

# Package: MigrationDetectR (via r-universe)

September 6, 2024

**Type** Package

**Title** Segment-Based Migration Detection Algorithm

**Version** 0.1.1

**Description** Detection of migration events and segments of continuous residence based on irregular time series of location data as published in Chi et al. (2020)  [<doi:10.1371/journal.pone.0239408>](https://doi.org/10.1371/journal.pone.0239408).

**License** Apache License (>= 2)

**Copyright** German Aerospace Center (DLR)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.2

**Depends** R (>= 3.5.0)

**Imports** assertthat, dplyr, lifecycle, lubridate, tibble, tidyr

**NeedsCompilation** no

**Author** Johannes Mast [aut, cre] (Author of R code and wrappers,  [<https://orcid.org/0000-0001-6595-5834>](https://orcid.org/0000-0001-6595-5834)), Guanghua Chi [aut] (Developer of the Algorithm,  [<https://orcid.org/0000-0003-0430-7483>](https://orcid.org/0000-0003-0430-7483)), German Aerospace Center DLR [cph, fnd]

**Maintainer** Johannes Mast <johannes.mast@dlr.de>

**Repository** CRAN

**Date/Publication** 2023-11-11 11:53:25 UTC

## Contents

detect_segments . . . . .	2
example_trace . . . . .	4
find_best_split . . . . .	4
identify_migrations . . . . .	5
MigrationDetectR . . . . .	7

---

detect_segments	<i>detect_segments</i>
-----------------	------------------------

---

### Description

A three step algorithm to identify segments of continuous presence within trace data

### Usage

```
detect_segments(
  locs,
  times,
  param_min_days = 2,
  param_prop_days = 0.05,
  param_window_size_days = 20,
  param_ol_min_frac = 1,
  return_intermediate_results = FALSE
)
```

### Arguments

**locs** (character). A vector containing locations corresponding to **times**.

**times** (POSIXct). A vector containing timestamps corresponding to **locs**.

**param\_min\_days** (Optional) numeric. The minimum length in days of a segment. Smaller segments will be eliminated during stage 1. Default is 2.

**param\_prop\_days** (Optional) numeric. The minimum fraction of days in a segment during which a user must have been observed at a location. Segments with a smaller proportion will be eliminated. Default is 0.05.

**param\_window\_size\_days** (Optional) numeric. The minimum forward window size in days for step 1. Observations separated by a smaller timespan will be connected. Default is 20.

**param\_ol\_min\_frac** (Optional) numeric. In step 3, do not remove the overlapped segment if the place it belongs to is the mode during the period of overlap and if the place contains more than this fraction of all occurrences. Default is 1 meaning that no segments can persist at overlaps.

**return\_intermediate\_results** (Optional) logical. Should the results of Step 1 and 2 be returned as well? If TRUE the result will be a list of length 3. Default is FALSE.

## Details

The first step in detecting migration requires detecting periods of time when an individual is continuously present in a single location, allowing for some margin of travel from that location (Chi et al., 2020).

These segments are identified from the time series of `locs` and `times` in a three-step procedure:

Step 1: **Identify** contiguous segments, with no gap exceeding `param_window_size_days` days, where the individual is present for at least `param_prop_days` percent of days in the segment, and the total length of the segment is at least `param_min_days`,

- Step 2: **Merge** segments if there are no segments in other locations between them.,
- Step 3: **Prune** overlapping time from segments, when an individual is associated with segments in multiple locations at a single point in time. overlapping segments may be allowed to persist if they contain at least `param_ol_min_frac` of all observations during the overlap period.

## Value

A `tibble` containing the detected segments as `interval` with their location as `Character`.

## Author(s)

Johannes Mast <Johannes.Mast@d1r.de>, based on the algorithm by Guanghua Chi <guanghua@berkeley.edu>

## References

Chi, Guanghua, Fengyang Lin, Guangqing Chi, and Joshua Blumenstock. 2020. "A General Approach to Detecting Migration Events in Digital Trace Data." Edited by Song Gao. PLOS ONE 15 (10): e0239408. <https://doi.org/10.1371/journal.pone.0239408>.

## Examples

```
trace <- MigrationDetectR::example_trace

# Detect segments
segments <-
  detect_segments(
    locs = trace$location,
    times = trace$timestamp,
    param_min_days = 3,
    param_prop_days = 0.06,
    param_window_size_days = 7,
    param_ol_min_frac = 0.5)
nrow(segments) # check the number of detected segments
head(segments) # check the segments
```

---

 example\_trace

*Example trace*


---

### Description

A dataset containing trace of a fictional migrant in the form of location - timestamp pairs.

### Usage

```
example_trace
```

### Format

A tibble with 200 rows and 2 variables:

**timestamp** POSIXct timestamps of observations

**location** Locations of observations for one of four places (Rivendell, Tatooine, Shangri-La, Narnia)

---

 find\_best\_split

*find\_best\_split*


---

### Description

Find the optimal point in time to split a migration interval

### Usage

```
find_best_split(locs, times, movement, from, to)
```

### Arguments

**locs** character, A vector of the locations of occurrences.

**times** POSIXct, A vector of the occurrences corresponding timestamps.

**movement** A lubridate interval for which the best split is to be found.

**from** character, the name of the location the movement ends at. Must match an element in locs.

**to** character, the name of the location the movement ends at. Must match an element in locs.

## Details

The function checks which occurrences by `locs` and `times` fall within the interval given by `movement`. It then tries to find the optimal point in time to split the interval, using the following two criteria:

- The point which minimizes the number of misclassified **days**, i.e., the number of days when the migrant appears at to before the migration date and days when the migrant appears at from after the migration date.
- In cases where multiple days yield the same number of misclassifications, we select the last timestamp as the migration date

## Value

A tibble of 2 fields:

- **split\_time**, a lubridate date time which indicates the time of the optimal split found
- **split\_correctness**, a numeric date time which indicates the split's ratio of the wrongly assigned days to the correctly assigned days

## Author(s)

Johannes Mast <Johannes.Mast@dlr.de>, based on the algorithm by Guanghua Chi <guanghua@berkeley.edu>

## References

Chi, Guanghua, Fengyang Lin, Guangqing Chi, and Joshua Blumenstock. 2020. "A General Approach to Detecting Migration Events in Digital Trace Data." Edited by Song Gao. PLOS ONE 15 (10): e0239408. <https://doi.org/10.1371/journal.pone.0239408>.

---

identify\_migrations    *identify\_migrations*

---

## Description

identify\_migrations

## Usage

```
identify_migrations(  
  segs,  
  locs,  
  min_res_length = 90,  
  occurrence_locs = NULL,  
  occurrence_times = NULL,  
  verbose = TRUE  
)
```

**Arguments**

segs	A vector of <a href="#">interval</a> , a vector of intervals which indicate residence segments, such as detected by <a href="#">detect_segments</a> .
locs	A vector of Character, locations which correspond to segs.
min_res_length	(Optional) Numeric. A vector of days indicating the minimum number of days for segs to qualify as viable origin or destination of a migration. Default is 90.
occurrence_locs	(Optional) Character. See occurrence_times
occurrence_times	(Optional) If provided along with the matching occurrence_locs, will try to find the optimal point in time during which the migration occurred.
verbose	(Optional) logical. Output optional messages? Default is TRUE.

**Value**

A tibble containing the detected true relocations, if any. Contains the fields:

- **from** Character, the location from which the relocations started.
- **to** Character, the location from which the relocations started.
- **movement\_interval** [interval](#) of the transitional period, starting with the end of the prior residency and ending with the start of the new residency.
- **movement\_length** The duration of the transitional period in seconds.
- **movement\_midrange** The midpoint of the transitional period.

If occurrence\_locs and occurrence\_times were provided, [find\\_best\\_split](#) will be used to detect the optimum split time, and the following columns will be added:

- **split\_time**, a [POSIXct](#) which indicates the time of the optimal split.
- **split\_correctness**, a numeric which as a quality measure of split\_time reports the split's ratio of wrongly assigned days to the correctly assigned days.

**Author(s)**

Johannes Mast <Johannes.Mast@dlr.de>, based on the algorithm by Guanghua Chi <guanghua@berkeley.edu>

**References**

Chi, Guanghua, Fengyang Lin, Guangqing Chi, and Joshua Blumenstock. 2020. "A General Approach to Detecting Migration Events in Digital Trace Data." Edited by Song Gao. PLOS ONE 15 (10): e0239408. <https://doi.org/10.1371/journal.pone.0239408>.

**Examples**

```
trace <- MigrationDetectR::example_trace
# Detect segments
segments <-
  detect_segments(
```

```
      locs = trace$location,
      times = trace$timestamp,
      param_min_days = 3,
      param_prop_days = 0.06,
      param_window_size_days = 7)
nrow(segments) # check the number of detected segments

migrations <-
identify_migrations(
  segs = segments$segments,
  locs = segments$locs,
  min_res_length = 90,
  occurrence_locs = trace$location,
  occurrence_times = trace$timestamp
)
nrow(migrations) # check the number of identified migrations
head(migrations) # check the detected migrations
```

---

MigrationDetectR

*MigrationDetectR: Segment-Based Migration Detection Algorithm*

---

## Description

Implementation of Migration detection algorithm, published by Chi et al. (2020) in *A general approach to detecting migration events in digital trace data* ([doi:10.1371/journal.pone.0239408](https://doi.org/10.1371/journal.pone.0239408))

## Changes in the R implementation

- Compared to the reference, this implementation works at the precision of seconds rather than days. At the expense of speed, it is applicable at finer timescales.
- The segment detector has the additional option to allow for one of several overlapping segments to persist if it contains the majority of occurrences during the overlap period.

## Usage

First, transform your data into the format required by the package: Two aligned vectors, one with locations and one with timestamps. Use the [detect\\_segments](#) function to identify segments of continuous residence. Then, use the [identify\\_migrations](#) function on the segments to detect migrations. To optionally determine the best split time, pass the original locations and timestamps vectors.

## Author(s)

Johannes Mast (R Implementation)

Guanghua Chi (Developer of the Algorithm)

# Index

## \* datasets

example\_trace, 4

detect\_segments, 2, 6, 7

example\_trace, 4

find\_best\_split, 4, 6

identify\_migrations, 5, 7

interval, 3, 6

MigrationDetectR, 7

POSIXct, 6

tibble, 3