Package 'PlateVision'

December 22, 2025

Title Automated qPCR Analysis and Visual Quality Control

Version 0.1.0

Description Directly pipes raw quantitative PCR (qPCR) machine outputs into downstream analyses using the comparative Ct (Delta-Delta Ct) method described by Livak and Schmittgen (2001) <doi:10.1006/meth.2001.1262>. Streamlines the workflow from 'Excel' export to publication-ready plots. Integrates unique visual quality control by reconstructing 96-well plate heatmaps, allowing users to instantly detect pipetting errors, edge effects, and outliers. Key features include automated error propagation, laboratory master mix calculations, and generation of bar charts and volcano plots.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr, readxl, ggplot2, stringr, plotly

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Shubham Love [aut, cre]

Maintainer Shubham Love <shubhamlove2101@gmail.com>

Repository CRAN

Date/Publication 2025-12-22 17:50:07 UTC

Contents

calculate_ddct *Calculate ddCt Statistics*

Description

Performs the complete Delta-Delta Ct method with error propagation.

Usage

```
calculate_ddct(data, ref_gene, control_group)
```

Arguments

data	Output from import_plate().
ref_gene	Name of the housekeeping gene (e.g., "GAPDH").
control_group	Name of the control condition (e.g., "WT").

Value

A comprehensive dataframe with Fold Changes, Log2FC, P-values, and Error bars.

Examples

```
# Mock data: 2 biological replicates per group to allow t-test
df <- data.frame(
  Sample = c("S1", "S2", "S3", "S4"),
  Group = c("WT", "WT", "Treated", "Treated"),
  Gene = c(rep("GAPDH", 4), rep("Target", 4)),
  Ct = c(20, 20.1, 20.2, 20.3,   # GAPDH (Consistent)
         25, 24.8, 21, 21.2)     # Target (Down in Treated)
)

calculate_ddct(df, ref_gene = "GAPDH", control_group = "WT")
```

calculate_reagents *Calculate Master Mix Reagents*

Description

Generates a recipe for your qPCR Master Mix based on sample count.

Usage

```
calculate_reagents(
  n_samples,
  n_genes,
  replicates = 3,
  rxn_volume = 20,
  dead_volume_pct = 10
)
```

Arguments

n_samples	Number of biological samples.
n_genes	Number of genes (targets + reference).
replicates	Number of technical replicates (usually 3).
rxn_volume	Total volume per well (e.g., 20 uL).
dead_volume_pct	Percentage of extra mix to prepare for pipetting error (default 10%).

Value

A data frame containing the mix recipe.

Examples

```
# Plan for 12 samples, 2 genes, standard 20uL reaction
calculate_reagents(n_samples = 12, n_genes = 2)

# Adjust for a 10uL reaction volume and 15% dead volume
calculate_reagents(12, 2, rxn_volume = 10, dead_volume_pct = 15)
```

---

check_replicates *Check Technical Replicates*

Description

Scans the experiment for wells where technical replicates disagree (high standard deviation).

Usage

```
check_replicates(data, sd_threshold = 0.5)
```

Arguments

data	Output from import_plate().
sd_threshold	Maximum allowed Standard Deviation between replicates (default 0.5).

Value

A dataframe of "Bad Wells" to investigate.

Examples

```
# Data with a good group (SD=0.1) and a bad group (SD=2.0)
df <- data.frame(
  Sample = c(rep("S1", 3), rep("S2", 3)),
  Gene = "GAPDH",
  Ct = c(20.0, 20.1, 20.2,  25.0, 25.0, 29.0) # S2 has an outlier
)

# Run Check
check_replicates(df, sd_threshold = 0.5)
```

`import_plate` *Import and Merge PCR Data*

Description

Reads the raw machine export and merges it with a user-defined layout map.

Usage

```
import_plate(raw_file, map_file, skip_rows = 0)
```

Arguments

`raw_file`	Path to the machine excel output (.xls or .xlsx).
`map_file`	Path to the user-defined CSV map (Cols: Well, Sample, Gene, Group).
`skip_rows`	Number of rows of metadata to skip in the raw file (default 0).

Value

A clean, merged tibble ready for analysis.

Examples

```
# Locate the sample data bundled with the package
my_raw <- system.file("extdata", "experiment_data.xlsx", package = "PlateVision")
my_map <- system.file("extdata", "plate_map.csv", package = "PlateVision")

# Run import
if(file.exists(my_raw) && file.exists(my_map)) {
  df <- import_plate(raw_file = my_raw, map_file = my_map)
  head(df)
}
```

plot_bars *Plot Fold Change (Bar Chart)*

Description

Aggregates biological replicates and plots the Group Mean +/- SEM.

Usage

```
plot_bars(results)
```

Arguments

results	Output from calculate_ddct().

Value

A ggplot object.

Examples

```
# Mock results data
results <- data.frame(
  Gene = c("GeneA", "GeneA", "GeneB", "GeneB"),
  Group = c("WT", "Treated", "WT", "Treated"),
  fold_change = c(1, 5, 1, 0.5)
)

plot_bars(results)
```

plot_volcano *Plot Volcano (Log2FC vs P-value)*

Description

Creates a volcano plot to identify significantly regulated genes.

Usage

```
plot_volcano(results, p_cutoff = 0.05, fc_cutoff = 1)
```

Arguments

results	Output from calculate_ddct().
p_cutoff	Significance threshold (default 0.05).
fc_cutoff	Log2 Fold Change threshold (default 1).

Value

A ggplot object.

Examples

```
# Mock results data
results <- data.frame(
  log2_fc = c(2.5, -3.0, 0.1, 1.5),
  p_val = c(0.001, 0.0001, 0.8, 0.04)
)

plot_volcano(results)
```

`view_plate` *Visualize Plate Layout (Heatmap)*

Description

Creates a physical map of the 96-well plate colored by Ct value or Sample.

Usage

```
view_plate(data, fill_var = "Ct", interactive = FALSE)
```

Arguments

`data`	The dataframe output from import_plate().
`fill_var`	The column to color the wells by (e.g., "Ct", "Sample", "Gene").
`interactive`	If TRUE, returns a plotly interactive graph. If FALSE, returns static ggplot.

Value

A ggplot object or plotly object.

Examples

```
# Create dummy data representing a partial plate
dummy_data <- data.frame(
  Well = c("A1", "A2", "A3", "B1", "B2", "B3"),
  Ct = c(20, 20.5, 19.8, 25, 24.5, 26),
  Sample = c(rep("Control", 3), rep("Treated", 3)),
  Gene = "GAPDH"
)

# View static heatmap
view_plate(dummy_data, fill_var = "Ct")

# View interactive heatmap (if library plotly is available)
view_plate(dummy_data, fill_var = "Sample", interactive = FALSE)
```

Index