

# Package ‘ppcseq’

June 5, 2026

**Title** Probabilistic Outlier Identification for RNA Sequencing  
Generalized Linear Models

**Version** 1.21.0

**Description** Relative transcript abundance has proven to be a valuable tool for understanding the function of genes in biological systems. For the differential analysis of transcript abundance using RNA sequencing data, the negative binomial model is by far the most frequently adopted. However, common methods that are based on a negative binomial model are not robust to extreme outliers, which we found to be abundant in public datasets. So far, no rigorous and probabilistic methods for detection of outliers have been developed for RNA sequencing data, leaving the identification mostly to visual inspection. Recent advances in Bayesian computation allow large-scale comparison of observed data against its theoretical distribution given in a statistical model. Here we propose ppcseq, a key quality-control tool for identifying transcripts that include outlier data points in differential expression analysis, which do not follow a negative binomial distribution. Applying ppcseq to analyse several publicly available datasets using popular tools, we show that from 3 to 10 percent of differentially abundant transcripts across algorithms and datasets had statistics inflated by the presence of outliers.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Biarch** true

**Depends** R (>= 4.1.0), rstan (>= 2.18.1)

**Imports** benchmarkme, dplyr, edgeR, foreach, ggplot2, graphics,  
lifecycle, magrittr, methods, parallel, purrr, Rcpp (>=  
0.12.0), RcppParallel (>= 5.0.1), rlang, rstantools (>= 2.1.1),  
stats, tibble, tidybayes, tidyr (>= 0.8.3.9000), utils

**LinkingTo** BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0),  
RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>=  
2.18.0)

**Suggests** knitr, testthat, BiocStyle, rmarkdown

**VignetteBuilder** knitr

**RdMacros** lifecycle

**biocViews** RNASeq, DifferentialExpression, GeneExpression,  
Normalization, Clustering, QualityControl, Sequencing,  
Transcription, Transcriptomics

**SystemRequirements** GNU make

**RoxygenNote** 7.2.3

**Roxygen** list(markdown = TRUE)

**URL** <https://github.com/stemangiola/ppcseq>

**BugReports** <https://github.com/stemangiola/ppcseq/issues>

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/ppcseq>

**git\_branch** devel

**git\_last\_commit** cac1c6d

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-06-04

**Author** Stefano Mangiola [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-7474-836X>>)

**Maintainer** Stefano Mangiola <mangiolastefano@gmail.com>

## Contents

ppcseq-package . . . . .	2
counts . . . . .	3
identify_outliers . . . . .	3
plot_credible_intervals . . . . .	5
<b>Index</b>	<b>7</b>

---

ppcseq-package                      *The 'ppcseq' package.*

---

## Description

Relative transcript abundance has proven to be a valuable tool for understanding the function of genes in biological systems. For the differential analysis of transcript abundance using RNA sequencing data, the negative binomial model is by far the most frequently adopted. However, common methods that are based on a negative binomial model are not robust to extreme outliers, which we found to be abundant in public datasets. So far, no rigorous and probabilistic methods for detection of outliers have been developed for RNA sequencing data, leaving the identification mostly to visual inspection. Recent advances in Bayesian computation allow large-scale comparison of observed data against its theoretical distribution given in a statistical model. Here we propose ppcseq, a key quality-control tool for identifying transcripts that include outlier data points in differential expression analysis, which do not follow a negative binomial distribution. Applying ppcseq to analyse several publicly available datasets using popular tools, we show that from 3 to 10 percent of differentially abundant transcripts across algorithms and datasets had statistics inflated by the presence of outliers.

## Usage

data(counts)

**Value**

See documentation

**References**

Mangiola S, Thomas E, Modrak M, Vehtari A, Papenfuss A (2021). “Probabilistic outlier identification for RNA sequencing generalized linear models.” *NAR Genomics and Bioinformatics*, 3(1), lqab005. <URL: <https://doi.org/10.1093/nargab/lqab005>>.

---

counts	<i>counts</i>
--------	---------------

---

**Description**

Contains an example dataset for ppcseq, including RNA sequencing

**Usage**

counts

**Format**

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 394821 rows and 9 columns.

---

identify_outliers	<i>identify_outliers main</i>
-------------------	-------------------------------

---

**Description**

This function runs the data modeling and statistical test for the hypothesis that a transcript includes outlier biological replicate.

**[Maturing]****Usage**

```
identify_outliers(  
  .data,  
  formula = ~1,  
  .sample,  
  .transcript,  
  .abundance,  
  .significance,  
  .do_check,  
  .scaling_factor = NULL,  
  percent_false_positive_genes = 1,  
  how_many_negative_controls = 500,  
  approximate_posterior_inference = TRUE,  
  approximate_posterior_analysis = TRUE,  
  draws_after_tail = 10,
```

```

save_generated_quantities = FALSE,
additional_parameters_to_save = c(),
cores = detect_cores(),
pass_fit = FALSE,
do_check_only_on_detrimental = length(parse_formula(formula)) > 0,
tol_rel_obj = 0.01,
just_discovery = FALSE,
seed = sample(seq_len(length.out = 999999), size = 1),
adj_prob_threshold_2 = NULL
)

```

### Arguments

<code>.data</code>	A tibble including a transcript name column   sample name column   read counts column   covariate columns   Pvalue column   a significance column
<code>formula</code>	A formula. The sample formula used to perform the differential transcript abundance analysis
<code>.sample</code>	A column name as symbol. The sample identifier
<code>.transcript</code>	A column name as symbol. The transcript identifier
<code>.abundance</code>	A column name as symbol. The transcript abundance (read count)
<code>.significance</code>	A column name as symbol. A column with the Pvalue, or other significance measure (preferred Pvalue over false discovery rate)
<code>.do_check</code>	A column name as symbol. A column with a boolean indicating whether a transcript was identified as differentially abundant
<code>.scaling_factor</code>	In case the scaling factor must not be calculated (TMM method) using the input data but provided. It is useful, for example, for pseudobulk single-cell where the scaling might depend on sample sequencing depth for all cells rather than a particular cell type.
<code>percent_false_positive_genes</code>	A real between 0 and 100. It is the aimed percent of transcript being a false positive. For example, <code>percent_false_positive_genes = 1</code> provide 1 percent of the calls for outlier containing transcripts that has actually not outliers.
<code>how_many_negative_controls</code>	An integer. How many transcript from the bottom non-significant should be taken for inferring the mean-overdispersion trend.
<code>approximate_posterior_inference</code>	A boolean. Whether the inference of the joint posterior distribution should be approximated with variational Bayes It confers execution time advantage.
<code>approximate_posterior_analysis</code>	A boolean. Whether the calculation of the credible intervals should be done semi-analytically, rather than with pure sampling from the posterior. It confers execution time and memory advantage.
<code>draws_after_tail</code>	An integer. How many draws should on average be after the tail, in a way to inform CI.
<code>save_generated_quantities</code>	A boolean. Used for development and testing purposes
<code>additional_parameters_to_save</code>	A character vector. Used for development and testing purposes

cores	An integer. How many cores to be used with parallel calculations.
pass_fit	A boolean. Used for development and testing purposes
do_check_only_on_detrimental	A boolean. Whether to test only for detrimental outliers (same direction as the fold change). It allows to test for less transcript/sample pairs and therefore higher the probability threshold.
tol_rel_obj	A real. Used for development and testing purposes
just_discovery	A boolean. Used for development and testing purposes
seed	An integer. Used for development and testing purposes
adj_prob_threshold_2	A boolean. Used for development and testing purposes

**Value**

A nested tibble tbl with transcript-wise information: sample\_wise\_data | plot | ppc samples failed | tot deleterious\_outliers

**Examples**

```
library(dplyr)

data("counts")

if(Sys.info()[['sysname']] == "Linux")
  result =
    counts %>%
    dplyr::mutate( is_significant = ifelse(symbol %in% c("SLC16A12", "CYP1A1", "ART3"), TRUE, FALSE) ) %>%
    ppcseq::identify_outliers(
      formula = ~ Label,
      sample, symbol, value,
      .significance = PValue,
      .do_check = is_significant,
      percent_false_positive_genes = 1,
      tol_rel_obj = 0.01,
      approximate_posterior_inference =TRUE,
      approximate_posterior_analysis =TRUE,
      how_many_negative_controls = 50,
      cores=1
    )
```

---

plot\_credible\_intervals

*plot\_credible interval for theoretical data distributions*

---

**Description**

Plot the data along the theoretical data distribution.

**Usage**

```
plot_credible_intervals(.data)
```

**Arguments**

`.data`            The tibble returned by `identify_outliers`

**Value**

A tibble with an additional plot column

**Examples**

```
library(dplyr)

data("counts")

if(Sys.info()[['sysname']] == "Linux"){
  result =
    counts %>%
    dplyr::mutate( is_significant = ifelse(symbol %in% c("SLC16A12", "CYP1A1", "ART3"), TRUE, FALSE) ) %>%
    ppcseq::identify_outliers(
      formula = ~ Label,
      sample, symbol, value,
      .significance = PValue,
      .do_check = is_significant,
      percent_false_positive_genes = 1,
      tol_rel_obj = 0.01,
      approximate_posterior_inference =TRUE,
      approximate_posterior_analysis =TRUE,
      how_many_negative_controls = 50,
      cores=1
    )

  result_plot = result %>% plot_credible_intervals()
}
```

# Index

## \* datasets

counts, [3](#)

counts, [3](#)

identify\_outliers, [3](#)

plot\_credible\_intervals, [5](#)

ppcseq (ppcseq-package), [2](#)

ppcseq-package, [2](#)