

Package: plateforce (via r-universe)

September 14, 2026

Title Force-Plate Jump Kinetics that Carry Their Method

Version 0.1.2

Description Computes jump kinetics from a vertical ground reaction force trace. Every value carries the method that produced it: the rule, its published parameters, its citation and the choices made upstream of it. Across 244 trials from Harrison (2011) <doi:10.25710/929y-2n46>, ten published ways of computing one jump height disagree by a median 3.51 cm, exceeding the study's 1.98 cm training effect, so the method is reported with the number. Authorship and licences of the bundled Rust sources are listed in the AUTHORS file of the installed package.

License Apache License (== 2.0)

URL <https://github.com/DrAlexHarrison/plateforce>

BugReports <https://github.com/DrAlexHarrison/plateforce/issues>

Depends R (>= 4.3.0)

Imports S7

Suggests testthat (>= 3.0.0), knitr, rmarkdown, jsonlite, withr

SystemRequirements Cargo (Rust's package manager), rustc >= 1.82

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Collate 'acquisition.R' 'rust.R' 'request.R' 'refusal.R' 'trial.R'
'provenance.R' 'measured.R' 'analyse.R' 'capability.R'
'coercion.R' 'describe.R' 'plate.R' 'quality.R' 'read.R'
'zzz.R' 'registry.R' 'spread.R' 'version.R'

Config/pak/sysreqs libclang-dev

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

Date/Publication 2026-08-15 03:35:57 UTC
RemoteUrl <https://github.com/DrAlexHarrison/plateforce>
RemoteRef HEAD
RemoteSha 04f43fbf3968ef7725e7398d13793b3a93fc8103
RemoteSubdir bindings/r

Contents

analyse_countermovement_jump	2
capability_json	4
describe	5
pf_acquisition	6
pf_acquisition_members	7
pf_bindings	7
pf_entry	8
pf_force	8
pf_plate	9
pf_plate_forget	10
pf_plate_save	10
pf_plates	11
pf_read_force_file	12
pf_registry	13
pf_signals	13
pf_spread	14
pf_trial	17
pf_value	18
pf_version	19
plateforce_refusal	19
Index	21

analyse_countermovement_jump

Analyse a countermovement jump

Description

Three landmark rules are named by their registry identifiers, and every parameter each one read travels in the result beside the number it produced.

Usage

```
analyse_countermovement_jump(
  trial,
  weighing = NULL,
  onset = NULL,
  takeoff = NULL,
  preset = NULL,
  gravity_meters_per_second_squared = NULL,
  body_mass_kilograms = NULL,
  weighing_parameters = NULL,
  onset_parameters = NULL,
  takeoff_parameters = NULL,
  weighing_options = NULL,
  onset_options = NULL,
  takeoff_options = NULL,
  weighing_start_index = NULL,
  onset_index = NULL,
  takeoff_index = NULL,
  touchdown_index = NULL,
  derived = NULL,
  conditioning = NULL,
  registry = NULL,
  registry_version = NULL
)
```

Arguments

trial	A trial, as returned by <code>pf_trial()</code> or <code>pf_read_force_file()</code> .
weighing	Registry id of the rule that establishes system weight.
onset	Registry id of the rule that places the start of the jump.
takeoff	Registry id of the rule that places the instant of takeoff.
preset	Name of a published pipeline, which binds the rules and the values its source states and leaves every construct that source is silent about to the caller. Every value it supplies is recorded as cited, naming the pipeline, so a result reached this way is a different record from one reached by typing the same numbers.
gravity_meters_per_second_squared	Gravitational acceleration. A bound parameter rather than a constant, because published tools disagree on it, and it appears in the record beside the number it moved.
body_mass_kilograms	The athlete's mass, which is not the weighed system mass: system weight includes any bar and bodyweight does not. Absent, no mass is on the record, and a mass that is not a finite number above zero is refused by name.
weighing_parameters, onset_parameters, takeoff_parameters	Named numeric lists, as the registry names each rule's parameters.

weighing_options, onset_options, takeoff_options	Named character lists, for the settings that are a choice between named alternatives rather than a number.
weighing_start_index, onset_index, takeoff_index, touchdown_index	Indices placed by hand. An override is a fact about what produced the number, so it travels in the record rather than replacing it.
derived	Rules for constructs computed from the landmarks, named by the construct id the registry declares: <code>list(analysis_window = "window_end.takeoff.detected", peak_force = "force.peak.gross")</code> . A rule that takes values is written <code>list(peak_force = list(method_id = "force.peak.estimator", parameters = list(averaging_window_seconds = 0.1)))</code> . A rule that reads what another one placed declines by name when that construct was not named.
conditioning	Rules and values for the phase that conditions the signal before the landmarks are placed, named by the construct id the registry declares: <code>list(conditioned_force_signal = "filter.none")</code> . Values are written <code>list(conditioned_force_signal = list(options = list(passband_edge = "none")))</code> , with the rule left out where the phase's own is wanted. The phase runs whether or not this is passed, so what it buys is the record naming the caller rather than the software.
registry	Directory holding the registry, as in <code>pf_registry()</code> .
registry_version	The revision of the registry data to cite in the result. Absent, the result names no pinned revision and reports the one the registry declares for itself. The two are recorded separately: a revision the caller cited and one the data claimed about itself are different facts.

Value

A countermovement_jump. Read one quantity out of it with `pf_value()`.

Examples

```
standing <- pf_trial(rep(700, 1200), sample_rate_hz = 1200)
result <- analyse_countermovement_jump(
  standing,
  weighing = "bwepoch.fixed_window",
  onset = "onset.threshold.noise_relative",
  takeoff = "takeoff.threshold.absolute_force"
)
pf_value(result, "system_weight_newtons")@value
result@warnings
```

capability_json

What this surface can be asked to do

Description

Every method available, the operations this package dispatches, the container formats it writes, and every way it can decline with the exit status a shell reads.

Usage

```
capability_json(registry = NULL)
```

Arguments

registry Path to a registry directory, or NULL to describe the registry this package ships. Every rule's published values are read from it, so a manifest and an analyse() call resolving different registries would report values the other would refuse. Resolved through the same registry_root() both use.

Details

The operations are the ones this package exports rather than a list forwarded from the engine.

Value

A single JSON string with sorted keys and no spacing, so a comparison against another surface is a plain diff.

Examples

```
substr(capability_json(), 1, 40)
```

describe

The account a number gives of itself

Description

The value, its unit, the rule that produced it, and every rule upstream of that one with the parameters each was bound to.

Usage

```
describe(x, ...)
```

Arguments

x A measured, as returned by `pf_value()`.
... Unused.

Details

The text is generated by the engine, so a number reads the same in an R session, a browser tab, a terminal and a file somebody emailed on.

Value

x, invisibly. The account is printed.

Examples

```
standing <- pf_trial(rep(700, 1200), sample_rate_hz = 1200)
result <- analyse_countermovejump(
  standing,
  weighing = "bwepoch.fixed_window",
  onset = "onset.threshold.noise_relative",
  takeoff = "takeoff.threshold.absolute_force"
)
describe(pf_value(result, "system_weight_newtons"))
```

pf_acquisition

What the plate and its settings were

Description

Five facts about the capture that no reanalysis recovers. A block missing any of them fingerprints as incomplete, so results from that trial are never declared to match another lab's.

Usage

```
pf_acquisition(
  filter_at_capture = NULL,
  tare_state = NULL,
  plate_natural_frequency_hz = NULL,
  floor_surface = NULL,
  firmware_version = NULL
)
```

Arguments

filter_at_capture	The filter the plate applied before the trace was written, which no later filtering undoes.
tare_state	Whether and when the plate was zeroed.
plate_natural_frequency_hz	The plate's natural frequency.
floor_surface	What the plate was standing on.
firmware_version	The plate firmware that wrote the trace.

Details

sample_rate_hz is the sixth member of the block and belongs to the trial, so it is stated in [pf_trial\(\)](#) rather than here.

Value

A named list, for [pf_trial\(\)](#) and [pf_read_force_file\(\)](#).

Examples

```
partial <- pf_acquisition(tare_state = "tared_before_trial")
trial <- pf_trial(rep(700, 1200), sample_rate_hz = 1200, acquisition = partial)
trial@acquisition_complete
trial@acquisition_missing
```

pf_acquisition_members

The members the acquisition block declares

Description

Named by the block itself, so a member that arrives upstream arrives here.

Usage

```
pf_acquisition_members()
```

Value

A character vector of member names.

Examples

```
pf_acquisition_members()
```

pf_bindings

The rules available for each step

Description

The rules available for each step

Usage

```
pf_bindings()
```

Value

A data frame with one row per rule, carrying the registry id it binds, the slot it fills, the construct it computes, and the entry it composes an operator onto when it does.

Examples

```
pf_bindings()[["id"]]
```

pf_entry	<i>One registry entry</i>
----------	---------------------------

Description

One registry entry

Usage

```
pf_entry(id, path = NULL)
```

Arguments

id	Canonical dotted identifier, for example "onset.threshold.noise_relative".
path	Directory holding the registry, as in pf_registry() .

Value

A list carrying the entry's parameters, citations, biases, failure block, disagreements and surfacing verdict, as the registry states them.

Examples

```
entry <- pf_entry("onset.threshold.noise_relative")
entry[["construct"]]
```

pf_force	<i>The force trace a trial holds</i>
----------	--------------------------------------

Description

The force trace a trial holds

Usage

```
pf_force(x)
```

Arguments

x	A trial.
---	----------

Value

A double vector of vertical ground reaction force in newtons.

Examples

```
quiet <- pf_trial(rep(700, 12), sample_rate_hz = 12)
length(pf_force(quiet))
```

pf_plate

One plate, saved here or stated from its members

Description

Naming acquisition states a plate from the members the caller holds, with no file behind it, which is how a plate travels between machines. Leaving it out reads the one this machine has saved under that name.

Usage

```
pf_plate(name, acquisition = NULL, plates_folder = NULL)
```

Arguments

name	The plate to read, or to call the stated one.
acquisition	The members it holds, from <code>pf_acquisition()</code> . Absent reads a saved one.
plates_folder	Keep saved plates here rather than beside this machine's other settings. A plate saved under a named folder travels with a dataset; one saved without travels with the person.

Details

The revision is taken from the members either way, so a plate stated from a colleague's answers and the same plate saved here carry one revision and one attribution.

Value

A list carrying the plate, its revision, where it is filed when it is, the block it holds and the members still to answer.

Examples

```
folder <- tempfile("plates")
pf_plate_save("lab-1", pf_acquisition(floor_surface = "concrete"), plates_folder = folder)
pf_plate("lab-1", plates_folder = folder)[["acquisition"]][["floor_surface"]]

# The same plate, stated rather than read.
pf_plate("lab-1", pf_acquisition(floor_surface = "concrete"))[["revision"]]
```

pf_plate_forget	<i>Remove a saved plate</i>
-----------------	-----------------------------

Description

Results already recorded against it carry its members and are unchanged.

Usage

```
pf_plate_forget(name, plates_folder = NULL)
```

Arguments

name	The plate to remove.
plates_folder	Keep saved plates here rather than beside this machine's other settings. A plate saved under a named folder travels with a dataset; one saved without travels with the person.

Value

A list naming the plate and the file that was removed.

Examples

```
folder <- tempfile("plates")
pf_plate_save("lab-1", pf_acquisition(tare_state = "tared"), plates_folder = folder)
pf_plate_forget("lab-1", plates_folder = folder)[["plate"]]
```

pf_plate_save	<i>Record a plate's settings</i>
---------------	----------------------------------

Description

Answers five questions once instead of at every analysis. A run told about a saved plate fills its acquisition block from it and records which plate and which revision of it the answers came off.

Usage

```
pf_plate_save(name, acquisition = NULL, plates_folder = NULL)
```

Arguments

name	What to call this plate. Letters, digits, - and _.
acquisition	What the plate and its settings were, from pf_acquisition() .
plates_folder	Keep saved plates here rather than beside this machine's other settings. A plate saved under a named folder travels with a dataset; one saved without travels with the person.

Details

Saving over a name changes what later runs are told. The revision that was there before comes back in `replaced_revision`, and each member whose answer moved in `replaced_members`, because a result recorded earlier rests on answers this machine no longer holds.

Value

A list carrying the plate, its revision, where it is filed, the block it holds and the members still to answer, with `replaced_revision` and `replaced_members` where a plate of that name was already saved.

Examples

```
folder <- tempfile("plates")
saved <- pf_plate_save(
  "lab-1",
  pf_acquisition(tare_state = "tared_before_trial"),
  plates_folder = folder
)
saved[["revision"]]
saved[["acquisition_missing"]]
```

pf_plates

Every plate this machine holds

Description

Every plate this machine holds

Usage

```
pf_plates(plates_folder = NULL)
```

Arguments

`plates_folder` Keep saved plates here rather than beside this machine's other settings. A plate saved under a named folder travels with a dataset; one saved without travels with the person.

Value

A list carrying `plates_folder` and one entry per saved plate.

Examples

```
folder <- tempfile("plates")
pf_plate_save("lab-1", pf_acquisition(tare_state = "tared"), plates_folder = folder)
length(pf_plates(plates_folder = folder)[["plates"]])
```

pf_read_force_file *Read a force file*

Description

The engine reads the file, so every interface uses the same force column and the same meaning for each row.

Usage

```
pf_read_force_file(
    path,
    sample_rate_hz = NULL,
    delimiter = NULL,
    force_column = NULL,
    sentinel_convention = "none",
    acquisition = NULL,
    plate = NULL,
    plates_folder = NULL
)
```

Arguments

path	Path to a delimited text export.
sample_rate_hz	The rate the trace was recorded at.
delimiter	The single character separating this file's columns.
force_column	Zero-based index of the column carrying vertical ground reaction force.
sentinel_convention	How this export writes a missing sample: "none", "zero" or "negative_one".
acquisition	What the plate and its settings were, from pf_acquisition() .
plate	A saved plate to fill the block from: a name this machine holds, or the list pf_plate() returns. A member in acquisition beside it is the answer that runs, and the result records what it replaced.
plates_folder	Where to look for plate, when it is a name. Absent reads the folder plateforce plate save writes to.

Details

Every choice the read rests on is stated by the caller and reported back. A rate that is guessed scales every velocity, displacement, impulse and rate of force development with it, and a column that is guessed can be the wrong one quietly, so neither has a default.

Value

A trial whose @read_report names the delimiter, the column, the rows read, the columns per row, the blank lines skipped, the samples that matched the sentinel convention, and the samples that carried no number at all.

pf_registry	<i>Load the method registry</i>
-------------	---------------------------------

Description

Load the method registry

Usage

```
pf_registry(path = NULL)
```

Arguments

path	Directory holding the registry. When absent, the registry shipped inside this package is used, unless PLATEFORCE_REGISTRY names another one.
------	--

Value

A registry. @census is one row per population, with each derived count beside the denominator it was taken over. The populations are separate counts and are never added together, so there is no total row and no total column. @digest identifies the files that were read, measured from their bytes. @declared_version is the revision the registry names about itself, empty where it names none. The two answer different questions: a revision is a name to cite and a digest says which bytes were behind it, and the revision lives beside the rules rather than among them, so a digest cannot recover it.

Examples

```
reg <- pf_registry()
reg@census
```

pf_signals	<i>The signals raised over one analysis</i>
------------	---

Description

Each names what was compared, the value it compared, the threshold it applied, and an action. A result with none is a result nothing was noticed about.

Usage

```
pf_signals(x)
```

Arguments

x	A countermovement_jump.
---	-------------------------

Value

A list of signals.

Examples

```
standing <- pf_trial(rep(700, 1200), sample_rate_hz = 1200)
result <- analyse_countermovement_jump(
  standing,
  weighing = "bwepoch.fixed_window",
  onset = "onset.threshold.noise_relative",
  takeoff = "takeoff.threshold.absolute_force"
)
pf_signals(result)
```

pf_spread

How far the method choice moves one number

Description

Sweeps a step's alternatives on one trial and reports the spread over them.

Usage

```
pf_spread(
  trial,
  quantity,
  slot = NULL,
  weighing = NULL,
  onset = NULL,
  takeoff = NULL,
  preset = NULL,
  method_ids = NULL,
  vary = NULL,
  gravity_meters_per_second_squared = NULL,
  body_mass_kilograms = NULL,
  weighing_parameters = NULL,
  onset_parameters = NULL,
  takeoff_parameters = NULL,
  weighing_options = NULL,
  onset_options = NULL,
  takeoff_options = NULL,
  weighing_start_index = NULL,
  onset_index = NULL,
  takeoff_index = NULL,
  touchdown_index = NULL,
  derived = NULL,
  conditioning = NULL,
```

```

    maximum_combinations = NULL,
    registry = NULL,
    registry_version = NULL
)

```

Arguments

trial	A trial, as returned by <code>pf_trial()</code> or <code>pf_read_force_file()</code> .
quantity	The engine's name for the quantity to sweep, for example "jump_height_from_takeoff_meters".
slot	One step whose rule is swept, or several. Several sweeps every combination of them, which is the question a number resting on more than one rule asks: the onset rule and the takeoff rule both move a jump height, and sweeping them one at a time reports neither the widest disagreement nor the narrowest.
weighing	Registry id of the rule that establishes system weight.
onset	Registry id of the rule that places the start of the jump.
takeoff	Registry id of the rule that places the instant of takeoff.
preset	Name of a published pipeline, which binds the rules and the values its source states and leaves every construct that source is silent about to the caller. Every value it supplies is recorded as cited, naming the pipeline, so a result reached this way is a different record from one reached by typing the same numbers.
method_ids	Registry identifiers to sweep. A character vector is the set for the one step named; a list keyed by step names the ids for each. Absent, every rule bound to that step is compared, read off the binding table rather than listed here.
vary	Settings to sweep, keyed by the step and the setting, as <code>list("onset.k" = c(2, 5, 10))</code> . Numbers or names per key: "epoch_impulse.convention" = <code>c("net", "gross")</code> compares two names, and "global.gravity_meters_per_second_squared" sweeps the value the run carries rather than one a rule reads. Written beside slot, the rules and the settings vary together.
gravity_meters_per_second_squared	Gravitational acceleration. A bound parameter rather than a constant, because published tools disagree on it, and it appears in the record beside the number it moved.
body_mass_kilograms	The athlete's mass, which is not the weighed system mass: system weight includes any bar and bodyweight does not. Absent, no mass is on the record, and a mass that is not a finite number above zero is refused by name.
weighing_parameters, onset_parameters, takeoff_parameters	Named numeric lists, as the registry names each rule's parameters.
weighing_options, onset_options, takeoff_options	Named character lists, for the settings that are a choice between named alternatives rather than a number.
weighing_start_index, onset_index, takeoff_index, touchdown_index	Indices placed by hand. An override is a fact about what produced the number, so it travels in the record rather than replacing it.

derived	Rules for constructs computed from the landmarks, named by the construct id the registry declares: <code>list(analysis_window = "window_end.takeoff.detected", peak_force = "force.peak.gross")</code> . A rule that takes values is written <code>list(peak_force = list(method_id = "force.peak.estimator", parameters = list(averaging_window_seconds = 0.1)))</code> . A rule that reads what another one placed declines by name when that construct was not named.
conditioning	Rules and values for the phase that conditions the signal before the landmarks are placed, named by the construct id the registry declares: <code>list(conditioned_force_signal = "filter.none")</code> . Values are written <code>list(conditioned_force_signal = list(options = list(passband_edge = "none")))</code> , with the rule left out where the phase's own is wanted. The phase runs whether or not this is passed, so what it buys is the record naming the caller rather than the software.
maximum_combinations	Cap on how many combinations run.
registry	Directory holding the registry, as in <code>pf_registry()</code> .
registry_version	The revision of the registry data to cite in the result. Absent, the result names no pinned revision and reports the one the registry declares for itself. The two are recorded separately: a revision the caller cited and one the data claimed about itself are different facts.

Value

A list carrying variants, one per combination with its label, its value and the reason it declined when it did, and the summary the sweep computed over them: minimum, maximum, median, spread_absolute, spread_percent_of_median, and succeeded and failed beside the count that was requested. A sweep over one combination carries no spread_absolute, because one number holds no disagreement.

Examples

```
standing <- pf_trial(rep(700, 1200), sample_rate_hz = 1200)
sweep <- pf_spread(
  standing,
  quantity = "system_weight_newtons",
  slot = "weighing",
  weighing = "bwepoch.fixed_window",
  onset = "onset.threshold.noise_relative",
  takeoff = "takeoff.threshold.absolute_force"
)
sweep[["succeeded"]]
sweep[["spread_absolute"]]

# Several steps at once, over every combination of their rules.
over_the_landmarks <- pf_spread(
  standing,
  quantity = "system_weight_newtons",
  slot = c("weighing", "onset"),
  weighing = "bwepoch.fixed_window",
  onset = "onset.threshold.noise_relative",
```

```

    takeoff = "takeoff.threshold.absolute_force"
  )
  over_the_landmarks[["combinations_run"]]

# The rules and a value inside them, on one call.
rules_and_a_value <- pf_spread(
  standing,
  quantity = "system_weight_newtons",
  slot = "weighing",
  vary = list("weighing.duration" = c(0.5, 1.0)),
  weighing = "bwepoch.fixed_window",
  onset = "onset.threshold.noise_relative",
  takeoff = "takeoff.threshold.absolute_force"
)
rules_and_a_value[["combinations_run"]]

```

pf_trial

*Build a trial from a force vector***Description**

Build a trial from a force vector

Usage

```

pf_trial(
  force_newtons,
  sample_rate_hz = NULL,
  sentinel_convention = "none",
  acquisition = NULL,
  plate = NULL,
  plates_folder = NULL
)

```

Arguments

force_newtons	Vertical ground reaction force, in newtons, as a double vector.
sample_rate_hz	The rate the trace was recorded at. A rate that is guessed scales every velocity, displacement, impulse and rate of force development with it, so this has no default.
sentinel_convention	How this export writes a missing sample: "none", "zero" or "negative_one". A sample matching the convention is counted and left where it is, because closing the gap would shift every timestamp after it. Samples carrying no number are counted separately, whatever convention is declared.
acquisition	What the plate and its settings were, from pf_acquisition() . A block missing any member makes every result from this trial carry acquisition_complete = FALSE.

plate	A saved plate to fill the block from: a name this machine holds, or the list <code>pf_plate()</code> returns. A member in acquisition beside it is the answer that runs, and the result records what it replaced.
plates_folder	Where to look for plate, when it is a name. Absent reads the folder plateforce plate save writes to.

Value

A trial.

Examples

```
quiet <- pf_trial(rep(700, 1200), sample_rate_hz = 1200)
quiet@sample_count
quiet@duration_seconds
```

pf_value	<i>One quantity out of an analysis</i>
----------	--

Description

One quantity out of an analysis

Usage

```
pf_value(x, quantity)
```

Arguments

x	A countermovement_jump.
quantity	The engine's name for the quantity, matched in full.

Value

A measured, the number with the rule that produced it attached.

pf_version	<i>The version of this package</i>
------------	------------------------------------

Description

The version of this package

Usage

```
pf_version()
```

Value

A single string, the package version.

Examples

```
pf_version()
```

plateforce_refusal	<i>Refusals</i>
--------------------	-----------------

Description

A rule that declines raises a condition rather than returning a sentinel. The condition carries the code, the method, the slot, the parameter, the value, the detail and what was available instead, so a caller branches on the fields rather than parsing a sentence.

Details

The class vector is `c("plateforce_<code>", "plateforce_refusal", "plateforce_error")`, so `tryCatch` handles one code or every refusal.

Read the fields with `[[`. `cn$method` returns the value of `method_id` on a plain condition, silently, because `$` partial matches on a list, so this package defines `$` on a refusal to require the whole name.

Examples

```
standing <- pf_trial(rep(700, 1200), sample_rate_hz = 1200)
condition <- tryCatch(
  analyse_countermovement_jump(
    standing,
    weighing = "bwepoch.fixed_window",
    onset = "onset.not.a.rule",
    takeoff = "takeoff.threshold.absolute_force"
  ),
  plateforce_refusal = identity
```

```
)  
condition[["code"]]  
condition[["message"]]
```

Index

`analyse_countermovement_jump`, [2](#)

`capability_json`, [4](#)

`describe`, [5](#)

`pf_acquisition`, [6](#)

`pf_acquisition()`, [9](#), [10](#), [12](#), [17](#)

`pf_acquisition_members`, [7](#)

`pf_bindings`, [7](#)

`pf_entry`, [8](#)

`pf_force`, [8](#)

`pf_plate`, [9](#)

`pf_plate()`, [12](#), [18](#)

`pf_plate_forget`, [10](#)

`pf_plate_save`, [10](#)

`pf_plates`, [11](#)

`pf_read_force_file`, [12](#)

`pf_read_force_file()`, [3](#), [6](#), [15](#)

`pf_registry`, [13](#)

`pf_registry()`, [4](#), [8](#), [16](#)

`pf_signals`, [13](#)

`pf_spread`, [14](#)

`pf_trial`, [17](#)

`pf_trial()`, [3](#), [6](#), [15](#)

`pf_value`, [18](#)

`pf_value()`, [4](#), [5](#)

`pf_version`, [19](#)

`plateforce_refusal`, [19](#)