

# Package: fz (via r-universe)

June 30, 2026

**Type** Package

**Title** R Wrapper for the 'funz-fz' Parametric Simulation Framework

**Version** 1.1

**Description** Provides R bindings to the 'funz-fz' Python package using 'reticulate'. The 'fz' framework wraps arbitrary simulation codes to run parameter sweeps, design-of-experiments studies, and iterative algorithm-driven analyses by substituting variable placeholders in text input files and collecting outputs into data frames. Calculators can run locally (shell), over SSH, or on 'SLURM' clusters. See <https://github.com/Funz/fz> for the underlying framework.

**License** BSD\_3\_clause + file LICENSE

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.3

**SystemRequirements** Python (>= 3.8), funz-fz Python package

**Imports** reticulate (>= 1.28)

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown

**Config/reticulate** list( packages = list( list(package = ``funz-fz") ) ) )

**VignetteBuilder** knitr

**URL** <https://github.com/Funz/fz.R>

**BugReports** <https://github.com/Funz/fz.R/issues>

**Roxygen** list(markdown = TRUE)

**Config/pak/sysreqs** libpng-dev python3

**Repository** <https://funz.r-universe.dev>

**Date/Publication** 2026-06-23 13:40:24 UTC

**RemoteUrl** <https://github.com/funz/fz.r>

**RemoteRef** HEAD

**RemoteSha** 4e6560fde857249d607ed740282408abc4d4a11d

## Contents

fz_available . . . . .	2
fz_install . . . . .	3
fzc . . . . .	4
fzd . . . . .	5
fzi . . . . .	6
fzl . . . . .	7
fzo . . . . .	8
fzr . . . . .	8
get_config . . . . .	10
get_interpreter . . . . .	10
get_log_level . . . . .	11
install . . . . .	11
install_algorithm . . . . .	12
install_model . . . . .	12
list_installed_algorithms . . . . .	13
list_installed_models . . . . .	14
list_models . . . . .	14
print_config . . . . .	15
reload_config . . . . .	15
set_interpreter . . . . .	16
set_log_level . . . . .	16
uninstall . . . . .	17
uninstall_algorithm . . . . .	18
uninstall_model . . . . .	18
<b>Index</b>	<b>20</b>

---

fz_available	<i>Check if fz Python Package is Available</i>
--------------	--

---

### Description

Checks whether the fz Python package is available in the current Python environment.

### Usage

```
fz_available()
```

### Value

Logical; TRUE if fz is available, FALSE otherwise.

## Examples

```
if (fz_available()) {
  message("fz is available!")
} else {
  message("Please install fz with fz_install()")
}
```

---

fz\_install

*Install the fz Python Package*

---

## Description

This function installs the fz Python package into a virtual environment or conda environment managed by reticulate.

## Usage

```
fz_install(method = "auto", conda = "auto", pip = TRUE, ...)
```

## Arguments

method	Installation method. Either "auto", "virtualenv", or "conda".
conda	Path to conda executable. Only used when method is "conda".
pip	Logical; use pip for installation? Default is TRUE.
...	Additional arguments passed to <code>reticulate::py_install()</code> .

## Value

NULL (invisibly). Called for side effects.

## Examples

```
## Not run:
# Install fz in a virtual environment
fz_install()

# Install in a conda environment
fz_install(method = "conda")

## End(Not run)
```

fzc

*fzc Function***Description**

Compiles input file(s) by replacing variable placeholders with values. Each unique combination of values is written to its own subdirectory inside `output_dir`, named `var1=val1,var2=val2,...`

**Usage**

```
fzc(input_path, input_variables, model, output_dir = "output")
```

**Arguments**

<code>input_path</code>	Path to input file or directory.
<code>input_variables</code>	Named list of variable values. Supply a vector of values to generate a full-factorial grid across variables.
<code>model</code>	Model definition dict or alias string.
<code>output_dir</code>	Output directory for compiled files. Default "output".

**Value**

NULL (invisibly). Called for side effects.

**Examples**

```
if (fz_available()) {
  tf <- tempfile(fileext = ".txt")
  writeLines(c("P = ${P~1.013}", "V = ${V~22.4}"), tf)

  model <- list(varprefix = "$", delim = "{", formulaprefix = "@",
               commentline = "#")
  out <- tempfile()

  fzc(tf, list(P = 2.0, V = 11.2), model, out)
  fzc(tf, list(P = c(1.0, 2.0), V = c(11.2, 22.4)), model, out)
}
```

---

`fzd`*fzd Function*

---

## Description

Runs an iterative design of experiments driven by an algorithm. Unlike `fzr` (which evaluates a fixed grid), `fzd` lets an algorithm adaptively choose which parameter combinations to evaluate, which is useful for sensitivity analysis, surrogate-model fitting, or optimization.

## Usage

```
fzd(  
    input_path,  
    input_variables,  
    model,  
    output_expression,  
    algorithm,  
    calculators = NULL,  
    algorithm_options = NULL,  
    analysis_dir = "analysis"  
)
```

## Arguments

<code>input_path</code>	Path to input file or directory.
<code>input_variables</code>	Named list of variable range strings of the form "[min;max]", e.g. <code>list(x = "[0;1]", y = "[-5;5]")</code> .
<code>model</code>	Model definition dict or alias string.
<code>output_expression</code>	Expression evaluated on the model outputs to produce the scalar quantity the algorithm optimizes or analyses, e.g. "result" or "out1 + 2 * out2".
<code>algorithm</code>	Path to the algorithm Python file, e.g. "algorithms/montecarlo_uniform.py".
<code>calculators</code>	Calculator specification(s). Default NULL.
<code>algorithm_options</code>	Algorithm options as a named list or semicolon-separated string, e.g. "batch_sample_size=10;seed=42". Default NULL.
<code>analysis_dir</code>	Analysis directory. Default "analysis".

## Value

Named list with the analysis results produced by the algorithm.

**Examples**

```

if (fz_available()) {
  tf <- tempfile(fileext = ".txt")
  writelines(c("x = ${x~0}", "y = ${y~0}"), tf)

  model <- list(
    varprefix = "$", delim = "{", formulaprefix = "@", commentline = "#",
    output = list(z = "grep z output.txt | cut -d= -f2")
  )

  result <- fzd(
    tf,
    list(x = "[0;1]", y = "[-5;5]"),
    model,
    output_expression = "z",
    algorithm = "algorithms/montecarlo_uniform.py",
    algorithm_options = "batch_sample_size=10;max_iterations=3"
  )
}

```

fzi

*fzi Function***Description**

Parses input file(s) to find variables, formulas, and static objects.

**Usage**

```
fzi(input_path, model)
```

**Arguments**

input_path	Path to input file or directory.
model	Model definition dict or alias string.

**Value**

Named list with variable names and their default values (or NULL).

**Examples**

```

if (fz_available()) {
  tf <- tempfile(fileext = ".txt")
  writelines(c("pressure = ${P~1.013}", "volume = ${V~22.4}"), tf)

  model <- list(varprefix = "$", delim = "{", formulaprefix = "@",
               commentline = "#")
}

```

```
vars <- fzi(tf, model)
}
```

---

fzl

*fzl Function*

---

### Description

Lists installed models and available calculators.

### Usage

```
fzl(models = "*", calculators = "*", check = FALSE)
```

### Arguments

models	Pattern to match models. Default "*" for all. Accepts glob patterns ("my*") or plain alias names.
calculators	Pattern to match calculators. Default "*" for all.
check	Logical; probe each calculator to verify it is reachable. Default FALSE.

### Value

Named list with two entries:

**models** Named list of installed model definitions.

**calculators** Named list of available calculators.

### Examples

```
if (fz_available()) {
  info <- fzl()
  names(info$models)
  names(info$calculators)

  info <- fzl(models = "Perfect*")
  info <- fzl(check = TRUE)
}
```

---

fzo *fzo Function*

---

### Description

Reads and parses output file(s) according to the model's output commands. Each matched directory is processed independently; the results are combined into a single list or data frame.

### Usage

```
fzo(output_path, model)
```

### Arguments

`output_path` Path or glob pattern matching one or more output directories. Subdirectories within matched directories are not processed.

`model` Model definition dict or alias string.

### Value

Named list or data frame of parsed output values.

### Examples

```
if (fz_available()) {
  out_dir <- file.path(tempdir(), "P=2,V=11.2")
  dir.create(out_dir, recursive = TRUE)
  writeLines("result = 42", file.path(out_dir, "output.txt"))

  model <- list(
    varprefix = "$", delim = "{", formulaprefix = "@", commentline = "#",
    output = list(result = "grep 'result' output.txt | cut -d= -f2")
  )

  values <- fzo(out_dir, model)
}
```

---

fzr *fzr Function*

---

### Description

Runs full parametric calculations over an input template. `fzr` combines `fzc`, calculator execution, and `fzo` into a single call: it compiles the template for every parameter combination, runs the model via the calculator(s), and collects all outputs into a data frame.

**Usage**

```
fzr(
  input_path,
  input_variables,
  model,
  results_dir = "results",
  calculators = NULL,
  callbacks = NULL,
  timeout = NULL
)
```

**Arguments**

input_path	Path to input file or directory.
input_variables	Named list of variable values (or vectors of values for a full-factorial grid), or a data frame where each row is one case.
model	Model definition dict or alias string.
results_dir	Results directory. Default "results".
calculators	Calculator specification(s). Strings of the form "sh://<command>" run a local shell command; "ssh://user@host" runs over SSH; NULL auto-detects installed calculators.
callbacks	Optional named list of callback functions.
timeout	Timeout in seconds per case. Default NULL (no timeout).

**Value**

Data frame (or named list) with one row per case and columns for each input variable and output quantity.

**Examples**

```
if (fzr_available()) {
  tf <- tempfile(fileext = ".sh")
  writeLines(c(
    "#!/bin/sh",
    "echo result = $(( ${x~0} + ${y~0} )) > output.txt"
  ), tf)

  model <- list(
    varprefix = "$", delim = "{", formulaprefix = "@", commentline = "#",
    output = list(result = "grep result output.txt | cut -d= -f2")
  )

  results <- fzr(tf, list(x = c(1L, 2L), y = 3L), model,
    calculators = "sh://bash input.sh")
}
```

---

**get\_config**                    *Get the Global Configuration*

---

**Description**

Returns the fz configuration object. Values are controlled by environment variables such as FZ\_LOG\_LEVEL, FZ\_MAX\_WORKERS, FZ\_MAX\_RETRIES, and FZ\_SHELL\_PATH.

**Usage**

```
get_config()
```

**Value**

A Python Config object. Access fields with \$, e.g. get\_config()\$max\_workers.

**Examples**

```
if (fz_available()) {  
  cfg <- get_config()  
  cfg$max_workers  
  cfg$max_retries  
}
```

---

**get\_interpreter**                    *Get the Current Interpreter*

---

**Description**

Returns the global formula interpreter used when evaluating formula expressions inside template files (e.g. "python" or "R").

**Usage**

```
get_interpreter()
```

**Value**

Character string naming the current interpreter.

**Examples**

```
if (fz_available()) {  
  get_interpreter()  
}
```

---

get_log_level	<i>Get the Current Log Level</i>
---------------	----------------------------------

---

**Description**

Returns the current logging verbosity level.

**Usage**

```
get_log_level()
```

**Value**

A log-level value (use `as.character()` to convert to a string such as "DEBUG", "INFO", "WARNING", "ERROR").

**Examples**

```
if (fz_available()) {  
  as.character(get_log_level())  
}
```

---

install	<i>Install a Model or Algorithm (generic)</i>
---------	---

---

**Description**

Generic alias: installs a model from a GitHub name, URL, or local zip file. Equivalent to [install\\_model](#).

**Usage**

```
install(source, global = FALSE)
```

**Arguments**

source	GitHub name (e.g. "Funz/Model-PerfectGas"), URL, or path to a local zip file.
global	Logical; install system-wide instead of user-level. Default FALSE.

**Value**

Named list with installation details.

**Examples**

```
if (fz_available()) {
  install("Funz/Model-PerfectGas")
}
```

---

install\_algorithm      *Install an Algorithm*

---

**Description**

Installs an algorithm from a GitHub repository name, URL, or local zip file into the user-level `~/ .fz/algorithms/` directory (or system-level when `global = TRUE`).

**Usage**

```
install_algorithm(source, global = FALSE)
```

**Arguments**

source	GitHub name (e.g. "Funz/Algorithm-MonteCarlo"), URL, or path to a local zip file.
global	Logical; install system-wide instead of user-level. Default FALSE.

**Value**

Named list with installation details (path, name, ...).

**Examples**

```
if (fz_available()) {
  install_algorithm("Funz/Algorithm-MonteCarlo")
}
```

---

install\_model      *Install a Model*

---

**Description**

Installs a model from a GitHub repository name, URL, or local zip file into the user-level `~/ .fz/models/` directory (or system-level when `global = TRUE`).

**Usage**

```
install_model(source, global = FALSE)
```

**Arguments**

source	GitHub name (e.g. "Funz/Model-PerfectGas"), URL, or path to a local zip file.
global	Logical; install system-wide instead of user-level. Default FALSE.

**Value**

Named list with installation details (path, id, ...).

**Examples**

```
if (fz_available()) {  
  install_model("Funz/Model-PerfectGas")  
}
```

---

list\_installed\_algorithms

*List Installed Algorithms*

---

**Description**

Returns details of all algorithms installed in ~/.fz/algorithms/.

**Usage**

```
list_installed_algorithms(global = FALSE)
```

**Arguments**

global	Logical; list system-level installs. Default FALSE.
--------	---

**Value**

Named list of installed algorithm definitions.

**Examples**

```
if (fz_available()) {  
  algos <- list_installed_algorithms()  
  names(algos)  
}
```

---

list\_installed\_models *List Installed Models*

---

**Description**

Returns details of all models installed in `~/ .fz/models/`.

**Usage**

```
list_installed_models(global = FALSE)
```

**Arguments**

global            Logical; list system-level installs. Default FALSE.

**Value**

Named list of installed model definitions.

**Examples**

```
if (fz_available()) {  
  models <- list_installed_models()  
  names(models)  
}
```

---

list\_models            *List Installed Models (alias)*

---

**Description**

Alias for [list\\_installed\\_models](#).

**Usage**

```
list_models(global = FALSE)
```

**Arguments**

global            Logical; list system-level installs. Default FALSE.

**Value**

Named list of installed model definitions.

**Examples**

```
if (fz_available()) {  
  names(list_models())  
}
```

---

print_config	<i>Print the Current Configuration</i>
--------------	--

---

**Description**

Prints all fz configuration values in a human-readable format, including which settings come from environment variables.

**Usage**

```
print_config()
```

**Value**

NULL (invisibly). Called for side effects.

**Examples**

```
if (fz_available()) {  
  print_config()  
}
```

---

reload_config	<i>Reload Configuration from Environment Variables</i>
---------------	--

---

**Description**

Re-reads all FZ\_\* environment variables and updates the live configuration. Useful after changing environment variables within the session.

**Usage**

```
reload_config()
```

**Value**

NULL (invisibly). Called for side effects.

**Examples**

```

if (fz_available()) {
  Sys.setenv(FZ_MAX_WORKERS = "8")
  reload_config()
  get_config()$max_workers
}

```

---

set\_interpreter      *Set the Interpreter*

---

**Description**

Sets the global formula interpreter for evaluating expressions inside template files.

**Usage**

```
set_interpreter(interpreter)
```

**Arguments**

interpreter      Character string: "python" or "R".

**Value**

NULL (invisibly). Called for side effects.

**Examples**

```

if (fz_available()) {
  set_interpreter("R")
  set_interpreter("python")
}

```

---

set\_log\_level      *Set the Log Level*

---

**Description**

Controls how much output fz emits during execution.

**Usage**

```
set_log_level(level)
```

**Arguments**

level            Character string or log-level object: one of "DEBUG", "INFO", "WARNING", "ERROR".

**Value**

NULL (invisibly). Called for side effects.

**Examples**

```
if (fz_available()) {
  set_log_level("DEBUG")
  set_log_level("WARNING")
  set_log_level("ERROR")
}
```

---

uninstall

*Uninstall a Model (generic)*

---

**Description**

Generic alias: removes a model by name. Equivalent to [uninstall\\_model](#).

**Usage**

```
uninstall(model_name, global = FALSE)
```

**Arguments**

model\_name      Name of the model to remove.  
global           Logical; remove from system-level install. Default FALSE.

**Value**

TRUE if removed, FALSE otherwise.

**Examples**

```
if (fz_available()) {
  uninstall("PerfectGas")
}
```

uninstall\_algorithm     *Uninstall an Algorithm*

---

**Description**

Removes a previously installed algorithm from `~/ .fz/algorithms/`.

**Usage**

```
uninstall_algorithm(algorithm_name, global = FALSE)
```

**Arguments**

`algorithm_name`     Name of the algorithm to remove.  
`global`             Logical; remove from system-level install. Default FALSE.

**Value**

TRUE if the algorithm was removed, FALSE otherwise.

**Examples**

```
if (fz_available()) {  
  uninstall_algorithm("MonteCarlo")  
}
```

---

uninstall\_model         *Uninstall a Model*

---

**Description**

Removes a previously installed model from `~/ .fz/models/`.

**Usage**

```
uninstall_model(model_name, global = FALSE)
```

**Arguments**

`model_name`         Name of the model to remove (e.g. "PerfectGas").  
`global`             Logical; remove from system-level install. Default FALSE.

**Value**

TRUE if the model was removed, FALSE otherwise.

**Examples**

```
if (fz_available()) {  
  uninstall_model("PerfectGas")  
}
```

# Index

fz\_available, 2  
fz\_install, 3  
fzc, 4, 8  
fzd, 5  
fzi, 6  
fzl, 7  
fzo, 8, 8  
fzr, 5, 8

get\_config, 10  
get\_interpreter, 10  
get\_log\_level, 11

install, 11  
install\_algorithm, 12  
install\_model, 11, 12

list\_installed\_algorithms, 13  
list\_installed\_models, 14, 14  
list\_models, 14

print\_config, 15

reload\_config, 15  
reticulate::py\_install(), 3

set\_interpreter, 16  
set\_log\_level, 16

uninstall, 17  
uninstall\_algorithm, 18  
uninstall\_model, 17, 18