

Package: saintexpress (via r-universe)

September 26, 2026

Title Pure R Implementation of SAINTExpress

Version 0.99.0

Description A pure R implementation of the SAINTExpress scoring engines for affinity purification-mass spectrometry data, covering both the spectral-count and intensity models, together with validation of SAINT input tables. The output mirrors the native SAINTExpress 'list.txt' format, so protein-protein interactions can be scored without external binaries. The underlying method is described in Teo et al. (2014) [doi:10.1016/j.jprot.2013.10.023](https://doi.org/10.1016/j.jprot.2013.10.023).

License GPL (>= 3)

URL <https://prolfqua.github.io/saintexpress/>,
<https://github.com/prolfqua/saintexpress>

BugReports <https://github.com/prolfqua/saintexpress/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests BiocStyle, knitr, nloptr, pkgdown, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports stats

VignetteBuilder knitr

biocViews MassSpectrometry, Proteomics, Software

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-08-27 14:27:48 UTC

RemoteUrl <https://github.com/BiocStaging/saintexpress>

RemoteRef HEAD

RemoteSha 8731469af7d23d85edde288f18d180ac6ce0800d

Contents

run_saint	2
validate_saint_input	3
Index	4

run_saint	<i>Run the pure-R SAINTexpress engine</i>
-----------	---

Description

Dispatches to the spectral-count or intensity scorer.

Usage

```
run_saint(si, mode = c("spc", "int"), optimizer = c("base", "nloptr"), ...)
```

Arguments

- si A named list with elements inter, prey, bait (data frames).
- mode "spc" for spectral counts, "int" for intensities.
- optimizer "base" (default; uses `stats::optim()`) or "nloptr" (uses NLOpt COBYLA via the optional **nloptr** package).
- ... Additional arguments forwarded to the internal scorers saint_spc_r() or saint_int_r() (e.g. R, L, frequency, p2p_mapping).

Value

A data frame in the SAINTexpress list.txt shape.

Examples

```
si <- list(
  inter = data.frame(
    ipId = c("IP1", "IP1", "IP2", "IP2", "IP3", "IP4"),
    baitId = c("BaitA", "BaitA", "BaitA", "BaitA", "Ctrl", "Ctrl"),
    preyId = c("Prey1", "Prey2", "Prey1", "Prey2", "Prey1", "Prey2"),
    quant = c(20, 1, 18, 1, 1, 1)
  ),
  prey = data.frame(
    preyId = c("Prey1", "Prey2"),
    preyLength = c(500, 500),
    preyGeneId = c("Gene1", "Gene2")
  ),
  bait = data.frame(
    ipId = c("IP1", "IP2", "IP3", "IP4"),
    baitId = c("BaitA", "BaitA", "Ctrl", "Ctrl"),
    CorT = c("T", "T", "C", "C")
  )
)
```

```
    )  
  )  
  run_saint(si, mode = "spc")
```

validate_saint_input	<i>Validate a SAINT input list</i>
----------------------	------------------------------------

Description

Checks that si is a list with exactly the elements inter, prey, bait, and that each is a data frame.

Usage

```
validate_saint_input(si)
```

Arguments

si	A named list with elements inter, prey, bait.
----	---

Value

TRUE invisibly on success; otherwise stops with a message.

Examples

```
si <- list(  
  inter = data.frame(ipId = "IP1", baitId = "BaitA", preyId = "Prey1"),  
  prey = data.frame(preId = "Prey1"),  
  bait = data.frame(ipId = "IP1", baitId = "BaitA", CorT = "T")  
)  
validate_saint_input(si)
```

Index

`run_saint`, [2](#)

`stats::optim()`, [2](#)

`validate_saint_input`, [3](#)