

Analysis of qRT-PCR Data to Identify the Most Stable Reference Gene Using gQuant

Abhay Kumar Pathak^{1, #, §}, Sukhad Kural^{2, #}, Shweta Singh^{3, #}, Lalit Kumar², Manjari Gupta^{1, *} and Garima Jain^{3, *}

¹CIMS, Institute of Science, Banaras Hindu University, Varanasi, India

²Department of Urology, Institute of Medical Science, Banaras Hindu University, Varanasi, India

³MIRNOW, BioNest, Banaras Hindu University, Varanasi, India

*For correspondence: garima.jain@bhu.ac.in; manjari@bhu.ac.in

#Contributed equally to this work

§Technical contact: pathakabhay@bhu.ac.in

Abstract

The accurate quantification of nucleic acid-based biomarkers, including long non-coding RNAs (lncRNAs), messenger RNAs (mRNAs), and microRNAs (miRNAs), is essential for disease diagnostics and risk assessment across the biological spectrum. Quantitative reverse transcription PCR (qRT-PCR) is the gold standard assay for the quantitative measurement of RNA expression levels, but its reliability depends on selecting stable reference targets for normalization. Yet, the lack of consensus on a universally accepted reference gene for a given sample type or species, despite being necessary for accurate quantification, presents a challenge to the broad application of such biomarkers. Various tools are currently being used to identify a stably expressed gene by using qRT-PCR data of a few potential normalizer genes. However, existing tools for normalizer gene selection are fraught with both statistical limitations and inadequate graphical user interfaces for data visualization. gQuant, the tool presented here, essentially overcomes these limitations. The tool is structured in two key components: the preprocessing component and the data analysis component. The preprocessing addresses missing values in the given dataset by the imputation strategies. After data preprocessing, normalizer genes are ranked using democratic strategies that integrate predictions from multiple statistical methods. The effectiveness of gQuant was validated through data available online as well as in-house data derived from urinary exosomal miRNA expression datasets. Comparative analysis against existing tools demonstrated that gQuant delivers more stable and consistent rankings of normalizer genes. With its promising performance, gQuant enhances the precision and reproducibility in the identification of normalizer genes across diverse research scenarios, addressing key limitations of RNA biomarker-based translational research.

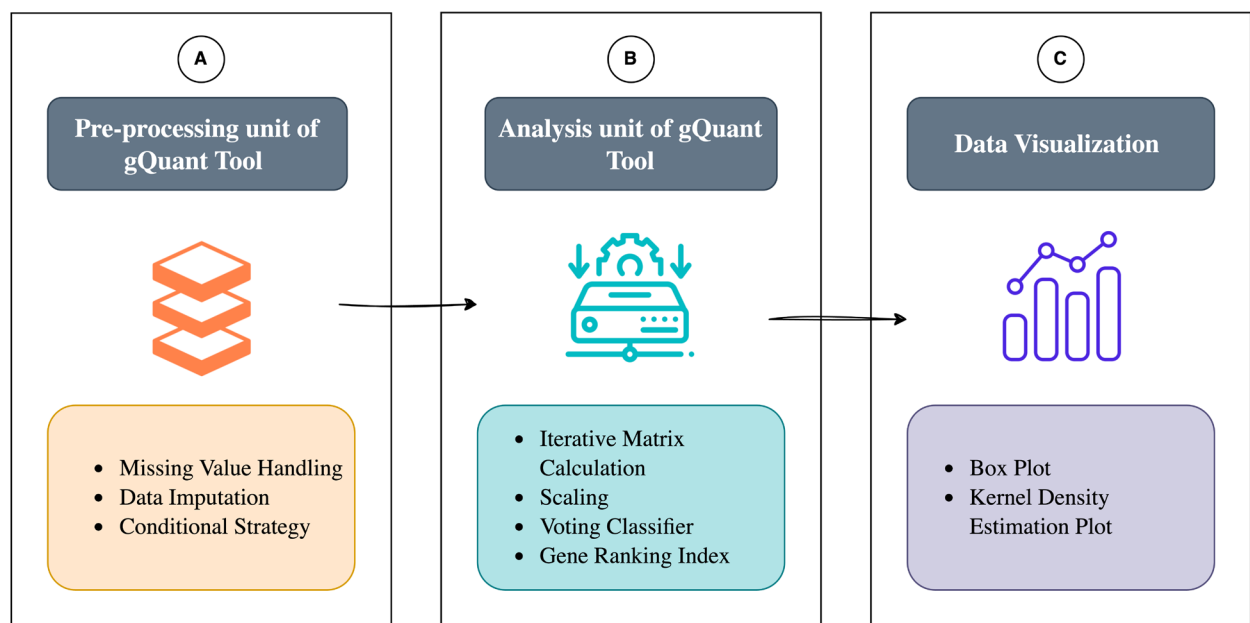
Key features

- Accurate reference gene selection: gQuant identifies the most stable gene in qRT-PCR datasets using a multi-metric approach including SD, GM, CV, and KDE.
- Robust missing data handling: Implements imputation and removal strategies to ensure data integrity and accurate normalizer selection.
- Bias-free ranking algorithm: Utilizes a voting-based classifier to provide fair and consistent ranking, overcoming limitations of weighted approaches.
- Comprehensive visualization: Offers boxplots and KDE plots for analyzing gene expression variability, aiding in data interpretation.

Keywords: Reference gene, Normalizer, Ranking algorithm, Voting classifier, qRT-PCR, Expression data, miRNAs

This protocol is used in: Sci Rep (2023), DOI: 10.1038/s41598-024-66770-y

Graphical overview



Comprehensive workflow. (A) Preprocessing unit of gQuant, including missing value handling, imputation, and filtering. (B) Analysis unit of gQuant, including matrix calculations, scaling, and multiple statistical methods. (C) Comprehensive visualization: Offers boxplots and KDE plots for analyzing expression variability in qRT-PCR data, aiding in data interpretation.

Background

Quantitative reverse transcription PCR (qRT-PCR) is the benchmark method for accurately measuring RNA expression levels. qRT-PCR becomes completely indispensable as a tool when it comes to using RNA-based biomarkers for diagnosis or prognosis [1,2]. The vital aspect of comparative RNA expression studies utilizing qRT-PCR is the judicious selection of normalizer or reference genes/sequences for accurate data normalization [3]. While internal reference genes or normalizer genes, such as *ACTB*, *GAPDH*, and 18S Ribosomal RNA [4], have been successfully employed at the tissue level for mRNA quantification, researchers also opt to choose specific normalizers for individual experiments, particularly where a universally accepted set of normalizers remains elusive, such as in miRNA-based assays [5,6]. Although many miRNA-based studies have used *RNU6* as the most commonly used reference gene, it has been reported to have variable expression and does not represent the optimal reference gene for miRNA analysis [7]. *RNU6* does not show the same biochemical properties as miRNA molecules, so the process of extracting, converting, and amplifying *RNU6* might not be as efficient as it is for miRNAs. The use of *RNU6* is inappropriate for serum and plasma as it is only present in serum or plasma as a result of coagulation or cell lysis.

miRNAs have emerged as promising biomolecules in diagnostic, prognostic, and therapeutic avenues in various cancers [8]. Their stability, abundance, and presence in body fluids, such as urine, make them ideal candidates for cancer detection and monitoring [9]. Among these, urinary extracellular vesicles (uEVs) have garnered significant attention due to their cargo of stable miRNAs, reflecting the molecular signatures of their parent cells [10]. One such example is the exploration of uEVs to detect miRNAs released by tumors, especially in urological cancers like bladder cancer (BCa) and prostate cancer (PCa), in the search for non-invasive methods to detect and monitor the disease [11]. Some of the most commonly used strategies are spike-in controls [12] like *cel-miR-39*, multiple endogenous miRNAs and their arithmetic means, the pair ratio method [13], and study-specific normalizers [14]. Such variable strategies raise significant concerns about the comparability and reliability of results across different studies. Therefore, it is essential to identify and validate a suitable normalizer for various studies, considering factors such as sample type, pre-analytical conditions, and post-analytical variables.

Existing tools like BestKeeper [15] use the standard deviation and coefficient of variation of Ct values to evaluate the stability of the candidate reference sequence, which is highly sensitive to outliers and does not deal with missing values. The Delta-Ct [16] algorithm uses intra-group variation, whereas NormFinder [17] uses a model-based approach that focuses on intergroup and intragroup variation, identifying the most stable reference gene. Another tool, geNorm [18], calculates the M-value for each gene, calculating pairwise variation with all the other candidate genes to identify the normalizer gene, which is also affected by the coregulation of genes. Many different tools use accumulated standard deviation, which also uses intergroup and intragroup variation assuming that the expression data is distributed normally and is therefore very sensitive to outliers. GenExpa [19] uses the coherence score to validate the reliability of the most stable reference gene, which is statistically sound but does not have a strategic mechanism to handle missing values and outliers. RefFinder, on the other hand, addresses a few of these challenges by integrating various existing tools like NormFinder, DeltaCt, BestKeeper, and GeNorm using a weighted approach that provides a robust selection of the most stable reference gene; yet it lacks the strength of handling missing values.

As such, these tools have limitations in addressing the unique challenges posed by presumptions of the nature of the data, statistical implications, null values, and graphical user interface for data visualization (Table 1).

Table 1. Detailed descriptions of existing limitations addressed by our proposed tool

Aspect	Limitations	Limitations addressed in gQuant tool
Missing values	Lack of methods to handle missing values, which raises concerns about final ranking.	A preprocessing unit addresses this issue and handles it strategically.
Different approaches	Using different statistical tools used by different methods leads to different interpretations of expression data.	Our tool employs multiple statistical methods, which enables multidimensional analysis from a distinctive angle.
Scaling	Literal values from different methods can cause dominance in weighted or other combined approaches.	The tool employs standard scaling to standardize the result in a range of 0 to 1, which cannot dominate the voting process.
Interactive graphics	Limited or no interactive graphics for gene ranking and expression data distribution.	Incorporates boxplots and KDE plots to visualize the distribution and density of data.
Democratic voting-based integration	Traditional tools like RefFinder use a weighted approach, which can lead to biases.	Our tool employs the democratic voting mechanism where each data point competes for the vote-based characteristics.

Therefore, the development of robust, consistent, and generalizable algorithms is essential for accurately identifying the most stable reference genes. Given this context, our effort to develop a tool to find the most stable reference gene and its validation on uEV-miRNAs represents a timely and crucial endeavor. In this cross-disciplinary research initiative, we synergized biological and mathematical fields.

While searching for a standard normalizer gene in a concurrent study, we screened selected miRNAs based on existing literature. This initial assessment involved experimental qRT-PCR, followed by analysis using existing tools like RefFinder. However, discrepancies in the results with current tools prompted us to develop a new, more robust analytical algorithmic model.

This paper introduces a novel and efficient algorithmic tool named gQuant, designed for analyzing qRT-PCR expression data to identify the most stable reference gene. We have validated our tool using a range of experimental data and publicly available datasets. Furthermore, we conducted a comparative analysis with current tools, pointing out their limitations, and detailed the methodological thoroughness that went into the development of this innovative analytical approach.

Tool description

The tool gQuant was developed in Python programming language using Jupyter Notebook Integrated Development Environment. Python is a well-known, widely used, free, and open-source programming language that is very efficient in data analysis and mathematical computations, thus providing a significant basis for qRT-PCR data analysis. This environment offers various automated libraries for statistical computation, mathematical operations, and data visualization. The related codes and libraries required to use this tool are available on a GitHub repository, which can be accessed using the link <https://github.com/ABHAYHBB/gQuant>.

For the gQuant tool, data needs to be processed in tabular format (Figure 1) for preprocessing, ranking of most stable reference genes, and data visualization. However, datasets can be used in different supported file formats like spreadsheets, comma-separated files, or text files in which table columns must represent the candidate reference genes and rows represent samples.

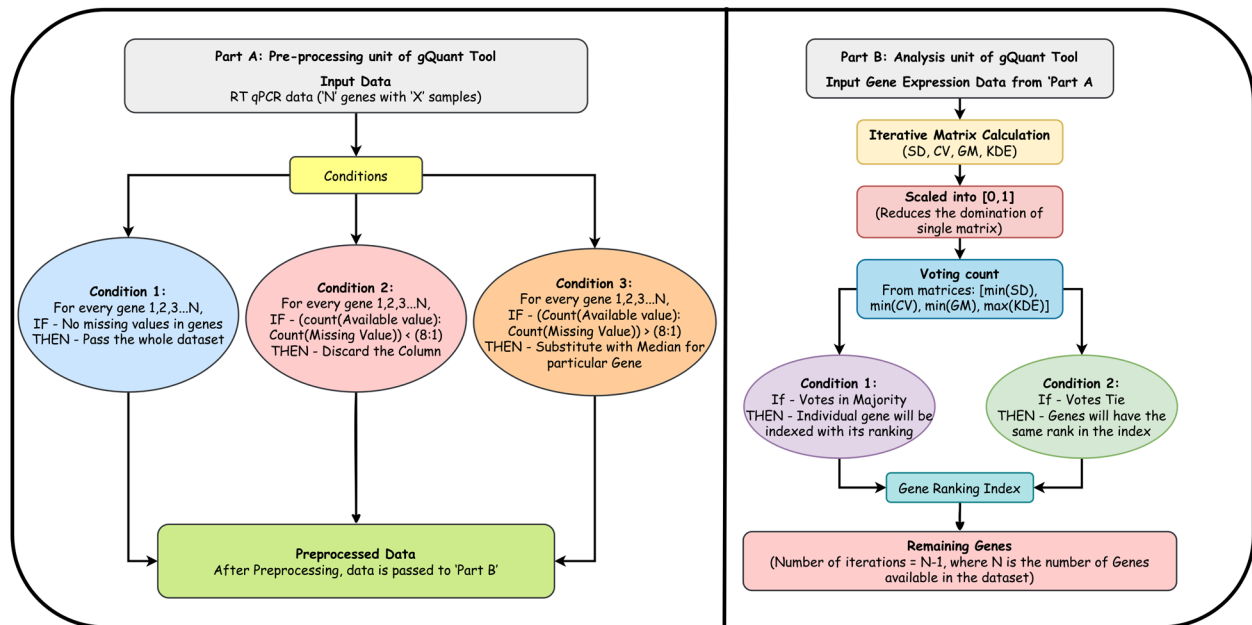


Figure 1. Workflow of gQuant tool. (Part A) Preprocessing unit of gQuant, which involves handling missing values, data imputation, and filtering to ensure data integrity before analysis. (Part B) Analysis unit of gQuant, where statistical computations are performed using multiple ranking metrics, followed by a voting-based integration approach to identify the most stable reference gene.

The functionality of gQuant is divided into two broad categories:

1. Preprocessing unit

Data preprocessing is an integral part of the gQuant tool, which initially processes the dataset. Once preprocessing is completed, statistical metrics are calculated, and the tool proceeds with ranking stable reference genes. In the initial preprocessing stage, the tool systematically processes the input dataset by identifying and quantifying missing or null values within each gene column. For every column, it calculates the ratio of available values to missing values (NA), which plays a crucial role in guiding the next steps of the workflow. The user must provide the ratio of available to missing value (NA) for each column for the preprocessing part (see Troubleshooting section). It decides whether an imputation process will happen or if a particular column will be excluded from the further ranking process due to the high number of missing values. Throughout this paper, we use a default ratio threshold of 8:1 to demonstrate the results. If this ratio exceeds a user-defined threshold, the tool engages in a missing value imputation process, replacing the missing values with the median of the available values in the corresponding column. This approach ensures that incomplete data can still contribute meaningfully to subsequent analyses. However, if the ratio falls below the threshold, the program deems the column unreliable and removes it from the dataset entirely. This mechanism iteratively evaluates all gene columns, employing an imputation strategy that assumes the dataset has different counts of missing values. The flexibility of this preprocessing step allows the threshold to be adjusted based on specific problem requirements; in our case, an 8:1 ratio was selected to balance data retention and accuracy. By methodically addressing missing values, this preprocessing procedure forms the first stage of the tool's operational framework, establishing a robust foundation for downstream analyses, as described in detail in Part A (Figure 1).

2. Analysis unit

To design a model that effectively integrates the advantages of current methods while addressing their

limitations, we employed a systematic approach to optimize the performance evaluation framework, enabling more precise and reliable gene selection. This process is elaborated in the following sections. The tool utilizes preprocessed data from Part A as input (Figure 1). It calculates all metrics and scales their values to the [0,1] range, ensuring no single metric disproportionately influences the voting process. The majority voting integration strategy is then applied to identify the most stable reference gene among the candidate reference genes. Two scenarios arise during this process: in the event of a tie in the first voting round, the tied genes are assigned equal rankings in the index; if a majority vote favors a specific gene, it is recorded in the ranking index. The remaining genes undergo repeated iterations until a single gene remains. Required mathematical notations for the gQuant tool algorithm and their descriptions are shown in Table 2.

Table 2. Mathematical notations

Symbol	Description
G	Sets of genes $G = \{G_1, G_2, \dots G_n\}$.
X	Matrix of gene expression values; X_{ij} denotes the expression value of gene i in j samples.
σ	Standard deviation vector $\sigma = \{\sigma_1, \sigma_2, \dots \sigma_n\}$, where σ_i is the standard deviation of gene i .
GM	Geometric mean vector $GM = \{GM_1, GM_2, \dots GM_n\}$ where GM_i is the standard deviation of gene i .
CV	Covariance vector $CV = \{CV_1, CV_2, \dots CV_n\}$, where CV_i is the standard deviation of gene i .
KDE	Kernel density estimation vector $KDE = \{KDE_1, KDE_2, \dots KDE_n\}$, where KDE_i is the standard deviation of gene i .
R	Ranking index list $R = \{(g_1, r_1), (g_2, r_2) \dots (g_n, r_n)\}$, where g_i is the ranking of the gene.

For an explanation of the algorithm of gQuant, Table 3 shows the step-by-step process, definitions, and technical details.

Table 3. Steps of the gene ranking algorithm

Step	Description
1. Data reading	Read the gene expression data, extracting gene names and their numerical values.
2. Scaling	Scale all the matrix vectors SD , GM , CV , and KED to ensure values are in [0,1] range, so no single metric dominates in the voting process.
3. Iterative calculation	For each remaining gene g_i , from G_1 to G_n , compute the following metrics: SD , GM , CV , and KDE . These metrics are calculated iteratively using their respective equations.
4. Voting	Each matrix elects the gene based on: Minimum value of SD , GM , and CV . Maximum value of KDE .
5. Majority rule	The individual gene who received most votes across all metrics is: Removed from the current dataset X . Assigned rank and added to the ranking index R . Rank counter r increased by one.
6. Tie breaking	If more than one gene received the same number of votes: All tied genes are removed from the dataset X together. All removed genes will be assigned the same rank in the ranking index.
7. Recursion	Steps 3–6 are repeated iteratively until one gene remains in the dataset.
8. Final index	The remaining gene/genes are added to R , with the final rank r .

Equipment

1. For optimal performance, gQuant requires a system with a multi-core processor (Intel Core i5 or equivalent and above) and 8 GB RAM and above to efficiently handle computation. The tool is compatible with Windows, macOS, and Linux-based operating systems supporting Python 3.7 or higher. Additionally, using the Conda distribution with Jupyter Notebook (version 6.0 and above, optional) is recommended for an integrated development environment.

Software and datasets

Python software environment (≥ 3.6) (<https://www.python.org/>)

Anaconda distribution with Jupyter Notebook integrated development environment (≥ 6.0 , optional) (<https://www.anaconda.com/>)

All data and code have been deposited to GitHub: <https://github.com/ABHAYHBB/gQuant>

1. Datasets can be prepared in different ways, and the scope of the protocol does not cover the processing of raw data. Generally, data should be cleaned, quality checked, and ready to be used.




2. Supported file formats for the tool can be, e.g., Excel files (.xlsx), comma-separated files (.csv), text files (.txt), and data frames (df).

3. After obtaining the qRT-PCR data, input source type does not matter, and the following requirements are the same:

- Columns of the dataset must represent the name of the gene/target and the row represents the sample.
- The name of the rows (samples) must be removed.
- For the best practice, avoid having spaces in the column names. It could cause unnecessary problems.
- If the dataset has missing values, it will not affect the result because it is carefully designed to handle it.

4. The datasets used here to show the example results are the same as what was used in the original article. These three datasets collectively validate gQuant across different molecular targets (mRNA vs. miRNA), sample types (cell lines vs. patient samples), and experimental constraints (small vs. large datasets, missing values). By demonstrating its robustness in diverse conditions, we confirm that gQuant can be reliably applied to various qRT-PCR-based normalization scenarios.

a. Dataset 1: Derived from research on OvCar-3 and PC-3 cancer cell lines, this dataset encompasses transcriptomic profiling of 84 genes pertinent to cell regulation and five conventional housekeeping genes (Figure 2). The data, accessible via GSE57888, serves as a benchmark for mRNA-based qRT-PCR analysis, offering validation for gQuant through non-normalized data. It contains un-normalized Ct values, where column names represent gene names, and rows correspond to samples without explicit row labels. Data also have undetermined or missing values. The file can be accessed using the link <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE57888>.

K18    28.47458798

	A	B	C	D	E	F	G	H	I
1	hsa-miR-346	hsa-miR-196b	hsa-miR-214	hsa-miR-643	hsa-miR-608	hsa-miR-376b	hsa-miR-944	hsa-miR-523	hsa-miR-938
2	24.91818341	32.53782541	29.55351474	32.55316453	28.93691752	33.44517513	40.00470059	33.56234394	33.83767025
3	32.17050872	36.06820533	30.96981682	34.5549374	28.91851853	31.89384873	36.7524326	33.17364223	32.64616679
4	26.65645551	32.60207395	30.56862453	33.8439056	30.52070819	38.10147152	Undetermined	34.474804	33.97168249
5	29.53351672	31.64085671	32.60324833	34.15398863	26.89834958	Undetermined	33.45947922	Undetermined	38.13253542
6	25.35759246	Undetermined	28.69919398	33.88817715	28.83450868	33.90737073	Undetermined	Undetermined	33.53501536
7	27.99376232	34.51247311	29.79337565	35.37969625	30.64556024	38.66110818	Undetermined	33.80967147	36.03938737
8	27.46895359	32.7877752	30.86969135	34.5743333	29.75371529	35.83678122	Undetermined	Undetermined	34.16281713
9	27.64538578	34.50077499	33.63255952	36.59854416	30.517021	Undetermined	Undetermined	36.33557022	34.49876755
10	26.22079973	32.15607049	30.94345633	34.18006617	29.59610727	33.92743822	Undetermined	32.97240551	33.07048459
11	25.87659665	31.40877868	29.85636516	34.61588475	29.08876604	33.89486817	37.91818122	35.16373584	32.28675814
12	29.53522356	32.3504539	33.86440878	37.18462673	29.96864606	34.84616114	34.77661195	Undetermined	38.38208239
13	28.6335419	33.80311025	31.90947707	37.23274615	27.69425966	38.73358857	Undetermined	Undetermined	35.1422153
14	26.66751364	32.58472713	34.88463249	33.08659486	28.1345116	37.2434108	37.1285603	33.67628943	33.49484495
15	28.19255487	31.75155895	34.74486159	Undetermined	25.11733635	38.3655435	Undetermined	Undetermined	Undetermined
16	27.43579976	30.36289808	31.23164093	Undetermined	32.08984777	36.1883601	32.53714977	Undetermined	35.17785986
17	28.30588348	35.19480854	35.42489052	Undetermined	29.71151035	37.7170097	Undetermined	Undetermined	35.86378639
18	30.333737	34.69129777	Undetermined	Undetermined	29.98094089	32.76678053	Undetermined	Undetermined	33.22051286
19	30.08627422	Undetermined	34.6790194	Undetermined	30.82644344	Undetermined	Undetermined	Undetermined	Undetermined
20	27.71251012	37.00234061	35.33910847	Undetermined	31.09386973	Undetermined	Undetermined	34.35388178	35.08485642
21	28.74572715	30.72523774	30.2113541	32.37274485	26.99621318	33.4592142	Undetermined	32.3842928	Undetermined
22	32.69024458	Undetermined	38.78974276	Undetermined	31.44605843	Undetermined	Undetermined	Undetermined	36.20581696
23	32.00582458	32.77105764	Undetermined	Undetermined	28.98922574	35.39636603	Undetermined	Undetermined	Undetermined
24	27.70761625	29.53337804	31.34107291	31.80378658	27.92441919	32.51341472	34.18360032	Undetermined	Undetermined
25	33.26175436	37.06751412	35.4086191	Undetermined	31.59174616	40.33681027	Undetermined	Undetermined	Undetermined
26	28.28598881	29.41931223	32.15792663	34.00571476	29.92033938	35.03181837	Undetermined	Undetermined	Undetermined
27	31.35003268	28.47439838	33.28150373	Undetermined	28.61988581	38.58828315	35.02634952	Undetermined	Undetermined
28	26.23686508	31.75749414	30.72523681	33.82493029	29.68616269	Undetermined	Undetermined	Undetermined	34.15213638
29	23.86833593	27.85594637	26.86390196	31.05264188	27.40024723	31.71238543	33.77712037	30.92730033	31.2255938
30	24.16697709	30.27915525	29.27874698	32.12773573	27.31549519	34.72365678	37.62009573	33.46622663	32.34027306
31	26.77087961	33.8722592	32.11820376	38.08512449	24.14371176	34.4693428	35.11830416	Undetermined	Undetermined
32	26.11761417	32.99521617	30.92439086	32.74703481	30.50347526	36.76856565	35.47802255	34.6649804	34.286752
33	26.15420042	28.73420065	28.58225475	33.51125637	30.45385857	33.69297545	35.57381922	Undetermined	33.35417717
34	24.73801086	30.97191336	30.28884215	32.87468296	29.97489018	35.17866913	Undetermined	34.00109849	33.00909051
35	25.85690492	32.51080977	30.84184699	34.36370333	31.1083376	36.12672162	37.30555069	34.67931182	33.92059737
36	26.6070564	32.96065666	32.29256315	36.30525463	31.44169908	37.65383287	Undetermined	35.66710657	33.37989309
37	26.13938544	31.98036401	31.50763594	33.87230847	29.96765714	34.42760914	40.88065703	33.21895286	33.7216135

◀ ▶ Sheet2 Sheet3 Sheet4 +

Figure 2. Example of expression data in Excel file (.xlsx). The dataset shown in the figure is Dataset 1 as defined in this article.

b. Dataset 2: We chose the dataset GSE239868 with more detailed variables, such as the expression of 1,066 human miRNAs and $n = 36$ tracheal aspirate samples, for a comparative efficiency assessment of gQuant. This database included data for a comparative analysis on the management of missing values and allowed us to test the appropriateness of our method on miRNA background. This dataset's unnormalized Ct values provide a strict testing environment. The file can be accessed using the link <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi>.

c. Dataset 3: In an endeavor to construct a more robust analytical framework, we decided to explore the potential of miRNA-based normalizers as an alternative to the well-established mRNA-based normalizers. Dataset 3 is also a testing environment for small datasets. Given the nascent state of miRNA-based normalization data, we focused specifically on uEVs. A comprehensive literature review and Qiagen's Human Urine Exosomes Focus miRCURY LNA Panel list were used to select the miRNA set used in this study for validation. This investigation led to the selection of *miR-16-5p*, *miR-10b-5p*, *30b-5p*, and *miR-30d-5p*. Additionally, upon evaluating the expression levels of the Let7c cluster genes from another ongoing study, we observed high stability for *let-7c-5p* across both diseased and control samples, leading to its inclusion in subsequent investigations. Therefore, our preliminary study proceeded to focus on the selected miRNAs: *let-7c-5p*, *miR-16-5p*, *miR-10b-5p*, *miR-30a-5p*, and *miR-30d-5p*. Using qRT-PCR, we assessed their expression in uEVs samples of BCa ($n = 9$), PCa ($n = 6$), and control

samples ($n = 3$). Among the evaluated miRNAs, *let-7c-5p* has emerged as the most promising candidate for normalizer functions. Subsequently, expression levels for *let-7c-5p* miRNAs in uEVs were quantified in an expanded sample set ($n = 30$) using qRT-PCR. Rigorous quality control measures, including primer efficiency and melt curve analysis, were implemented, and any data with Ct values over 38 were excluded. The stability of the miRNA expression was validated using gQuant, solidifying *let-7c-5p*'s potential as a reliable normalizer. This dataset was uploaded to the gQuant tool GitHub repository (<https://github.com/ABHAYHBB/gQuant>) (Dataset S1. Dataset 3).

Procedure

A. Installing Python and Jupyter

Installing Python and Jupyter Notebook is not covered by the presented protocol; however, detailed guidance on how to install them can be found on their respective official website.

Python: <https://www.python.org/>

Jupyter Notebook: <https://www.anaconda.com/>

B. Installing the required dependencies for the tool (File S1)

The gQuant tool has a few dependencies that need to be installed.

1. Data preprocessing: These libraries are used for data manipulation and handling missing values.

Pandas: `!pip install pandas`

NumPy: `!pip install numpy`

2. gQuant tool computation: These libraries are used for statistical computations, scaling, and voting classifiers.

SciPy: `!pip install scipy`

Scikit-Learn: `!pip install scikit-learn`

3. Data visualization: These libraries are used for generating publication-ready interactive and static visualizations.

Bokeh: `!pip install scikit-learn`

Plotly: `!pip install scikit-learn`

C. Importing the qRT-PCR expression data

Code is given below for various supported file formats:

1. Spreadsheet (Excel):

```
import pandas as pd
df_excel = pd.read_excel('path_to_your_file.xlsx', sheet_name = 0, header = 0)
```

Parameter explanation: Change this parameter as your path and file name “'path_to_your_file.xlsx”,

provide sheet name in this parameter “`sheet_name = 0`” (it can either be a number or name of the sheet), and the header parameter “`header = 0`” should be set to 0 as it reads the first row as a column name.

2. Text file (plain text):

```
df_text = pd.read_csv('path_to_your_file.txt', delimiter = '\t', header = 0)
```

Parameter explanation: Change this parameter as your path and file name “`'path_to_your_file.txt'`” and the delimiter parameter “`delimiter = '\t'`” is used to set the delimiter sign, which is used in the text file.

3. Comma-separated (CSV) files:

```
df_csv = pd.read_csv('path_to_your_file.csv', header=0)
```

Parameter explanation: Change this parameter as your path and file name “`path_to_your_file.txt`” and the header parameter “`header=0`” is set to 0, assuming the first row has the column names.

D. Data preprocessing

1. For data preprocessing, the user needs to import the libraries that have been installed earlier in this section.

```
import pandas as pd
import numpy as np
```

2. Copy the function code “`preprocessing.py`” from the GitHub repository of the tool, paste it into the Jupyter notebook, and run the function (Figure 3). Figure 4 presents the dummy expression data with missing values (NAs).

3. To use the preprocessing function, use the function call syntax with the imported file name as “`df`”.

```
results, df1 = preprocessing(df)
```

4. To display the results, use the following syntax

```
print("Overall Results:", results)
```

to print the overall results showing the count ratio of missing values to available values of the expression data (Figure 5).

```
print("Updated File:", df1)
```

This will print the initial samples of preprocessed data named “`df1`” (Figure 6).

gQuant / preprocessing.py

ABHAYHBB Update preprocessing.py

Code Blame 54 lines (39 loc) · 2.23 KB

```

1  import pandas as pd
2  import numpy as np
3
4  def preprocessing(df):
5      """
6      Analyze a DataFrame to calculate the available-to-missing value ratio for each column
7
8      Parameters:
9      df (pd.DataFrame): Input DataFrame to analyze.
10
11     Returns:
12     results (pd.DataFrame): DataFrame with Available_Count, Missing_Count, and Available_
13     df1 (pd.DataFrame): Subset of columns with Available_to_Missing_Ratio >= 8 or with nc
14     """
15
16     # Define a function to check if a value is not a number (NaN or any non-numeric type)
17     def is_not_number(x):
18         return not (isinstance(x, (int, float)) or pd.isna(x))
19
20     # Apply the function across the DataFrame to identify non-numeric or missing values
21     not_number = df.applymap(is_not_number)
22
23     # Count 'True' values returned by the function (which indicate non-numeric values)
24     not_number_count = not_number.sum()
25
26     # Count the numeric and non-missing values in each column
27     numeric_and_available_count = df.applymap(lambda x: isinstance(x, (int, float)) and r
28
29     # Calculate the ratio of available values to missing values for each column
30     available_to_missing_ratio = numeric_and_available_count / not_number_count.replace(
31

```

Figure 3. Snapshot of the GitHub repository of gQuant showing preprocessing function code

	Gene1	Gene2	Gene3	Gene4	Gene5	Gene6	Gene7
0	1.2	1.1	1.0	1.3	1.5	1.4	1.6
1	2.3	2.2	2.1	2.4	2.6	2.5	2.7
2	3.4	NaN	3.2	3.5	3.7	3.6	3.8
3	4.5	4.4	NaN	4.6	4.8	4.7	4.9
4	5.6	5.5	5.3	5.7	5.9	5.8	5.0
5	6.7	6.6	6.4	6.8	6.0	6.9	6.1
6	7.8	7.7	7.5	7.9	7.1	7.0	7.2
7	8.9	8.8	8.6	8.0	8.2	8.1	8.3
8	9.0	9.9	NaN	9.1	9.3	9.2	9.4
9	10.1	NaN	10.7	10.2	10.4	10.3	10.5
10	11.2	11.0	11.8	11.3	11.5	11.4	11.6
11	12.3	12.1	12.9	12.4	12.6	12.5	12.7
12	NaN	13.2	NaN	13.5	13.7	13.6	13.8
13	14.5	14.3	14.1	14.6	14.8	14.7	14.9
14	15.6	15.4	15.2	15.7	15.9	15.8	15.0

Figure 4. Initial dummy expression data with missing values

	Available_Count	Missing_Count	Available_to_Missing_Ratio
Column			
Gene1	14	1	14.0
Gene2	13	2	6.5
Gene3	12	3	4.0
Gene4	15	0	NaN
Gene5	15	0	NaN
Gene6	15	0	NaN
Gene7	15	0	NaN

Figure 5. Output of the preprocessing function on dummy expression data as a data frame “result”, which uses an exclusion ratio threshold of 8:1 (available values to missing values)

	Gene1	Gene4	Gene5	Gene6	Gene7
0	1.2	1.3	1.5	1.4	1.6
1	2.3	2.4	2.6	2.5	2.7
2	3.4	3.5	3.7	3.6	3.8
3	4.5	4.6	4.8	4.7	4.9
4	5.6	5.7	5.9	5.8	5.0
5	6.7	6.8	6.0	6.9	6.1
6	7.8	7.9	7.1	7.0	7.2
7	8.9	8.0	8.2	8.1	8.3
8	9.0	9.1	9.3	9.2	9.4
9	10.1	10.2	10.4	10.3	10.5
10	11.2	11.3	11.5	11.4	11.6
11	12.3	12.4	12.6	12.5	12.7
12	7.5	13.5	13.7	13.6	13.8
13	14.5	14.6	14.8	14.7	14.9
14	15.6	15.7	15.9	15.8	15.0

Figure 6. Final data frame with filtered columns and data imputation for “Gene1”

5. The preprocessing function code given in the tool uses a default exclusion ratio threshold of 8:1 of available values to missing values. This ratio can be changed according to the user’s needs. To change this ratio, the user can modify the function code as follows:

```
df1=results[(results['Available_to_Missing_Ratio']>=8)|(results['Missing_Count'] == 0)]
```

6. To save this “df1” file in CSV format, use this syntax

```
df1.to_csv('df1.csv', index=True)
```

E. gQuant tool

1. To use the gQuant tool, the user needs to import the necessary libraries that were installed earlier in this section.

```
import pandas as pd
import numpy as np
```

```
from scipy.stats import gmean
from sklearn.neighbors import KernelDensity
from sklearn.preprocessing import MinMaxScaler
```

2. Copy the function code “tool.py” from the GitHub repository, paste it into the Jupyter notebook, and run the function (Figure 7).

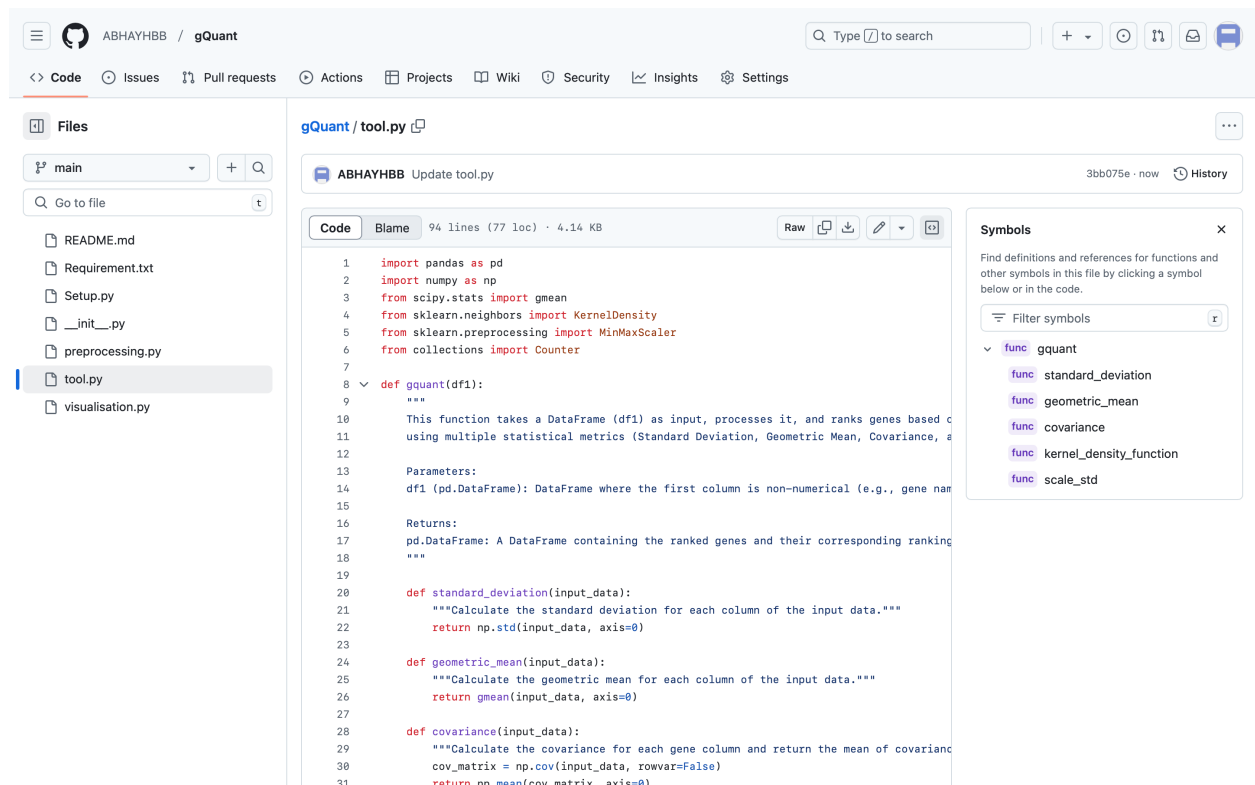


Figure 7. Snapshot of the GitHub repository of gQuant, showing the tool function code

3. After running the tool function “tool.py”, use the syntax given below to call the function passing the data frame “df1”, which was preprocessed and downloaded from the preprocessing function.

```
ranking_df = gquant(df1)
```

After calling the function, the output will be in table format (Figure 8). If there are multiple genes, all the rows of the table may not be visible. To see the full ranking table, use the next instruction.

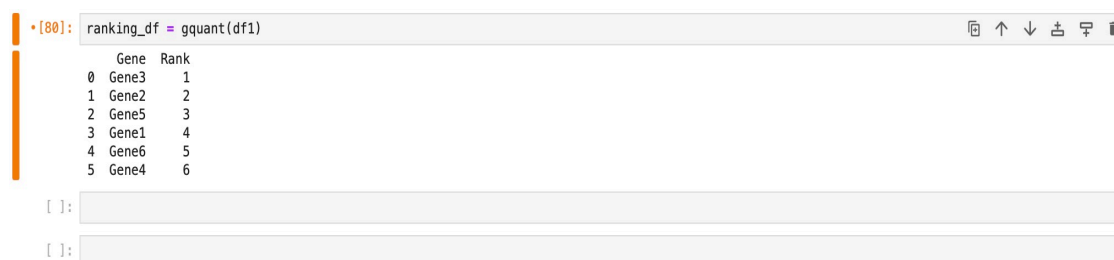


Figure 8. Example output of gQuant tool

4. To save the ranking table that was generated by the “tool.py” function, use the following syntax:

```
ranking_df.to_csv('ranking_table.csv', index=False)
```

F. Data visualization

1. For data visualization (boxplot and KDE plot), import the libraries that were installed earlier in this section.

```
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
import numpy as np
from scipy.stats import norm
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import LabelSet, ColumnDataSource
```

2. Copy the function code “visualization.py” from the GitHub repository, paste it on the Jupyter notebook, and run the function (Figure 9).

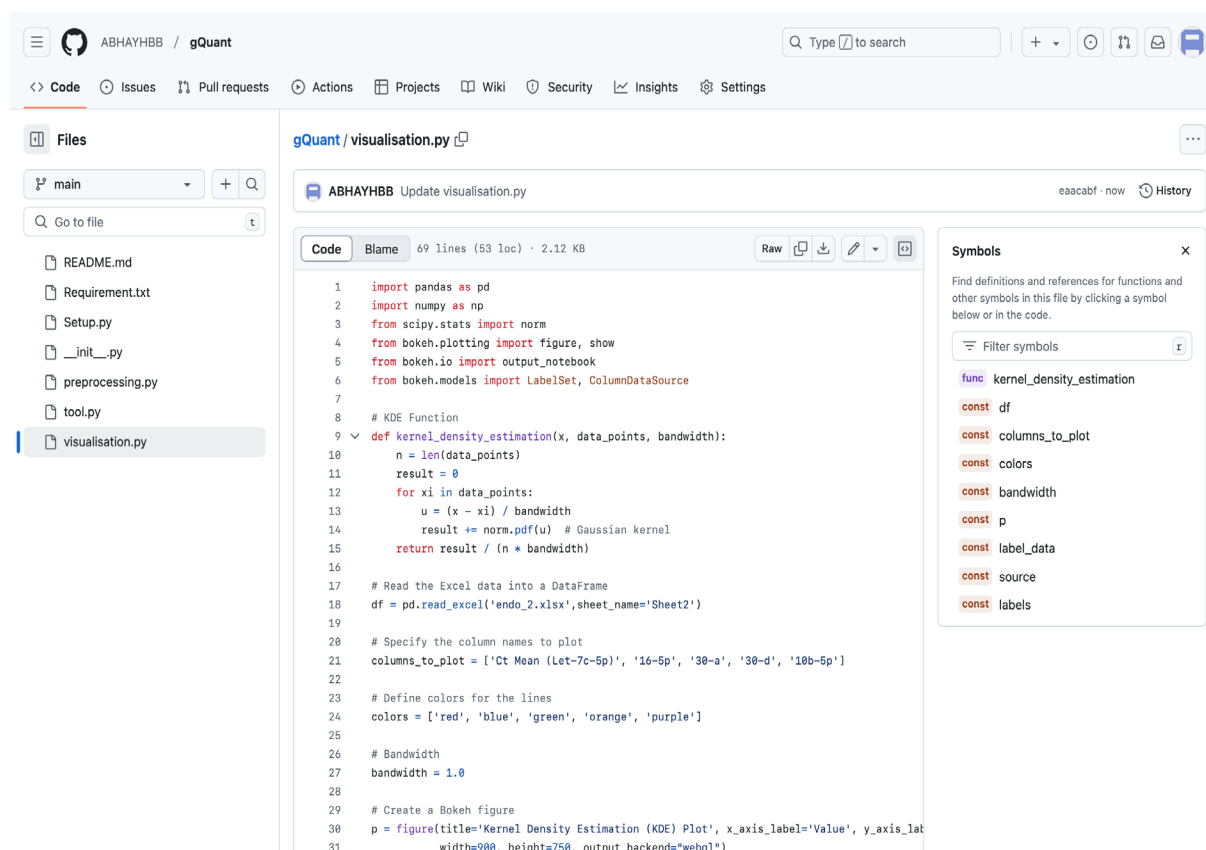


Figure 9. Snapshot of the GitHub repository of gQuant, showing the visualization function code

3. To create a boxplot and KDE plot, the user needs to call the boxplot function as given below:

```
create_box_plot(df1)
```

```
create_kde_plot(df1)
```

After calling these visualization functions, the user will see the output as two different plots based on the dataset. The high-quality .png images will be saved on the local file system and named “kde_plot.png” and “box_plot.png”. The boxplot in Figure 10 shows the expression differences and distributions of Dataset 3.

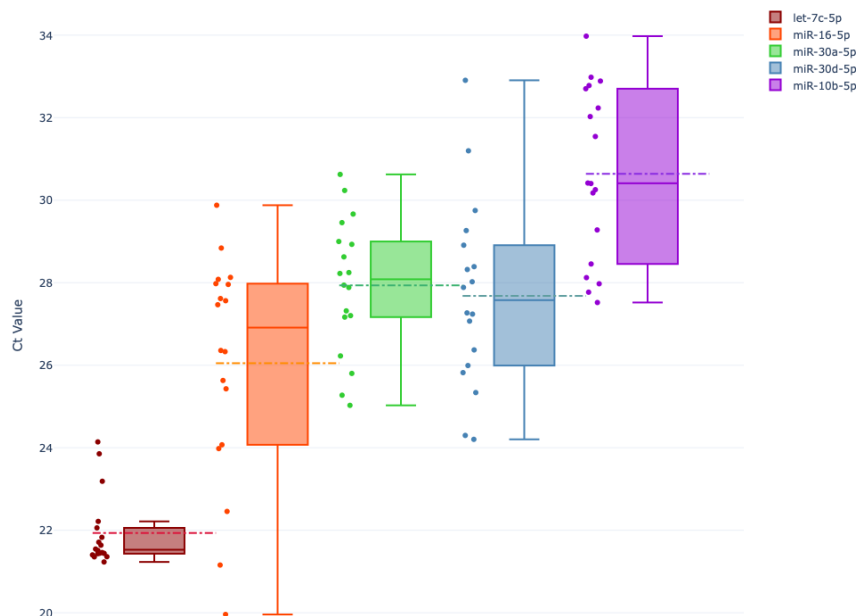


Figure 10. Expression levels presented in the form of boxplot using Dataset 3

The KDE plot in Figure 11 demonstrates the density of the data points applied on Dataset 3.

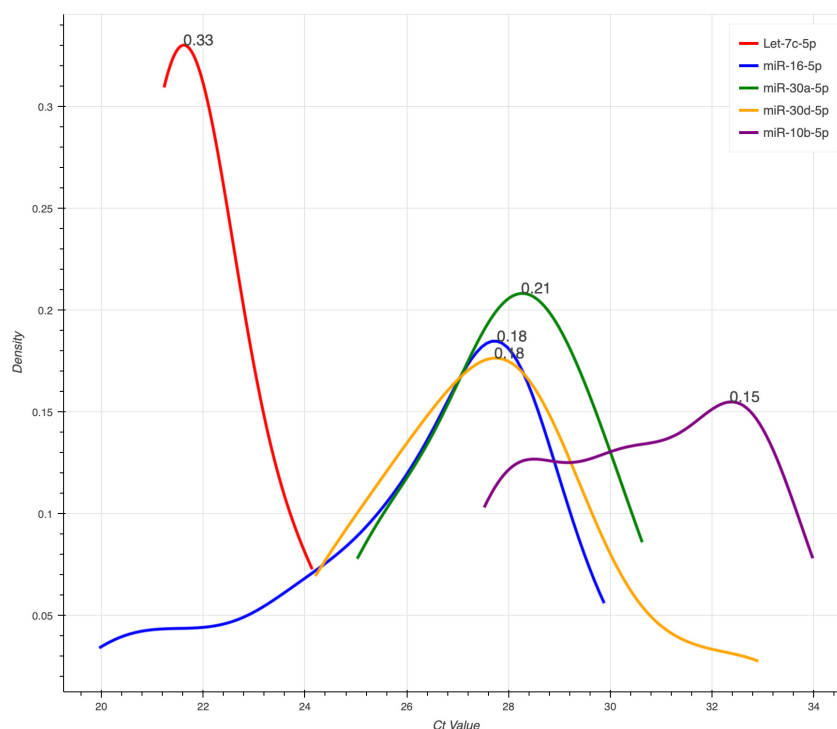


Figure 11. Kernel density estimation level plot using Dataset 3

Data analysis

The step-by-step guide “Tutorial Document” is provided in File S1, which is user-friendly even for beginners in Python programming. This also provides options for exporting ready-to-publish figures and tables.

Validation of protocol

This protocol or parts of it has been used and validated in the following research article:

- Pathak et al. [20]. Development of a robust and generalizable algorithm "gQuant" for accurate normalizer gene selection in qRT-PCR analysis. *Scientific Reports*. (Figure 2 and Tables 1, 2, 3, and 4).

General notes and troubleshooting

General notes

1. The gQuant tool is created using Python programming environment. Users should have a basic understanding of the Python environment and any integrated development environment (preferred: Jupyter notebook).
2. Before using the gQuant tool, make sure you have the dataset in the right format.
3. A Python environment setup is necessary to utilize the tool efficiently. The user can search in the gQuant GitHub repository for dependencies. Make sure that you have all possible libraries installed and imported in the correct version as specified.
4. Be aware of the compatibility of the environment; updating the software and libraries reduces error occurrences when running the algorithm code.
5. The visualizations created by the functions given here create high-quality pictures. Users can also change the format with small changes in the code. This type of information is not included in this protocol.
6. For optimal stability ranking, we recommend using at least three technical replicates per gene and three biological replicates per group, although gQuant can process datasets of any size and gene count.
7. To mitigate sample size constraints, publicly available datasets can be integrated into the analysis. However, users should carefully consider the following criteria to ensure data compatibility and reliability:
 - Ensuring comparable experimental conditions.
 - Checking data normalization and quality control steps.
 - Addressing potential batch effects that could introduce bias.
 - By supplementing small-scale experimental data with validated public datasets, researchers could improve the robustness of stability rankings in cases where the number of experimental replicates is limited.
8. gQuant is compatible with Ct data obtained from both SYBR Green and TaqMan probe-based qRT-PCR assays. The tool operates on unnormalized Ct values using robust statistical methods that are independent of the probe chemistry. While experimental validation has been primarily conducted with SYBR Green data, the underlying statistical framework ensures consistent performance across both detection methods.

Troubleshooting

Problem 1: Error in missing value imputations.

Possible cause: A dataset containing non-numeric entities or incorrectly formatted values.

Solution: Make sure that all the entries are in numerical format; if not, make sure to format them correctly.

Problem 2: Incorrect gene rank output.

Possible cause: Improper preprocessing of data.

Solution: Verify that data holds the desired format of information and has been preprocessed using the preprocessing tool. Also, make sure that column-to-gene structure is being maintained in the process.

Problem 3: Failed to generate visualizations (boxplot/KDE plot).

Possible cause: Missing libraries or incorrect function.

Solution: Ensure that all the required libraries were installed and imported in the correct way. Make sure that the correct data is being passed to visualization functions.

Problem 4: Tool crashes while performing gene ranking of large amounts of expression data.

Possible cause: Insufficient memory/computational resources.

Solution: Optimize by removing duplicates (if any). Use a high-end machine with good computational resources.

Problem 5: No output or empty ranking table.

Possible cause: Dataset filtering during preprocessing due to high missing value ratios.

Solution: Adjust to the appropriate threshold. Make sure that columns are not entirely missing.

Problem 6: Handling Ct variability and setting thresholds in gQuant.

Possible cause: Ct values in qRT-PCR datasets can be influenced by various experimental factors, such as sample quality, RNA extraction methods, qPCR platforms, fluorophores, and primer efficiencies. Additionally, unnormalized Ct values require users to define appropriate thresholds for data processing, which may introduce inconsistencies if not properly managed.

Solution: To address this, we provide the best practices for users to enhance data comparability and improve analysis consistency when using gQuant. Users should follow these guidelines:

- Use the same qPCR instrument and reagents for all samples to reduce inter-run variability.
- Maintain consistent RNA extraction and reverse transcription protocols to prevent batch effects.
- Verify primer efficiency and specificity through melt curve analysis and standard curve generation.
- Avoid including Ct values from degraded or low-quality RNA samples, as these can introduce bias in the normalization process.
- gQuant automatically imputes missing Ct values if their presence is below a user-defined threshold. However, users should manually inspect missing values to determine if imputation is appropriate or if certain genes should be excluded. The default gQuant setting removes genes where the available-to-missing value ratio is below 8:1. Users can adjust this threshold based on dataset size and experimental constraints.
- Identify and remove extreme Ct values that could arise from technical errors.
- While gQuant operates on raw Ct values, users should ensure that their dataset does not include systematic biases.
- Exclude Ct values above 38, as these indicate low template presence and unreliable amplification.

Supplementary information

The following supporting information can be downloaded [here](#):

1. Dataset S1. Dataset 3
2. File S1. Tutorial Document (Step-by-step guideline for using the gQuant tool)

Terminology clarification

- Candidate reference sequence: Refers to any target or miRNA being analyzed with RT-PCR.
- Candidate reference sequence, reference gene, normalizer: These terms are used interchangeably throughout the manuscript.
- Most stable reference gene: Specifically used when discussing “candidate reference sequences” that are stably expressed across samples.
- mRNA-based normalization: Using genes as reference targets.
- miRNA-based normalization: May involve intergenic or non-coding RNA sequences.
- Missing values (NA): Refers to the absence of data in a dataset, represented as NaN, null, or NA.
- Multiple statistical methods: Refers to various techniques used to analyze data, such as SD, CV, KDE, and GM to draw insights from the data.
- Python: A high-level programming language widely used in data science, AI, and machine learning for its simplicity and extensive libraries.
- Jupyter: An open-source web application for creating and sharing documents that contain live code, equations, visualizations, and narrative text, commonly used for data analysis and scientific computing.
- Library: A collection of pre-written code in Python or other languages that provides specific functionality, such as NumPy, Pandas, and Matplotlib for data manipulation and analysis.

Acknowledgments

Conceptualization, Abhay Kumar Pathak, Garima Jain.; Investigation, Abhay Kumar Pathak, Garima Jain.; Writing—Original Draft, Abhay Kumar Pathak, Sukhad Kural, Shweta Singh, Garima Jain.; Writing—Review & Editing, Abhay Kumar Pathak, Garima Jain; Funding acquisition, Garima Jain.; Supervision, Garima Jain, Lalit Kumar, Manjari Gupta.

The authors would like to extend their gratitude to the Institute of Eminence (IoE), Banaras Hindu University for their financial support through IoE seed grant; they are also thankful to ACG JanCare fund for their financial support. We acknowledge the MIRNOW LLP for providing a Junior Research Fellowship (JRF) and Banaras Hindu University for a Non-Net fellowship.

This tool was published as an original article in Scientific Reports (Pathak et al. [20], DOI: 10.1038/s41598-024-66770-y).

Competing interests

The authors declare no conflicts of interest.

Ethical considerations

Dataset 1 and Dataset 2 were obtained from GEO (Gene Expression Omnibus), a public dataset. For the samples considered as Dataset 3, informed consent for participation was obtained from all individuals. Study protocols and samples were approved by the Human Ethical Committee, Institute of Science, BHU (Registration number ECR/226/indt/UP/2014/RR-22) and the Institutional Ethics Committee, Institute of Medical Sciences, BHU (Registration number ECR/526/inst/UP/2014/RR-20), confirming that all experiments were performed in accordance with relevant guidelines and regulations.

Received: January 09, 2025; Accepted: April 02, 2025; Available online: April 11, 2025; Published: May 05, 2025

References

1. Ye, J., Xu, M., Tian, X., Cai, S. and Zeng, S. (2019). Research advances in the detection of miRNA. *J Pharm Anal.* 9(4): 217–226. <https://doi.org/10.1016/j.jpha.2019.05.004>
2. Harshitha, R. and Arunraj, D. R. (2021). Real-time quantitative PCR: A tool for absolute and relative quantification. *Biochem Mol Biol Educ.* 49(5): 800–812. <https://doi.org/10.1002/bmb.21552>
3. Smith, T. A. D., AbdelKarem, O. A., Irlam-Jones, J. J., Lane, B., Valentine, H., Bibby, B. A. S., Denley, H., Choudhury, A. and West, C. M. L. (2020). Selection of endogenous control genes for normalising gene expression data derived from formalin-fixed paraffin-embedded tumour tissue. *Sci Rep.* 10(1): 17258. <https://doi.org/10.1038/s41598-020-74380-7>
4. Suzuki, T., Higgins, P. J. and Crawford, D. R. (2000). Control Selection for RNA Quantitation. *BioTechniques.* 29(2): 332–337. <https://doi.org/10.2144/00292rv02>
5. Shen, Y., Li, Y., Ye, F., Wang, F., Wan, X., Lu, W. and Xie, X. (2011). Identification of miR-23a as a novel microRNA normalizer for relative quantification in human uterine cervical tissues. *Exp Mol Med.* 43(6): 358–366. <https://doi.org/10.3858/emmm.2011.43.6.039>
6. Veryaskina, Y. A., Titov, S. E. and Zhimulev, I. F. (2022). Reference Genes for qPCR-Based miRNA Expression Profiling in 14 Human Tissues. *Med Princ Pract.* 31(4): 322–332. <https://doi.org/10.1159/000524283>
7. Jacobsen, K. S., Nielsen, K. O., Winther, T. N., Glebe, D., Pociot, F. and Høgh, B. (2016). Identification of valid reference genes for microRNA expression studies in a hepatitis B virus replicating liver cell line. *BMC Res Notes.* 9(1): 38. <https://doi.org/10.1186/s13104-016-1848-2>
8. Bhatia, A., Upadhyay, A. K. and Sharma, S. (2023). miRNAs are now starring in “No Time to Die: Overcoming the chemoresistance in cancer.” *IUBMB Life.* 75(3): 238–256. <https://doi.org/10.1002/iub.2652>
9. Condrat, C. E., Thompson, D. C., Barbu, M. G., Bugnar, O. L., Boboc, A., Cretoiu, D., Suci, N., Cretoiu, S. M. and Voinea, S. C. (2020). miRNAs as Biomarkers in Disease: Latest Findings Regarding Their Role in Diagnosis and Prognosis. *Cells.* 9(2): Article 2. <https://doi.org/10.3390/cells9020276>

10. Jahromi, F. N. A., Dowran, R. and Jafari, R. (2023). Recent advances in the roles of exosomal microRNAs (exomiRs) in hematologic neoplasms: pathogenesis, diagnosis, and treatment. *Cell Commun Signal.* 21(1): 88. <https://doi.org/10.1186/s12964-023-01102-7>
11. Piao, X.-M., Cha, E.-J., Yun, S. J. and Kim, W.-J. (2021). Role of Exosomal miRNA in Bladder Cancer: A Promising Liquid Biopsy Biomarker. *Int J Mol Sci.* 22(4): Article 4. <https://doi.org/10.3390/ijms22041713>
12. Sewer, A., Gubian, S., Kogel, U., Veljkovic, E., Han, W., Hengstermann, A., Peitsch, M. C. and Hoeng, J. (2014). Assessment of a novel multi-array normalization method based on spike-in control probes suitable for microRNA datasets with global decreases in expression. *BMC Res Notes.* 7(1): 302. <https://doi.org/10.1186/1756-0500-7-302>
13. Lekchnov, E. A., Amelina, E. V., Bryzgunova, O. E., Zaporozhchenko, I. A., Konoshenko, M. Y., Yarmoschuk, S. V., Murashov, I. S., Pashkovskaya, O. A., Gorizkii, A. M., Zheravin, A. A., et al. (2018). Searching for the Novel Specific Predictors of Prostate Cancer in Urine: The Analysis of 84 miRNA Expression. *Int J Mol Sci.* 19(12): Article 12. <https://doi.org/10.3390/ijms19124088>
14. Jain, G., Das, P., Ranjan, P., Neha, Valderrama, F. and Cieza-Borrella, C. (2023). Urinary extracellular vesicles miRNA—A new era of prostate cancer biomarkers. *Front Genet.* 14. <https://doi.org/10.3389/fgene.2023.1065757>
15. Pfaffl, M. W., Tichopad, A., Prgomet, C. and Neuvians, T. P. (2004). Determination of stable housekeeping genes, differentially regulated target genes and sample integrity: BestKeeper – Excel-based tool using pair-wise correlations. *Biotechnol Lett.* 26(6): 509–515. <https://doi.org/10.1023/B:BILE.0000019559.84305.47>
16. Livak, K. J. and Schmittgen, T. D. (2001). Analysis of relative gene expression data using real-time quantitative PCR and the 2(-Delta Delta C(T)) Method. *Methods.* 25(4): 402–408. <https://doi.org/10.1006/meth.2001.1262>
17. Köhler, M., Leitsch, D., Müller, N. and Walochnik, J. (2020). Validation of reference genes for the normalization of RT-qPCR gene expression in *Acanthamoeba* spp. *Sci Rep.* 10(1): 10362. <https://doi.org/10.1038/s41598-020-67035-0>
18. St-Pierre, J., Grégoire, J.-C. and Vaillancourt, C. (2017). A simple method to assess group difference in RT-qPCR reference gene selection using GeNorm: The case of the placental sex. *Sci Rep.* 7(1): 16923. <https://doi.org/10.1038/s41598-017-16916-y>
19. Hoja-Łukowicz, D., Maciążek, D., Kościelniak, P. and Janik, M. E. (2022). Innovative GenExpA software for selecting suitable reference genes for reliable normalization of gene expression in melanoma. *Sci Rep.* 12(1): 3331. <https://doi.org/10.1038/s41598-022-07257-6>
20. Pathak, A. K., Kural, S., Singh, S., Kumar, L., Yadav, M., Gupta, M., Das, P. and Jain, G. (2024). Development of a robust and generalizable algorithm. *Sci Rep.* 14(1): 18774. <https://doi.org/10.1038/s41598-024-66770-y>