Package 'FracFixR'

May 11, 2026

Type Package

Title Compositional Statistical Framework for RNA Fractionation Analysis

Version 1.1.0

Date 2026-05-11

Description A compositional statistical framework for absolute proportion estimation between fractions in RNA sequencing data. 'FracFixR' addresses the fundamental challenge in fractionated RNA-seq experiments where library preparation and sequencing depth obscure the original proportions of RNA fractions. It reconstructs original fraction proportions using non-negative linear regression, estimates the ``lost'' unrecoverable fraction, corrects individual transcript frequencies, and performs differential proportion testing between conditions. Supports any RNA fractionation protocol including polysome profiling, sub-cellular localization, and RNA-protein complex isolation.

License CC BY 4.0

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports future.apply (>= 1.8.1), nnls (>= 1.4), ggplot2 (>= 3.3.5), dplyr (>= 1.0.7), RColorBrewer (>= 1.1-2), tidyr (>= 1.1.3), matrixStats (>= 0.60.0), aod (>= 1.3.1), parallelly (>= 1.30.0), stats, utils, future, grDevices

Suggests testthat (>= 3.0.0), knitr, rmarkdown

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Alice Cleynen [aut, cre] (ORCID: <https://orcid.org/0000-0001-8083-0204>), Agin Ravindran [aut], Nikolay Shirokikh [aut] (ORCID: <https://orcid.org/0000-0001-8249-358X>)

Maintainer Alice Cleynen <alice.cleynen@cnrs.fr>

Repository CRAN

Date/Publication 2026-05-11 03:40:02 UTC

Contents

`DiffPropTest` *DiffPropTest: Statistical Testing for Differential Proportions*

Description

Performs statistical testing to identify transcripts with significantly different proportions between conditions in specified fraction(s). Implements three test options: GLM (most powerful), Logit, and Wald.

Usage

```
DiffPropTest(NormObject, Conditions, Types, Test = c("GLM", "Logit", "Wald"))
```

Arguments

`NormObject`	Output from FracFixR() function
`Conditions`	Character vector of exactly 2 conditions to compare
`Types`	Character vector of fraction type(s) to analyze. Can be single fraction or multiple (will be combined)
`Test`	Statistical test to use: "GLM", "Logit", or "Wald"

Details

- GLM: Uses binomial generalized linear model (most statistically powerful)
- Logit: Faster alternative using logit transformation
- Wald: Beta-binomial Wald test for overdispersed count data

Value

Data frame with columns:

- transcript: transcript identifier
- mean_success_cond1/2: mean proportions in each condition
- mean_diff: difference in proportions
- log2FC: log2 fold change
- pval: p-value from statistical test
- padj: FDR-adjusted p-value

Examples

```
data(example_counts)
data(example_annotation)

# Run FracFixR
results <- FracFixR(example_counts, example_annotation, parallel=FALSE)
# Run differential testing
diff_results <- DiffPropTest(results,
                             Conditions = c("Control", "Treatment"),
                             Types = "Heavy_Polysome",
                             Test = "GLM")
```

---

example_annotation *Example annotation data frame*

Description

A data frame containing sample annotations for the example_counts matrix. Describes the experimental design with conditions, fraction types, and replicates.

Usage

```
example_annotation
```

Format

A data frame with 12 rows and 4 columns:

Sample Sample identifier matching column names in example_counts

Condition Experimental condition (Control or Treatment)

Type Fraction type (Total, Light_Polysome, or Heavy_Polysome)

Replicate Replicate identifier (Rep1 or Rep2)

Source

Simulated data generated for package examples

Examples

```
data(example_annotation)
head(example_annotation)
table(example_annotation$Condition, example_annotation$Type)
```

`example_counts` *Example RNA-seq count matrix*

Description

A matrix containing simulated RNA-seq counts for 100 genes across 12 samples. The data simulates a polysome profiling experiment with two conditions (Control and Treatment) and three fractions (Total, Light_Polysome, Heavy_Polysome).

Usage

```
example_counts
```

Format

A numeric matrix with 100 rows (genes) and 12 columns (samples):

rows Gene identifiers (Gene1 to Gene100)

columns Sample identifiers (Sample1 to Sample12)

Source

Simulated data generated for package examples

Examples

```
data(example_counts)
dim(example_counts)
head(example_counts[, 1:6])
```

`FracFixR` *FracFixR: Main Function for Fraction Correction*

Description

This is the core function that implements the FracFixR framework. It takes raw count data from total and fractionated samples and reconstructs the original fraction proportions through compositional modeling.

Usage

```
FracFixR(MatrixCounts, Annotation, st1 = 0.6, st2 = 0.999, parallel = TRUE)
```

Arguments

`MatrixCounts`	A numeric matrix of raw transcript/gene counts with: • Rows: transcripts/genes (must have rownames) • Columns: samples (must have colnames matching Annotation$Sample)
`Annotation`	A data.frame with required columns: • Sample: sample identifiers matching column names in MatrixCounts • Condition: experimental condition (e.g., "Control", "Treatment") • Type: fraction type (must include at least one "Total") • Replicate: replicate identifier
`parallel`	A boolean indicating whether to use parallel processing of the transcripts (default=TRUE).
`st1`	Lower quantile threshold (between 0 and 1) for selecting informative transcripts for the NNLS regression fit (default = 0.6). Transcripts below this quantile of Total abundance are considered too noisy for reliable regression.
`st2`	Upper quantile threshold (between 0 and 1) for selecting informative transcripts for the NNLS regression fit (default = 0.999). Transcripts above this quantile are potential outliers and are excluded from the regression.

Details

The function works by:

1. Filtering transcripts based on presence in Total samples
2. For each condition and replicate, fitting NNLS regression
3. Estimating global fraction weights and individual transcript proportions
4. Calculating the "lost" unrecoverable fraction

Value

A list containing:

- OriginalData: filtered input count matrix
- Annotation: input annotation data
- Propestimates: matrix of proportion estimates (values between 0 and 1)
- NewData: corrected count matrix (proportions multiplied by predicted total, rounded)
- Coefficients: data.frame of regression coefficients
- Fractions: data.frame of estimated fraction proportions
- plots: list of diagnostic plots

References

Cleynen et al. FracFixR: A compositional statistical framework for absolute proportion estimation between fractions in RNA sequencing data.

Examples

```
# Load example data
data(example_counts)
data(example_annotation)

# Run FracFixR
results <- FracFixR(example_counts, example_annotation, parallel=FALSE)

# View fraction proportions
print(results$Fractions)
```

---

get_corrected_counts *get_corrected_counts: Retrieve the Corrected Count Matrix*

Description

Returns the corrected count matrix embedded in a `FracFixR()` result object. This matrix is computed internally by multiplying each transcript's proportion estimate by the predicted total abundance for that replicate, providing counts that are corrected for compositional bias while remaining on the original count scale.

If you need to re-scale using the raw (observed) Total counts instead of the NNLS-predicted totals, multiply `fracfixr_results$Propestimates` by the corresponding column of `fracfixr_results$OriginalData` manually.

Usage

```
get_corrected_counts(fracfixr_results)
```

Arguments

`fracfixr_results`
: Output list from FracFixR, which must contain the element `NewData` (present in results from version >= 1.1.0).

Value

A numeric matrix with the same dimensions as `fracfixr_results$Propestimates`. Non-Total columns contain corrected counts (rounded integers, proportion estimate multiplied by the NNLS-predicted total abundance). The Total column contains the NNLS-predicted total abundance itself.

Examples

```
data(example_counts)
data(example_annotation)

results <- FracFixR(example_counts, example_annotation, parallel = FALSE)
corrected <- get_corrected_counts(results)
head(corrected)
```

PlotComparison *PlotComparison: Create Volcano Plot for Differential Results*

Description

Generates avolcano plot showing transcripts with significant differential proportions between conditions.

Usage

```
PlotComparison(DiffPropResult, Conditions = NULL, Types = NULL, cutoff = NULL)
```

Arguments

`DiffPropResult`	Output from DiffPropTest() function
`Conditions`	Character vector of conditions being compared
`Types`	Character vector of fraction types analyzed
`cutoff`	Optional y-axis maximum for plot

Value

Volcano plot-type object

Examples

```
data(example_counts)
data(example_annotation)

# Run FracFixR
results <- FracFixR(example_counts, example_annotation,parallel=FALSE)
# Run differential testing
diff_results <- DiffPropTest(results,
                             Conditions = c("Control", "Treatment"),
                             Types = "Heavy_Polysome",
                             Test = "GLM")
# Create volcano plot
volcano <- PlotComparison(diff_results,
                          Conditions = c("Control", "Treatment"),
                          Types = "Heavy_Polysome")
```

`PlotFractions` *PlotFractions: Visualize Fraction Proportions*

Description

Creates a stacked bar plot showing the distribution of RNA across fractions for each replicate, including the "lost" fraction.

Usage

```
PlotFractions(FracFixed)
```

Arguments

`FracFixed`	Output from FracFixR() function

Value

ggplot2 object showing fraction proportions

Examples

```
data(example_counts)
data(example_annotation)

# Run FracFixR
results <- FracFixR(example_counts, example_annotation, parallel=FALSE)
# Create fraction plot
frac_plot <- PlotFractions(results)
# Save plot with ggsave("fractions.pdf", frac_plot, width = 10, height = 8)
```

polysome_annotation *Polysome profiling annotation example*

Description

An alternative annotation data frame for polysome profiling experiments with monosome and polysome fractions.

Usage

```
polysome_annotation
```

Format

A data frame with 12 rows and 4 columns:

Sample Sample identifier

Condition Experimental condition (Control or Stress)

Type Fraction type (Total, Monosome, or Polysome)

Replicate Replicate identifier (Rep1 or Rep2)

Source

Simulated data generated for package examples

Examples

```
data(polysome_annotation)
head(polysome_annotation)
```

subcellular_annotation *Subcellular fractionation annotation example*

Description

An annotation data frame for subcellular fractionation experiments with nuclear and cytoplasmic fractions.

Usage

```
subcellular_annotation
```

Format

A data frame with 12 rows and 4 columns:

Sample Sample identifier

Condition Experimental condition (WT or Mutant)

Type Fraction type (Total, Nuclear, or Cytoplasmic)

Replicate Replicate identifier (Rep1 or Rep2)

Source

Simulated data generated for package examples

Examples

```
data(subcellular_annotation)
head(subcellular_annotation)
```

Index