# Package 'nfer'

July 22, 2025

**Title** Event Stream Abstraction using Interval Logic

**Version** 1.1.3

**Description** This is the R API for the 'nfer' formalism (<http://nfer.io/>). 'nfer' was developed to specify event stream abstractions for spacecraft telemetry such as the Mars Science Laboratory. Users write rules using a syntax that borrows heavily from Allen's Temporal Logic that, when applied to an event stream, construct a hierarchy of temporal intervals with data. The R API supports loading rules from a file or mining them from historical data. Traces of events or pools of intervals are provided as data frames.

**License** GPL (>= 3)

**Encoding** UTF-8

**URL** http://nfer.io/

**BugReports** https://bitbucket.org/seanmk/nfer/issues

**RoxygenNote** 7.2.3

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

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Sean Kauffman [aut, cre] (ORCID: <https://orcid.org/0000-0001-6341-3898>)

**Maintainer** Sean Kauffman <seank@cs.aau.dk>

**Repository** CRAN

**Date/Publication** 2023-04-12 10:00:02 UTC

## Contents

---

`apply` *Apply a loaded nfer specification to a dataframe of events.*

---

## Description

This function obtains the interval abstractions generated by applying the nfer specification to the events.

## Usage

```
apply(handle, events)
```

## Arguments

| | |
|---|---|
| `handle` | The loaded nfer specification (using `nfer::load`, or `nfer::learn`) |
| `events` | The dataframe of events to abstract using nfer. |

## Details

Event traces are passed as dataframes with at least two columns. The first two columns contain event names and timestamps, respectively. Names should be strings. Timestamps should be integers or strings, ideally, but may be numeric. Subsequent columns contain data, where the column name is the data key. The column value will be the data value for each event, where NA means no value is present.

Example dataframe events:

```
| name | timestamp | x |    y |
-------------------------------
| foo  | 123       | 2 | NA   |
| bar  | 987       | 3 | TRUE |
```

The result of the function is also a dataframe, but this contains intervals. The difference is that it has an extra timestamp column. Column two is now the start time of the interval, and column 3 is the end time. Name is still column one and columns 4+ are still data.

Dataframe interval output:

```
| name | start | end | z  |
---------------------------
| far  | 123   | 987 | NA |
| baf  | 654   | 987 | 3  |
```

## Value

A dataframe containing intervals.

## Examples

```
ops <- nfer::load(system.file("extdata", "ops.nfer", package = "nfer"))
events <- nfer::read(system.file("extdata", "ops.events", package = "nfer"))
intervals <- nfer::apply(ops, events)
```

---

`learn` *Learn an nfer specification from an event trace.*

---

## Description

This function applies the nfer mining algorithm to a trace to attempt to learn a specification. It will only learn before rules and will only find those when there is periodic behavior in the trace. This works best for system traces of real-time systems where events represent scheduled, periodic behavior. At this time, only the event names will be considered while any data is ignored.

## Usage

```
learn(events, loglevel = 0)
```

## Arguments

| | |
|---|---|
| `events` | The dataframe containing events. |
| `loglevel` | (Optional) The logging level to set (0-4), where 0 is only warnings and errors and 3 is debug. Default is 0. |

## Details

Event traces are passed as dataframes with at least two columns. The first two columns contain event names and timestamps, respectively. Names should be strings. Timestamps should be integers or strings, ideally, but may be numeric. Subsequent columns will be ignored by this function, for now.

Example dataframe events:

```
| name | timestamp | x |    y |
-------------------------------
| foo  | 123       | 2 | NA   |
| bar  | 987       | 3 | TRUE |
```

## Value

A handle to a learned nfer specification, loaded into R.

## Examples

```
events <- nfer::read(system.file("extdata", "qnx.events", package = "nfer"))
learned <- nfer::learn(events)
learned <- nfer::learn(events, loglevel=1)
```

load *Load an nfer specification from a file.*

## Description

This function loads an existing nfer specification, stored in a file, into R in a format that can be used by the `nfer::apply` function. See http://nfer.io/ for information on how to write nfer specifications.

## Usage

```
load(specfile, loglevel = 0)
```

## Arguments

| | |
|---|---|
| `specfile` | The name of file containing the nfer specification to load. |
| `loglevel` | (Optional) The logging level to set (0-4), where 0 is only warnings and errors and 3 is debug. Default is 0. |

## Details

Nfer specifications consist of rules that describe a relationship between pairs of intervals. These relationships describe new intervals that can then be used to describe more intervals. The result is a hierarchy of intervals. Although the algorithm takes intervals as its input, event traces are actually the target, so events are converted to intervals with zero duration.

## Value

A handle to the nfer specification, loaded into R.

## Examples

```
test <- nfer::load(system.file("extdata", "ops.nfer", package = "nfer"))
ssps <- nfer::load(system.file("extdata", "ssps.nfer", package = "nfer"), loglevel=1)
```

nfer *nfer: A package for inferring interval abstractions of event traces.*

## Description

The nfer package provides an R interface to the nfer language for abstracting event traces into a hierarchy of intervals with data.

## nfer functions

load, learn, read, apply

read *Reads an nfer event file into an R dataframe.*

## Description

The nfer event format is not read easily into R. This function essentially simplifies sharing event traces between the nfer command-line tool and the R package.

## Usage

```
read(event_file)
```

## Arguments

| | |
|---|---|
| event_file | The name of the file containing nfer formatted events. |

## Details

Nfer event files are formatted as follows: Each row is a pipe-separated list of either 2 or 4 values represents one event. The first column is the event name. The second is the event timestamp (integer). The third and fourth columns are used when the event carries data. The third column is a semicolon separated list of data keys. The fourth column is a semicolon separate list of data values in the same order as the keys.

Example event file contents:

```
foo|123|x|2
bar|987|x;y|3;true
```

The result of reading such a file with this function is a dataframe where the first column contains event names, the second contains event timestamps, and subsequent columns contain data. The data is formatted with one column per name. Events that don't carry some data key will have an NA value for that column.

Resulting dataframe of event:

```
| name | timestamp | x |    y |
-------------------------------
| foo  | 123       | 2 | NA   |
| bar  | 987       | 3 | TRUE |
```

## Value

A dataframe containing the events.

## Examples

```
events <- nfer::read(system.file("extdata", "ops.events", package = "nfer"))
```

# Index