

# Package ‘sparseLM’

May 9, 2026

**Title** Interface to the 'sparseLM' Levenberg-Marquardt Library

**Version** 0.5

**Description** Provides an R interface to the 'sparseLM' C library for large-scale nonlinear least squares problems with arbitrarily sparse Jacobians. The underlying solver implements a sparse variant of the Levenberg-Marquardt algorithm for minimizing sum-of-squares objective functions, supports user-supplied analytic Jacobians or finite-difference approximation, and is designed to exploit sparsity for improved memory use and performance. This package exposes the solver in R and uses sparse matrix classes and the 'CHOLMOD' sparse Cholesky factorization routines through the 'Matrix' package interface. Methods from the C library are described in Lourakis (2010) <[doi:10.1007/978-3-642-15552-9\\_4](https://doi.org/10.1007/978-3-642-15552-9_4)>.

**URL** <https://github.com/smith-group/sparseLM>,  
<https://smith-group.github.io/sparseLM/>

**Depends** Matrix

**Imports** methods

**LinkingTo** Matrix

**License** GPL-2

**Encoding** UTF-8

**NeedsCompilation** yes

**RoxygenNote** 7.3.3

**Author** Colin Smith [aut, cre] (ORCID: <<https://orcid.org/0000-0002-4651-167X>>),  
Manolis Lourakis [aut] (ORCID: <<https://orcid.org/0000-0003-4596-5773>>)

**Maintainer** Colin Smith <[colin.smith@wesleyan.edu](mailto:colin.smith@wesleyan.edu)>

**Repository** CRAN

**Date/Publication** 2026-03-30 09:30:08 UTC

## Contents

add_assign_col_inplace . . . . .	2
sparselm . . . . .	3
sparselm.opts . . . . .	5

---

 add\_assign\_col\_inplace

*Add values to an existing sparse column in place*


---

### Description

Adds value to the existing nonzero entries of column *j* at rows *i*. Only existing nonzeros are updated; no new nonzeros are created. `add_assign_col_inplace_unsafe` skips the input validation performed by `add_assign_col_inplace`. When `validate = TRUE`, the function checks that *i* is strictly increasing, in range, and contains no NA; that value contains no NA/NaN; and that the column pointer *p* is nondecreasing.

### Usage

```
add_assign_col_inplace(x, i, j, value, validate = TRUE)
```

```
add_assign_col_inplace_unsafe(x, i, j, value)
```

### Arguments

<code>x</code>	dgCMatrix to be assigned
<code>i</code>	rows to be assigned in strictly increasing order
<code>j</code>	single integer giving column to assign
<code>value</code>	numeric vector (same length as <i>i</i> ) to increment rows <i>i</i> of column <i>j</i> of <i>x</i>
<code>validate</code>	logical scalar; if TRUE, validates that <i>i</i> is strictly increasing and within bounds

### Value

value passed to *x*

### Examples

```
if (requireNamespace("Matrix", quietly = TRUE)) {
  x <- Matrix::Matrix(0, nrow = 4, ncol = 2, sparse = TRUE)
  x[1, 1] <- 1
  x[3, 1] <- 2
  # value is the same length as i
  add_assign_col_inplace(x, i = c(1L, 3L), j = 1L, value = c(10, -5))
}
```

---

sparselm	<i>Nonlinear Least Squares Fit with Sparse Levenberg-Marquardt Algorithm</i>
----------	--

---

**Description**

Nonlinear Least Squares Fit with Sparse Levenberg-Marquardt Algorithm

**Usage**

```
sparselm(
  p,
  x,
  func,
  fjac,
  Jnnz,
  nconvars = 0,
  itmax = 100,
  opts = sparselm.opts(),
  dif = FALSE,
  ...
)
```

**Arguments**

p	initial parameter estimates length nvars
x	measurement vector length nobs; length(x) must be at least length(p)
func	functional relation describing measurements given a parameter vector p, returning vector length nobs
fjac	function to supply the nonzero pattern of the sparse Jacobian of func and optionally evaluate it at p, returning nobs by nvars dgCMatrix
Jnnz	number of nonzeros for the Jacobian J
nconvars	number of constrained variables (currently reserved for future use)
itmax	maximum number of iterations
opts	minim. options mu, epsilon1, epsilon2, epsilon3, delta, spsolver
dif	logical indicating whether to use finite differences
...	additional arguments passed to func and fjac

**Value**

list with four elements: par, niter, info, and term

**Examples**

```

set.seed(1)
t <- seq(0, 10, length.out = 80)
g <- function(A, mu, s, x) A * exp(-0.5 * ((x - mu) / s)^2)
y <- g(3, 3, 0.7, t) + g(2, 7, 1.0, t) + rnorm(length(t), sd = 0.2)
p0 <- c(2.5, 3.2, 0.8, 1.5, 6.8, 1.2)

y_obs <- y
mask1 <- abs(t - p0[2]) <= 3 * p0[3]
mask2 <- abs(t - p0[5]) <= 3 * p0[6]
mask <- mask1 | mask2
y[!mask] <- 0

idx1 <- which(mask1); idx2 <- which(mask2)
i <- c(idx1, idx1, idx1, idx2, idx2, idx2)
j <- c(rep(1, length(idx1)), rep(2, length(idx1)), rep(3, length(idx1)),
      rep(4, length(idx2)), rep(5, length(idx2)), rep(6, length(idx2)))
Jpat <- sparseMatrix(i = i, j = j, x = 1, dims = c(length(t), 6))

func <- function(p, t, ...) {
  f <- g(p[1], p[2], p[3], t) + g(p[4], p[5], p[6], t)
  f[!mask] <- 0
  f
}

fjac <- function(p, t, ...) {
  A1 <- p[1]; mu1 <- p[2]; s1 <- p[3]
  A2 <- p[4]; mu2 <- p[5]; s2 <- p[6]
  e1 <- exp(-0.5 * ((t[idx1] - mu1) / s1)^2)
  e2 <- exp(-0.5 * ((t[idx2] - mu2) / s2)^2)
  J <- Jpat
  J@x <- c(e1, A1 * e1 * ((t[idx1] - mu1) / s1^2),
          A1 * e1 * ((t[idx1] - mu1)^2 / s1^3),
          e2, A2 * e2 * ((t[idx2] - mu2) / s2^2),
          A2 * e2 * ((t[idx2] - mu2)^2 / s2^3))
  J
}

fit <- sparselm(p0, y, func, fjac,
               Jnnz = length(i),
               nconvars = 0, t = t)

Jpat
fit$par

f0 <- g(p0[1], p0[2], p0[3], t) + g(p0[4], p0[5], p0[6], t)
f1 <- g(fit$par[1], fit$par[2], fit$par[3], t) + g(fit$par[4], fit$par[5], fit$par[6], t)
plot(t, y_obs, pch = 16, cex = 0.6, col = "black", xlab = "t", ylab = "y")
lines(t, f0, col = "blue")
lines(t, f1, col = "red")

```

---

`sparselm.opts`*Sparse Levenberg-Marquardt Algorithm Options*

---

**Description**

Sparse Levenberg-Marquardt Algorithm Options

**Usage**

```
sparselm.opts(  
  mu = 0.001,  
  epsilon1 = 1e-12,  
  epsilon2 = 1e-12,  
  epsilon3 = 1e-12,  
  delta = 1e-06  
)
```

**Arguments**

<code>mu</code>	scale factor for initial mu
<code>epsilon1</code>	stopping threshold for $\ J^T \text{ell\_inf}\ $
<code>epsilon2</code>	stopping threshold for $\ dp\ _2$
<code>epsilon3</code>	stopping threshold for $\ \text{ell}_2\ $
<code>delta</code>	step used in difference approximation to the Jacobian; if negative, central differences are used instead of forward differences

**Value**numeric vector of length 6 with the above options and `spsolver=1` (SuiteSparse CHOLMOD)

# Index

`add_assign_col_inplace`, [2](#)  
`add_assign_col_inplace_unsafe`  
    (`add_assign_col_inplace`), [2](#)

`sparselm`, [3](#)  
`sparselm.opts`, [5](#)