold": 30
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
```

```
            "y_value": "FastQC_percent_duplicates"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "FastQC_percent_gc"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "FastQC_total_sequences"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "FastQC_avg_sequence_length"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "FastQC_percent_fails"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "Bcftools_Stats_tstv"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "QualiMap_percentage_aligned"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "QualiMap_30_x_pc"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "QualiMap_mapped_reads"
        }
```

```
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_box_whisker_plot",
        "chart_properties": {
            "y_value": "Bcftools_Stats_number_of_MNPs"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_box_whisker_plot",
        "chart_properties": {
            "y_value": "Bcftools_Stats_number_of_SNPs"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_box_whisker_plot",
        "chart_properties": {
            "y_value": "Bcftools_Stats_number_of_indels"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_box_whisker_plot",
        "chart_properties": {
            "y_value": "Bcftools_Stats_number_of_records"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "QualiMap_total_reads"
        }
    },
    {
        "table_name": "chronqc_stats_data",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
            "y_value": "Samtools_Flagstat_mapped_passed"
        }
    }
]
```

Below is an example of customized ChronQC config file

```
[
  {
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
          "chart_title": "% of Duplicates in On-target Sites (per run)",
          "y_value": "Duplicates",
          "y_label": "%  of Duplicates"
        }
```

```
  },
  {
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
          "chart_title": "Average Mapping Quality of On-target Sites (per run)",
          "y_value": "MappingQuality",
          "y_label": "MappingQuality"
        }
  },
  {
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_absolute_threshold",
        "chart_properties": {
          "chart_title": "Average Base Quality Scores in On-target Sites (per run)",
          "y_value": "BaseQuality",
          "lower_threshold": 30,
          "y_label": "Average Base Quality Score"
        }
  },
  {
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
          "chart_title": "Number of Bases in Reads within On-target Sites (per run)",
          "y_value": "BasesOfReads",
          "y_label": "Bases Of Reads"
        }
  },
  {
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
          "chart_title": "% of Bases in Reads within On-target Sites (per run)",
          "y_value": "%BasesofReads",
          "y_label": "% of Bases of Reads"
        }
  },
  {
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_absolute_threshold",
        "chart_properties": {
          "chart_title": "Depth Median (per run)",
          "y_value": "Depth",
          "lower_threshold": 200,
          "y_label": "Depth Median (per run)"
        }
  },
```

```
{
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "HCT",
        "chart_type": "time_series_with_absolute_threshold",
        "chart_properties": {
          "chart_title": "Depth Median (HCT)",
          "y_value": "Depth",
          "lower_threshold": 200,
          "y_label": "Depth Median"
        }
},
{
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
          "chart_title": "GC Content % (per run)",
          "y_value": "GCContent",
          "y_label": "GC Content % (per run)"
        }
},
{
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_percentage_category",
        "chart_properties": {
          "chart_title": "% of Samples that passed VCS QC (per run)",
          "y_value": "vcs_coverage_qc",
          "y_label": "% Samples in library",
          "category": "PASS"
        }
},
{
        "table_name": "Production_Run_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_percentage_of_samples_above_threshold",
        "chart_properties": {
          "chart_title": "% of Samples in a run with >= 200 depth (per run)",
          "y_value": "Depth",
          "threshold": 200,
          "y_label": "% Samples in library"
        }
},
{
        "table_name": "SNPs_Indels_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_box_whisker_plot",
        "chart_properties": {
          "chart_title": "Number of SNPs found in Samples Over Time",
          "y_value": "Number",
          "Type": "SNPs",
          "y_label": "Number of SNPs found in each sample"
        }
},
```

```
  {
        "table_name": "SNPs_Indels_Stats_Summary",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_box_whisker_plot",
        "chart_properties": {
          "chart_title": "Number of indels found in Samples Over Time",
          "y_value": "Number",
          "Type": "Indels",
          "y_label": "Number of indels found in each sample"
        }
  },
  {
        "table_name": "Ti_Tv_Ratio_Stats",
        "include_samples": "all",
        "exclude_samples": "HCT, NTC",
        "chart_type": "time_series_with_mean_and_stdev",
        "chart_properties": {
          "chart_title": "Transition to Transversion Ratio of Samples Over Time (per␣
↪run)",
          "y_value": "Number",
          "y_label": "Ti/Tv Ratio"
        }
  },
  {
        "table_name": "Ti_Tv_Ratio_Stats",
        "include_samples": "HCT",
        "chart_type": "time_series_with_absolute_threshold",
        "chart_properties": {
          "chart_title": "Transition to Transversion Ratio of Positive Control (HCT)␣
↪Over Time (per run)",
          "y_value": "Number",
          "y_label": "Positive Control (HCT) Ti/Tv Ratio",
          "lower_threshold": 1.4,
          "upper_threshold": 1.78
        }
  },
  {
        "table_name": "SNPs_Indels_Stats_Summary",
        "include_samples": "HCT",
        "chart_type": "time_series_with_absolute_threshold",
        "chart_properties": {
          "chart_title": "Numbers of SNPs in Positive Control (HCT) Over Time",
          "y_value": "Number",
          "lower_threshold": 6580,
          "upper_threshold": 9728,
          "Type": "SNPs",
          "y_label": "Numbers of SNPs in Positive Control (HCT) Over Time"
        }
  },
  {
        "table_name": "SNPs_Indels_Stats_Summary",
        "include_samples": "HCT",
        "chart_type": "time_series_with_absolute_threshold",
        "chart_properties": {
          "chart_title": "Numbers of Indels in Positive Control (HCT) Over Time",
          "y_value": "Number",
          "lower_threshold": 1521,
```

```
          "upper_threshold": 1960,
          "Type": "Indels",
          "y_label": "Numbers of Indels in Positive Control (HCT) Over Time"
        }
  },
  {
        "table_name": "VCS_Stats_Summary",
        "include_samples": "all",
        "chart_type": "time_series_with_bar_line_plot",
        "chart_properties": {
          "y_value": "Gene",
          "categories": "KRAS, KIT, BRAF, PDGFRA, NRAS"
          }
  },
  {
        "table_name": "VCS_Stats_Summary",
        "include_samples": "all",
        "chart_type": "time_series_with_stacked_bar_plot",
        "chart_properties": {
          "y_value": "Gene",
          "categories": "KRAS, KIT, BRAF, PDGFRA, NRAS"
          }
  }
]
```

2.8 ChronQC Plots

ChronQC currently supports seven types of charts. The seven chart types were selected based on our experience as a CAP-accredited laboratory. The different chart types are associated with different QC tracking features based on Westgard rules for clinical validity (e.g. demarcating ± 2 standard deviations) (Westgard, J.O. et al. 1981), laboratory-defined thresholds, and historical QC observations within a specified period. ChronQC plots can assist in identifying trends, bias, and excessive scatter in the clinical data, so that corrective and preventive actions can be taken to ensure that patient results remain clinically valid.

Chart type	Description	Use case
Time series plot with mean and standard deviation	A time series plot of numerical data with historical runs. Rolling mean and ± 2 standard deviations are shown.	Can be used to track metrics such as total number of reads. The window to compute rolling mean and ± 2 standard deviations can be set to either a specified duration (e.g. runs in past 1 year) or number of historical runs (e.g. past 10 runs).
Time series plot with absolute threshold	A time series plot of numerical data with user-defined lower and upper thresholds.	Can be used to track metrics such as depth of coverage, Ti/Tv ratio, and GC content per sample. Lower and upper thresholds can be based the clinical validation experiment or empirical values.
Time series plot with percentage of samples above threshold	A time series plot representing percentage of numerical data above the user defined threshold.	Can be used to track metrics such as percentage of samples in a run that exceed a specified coverage depth. The threshold can be based the clinical validation experiment.
Time series plot with percentage of samples with a category label	A time series plot of categorical data representing % of samples in a run with y-value is equal to category.	Can be used to track percentage of samples in a run with a certain label. E.g. % of samples labeled "PASS" based on laboratory-defined QC metrics.
Time series box-and-whisker plot of the numerical data	A monthly time series box-and-whisker plot of numerical data.	Can be used to track number of single nucleotide variants (SNVs) and indels observed in a month.
Time series with stacked bar plot of the categorical data	A monthly time series stacked bar plot of categorical data.	Can be used to track number of mutations in clinically actionable genes per month.
Time series with bar and line plot of the categorical data	A monthly time series bar and line plot of categorical data.	Can be used to track number of mutations in clinically actionable genes per month.

2.9 Plot options

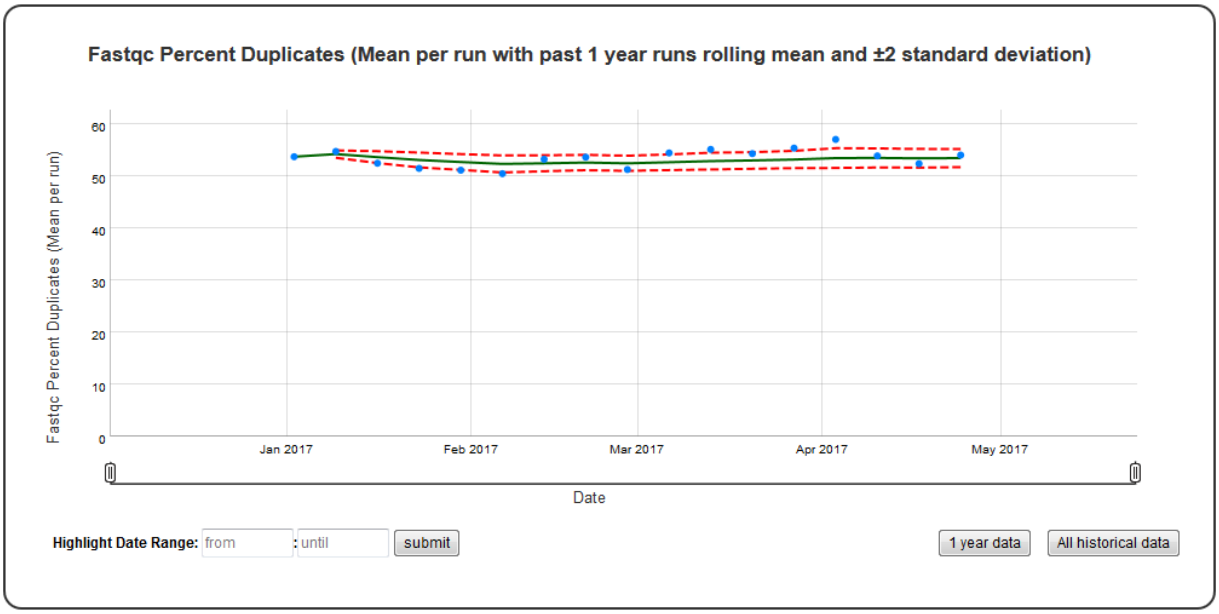
Following options are available for each chart type and can be set in the JSON file

Option	Type	Use
table_name	String (Optional)	This is used to get data from the SQLite table. Default is "chronqc_stats_data".
include_samples	String (Optional)	This is used to select samples from the SQLite table. It can be either "all" or a list of sample names. If "all" all samples are selected. If a list of strings, samples matching the elements in a comma-delimited list will be selected. String matching is partial. For example, "HCT15, NTC" would include samples called HCT15, NTC, NTC1, NTC2. Default is "all".
exclude_samples	String (Optional)	This is used to exclude samples from plotting. It can be either a string or a list of sample names. Samples matching the string or elements in a list will not be plotted. String matching is partial and case-sensitive. For example, "Control" would exclude samples named Control, Control1, 1Control, etc. Default is empty string.
chart_type	String (Required)	**This is used to specify chart type. Possible values are** 1. time_series_with_mean_and_stdev 2. time_series_with_absolute_threshold 3. time_series_with_percentage_of_samples_a 4. time_series_with_percentage_category 5. time_series_with_box_whisker_plot 6. time_series_with_stacked_bar_plot 7. time_series_with_bar_line_plot

2.10 Time series plot with mean and standard deviation

A time series plot of numerical data with rolling mean and standard deviation. Numerical data in `y_value` column of the SQLite table defined by `table_name` is used to plot this graph. SQLite table must have; `Run, Sample, Date, y_value` columns to generate the plot. In case of `per_sample` graph `Run` column is not required. For `per_sample` graph if only `Run` column is present in the table, `Run` column is used to generate plots.

2.10.1 Example Plot



2.10.2 Chart Properties

Option	Type	Use
Chart_title	String (Optional)	This is used to create the title of the chart. Default is "{y_label} (Mean per run with {window} rolling mean and ± standard deviation)". E.g. "Fastqc Percent Duplicates (Mean per run with past 1 year runs rolling mean and ±2 standard deviation)".
y_value	String (Required)	Column header in SQLite table. The column should contain numerical data. This data is plotted on the y-axis. E.g. "FastQC_percent_duplicates".
y_label	String (Optional)	This is used to create the y-axis label in the chart. Default is "Mean {y_value} per run". E.g. "FastQC Percent Duplicates".
window	Integer or String (Optional)	Window can be an integer (n) or number of days in the format "365D" (d). If an integer (n) is is specified, rolling mean and standard deviation is computed based on past "n" runs. If number of days (d) is specified, rolling mean and standard deviation is computed based on runs in the past "d". E.g. 1. window="10", this will compute rolling mean and standard deviation based on past 10 runs. 2. window="365D", this will compute rolling mean and standard deviation on the runs in past 365 days. Default is "365D".
per_sample_plot	Boolean (Optional)	Plot per sample graph. Default is "False". If set to "True" per sample graph will be plotted.

Example JSON entry (minimum):

```
[
 {
  "chart_type": "time_series_with_mean_and_stdev",
  "chart_properties": {
```

```
      "y_value": "FastQC_percent_duplicates"
      }
 },
]
```

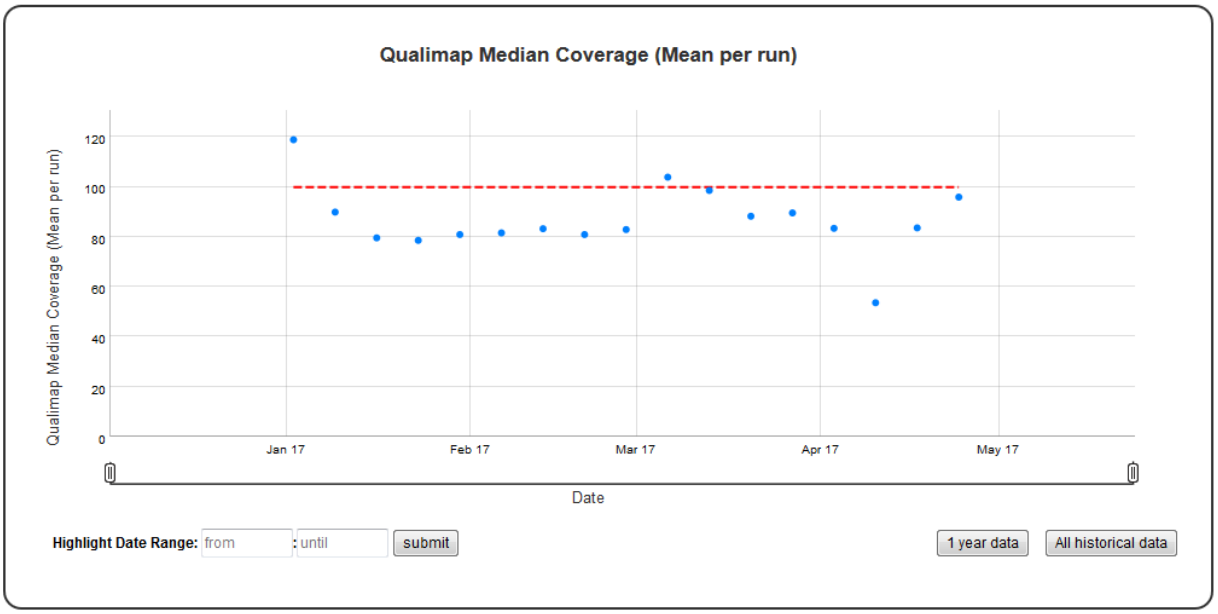
Example JSON entry (full) to plot all samples excluding HCT15 and NTC:

```
[
 {
  "table_name": "Production_Run_Stats_Summary",
  "include_samples": "all",
  "exclude_samples": "HCT15, NTC",
  "chart_type": "time_series_with_mean_and_stdev",
  "chart_properties": {
       "chart_title": "Fastqc Percent Duplicates (Mean per run with past 1 year runs␣
↪rolling mean and &plusmn standard deviation)",
       "y_value": "FastQC_percent_duplicates",
       "y_label": "FastQC Percent Duplicates",
       "window" : "10",
       "per_sample": "False"
  }
 }
]
```

2.11 Time series plot with absolute threshold

A time series plot of numerical data with user defined lower and upper thresholds. Numerical data in `y_value` column of the SQLite table defined by `table_name` is used to plot this graph. SQLite table must have; `Run, Sample, Date, y_value` columns to generate the plot. In case of `per_sample` graph `Run` column is not required. For `per_sample` graph if only `Run` column is present in the table, `Run` column is used to generate plots.

2.11.1 Example Plot



2.11.2 Chart Properties

Option	Type	Use
Chart_title	String (Optional)	This is used to create the title of the chart. Default is "{y_label} (Mean per run)". E.g. "Qualimap Median Coverage (Mean per run)".
y_value	String (Required)	Column header in SQLite table. The column should contain numeric data. This data is plotted on the y-axis. E.g. "Depth".
y_label	String (Optional)	This is used to create the y-axis label in the chart. Default is "{y_value} (Mean per run)". E.g. "Qualimap Median Coverage (Mean per run)".
lower_threshold	Integer (Optional)	This is used to create lower threshold line on the chart. E.g. 100.
upper_threshold	Integer (Optional)	This is used to create upper threshold line on the chart. E.g. 300.
per_sample	Boolean (Optional)	Plot per sample graph. Default is "False". If set to "True" per sample graph will be plotted.

Example JSON entry (minimum):

```
[
 {
  "chart_type": "time_series_with_absolute_threshold",
  "chart_properties": {
      "y_value": "Depth",
      "lower_threshold": 100,
      }
 }
]
```

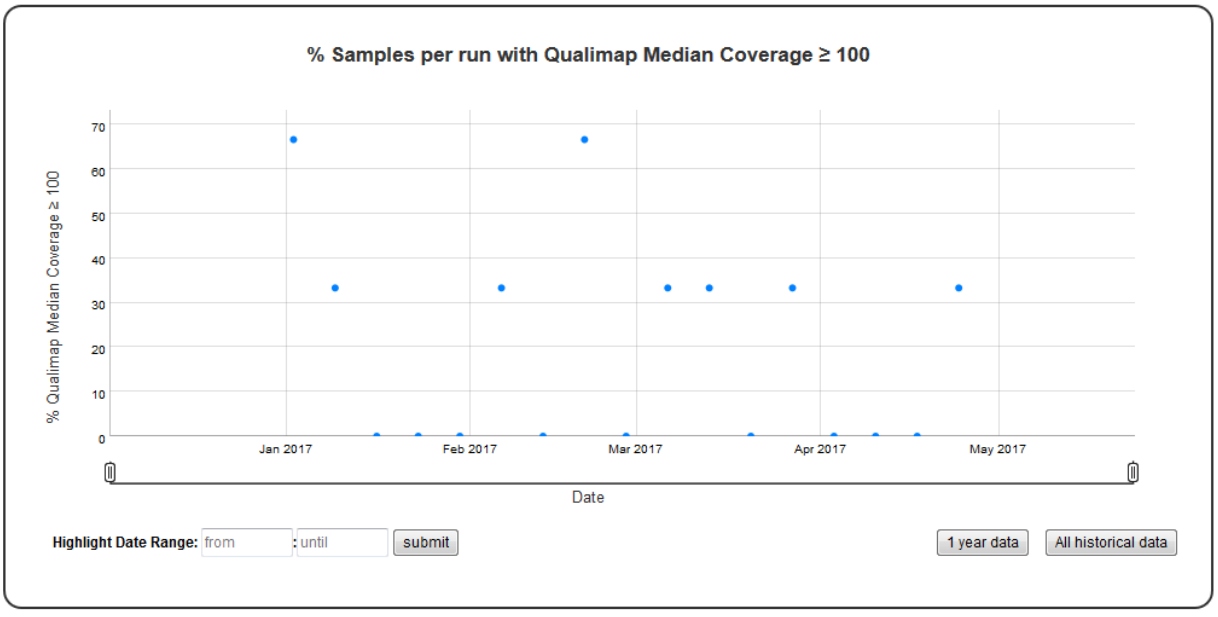
Example JSON entry (full) to plot all samples excluding HCT15 and NTC:

```
[
 {
  "table_name": "Production_Run_Stats_Summary",
  "include_samples": "all",
  "exclude_samples": "HCT15, NTC",
  "chart_type": "time_series_with_absolute_threshold",
  "chart_properties": {
      "chart_title": "Qualimap Median Coverage (Mean per run)",
      "y_value": "Depth",
      "lower_threshold": 100,
      "y_label": "Qualimap Median Coverage (Mean per run)",
      "per_sample": "True"
      }
 }
]
```

2.12 Time series plot with percentage of samples above threshold

A time series plot representing percentage of numerical data in `y_value` column of the SQLite table defined by `table_name` above the user defined threshold. SQLite table must have; `Run, Sample, Date, y_value` columns to generate the plot.

2.12.1 Example Plot



2.12.2 Chart Properties

Option	Type	Use
Chart_title	String (Optional)	This is used to create the title of the chart. Default is "% Samples per run with {y_label} {threshold}". E.g. "% Samples per run with Qualimap Median Coverage 100".
y_value	String (Required)	Column header in SQLite table. The column should contain numeric data. This data is plotted on the y-axis. E.g. "Depth".
y_label	String (Optional)	This is used to create the y-axis label in the chart. Default is "% Samples per run with {y_value} {threshold}". E.g. "Qualimap Median Coverage 100".
threshold	Integer (Required)	This is used to compute % of values above the threshold. E.g. 100.

Example JSON entry (minimum):

```
[
 {
  "chart_type": "time_series_with_percentage_of_samples_above_threshold",
  "chart_properties": {
        "y_value": "QualiMap_median_coverage",
        "threshold": 100
        }
 }
]
```

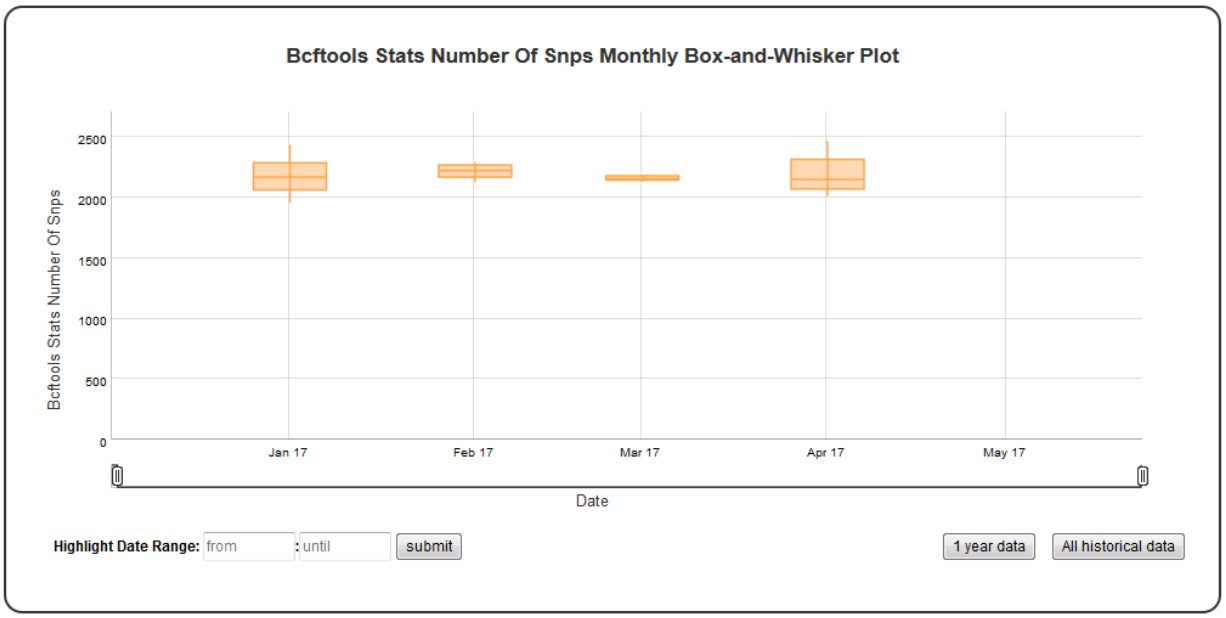
Example JSON entry to plot all samples exluding HCT15 and NTC (full):

```
[
 {
  "table_name": "Production_Run_Stats_Summary",
  "include_samples": "all",
  "exclude_samples": "HCT15, NTC",
  "chart_type": "time_series_with_percentage_of_samples_above_threshold",
  "chart_properties": {
        "chart_title": "% Samples per run with Qualimap Median Coverage &ge; 100",
        "y_value": "Depth",
        "threshold": 100,
        "y_label": "Qualimap Median Coverage &ge; 100"
        }
 }
]
```

2.13 Time series Box-and-Whisker plot of the numerical data

A time series Box-and-Whisker plot of numerical data. Numerical data in `y_value` column of the SQLite table defined by `table_name` is used to plot this graph. SQLite table must have; `Sample, Date, y_value` columns to generate the plot. If `Run` column is present instead of `Sample` column in the table, `Run` column is used to generate plots.

2.13.1 Example Plot



2.13.2 Chart Properties

Option	Type	Use
Chart_title	String (Optional)	This is used to create the title of the chart. Default is "{y_label} Monthly Box-and-Whisker Plot". E.g. "Bcftools Stats Number Of Snps Monthly Box-and-Whisker Plot".
y_value	String (Required)	Column header in SQLite table. The column should contain numeric data. This data is plotted on the y-axis. E.g. "Number".
y_label	String (Optional)	This is used to create the y-axis label in the chart. Default is "{y_label}". E.g. "Bcftools Stats Number Of Snps".
Type	String (Optional)	This is used to select subset of rows from the SQLite table's "Type" columns. E.g. "SNPs".
lower_threshold	Integer (Optional)	This is used to create lower threshold line on the chart. E.g. 100.
upper_threshold	Integer (Optional)	This is used to create upper threshold line on the chart. E.g. 300.

Example JSON entry (minimum):

```
[
 {
  "chart_type": "time_series_with_box_whisker_plot",
  "chart_properties": {
      "y_value": "Bcftools_Stats_number_of_SNPs"
```

```
        }
  }
]
```

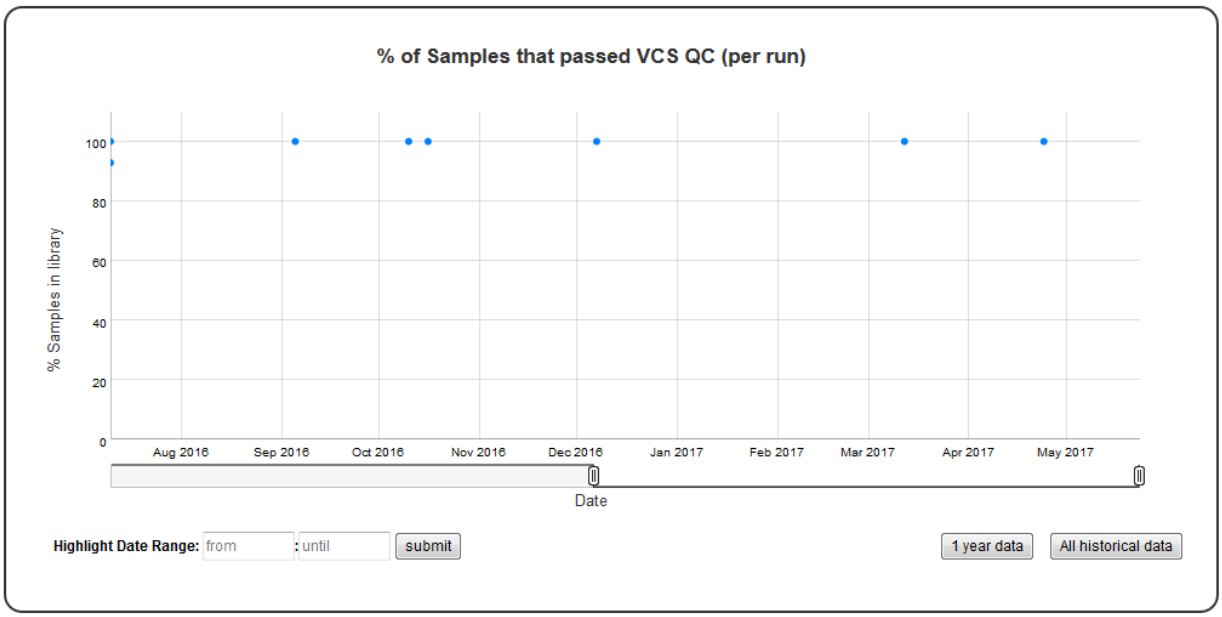
Example JSON entry (full) to plot all samples excluding HCT15 and NTC

```
[
  {
   "table_name": "SNPs_Indels_Stats_Summary",
   "include_samples": "all",
   "exclude_samples": "HCT15, NTC",
   "chart_type": "time_series_with_box_whisker_plot",
   "chart_properties": {
       "chart_title": "Bcftools Stats Number Of Snps Monthly Box-and-Whisker Plot",
       "y_value": "Number",
       "Type": "SNPs",
       "y_label": "Bcftools Stats Number Of Snps",
       "lower_threshold": 10000,
       "upper_threshold": 15000
       }
  }
]
```

2.14 Time series plot with percentage category

A time series plot of categorical data in `y_value` column of the SQLite table defined by `table_name`. Represents % of samples in a run with `y_value` is equal to category (defined by `category`). SQLite table must have; `Run, Sample, Date, y_value` columns to generate the plot.

2.14.1 Example Plot



2.14.2 Chart Properties

Option	Type	Use
Chart_title	String (Optional)	This is used to create the title of the chart. Default is "% Samples per run with {y_label} = {category}". E.g. "% of Samples that passed VCS QC (per run)".
y_value	String (Required)	Column header in SQLite table. % of samples with y_value = category is plotted on y-axis. E.g. "vcs_coverage_qc".
y_label	String (Optional)	This is used to create the y-axis label in the chart. Default is "% {y_value} = {category}". E.g. "% Samples in library".
category	String (Required)	This is used to calculate the % of samples = "category". String matching is done by ignoring the case for values. Default is "PASS". E.g. "PASS".

Example JSON entry (minimum):

```
[
 {
  "chart_type": "time_series_with_percentage_category",
  "chart_properties": {
       "y_value": "vcs_coverage_qc",
       }
 }
]
```

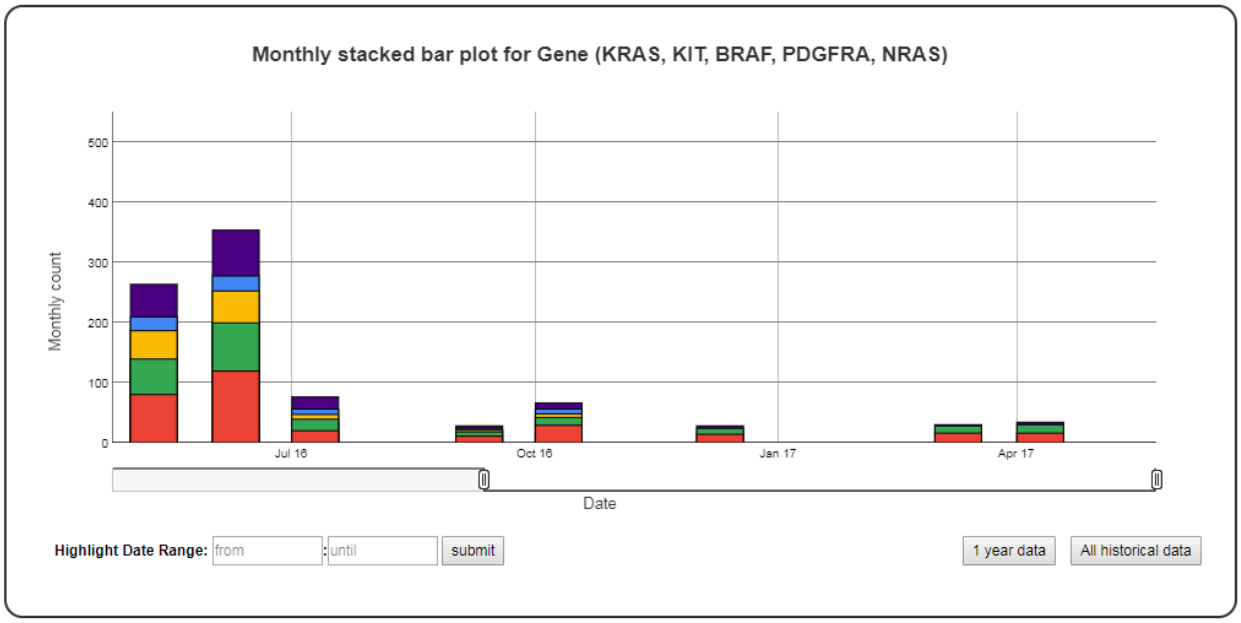
Example JSON entry (full) to plot all samples excluding HCT15 and NTC:

```
[
 {
  "table_name": "Production_Run_Stats_Summary",
  "include_samples": "all",
  "exclude_samples": "HCT15, NTC",
  "chart_type": "time_series_with_percentage_category",
  "chart_properties": {
       "chart_title": "% of Samples that passed VCS QC (per run)",
       "y_value": "vcs_coverage_qc",
       "y_label": "% Samples in library",
       "category": "PASS"
       }
 }
]
```

2.15 Time series stacked bar plot of the categorical data

A time series stacked bar plot of the categorical data. Count of categorical `y_value` values of `categories` in the SQLite table defined by `table_name` is used to plot this graph. SQLite table must have; `Sample, Date, y_value` columns to generate the plot. If `Run` column is present instead of `Sample` column in the table, `Run` column is used to generate plots.

2.15.1 Example Plot



2.15.2 Chart Properties

Option	Type	Use
Chart_title	String (Optional)	This is used to create the title of the chart. Default is "Monthly stacked bar plot for {y_label} ({categories})". E.g. "Monthly stacked bar plot for Gene (KRAS, KIT, BRAF, PDGFRA, NRAS)".
y_value	String (Required)	Column header in SQLite table. The column should contain categorical data. Count of categories defined by "categories" option is plotted on the y-axis. E.g. "Gene".
y_label	String (Optional)	This is used to create the y-axis label in the chart. Default is "Monthly count".
categories	String (Required)	This is used to select subset of categories from the SQLite table's y_value column. Maximum 10 categories can be specified in a single graph. To track more than 10 categories create multiple graphs. E.g. "KRAS, KIT, BRAF, PDGFRA, NRAS".

Example JSON entry (minimum):

```
[
 {
  "chart_type": "time_series_with_stacked_bar_plot",
  "chart_properties": {
      "y_value": "Gene",
      "categories": "KRAS, KIT, BRAF, PDGFRA, NRAS"
      }
 }
]
```

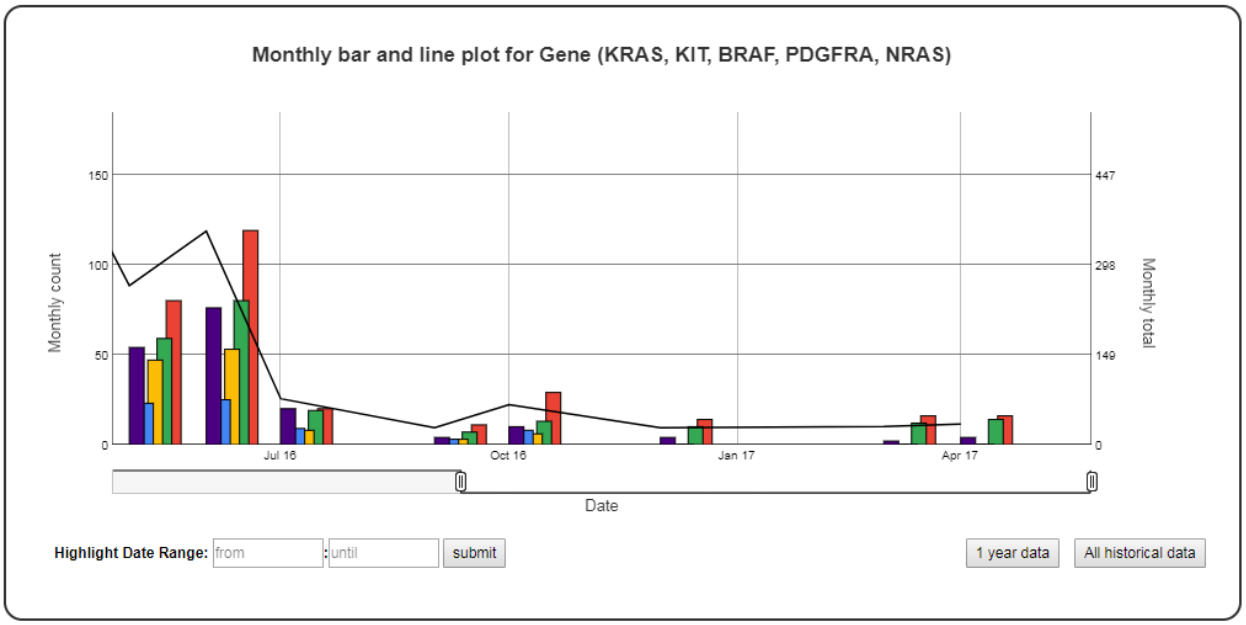
Example JSON entry (full) to plot all samples excluding NTC

```
[
 {
  "table_name": "VCS_Stats_Summary",
  "include_samples": "all",
  "exclude_samples": "NTC",
  "chart_type": "time_series_with_stacked_bar_plot",
  "chart_properties": {
       "chart_title": "Monthly bar and line plot for Gene (KRAS, KIT, BRAF, PDGFRA,␣
↪NRAS)",
       "y_value": "Gene",
       "categories": "KRAS, KIT, BRAF, PDGFRA, NRAS",
       "y_label": "Monthly count"
       }
 }
]
```

2.16 Time series bar line plot of the categorical data

A time series bar line plot of the categorical data. Count of categorical `y_value` values of `categories` in the SQLite table defined by `table_name` is used to plot this graph. SQLite table must have; `Sample`, `Date`, `y_value` columns to generate the plot. If `Run` column is present instead of `Sample` column in the table, `Run` column is used to generate plots.

2.16.1 Example Plot



2.16.2 Chart Properties

Option	Type	Use
Chart_title	String (Optional)	This is used to create the title of the chart. Default is "Monthly bar and line plot for {y_label} ({categories})". E.g. "Monthly bar and line plot for Gene (KRAS, KIT, BRAF, PDGFRA, NRAS)".
y_value	String (Required)	Column header in SQLite table. The column should contain categorical data. Count of categories defined by "categories" option is plotted on the y1-axis and sum is plotted on the y2-axis. E.g. "Gene".
y_label	String (Optional)	This is used to create the y1-axis label in the chart. Default is "Monthly count".
y_label2	String (Optional)	This is used to create the y2-axis label in the chart. Default is "Monthly total".
categories	String (Required)	This is used to select subset of categories from the SQLite table's y_value column. Maximum 10 categories can be specified in a single graph. To track more than 10 categories create multiple graphs. E.g. "KRAS, KIT, BRAF, PDGFRA, NRAS".

Example JSON entry (minimum):

```
[
 {
  "chart_type": "time_series_with_bar_line_plot",
  "chart_properties": {
      "y_value": "Gene",
      "categories": "KRAS, KIT, BRAF, PDGFRA, NRAS"
      }
 }
]
```

Example JSON entry (full) to plot all samples excluding NTC

```
[
 {
  "table_name": "VCS_Stats_Summary",
  "include_samples": "all",
  "exclude_samples": "NTC",
  "chart_type": "time_series_with_bar_line_plot",
  "chart_properties": {
      "chart_title": "Monthly bar and line plot for Gene (KRAS, KIT, BRAF, PDGFRA,␣
↪NRAS)",
      "y_value": "Gene",
      "categories": "KRAS, KIT, BRAF, PDGFRA, NRAS",
      "y_label": "Monthly count",
      "y_label2": "Monthly total"
      }
 }
]
```

2.17 Automating ChronQC (chrongen)

ChronQC plot generation can be automated in two stages,

1. Use "chronqc database" command as part of bioinformatics pipeline to update the ChronQC SQLite database (chronqc.stats.sqlite) with statistics .
2. The command "chronqc chrongen" can be used for the automation of generation of ChronQC plots from a ChronQC statistics database (chronqc.stats.sqlite) or custom SQLite database. The database must contain information on sequencing runs, run dates, and laboratory or bioinformatics QC metrics.

The settings for generating ChronQC plot can be specified in a configuration file (.ini). An email notification will be sent out to the users after the plots are generated. This script also generates a log event file to record the ChronQC commands that have been used.

Edit the configuration file with the panel name and json file name to generate ChronQC plots.

Table of Contents

2.18 Requirements

- Python 2.6 and above
- ChronQC 1.0.4 and above
- ChronQC json file
- crontab configuration

2.19 Execute it

To run it, execute the command below:

```
chonqc chrongen <.ini configuration file>
```

To run it on crontab scheduler periodically (etc. every month):

2.20 I / O

2.20.1 INPUT: ChronGen .ini Configuration File

The headers and parameters shown below are mandatory. All paths should be **absolute**.

```
[email]
TO = <email 1>, <email 2>
HOST = <from email address>
CC = <cc email 1>, <cc email 2>
SMTP_SERVER = <smpt server ip address>

[template]
SUBJECT = [ Monthly QC statistics ] Month of %s
NOTICE = <br>Dear Users,</br> <p><br>ChronQC plots are ready for viewing in:  <br>%s</
↪br></br></p><br>Thank you.</br><br>*** This is an  automated mail, please do not␣
↪reply ***</br>

[chronqc]
DATABASE = <database path>
GEN_CMD = chronqc plot -o %s %s %s %s

[chronqc_json]
<panel name 1> = <panel 1 .json path>
<panel name 2> = <panel 2 .json path>

[iomanip]
DESTINATION = <ChronQC output directory>
```

2.20.2 OUTPUT: ChronQC Graphs in Dated folder | Log File

A output folder named based on the date format: 'DD_MON_YYYY' will be created in the directory specified by "iomanip"'s DESTINATION tag in the .ini config file:

```
[iomanip]
DESTINATION = <ChronQC output directory>
```

The output ChronQC HTML files are stored in this the folder.

A log file detailing the events of the ChronGen process will be present in the config directory of this script.

2.21 Citation

ChronQC: A Quality Control Monitoring System for Clinical Next Generation Sequencing Nilesh R. Tawari, Justine Jia Wen Seow, Dharuman Perumal, Jack L. Ow, Shimin Ang, Arun G. Devasia, Pauline C. Ng Bioinformatics, In press, https://doi.org/10.1093/bioinformatics/btx843

2.22 License

This project is licensed under the MIT License - see the LICENSE.md file for details

2.23 FAQs

2.23.1 Q1: How to automate ChronQC for production?

ChronQC can be automated completely. For details on how to automate ChronQC plot generation see, automate_chronqc.

2.23.2 Q2: How to report bugs?

We encourage users to report bugs by creating Git Issue. Alternatively, users can send email to the developers at: tawari.nilesh@gmail.com

2.23.3 Q3: How to request new plots?

New plots can be added to ChronQC based on the users request. Users can request new plots through Git Issue. Alternatively, users can send email to the developers at: tawari.nilesh@gmail.com