# Package: trace (via r-universe)

November 18, 2026

**Title** Tandem Repeat Analysis by Capillary Electrophoresis

**Version** 1.0.0

**Description** A pipeline for short tandem repeat instability analysis from fragment analysis data. Inputs of fsa files or peak tables, and a user supplied metadata data-frame. The package identifies ladders, calls peaks, identifies the modal peaks, calls repeats, then calculates repeat instability metrics (e.g. expansion index from Lee et al. (2010) <doi:10.1186/1752-0509-4-29>).

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Suggests** knitr, dplyr, ggplot2, shinytest2, testthat (>= 3.0.0), rmarkdown

**Config/testthat/edition** 3

**Imports** graphics, grDevices, lme4, methods, mgcv, plotly, pracma, seqinr, shiny, stats, utils, yaml

**VignetteBuilder** knitr

**Depends** R (>= 3.5)

**LazyData** true

**URL** https://zachariahmclean.github.io/trace/

**Config/pak/sysreqs** cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

**Repository** https://zachariahmclean.r-universe.dev

**Date/Publication** 2025-12-23 15:29:15 UTC

**RemoteUrl** https://github.com/zachariahmclean/trace

**RemoteRef** HEAD

**RemoteSha** ace54658ca1bdc98f06f5a0ea5fd9fedc6518053

# Contents

calculate_instability_metrics . . . 2
cell_line_fsa_list . . . 5
example_data . . . 6
example_data_repeat_table . . . 6
extract_alleles . . . 7
extract_fragments . . . 7
extract_ladder_summary . . . 8
extract_repeat_correction_summary . . . 9
extract_trace_table . . . 10
fix_ladders_interactive . . . 10
fragments . . . 12
genemapper_table_to_fragments . . . 15
load_config . . . 16
metadata . . . 17
plot_batch_correction_samples . . . 17
plot_data_channels . . . 19
plot_fragments . . . 19
plot_ladders . . . 20
plot_repeat_correction_model . . . 21
plot_traces . . . 22
read_fsa . . . 23
remove_fragments . . . 24
repeat_table_to_fragments . . . 25
size_table_to_fragments . . . 26
trace . . . 27

**Index** **32**

---

`calculate_instability_metrics`

*Calculate Repeat Instability Metrics*

---

## Description

This function computes instability metrics from a list of fragments data objects.

## Usage

```
calculate_instability_metrics(
  fragments_list,
  peak_threshold = 0.05,
  window_around_index_peak = c(NA_real_, NA_real_),
  percentile_range = c(0.5, 0.75, 0.9, 0.95),
  repeat_range = c(2, 5, 10, 20),
  index_modal_signal_threshold = NA_real_,
  index_signal_sum_threshold = NA_real_
)
```

## Arguments

| | |
|---|---|
| `fragments_list` | A list of "fragments" objects representing fragment data. |
| `peak_threshold` | A single numeric value between 0 and 1 for the threshold of peak signals to be considered in the calculations, relative to the modal peak signal of the expanded allele. |
| `window_around_index_peak` | A numeric vector (length 2) defining the range around the index peak. First number specifies repeats before the index peak, second after. For example, `c(-5, 40)` around an index peak of 100 would analyze repeats 95 to 140. The sign of the numbers does not matter (The absolute value is found). |
| `percentile_range` | A numeric vector of percentiles to compute (e.g., c(0.5, 0.75, 0.9, 0.95)). |
| `repeat_range` | A numeric vector specifying ranges of repeats for the inverse quantile computation. |
| `index_modal_signal_threshold` | A single numeric value for the minimum signal of the modal peak for the index samples (basically a quality control for the samples used to set the index peak or to calculate average_repeat_change or instability_index_change). This is only relevant when grouped = TRUE for the index peak assignment. |
| `index_signal_sum_threshold` | A single numeric value for the minimum sum of all peaks for each index sample (basically a quality control for the samples used to set the index peak or to calculate average_repeat_change or instability_index_change). This is only relevant when grouped = TRUE for the index peak assignment. |

## Details

Each of the columns in the supplied dataframe are explained below:

**General Information:**

- `unique_id`: A unique identifier for the sample (usually the fsa file name).

**Quality Control:**

- `QC_comments`: Quality control comments.
- `QC_modal_peak_signal`: Quality control status based on the modal peak signal (Low < 500, very low < 100).
- `QC_peak_number`: Quality control status based on the number of peaks (Low < 20, very low < 10).
- `QC_off_scale`: Quality control comments for off-scale peaks. Potential peaks that are off-scale are given. However, a caveat is that this could be from any of the channels (ie it could be from the ladder channel but is the same scan as the given repeat).

**settings used:**

- `peak_threshold`: THe peak_threshold parameter used.
- `lower_repeat_threshold`: The lower repeat limit based of the index repeat of each sample.
- `upper_repeat_threshold`: The upper repeat limit based of the index repeat of each sample.

- index_modal_signal_threshold: The index_modal_signal_threshold parameter used.
- index_signal_sum_threshold: The index_signal_sum_threshold parameter used.

**General sample metrics:**

- modal_peak_repeat: The repeat size of the modal peak.
- modal_peak_signal: The signal of the modal peak.
- index_peak_repeat: The repeat size of the index peak (the repeat value closest to the modal peak of the index sample).
- index_weighted_mean_repeat: The weighted mean repeat size (weighted on the signal of the peaks) of the index sample.
- n_peaks_total: The total number of peaks in the repeat table.
- n_peaks_analysis_subset: The number of peaks in the analysis subset.
- n_peaks_analysis_subset_expansions: The number of expansion peaks in the analysis subset.
- min_repeat: The minimum repeat size in the analysis subset.
- max_repeat: The maximum repeat size in the analysis subset.
- mean_repeat: The mean repeat size in the analysis subset.
- weighted_mean_repeat: The weighted mean repeat size (weight on peak signal) in the analysis subset.
- median_repeat: The median repeat size in the analysis subset.
- max_signal: The maximum peak signal in the analysis subset.
- sum_signal: The sum of the peak signal in the analysis subset.
- max_delta_neg: The maximum negative delta to the index peak.
- max_delta_pos: The maximum positive delta to the index peak.
- skewness: The skewness of the repeat size distribution.
- kurtosis: The kurtosis of the repeat size distribution.

**Repeat instability metrics:**

- modal_repeat_change: The difference between the modal repeat and the index repeat.
- average_repeat_change: The weighted mean of the sample (weighted by peak signal) subtracted by the weighted mean repeat of the index sample(s).
- instability_index_change: The instability index of the sample subtracted by the instability index of the index sample(s). This will be very similar to the average_repeat_change, with the key difference of instability_index_change being that it is an internally calculated metric for each sample, and therefore the random slight fluctuations of bp size (or systematic if across plates for example) will be removed. However, it requires the index peak to be correctly set for each sample, and if set incorrectly, can produce large arbitrary differences.
- instability_index: The instability index based on peak signal and distance to the index peak. (See Lee et al., 2010, doi:10.1186/17520509429).
- instability_index_abs: The absolute instability index. The absolute value is taken for the "Change from the main allele".
- expansion_index: The instability index for expansion peaks only.
- contraction_index: The instability index for contraction peaks only.

- `expansion_ratio`: The ratio of expansion peaks' signals to the main peak signal. Also known as "peak proportional sum" (See Genetic Modifiers of Huntington's Disease (GeM-HD) Consortium, 2019, doi:10.1016/j.cell.2019.06.036).
- `contraction_ratio`: The ratio of contraction peaks' signals to the main peak signal.
- `expansion_percentile_*`: The repeat size at specified percentiles of the cumulative distribution of expansion peaks.
- `expansion_percentile_for_repeat_*`: The percentile rank of specified repeat sizes in the distribution of expansion peaks.

## Value

A data.frame with calculated instability metrics for each sample.

## Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
test_fragments <- trace(fsa_list, grouped = TRUE, metadata_data.frame = metadata)

test_metrics_grouped <- calculate_instability_metrics(
  fragments_list = test_fragments,
  peak_threshold = 0.05,
  window_around_index_peak = c(-40, 40)
)
```

---

# `cell_line_fsa_list` *A list of fsa files*

---

## Description

A list of fsa files read into R using trace::read_fsa() that for example data

## Usage

```
cell_line_fsa_list
```

## Format

`cell_line_fsa_list`:

A list with 92 elements, each one being the contents of an fsa file:

## Source

doi:10.1038/s41467024474850

---

`example_data` *example_data*

---

## Description

example_data is genemapper output peak table

## Usage

```
example_data
```

## Format

`example_data`**:**
A genemapper output dataframe

## Source

doi:10.1038/s41467024474850

---

`example_data_repeat_table`
*example_data_repeat_table*

---

## Description

example_data_repeat_table is data with repeats called

## Usage

```
example_data_repeat_table
```

## Format

`example_data_repeat_table`**:**
A genemapper output dataframe

## Source

doi:10.1038/s41467024474850

## extract_alleles *Extract Modal Peaks*

### Description

Extracts modal peak information from each sample in a list of fragments.

### Usage

```
extract_alleles(fragments_list)
```

### Arguments

`fragments_list` A list of fragments objects containing fragment data.

### Value

A dataframe containing modal peak information for each sample

### Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
test_fragments <- trace(fsa_list, grouped = TRUE, metadata_data.frame = metadata)

extract_alleles(test_fragments)
```

## extract_fragments *Extract All Fragments*

### Description

Extracts peak data from each sample in a list of fragments.

### Usage

```
extract_fragments(fragments_list)
```

### Arguments

`fragments_list` A list of fragments objects containing fragment data.

### Value

A dataframe containing peak data for each sample

## Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
test_fragments <- trace(fsa_list, grouped = TRUE, metadata_data.frame = metadata)

extract_fragments(test_fragments)
```

---

`extract_ladder_summary` *Extract ladder summary*

---

## Description

Extract a table summarizing the ladder models

## Usage

```
extract_ladder_summary(fragments_list, sort = FALSE)
```

## Arguments

| | |
|---|---|
| `fragments_list` | a list of fragments trace objects |
| `sort` | A logical statement for if the samples should be ordered by average ladder R-squared. |

## Details

The ladder peaks are assigned using a custom algorithm that maximizes the fit of detected ladder peaks and given base-pair sizes. This function summarizes the R-squared values of these individual correlations.

## Value

a dataframe of ladder quality information

## Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
test_fragments <- trace(fsa_list, grouped = TRUE, metadata_data.frame = metadata)

  extract_ladder_summary(test_fragments, sort = TRUE)
```

`extract_repeat_correction_summary` *Extract repeat correction summary*

## Description

Extracts a table summarizing the model used to correct repeat length

## Usage

```
extract_repeat_correction_summary(fragments_list)
```

## Arguments

| | |
|---|---|
| `fragments_list` | A list of fragments class objects obtained from the `call_repeats()` function when the `correction = "repeat"` parameter is used. |

## Details

For each of the samples used for repeat correction, this table pulls out the modal repeat length called by the model (`allele_repeat`), how far that sample is on average from the linear model in repeat units by finding the average residuals (`avg_residual`), and the absolute value of the `avg_residual` (`abs_avg_residual`)

## Value

A data.frame

## Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
test_fragments <- trace(
   fsa_list,
   grouped = TRUE,
   metadata_data.frame = metadata,
   correction = "repeat",
   show_progress_bar = FALSE
)

# finally extract repeat correction summary
extract_repeat_correction_summary(test_fragments)
```

## extract_trace_table *Extract traces*

### Description

Extract the raw trace from a list of fragments objects

### Usage

```
extract_trace_table(fragments_list)
```

### Arguments

| | |
|---|---|
| fragments_list | a list of fragments objects |

### Value

A dataframe of the raw trace data. Each row representing a single scan.

### Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
test_fragments <- trace(fsa_list, grouped = TRUE, metadata_data.frame = metadata)

extracted_traces <- extract_trace_table(test_fragments)
```

## fix_ladders_interactive *Fix ladders interactively*

### Description

An app for fixing ladders

### Usage

```
fix_ladders_interactive(fragment_trace_list)
```

### Arguments

| | |
|---|---|
| fragment_trace_list | A list of fragments objects containing fragment data |

## Details

This function helps you fix ladders that are incorrectly assigned. Run `fix_ladders_interactive()` and provide output from `find_ladders`. In the app, for each sample, click on line for the incorrect ladder size and drag it to the correct peak.

Once you are satisfied with the ladders for all the broken samples, click the download button to generate a file that has the ladder correction data. Read this file back into R using readRDS, then use `fix_ladders_manual()` and supply the ladder correction data as `ladder_df_list`. This allows the manually corrected data to be saved and used within a script so that the correct does not need to be done every time. An example of what you would need to do:

ladder_df_list <- readRDS('path/to/exported/data.rds') test_ladders_fixed <- fix_ladders_manual(test_ladders_broken, ladder_df_list)

## Value

interactive shiny app for fixing ladders

## See Also

`fix_ladders_manual()`, `find_ladders()`

## Examples

```
# to create an example, lets brake one of the ladders
brake_ladder_list <-  list(
   "20230413_A08.fsa" = data.frame(
    size = c(35, 50, 75, 100, 139, 150, 160, 200, 250, 300, 340, 350, 400, 450, 490, 500),
    scan = c(1544, 1621, 1850, 1912, 2143, 2201, 2261, 2506, 2805, 3135, 3380, 3442, 3760,
              4050, 4284, 4332)
   )
 )

fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
test_fragments <- trace(fsa_list, ladder_df_list = brake_ladder_list)

if (interactive()) {
  fix_ladders_interactive(test_fragments)
}

# once you have corrected your ladders in the app,
# export the data for incorporation into the script.
# You can then re-import the data and fix ladders as described in the help details.
```

`fragments` *fragments object*

## Description

An R6 Class representing a fragments object.

## Details

This is the parent class of both fragments and fragments object. The idea is that shared fields and methods are both inherited from this object, but it is not itself directly used.

## Public fields

`unique_id` unique id of the sample usually the file name

`input_method` pathway used to import data (either 'fsa', 'size', or 'repeats')

`metrics_group_id` sample grouping for metrics calculations. Associated with `add_metadata()`.

`metrics_baseline_control` logical to indicate if sample is the baseline control. Associated with `add_metadata()`.

`batch_run_id` fragment analysis run. Associated with `add_metadata()`.

`batch_sample_id` An id for the sample used as size standard for repeat calculation. Associated with `add_metadata()`.

`batch_sample_modal_repeat` Validated repeat length for the modal repeat repeat in that sample. Associated with `add_metadata()`.

`fsa` The whole fsa file, output from seqinr::read.abif()

`raw_ladder` The raw data from the ladder channel

`raw_data` The raw data from the sample channel

`scan` The scan number

`off_scale_scans` vector indicating which scales were too big and off scale. Note can be in any channel

`ladder_df` A dataframe of the identified ladder from `find_ladders()`. Scan is the scan number of peak and size is the associated bp size.

`ladder_total_combinations_tested` A numeric value indicating how many total combinations were tested during ladder fit

`trace_bp_df` A dataframe of bp size for every scan from `find_ladders()`.

`peak_table_df` A dataframe containing the fragment peak level information.

`repeat_table_df` A dataframe containing the fragment peak level information with the repeat size added. May or may not be the same as peak_table_df depending on what options are chosen in `call_repeats`.

## Methods

### Public methods:

- fragments$new()
- fragments$print()
- fragments$plot_trace()
- fragments$plot_ladder()
- fragments$plot_data_channels()
- fragments$get_allele_peak()
- fragments$set_allele_peak()
- fragments$get_index_peak()
- fragments$set_index_peak()
- fragments$plot_fragments()
- fragments$clone()

**Method** `new()`**:** initialization function that is not used since the child classes are the main object of this package.

*Usage:*

```
fragments$new(unique_id, input_method, object)
```

*Arguments:*

`unique_id` unique_id

`input_method` pathway used to import data (either 'fsa', 'size', or 'repeats')

`object` object to be inserted into either 'fsa', 'peak_table_df', or 'repeat_table_df'

**Method** `print()`**:** A function to print informative information to the console

*Usage:*

```
fragments$print()
```

**Method** `plot_trace()`**:** plot the trace data

*Usage:*

```
fragments$plot_trace(
  show_peaks = TRUE,
  x_axis = NULL,
  ylim = NULL,
  xlim = NULL,
  signal_color_threshold = 0.05,
  plot_title = NULL
)
```

*Arguments:*

`show_peaks` A logical to say if the called peaks should be plotted on top of the trace. Only valid for fragments objects.

`x_axis` Either "size" or "repeats" to indicate what should be plotted on the x-axis.

`ylim` numeric vector length two specifying the y axis limits

`xlim` numeric vector length two specifying the x axis limits

`signal_color_threshold` A threshold value to colour the peaks relative to the tallest peak.

`plot_title` A character string for setting the plot title. Defaults to the unique id of the object

*Returns:* A base R plot

**Method** `plot_ladder()`**:** plot the ladder data

*Usage:*

```
fragments$plot_ladder(xlim = NULL, ylim = NULL, plot_title = NULL)
```

*Arguments:*

`xlim` numeric vector length two specifying the x axis limits

`ylim` numeric vector length two specifying the y axis limits

`plot_title` A character string for setting the plot title. Defaults to the unique id of the object

*Returns:* A base R plot

**Method** `plot_data_channels()`**:** plot the raw data channels in the fsa file. It identifies every channel that has "DATA" in its name.

*Usage:*

```
fragments$plot_data_channels()
```

*Returns:* A base R plot

**Method** `get_allele_peak()`**:** This returns a list with the allele information for this object.

*Usage:*

```
fragments$get_allele_peak()
```

**Method** `set_allele_peak()`**:** This sets a single allele size/repeat. It searches through the appropriate peak table and finds the closest peak to the value that's provided.

*Usage:*

```
fragments$set_allele_peak(allele, unit, value)
```

*Arguments:*

`allele` Either 1 or 2, indicating which allele information should be set. Allele 1 is the only one used for repeat instability metrics calculations.

`unit` Either "size" or "repeats" to indicate if the value you're providing is bp size or repeat length.

`value` Numeric vector (length one) of the size/repeat length to set.

**Method** `get_index_peak()`**:** This returns a list with the index peak information for this object.

*Usage:*

```
fragments$get_index_peak()
```

**Method** `set_index_peak()`**:** This sets the index repeat length. It searches through the repeat table and finds the closest peak to the value that's provided.

*Usage:*

```
fragments$set_index_peak(value)
```

*Arguments:*

`value` Numeric vector (length one) of the repeat length to set as index peak.

**Method** `plot_fragments()`**:** This plots the peak/repeat table as a histogram

*Usage:*

```
fragments$plot_fragments(ylim = NULL, xlim = NULL, plot_title = NULL)
```

*Arguments:*

`ylim` numeric vector length two specifying the y axis limits

`xlim` numeric vector length two specifying the x axis limits

`plot_title` A character string for setting the plot title. Defaults to the unique id of the object

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
fragments$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

---

`genemapper_table_to_fragments` *Convert Genemapper Peak Table to fragments class*

---

## Description

This function converts a genemapper peak table data frame into a list of fragments objects.

## Usage

```
genemapper_table_to_fragments(
  df,
  dye_channel,
  min_size_bp = 200,
  max_size_bp = 1000
)
```

## Arguments

| | |
|---|---|
| `df` | A data frame from genemapper containing the peak data. |
| `dye_channel` | A character string indicating the Genemapper channel to extract data from. This is used to filter 'Dye.Sample.Peak' for the channel containing the data. For example, 6-FAM is often "B" while ladder is "O". |
| `min_size_bp` | Numeric value indicating the minimum size of the peak table to import. |
| `max_size_bp` | Numeric value indicating the maximum size of the peak table to import. |

### Details

This function takes a peak table data frame (eg. Genemapper 5 output) and converts it into a list of fragment objects. It uses the "Sample.File.Name" as the unique id, so make sure that each file as a unique name. Column names should contain: "Dye.Sample.Peak", "Sample.File.Name", "Allele", "Size", "Height".

### Value

A list of fragments objects.

### See Also

`repeat_table_to_fragments()`, `size_table_to_fragments()`, `read_fsa()`

### Examples

```
gm_raw <- trace::example_data

test_fragments <- genemapper_table_to_fragments(
  gm_raw,
  dye_channel = "B",
  min_size_bp = 400
)
```

---

## load_config *Load Trace Config File*

---

### Description

Load configuration file that can be passed to individual trace modules.

### Usage

```
load_config(config_file = NULL)
```

### Arguments

| | |
|---|---|
| `config_file` | The file path to a YAML file containing a full list of parameters. If NULL, it will load a default YAML. |

### Details

Use this function to load in a config file for passing to individual trace modules. This is only required if creating a custom pipeline rather than the main function. This provides a central place to adjust parameters for the pipeline. Either edit the YAML directly, or modify the object (which is basically just a list).

Use the following command to make a copy of the YAML file: `file.copy(system.file("extdata/trace_config.yaml", package = "trace"), ".")`.

### Value

A trace_config object.

### Examples

```
config <- load_config()
```

---

## metadata *metadata*

---

### Description

This is a dataframe containing the metadata information for the example data

### Usage

```
metadata
```

### Format

`metadata`:
A genemapper output dataframe

### Source

doi:10.1038/s41467024474850

---

## plot_batch_correction_samples *Plot correction samples*

---

### Description

Plot the overlapping traces of the batch control samples

### Usage

```
plot_batch_correction_samples(fragments_list, selected_sample, xlim = NULL)
```

## Arguments

| | |
|---|---|
| fragments_list | A list of fragments objects containing fragment data. must have trace information. |
| selected_sample | A character vector of batch_sample_id for a subset of samples to plot. Or alternatively supply a number to select batch sample by position in alphabetical order. |
| xlim | the x limits of the plot. A numeric vector of length two. |

## Details

A plot of the raw signal by bp size or repeats for the batch correction samples.

When plotting the traces before repeat correction, we do not expect the samples to be closely overlapping due to run-to-run variation. After repeat correction, the traces should be basically overlapping.

These plots are made using base R plotting. Sometimes these fail to render in the viewing panes of IDEs (eg you get the error 'Error in `plot.new()`: figure margins too large)'. If this happens, try saving the plot as a pdf using traditional approaches (see grDevices::pdf).

## Value

plot of batch corrected samples

## See Also

`call_repeats()` for more info on batch correction.

## Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
fragments_list <- trace(fsa_list, metadata_data.frame = metadata, correction = "batch")


# traces of bp size shows traces at different sizes
plot_batch_correction_samples(
  fragments_list,
  selected_sample = "S-21-212", xlim = c(100, 120)
)
```

plot_data_channels *plot_data_channels*

## Description

Plot the raw data from the fsa file

## Usage

```
plot_data_channels(fragments_list, sample_subset = NULL, n_facet_col = 1)
```

## Arguments

| | |
|---|---|
| fragments_list | A list of fragments objects. |
| sample_subset | A character vector of unique ids for a subset of samples to plot |
| n_facet_col | A numeric value indicating the number of columns for faceting in the plot. |

## Details

A plot of the raw data channels in the fsa file.

These plots are made using base R plotting. Sometimes these fail to render in the viewing panes of IDEs (eg you get the error 'Error in `plot.new()`: figure margins too large)'. If this happens, try saving the plot as a pdf using traditional approaches (see grDevices::pdf). To get it to render in the IDE pane, trying matching `n_facet_col` to the number of samples you're attempting to plot, or using `sample_subset` to limit it to a single sample.

## Value

a plot of the raw data channels

## Examples

```
plot_data_channels(cell_line_fsa_list[1])
```

plot_fragments *Plot Peak Data*

## Description

Plots peak data from a list of fragments.

### Usage

```
plot_fragments(
  fragments_list,
  n_facet_col = 1,
  sample_subset = NULL,
  xlim = NULL,
  ylim = NULL
)
```

### Arguments

| | |
|---|---|
| fragments_list | A list of fragments objects containing fragment data. |
| n_facet_col | A numeric value indicating the number of columns for faceting in the plot. |
| sample_subset | A character vector of unique ids for a subset of samples to plot |
| xlim | the x limits of the plot. A numeric vector of length two. |
| ylim | the y limits of the plot. A numeric vector of length two. |

### Value

A plot object displaying the peak data.

### Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
fragments_list <- trace(fsa_list)

plot_fragments(fragments_list[1])
```

---

## plot_ladders *Plot ladder*

---

### Description

Plot the ladder signal

### Usage

```
plot_ladders(
  fragments_list,
  n_facet_col = 1,
  sample_subset = NULL,
  xlim = NULL,
  ylim = NULL
)
```

### Arguments

| | |
|---|---|
| fragments_list | A list of fragments objects containing fragment data. |
| n_facet_col | A numeric value indicating the number of columns for faceting in the plot. |
| sample_subset | A character vector of unique ids for a subset of samples to plot |
| xlim | the x limits of the plot. A numeric vector of length two. |
| ylim | the y limits of the plot. A numeric vector of length two. |

### Value

a plot of ladders

### Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
fragments_list <- trace(fsa_list)

# Manually inspect the ladders
plot_ladders(fragments_list[1])
```

---

## plot_repeat_correction_model
*Plot Repeat Correction Model*

---

### Description

Plots the results of the repeat correction model for a list of fragments.

### Usage

```
plot_repeat_correction_model(
  fragments_list,
  batch_run_id_subset = NULL,
  n_facet_col = 1
)
```

### Arguments

| | |
|---|---|
| fragments_list | A list of fragments class objects obtained from the call_repeats() function when the correction = "repeat" parameter is used. |
| batch_run_id_subset | A character vector for a subset of batch_sample_id to plot. Or alternatively supply a number to select batch sample by position in alphabetical order. |
| n_facet_col | A numeric value indicating the number of columns for faceting in the plot. |

## Details

This function makes plots for the model used to correct samples for each `batch_run_id`. The repeat correction algorithm assigns the user supplied repeat length to the modal peak of the sample, then pulls out a set of robust neighboring peaks to help get enough data to build an accurate linear model for the relationship between base-pair size and repeat length. So on this plot, each dot is an individual peak, with the colour indicating each sample, with the y-axis is the repeat length called from the user-supplied value in the metadata and the value assigned to each peak, with the x-axis showing the corresponding base-pair size.

## Value

A base R graphic object displaying the repeat correction model results.

## Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
fragments_list <- trace(fsa_list, metadata_data.frame = metadata, correction = "repeat")

# traces of bp size shows traces at different sizes
plot_repeat_correction_model(
  fragments_list,
  batch_run_id_subset = "20230414"
)
```

---

# plot_traces *Plot sample traces*

---

## Description

Plot the raw trace data

## Usage

```
plot_traces(
  fragments_list,
  show_peaks = TRUE,
  n_facet_col = 1,
  sample_subset = NULL,
  xlim = NULL,
  ylim = NULL,
  x_axis = NULL,
  signal_color_threshold = 0.05
)
```

### Arguments

| | |
|---|---|
| fragments_list | A list of fragments or fragments objects containing fragment data. |
| show_peaks | If peak data are available, TRUE will plot the peaks on top of the trace as dots. |
| n_facet_col | A numeric value indicating the number of columns for faceting in the plot. |
| sample_subset | A character vector of unique ids for a subset of samples to plot |
| xlim | the x limits of the plot. A numeric vector of length two. |
| ylim | the y limits of the plot. A numeric vector of length two. |
| x_axis | A character indicating what should be plotted on the x-axis, chose between size or repeats. If neither is selected, an assumption is made based on if repeats have been called. |
| signal_color_threshold | Threshold relative to tallest peak to color the dots (blue above, purple below). |

### Details

A plot of the raw signal by bp size. Red vertical line indicates the scan was flagged as off-scale. This is in any channel, so use your best judgment to determine if it's from the sample or ladder channel.

If peaks are called, green is the tallest peak, blue is peaks above the signal threshold (default 5%), purple is below the signal threshold. If force_whole_repeat_units is used within call_repeats(), the called repeat will be connected to the peak in the trace with a horizontal dashed line.

The index peak will be plotted as a vertical dashed line when it has been set using assign_index_peaks().

### Value

plot traces from fragments object

### Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())
# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
fragments_list <- trace(fsa_list)

plot_traces(fragments_list, xlim = c(105, 150))
```

---

## read_fsa *Read fsa file*

---

### Description

Read fsa file into memory and create fragments object

### Usage

```
read_fsa(files)
```

### Arguments

| | |
|---|---|
| files | a chr vector of fsa file names. For example, return all the fsa files in a directory with 'list.files("example_directory/", full.names = TRUE, pattern = ".fsa")'. |

### Details

read_fsa is just a wrapper around `seqinr::read.abif()` that reads the fsa file into memory and stores it inside a fragments object. That enables you to use the next function `find_ladders()`.

### Value

A list of fragments objects

### See Also

`find_ladders()`, `plot_data_channels()`

### Examples

```
fsa_file <- read_fsa(system.file("abif/2_FAC321_0000205983_B02_004.fsa", package = "seqinr"))
plot_data_channels(fsa_file)
```

---

## remove_fragments *Remove Samples from List*

---

### Description

A convenient function to remove specific samples from a list of fragments.

### Usage

```
remove_fragments(fragments_list, samples_to_remove)
```

### Arguments

| | |
|---|---|
| fragments_list | A list of fragments objects containing fragment data. |
| samples_to_remove | A character vector containing the unique IDs of the samples to be removed. |

### Value

A modified list of fragments with the specified samples removed.

## Examples

```
gm_raw <- trace::example_data

test_fragments <- genemapper_table_to_fragments(
  gm_raw,
  dye_channel = "B",
  min_size_bp = 300
)

all_fragment_names <- names(test_fragments)

# pull out unique ids of samples to remove
samples_to_remove <- all_fragment_names[c(1, 5, 10)]

samples_removed <- remove_fragments(test_fragments, samples_to_remove)
```

---

`repeat_table_to_fragments` *Convert Repeat Table to Repeats Fragments*

---

## Description

This function converts a repeat table data frame into a list of fragments. class.

## Usage

```
repeat_table_to_fragments(df, min_repeat = 0, max_repeat = 1000)
```

## Arguments

| | |
|---|---|
| `df` | A data frame containing the repeat data with the columns "unique_id", "repeats", "signal". |
| `min_repeat` | minimum repeat size |
| `max_repeat` | maximum repeat size |

## Details

This function takes a repeat table data frame and converts it into a list of fragments objects. The column names must be "unique_id" (unique sample id), "repeats" (number of repeats), and "signal" (the signal associated with the number of repeats. for example a frequency count of reads.). The dataframe should be long (eg bind rows if the data are separate).

## Value

A list of fragments objects.

## Examples

```
repeat_table <- trace::example_data_repeat_table
test_fragments <- repeat_table_to_fragments(repeat_table)
```

---

`size_table_to_fragments` *Convert Size Table to Fragments*

---

## Description

This function converts a size table data frame into a list of fragments. class.

## Usage

```
size_table_to_fragments(df, min_size_bp = 200, max_size_bp = 1000)
```

## Arguments

| | |
|---|---|
| `df` | A data frame containing the size data with the columns "unique_id", "size", "signal". |
| `min_size_bp` | Numeric value indicating the minimum size of the peak table to import. |
| `max_size_bp` | Numeric value indicating the maximum size of the peak table to import. |

## Details

This function takes a size table data frame and converts it into a list of fragments objects. The column names must be "unique_id" (unique sample id), "size" (base pair size), and "signal" (the signal associated with fragment). The dataframe should be long (eg bind rows if the data are separate).

## Value

A list of fragments objects.

## See Also

`repeat_table_to_fragments()`, `size_table_to_fragments()`, `read_fsa()`

## Examples

```
size_table <- trace::example_data
colnames(size_table)[c(2, 5, 6)] <- c("unique_id", "size", "signal")
fragments_list <- size_table_to_fragments(size_table)
```

trace *Main function for sample processing*

## Description

The main function for the trace package that handles processing of samples through the pipeline ready for the calculation of repeat instability metrics.

## Usage

```
trace(
  fragments_list,
  metadata_data.frame = NULL,
  index_override_dataframe = NULL,
  ladder_df_list = NULL,
  config_file = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| fragments_list | A list of fragments objects containing fragment data, generated with either read_fsa(), size_table_to_fragments(), genemapper_table_to_fragments(), or repeat_table_to_fragment |
| metadata_data.frame | metadata passed to add_metadata() for grouping samples for metrics calculations or batch correction. |
| index_override_dataframe | A data.frame to manually set index peaks. Column 1: unique sample IDs, Column 2: desired index peaks (the order of the columns is important since the information is pulled by column position rather than column name). Closest peak in each sample is selected so the number needs to just be approximate. Default: NULL. See assign_index_peaks(). |
| ladder_df_list | A list of dataframes, with the names being the unique id and the value being a dataframe. The dataframe has two columns, size (indicating the bp of the standard) and scan (the scan value of the ladder peak). It's critical that the element name in the list is the unique id of the sample. Either manually figure out what scan the ladder peaks should be and generate the list, or use fix_ladders_interactive() to interactively generate the ladder_df_list. |
| config_file | The file path to a YAML file containing a list of parameters. This provides a central place to adjust parameters for the pipeline. Use the following command to make a copy of the default YAML file: file.copy(system.file("extdata/trace_config.yaml", package = "trace"), "."). The YAML file does not have to contain a complete list of parameters. Parameters directly passed to this function via ... will overwrite values in the config_file. |

... additional parameters from any of the functions in the pipeline detailed below may be passed to this function.These parameters are grouped by functionality:

**Ladder-related parameters:**

- `ladder_channel`: string, which channel in the fsa file contains the ladder signal. Default: `"DATA.105"`.
- `signal_channel`: string, which channel in the fsa file contains the data signal. Default: `"DATA.1"`.
- `ladder_sizes`: numeric vector, bp sizes of ladder used in fragment analysis. Default: `c(50, 75, 100, 139, 150, 160, 200, 250, 300, 340, 350, 400, 450, 490, 500)`.
- `ladder_start_scan`: single numeric indicating the scan number to start looking for ladder peaks (only required when ladder signal does not have large spike at start). Usually this can be automatically found (when set to NA) through the detection of the large spike at the start of the signal. Default: `NA`.
- `minimum_ladder_signal`: single numeric for minimum signal of peak from the raw signal. If not set, the ladder will be fit to a set of the tallest peaks in the ladder channel. Default: `NA`.
- `ladder_assign_left_to_right`: single logical for if the ladder should be assigned from the smallest base pair size to largest (TRUE), or if the order should be reversed and assigned from largest to smallest (FALSE), which can be helpful since the end often has cleaner signal than the start. Default: `TRUE`.
- `ladder_selection_window`: single numeric for the ladder assigning algorithm. We iterate through the scans in blocks and test their linear fit (We can assume that the ladder is linear over a short distance). This value defines how large that block of peaks should be. Larger values should be better because the fit is tested in greater context, but larger numbers will make the fit increasingly slower. Default: `5`.
- `ladder_top_n_branching`: single numeric. The ladder assigning algorithm branches as it tests the various combinations. This value defines how many branches should be created. If the correct combination is not found, you could try increasing this value, but it will make it increasingly slower. Default: `5`.
- `ladder_branching_r_squared_threshold`: single numeric. The branches of the ladder assigning algorithm are pruned by R-squared values above this threshold to discard fits that are not promising. If the correct combination is not found, you could try decreasing this value, but it will make it increasingly slower. Default: `0.99`.
- `min_scan`: single numeric indicating the lower scan limit to filter out scans below. Default: `NA`.
- `max_scan`: single numeric indicating the upper scan limit to filter out scans above. Default: `NA`.
- `max_combinations`: single numeric indicating what is the maximum number of ladder combinations that should be tested. Default: `2500000`.
- `warning_rsq_threshold`: single numeric for the value for which this function will warn you when parts of the ladder have R-squared values below

the specified threshold. Default: `0.998`.

- `show_progress_bar`: single logical for showing progress bar. Default: `TRUE`.

**Peak-finding parameters:**

- `smoothing_window`: Single numeric value for signal smoothing window size passed to pracma::savgol(). Default: `21`.
- `minimum_peak_signal`: Single numeric value for the minimum peak signal for a valid peak. To have no minimum signal set as "-Inf". Default: `20`
- `min_bp_size`: Single numeric value for minimum bp size of peaks to consider. Default: `200`.
- `max_bp_size`: Single numeric value for maximum bp size of peaks to consider. Default: `1000`.
- `peak_scan_ramp`: Single numeric value to indicate how many scans (increasing in signal) should be either side of the peak maxima. Default: `5`.

**Allele-calling parameters:**

- `number_of_alleles`: Number of alleles to be returned for each fragment. Must either be 1 or 2. Default: `1`.
- `peak_region_size_gap_threshold`: Gap threshold for identifying peak regions. Default: `6`.
- `peak_region_signal_threshold_multiplier`: Multiplier for the peak signal threshold. Default: `1`.

**Repeat-calling parameters:**

- `assay_size_without_repeat`: An integer specifying the assay size without repeat for repeat calling. Default: `87`.
- `repeat_size`: An integer specifying the repeat size for repeat calling. Default: `3`.
- `correction`: A character vector of either "batch" to carry out a batch correction from common samples across runs (known repeat length not required), or "repeat" to use samples with validated modal repeat lengths to correct the repeat length. Default: `"none"`.
- `force_whole_repeat_units`: A logical value specifying if the peaks should be forced to be whole repeat units apart. Default: `FALSE`.
- `force_repeat_pattern`: A logical value specifying if the peaks should be re-called to fit the specific repeat unit pattern. Default: `FALSE`.
- `force_repeat_pattern_size_period`: A numeric value to set the peak periodicity bp size. Default: `2.79`.
- `force_repeat_pattern_size_window`: A numeric value for the size window when assigning the peak. Default: `0.5`.

**Index peak assignment parameters:**

- `grouped`: Logical value indicating whether samples should be grouped to share a common index peak. Default: `FALSE`.

## Details

This function processes samples through the full pipeline, applying the library of functions within this package. Parameters can be adjusted either by passing them directly to this function or by editing a configuration file and supplying it to `config_file`. Below is a breakdown of the pipeline stages and their key functionalities:

**Ladder-related parameters:** The ladder is used to calibrate the base pair (bp) sizes of fragments. The ladder peaks are identified in the ladder channel using `find_ladders`, and a generalized additive model (GAM) with cubic regression splines is used to fit the relationship between scan numbers and bp sizes. This allows for accurate interpolation of bp sizes for all scans. Manual inspection of ladder assignments is recommended to ensure correctness. If ladders are broken, first try and adjust parameters. In a last resort, ladders may be manually set by using the `ladder_df_list` parameter. To generate this list, use the helper shiny app `fix_ladders_interactive`.

**Peak-finding parameters:** Fragment peaks are identified in the continuous trace data using `find_fragments`. The signal is smoothed using a Savitzky-Golay filter, and peaks are detected based on their signal intensity and bp size. Parameters such as `min_bp_size` and `max_bp_size` allow filtering peaks outside the desired range, while `peak_scan_ramp` controls the number of scans around the peak maxima.

**Allele-calling parameters:** The main alleles within each fragment are identified by clustering peaks into "peak regions" using `find_alleles`. The tallest peak in each region is selected as the main allele. If `number_of_alleles` is set to 2, the two tallest peaks in their respective regions are selected, with the larger repeat size assigned as the main allele.

**Repeat-calling parameters:** Repeat lengths are calculated using `call_repeats`, based on the assay size without the repeat and the specified repeat size. Options include batch correction to account for run-to-run variability and repeat correction using samples with known modal repeat lengths. The `force_whole_repeat_units` option ensures repeat lengths are whole numbers, while `force_repeat_pattern` re-calls peaks to fit the expected repeat pattern.

**Index peak assignment parameters:** The index peak is the reference repeat length used for instability metrics calculations. It is typically the inherited repeat length or the modal repeat length at a starting time point. Samples can be grouped to share a common index peak using `assign_index_peaks`, or the index peak can be manually overridden using `index_override_dataframe`.

**Metadata:** Metadata is used to group samples for metrics calculations, batch correction, or repeat length validation. Use `add_metadata` to add metadata to the fragments list. Required (even if the column is left blank) metadata fields include:

- `batch_run_id`: Groups samples by fragment analysis run.
- `batch_sample_id`: Links samples across batches for batch or repeat correction.
- `batch_sample_modal_repeat`: Specifies the validated repeat length for samples used in repeat correction.
- `metrics_group_id`: Groups samples for shared index peak assignment.
- `metrics_baseline_control`: Identifies samples used as baseline controls for index peak assignment.

## Value

A list of fragments objects ready for calculation of instability metrics using `calculate_instability_metrics()`

## Examples

```
fsa_list <- lapply(cell_line_fsa_list, function(x) x$clone())

# import data with read_fsa() to generate an equivalent list to cell_line_fsa_list
fragments_list <- trace(fsa_list, grouped = TRUE, metadata_data.frame = metadata)

metrics <- calculate_instability_metrics(
  fragments_list,
  peak_threshold = 0.05,
  window_around_index_peak = c(-40, 40),
  percentile_range = c(0.5, 0.75, 0.9, 0.95),
  repeat_range = c(2, 5, 10, 20)
)
```

# Index


∗ **datasets**
    cell_line_fsa_list, 5
    example_data, 6
    example_data_repeat_table, 6
    metadata, 17

add_metadata, *30*
add_metadata(), *27*
assign_index_peaks, *30*
assign_index_peaks(), *27*

calculate_instability_metrics, 2
calculate_instability_metrics(), *30*
call_repeats, *30*
call_repeats(), *9*, *18*, *21*, *23*
cell_line_fsa_list, 5

example_data, 6
example_data_repeat_table, 6
extract_alleles, 7
extract_fragments, 7
extract_ladder_summary, 8
extract_repeat_correction_summary, 9
extract_trace_table, 10

find_alleles, *30*
find_fragments, *30*
find_ladders, *30*
find_ladders(), *11*, *24*
fix_ladders_interactive, 10, *30*
fix_ladders_interactive(), *27*
fix_ladders_manual(), *11*
fragments, 12

genemapper_table_to_fragments, 15
genemapper_table_to_fragments(), *27*

load_config, 16

metadata, 17

plot_batch_correction_samples, 17
plot_data_channels, 19
plot_data_channels(), *24*
plot_fragments, 19
plot_ladders, 20
plot_repeat_correction_model, 21
plot_traces, 22

read_fsa, 23
read_fsa(), *16*, *26*, *27*
remove_fragments, 24
repeat_table_to_fragments, 25
repeat_table_to_fragments(), *16*, *26*, *27*

seqinr::read.abif(), *24*
size_table_to_fragments, 26
size_table_to_fragments(), *16*, *26*, *27*

trace, 27