

Package ‘podcleaner’

July 23, 2025

Title Legacy Scottish Post Office Directories Cleaner

Version 0.1.2

Maintainer Olivier Bautheac <olivier.bautheac@strath.ac.uk>

Description Attempts to clean optical character recognition (OCR) errors in legacy Scottish Post Office Directories. Further attempts to match records from trades and general directories.

Imports dplyr (>= 1.0.7), fuzzyjoin (>= 0.1.6), magrittr (>= 2.0.1), progress (>= 1.2.2), purrr (>= 0.3.4), readr (>= 2.0.2), rlang (>= 0.4.12), stringi (>= 1.7.5), stringr (>= 1.4.0), tibble (>= 3.1.5), tidyr (>= 1.1.4)

License GPL (>= 3)

Encoding UTF-8

Language en-GB

RoxygenNote 7.1.2

Suggests testthat

NeedsCompilation no

Author Olivier Bautheac [aut, cre],
University of Strathclyde [cph, fnd]

Repository CRAN

Date/Publication 2022-01-11 20:02:42 UTC

Contents

clean_address_attached_words	4
clean_address_body	4
clean_address_ends	5
clean_address_mac	5
clean_address_names	6
clean_address_number	6
clean_address_others	7
clean_address_places	7

clean_address_possessives	8
clean_address_post_clean	8
clean_address_pre_clean	9
clean_address_saints	9
clean_address_suffixes	10
clean_address_worksites	10
clean_forename	11
clean_forename_punctuation	11
clean_forename_separate_words	12
clean_forename_spelling	12
clean_mac	13
clean_name_ends	13
clean_occupation	14
clean_parentheses	14
clean_specials	15
clean_string_ends	15
clean_surname	16
clean_surname_punctuation	16
clean_surname_spelling	17
clean_title	17
combine_get_address_house_type	18
combine_has_match_failed	18
combine_label_failed_matches	19
combine_label_if_match_failed	19
combine_make_match_string	20
combine_match_general_to_trades	20
combine_match_general_to_trades_plain	22
combine_match_general_to_trades_progress	23
combine_no_trade_address_to_random_string	24
combine_random_string_if_no_address	25
combine_random_string_if_pattern	25
general_clean_directory	26
general_clean_directory_plain	27
general_clean_directory_progress	27
general_clean_entries	28
general_fix_structure	29
general_move_house_to_address	29
general_repatriate_occupation_from_address	30
general_split_address_numbers_bodies	31
general_split_trade_addresses	32
general_split_trade_house_addresses	33
globals_address_names	33
globals_ampersand	34
globals_ampersand_vector	34
globals_and_double_quote	35
globals_and_single_quote	35
globals_forenames	36
globals_general_colnames	36

globals_macs	37
globals_numbers	37
globals_occupations	38
globals_places_raw	38
globals_places_regex	39
globals_regex_address_house_body_number	39
globals_regex_address_prefix	40
globals_regex_and_filter	40
globals_regex_and_match	41
globals_regex_get_address_house_type	41
globals_regex_house_split_trade	42
globals_regex_house_to_address	42
globals_regex_irrelevants	43
globals_regex_occupation_from_address	43
globals_regex_split_address_body	44
globals_regex_split_address_empty	44
globals_regex_split_address_numbers	45
globals_regex_split_trade_addresses	45
globals_regex_titles	46
globals_saints	46
globals_suffixes	47
globals_surnames	47
globals_titles	48
globals_trades_colnames	48
globals_union_colnames	49
globals_worksites	49
trades_clean_directory	50
trades_clean_directory_plain	51
trades_clean_directory_progress	51
trades_clean_entries	52
utils_clean_address	53
utils_clean_addresses	54
utils_clean_address_body	54
utils_clean_address_ends	55
utils_clean_address_number	56
utils_clean_ends	56
utils_clean_names	57
utils_clean_occupations	58
utils_clear_content	58
utils_clear_irrelevants	59
utils_execute	60
utils_format_directory_raw	61
utils_gsub_if_found	62
utils_IO_load	63
utils_IO_path	63
utils_IO_write	64
utils_is_address_missing	65
utils_label_address_if_missing	65

utils_label_missing_addresses	66
utils_load_directories_csv	67
utils_make_file	68
utils_make_path	68
utils_mutate_across	69
utils_mute	69
utils_paste_if_found	70
utils_regmatches_if_found	71
utils_regmatches_if_not_empty	72
utils_remove_address_prefix	73
utils_split_and_name	74
utils_squish_all_columns	75

Index	76
--------------	-----------

clean_address_attached_words	
	<i>Clean attached words in address entry(/ies)</i>

Description

Attempts to separate attached words in provided address entry(/ies).

Usage

clean_address_attached_words(addresses)

Arguments

addresses A character string vector of address(es).

Value

A character string vector of address(es) cleaned of attached words.

clean_address_body	<i>Clean address entry(/ies) body</i>
--------------------	---------------------------------------

Description

Attempts to clean body of address entry(/ies) provided.

Usage

clean_address_body(addresses)

Arguments

addresses A character string vector of address(es).

Value

A character string vector of address(es) with cleaned bodies.

clean_address_ends	<i>Clean ends in address entry(/ies)</i>
--------------------	--

Description

Attempts to clean ends in provided address entry(/ies).

Usage

```
clean_address_ends(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A character string vector of address(es) with clean ends.

clean_address_mac	<i>Standardise "Mac" prefix in address entry(/ies)</i>
-------------------	--

Description

Attempts to standardise "Mac" prefix in provided address entry(/ies).

Usage

```
clean_address_mac(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A character string vector of addresses with clean "Mac" prefix(es).

clean_address_names	<i>Clean place name(s) in address entry(/ies)</i>
---------------------	---

Description

Attempts to clean place names in provided address entry(/ies).

Usage

```
clean_address_names(addresses)
```

Arguments

addresses	A character string vector of address(es).
-----------	---

Value

A character string vector of address(es) with clean name(s).

clean_address_number	<i>Clean address entry numbers</i>
----------------------	------------------------------------

Description

Attempts to clean number of address entry(/ies) provided.

Usage

```
clean_address_number(addresses)
```

Arguments

addresses	A character string vector of address(es).
-----------	---

Value

A character string vector of address(es) with cleaned numbers.

clean_address_others	<i>Miscellaneous cleaning operations in address entry(/ies)</i>
----------------------	---

Description

Carries out miscellaneous cleaning operations in provided address entry(/ies).

Usage

```
clean_address_others(addresses)
```

Arguments

addresses	A character string vector of address(es).
-----------	---

Value

A character string vector of clean address(es).

clean_address_places	<i>Clean places in address entry(/ies)</i>
----------------------	--

Description

Attempts to clean places in provided address entry(/ies): street, road, place, quay, etc.

Usage

```
clean_address_places(addresses)
```

Arguments

addresses	A character string vector of address(es).
-----------	---

Value

A character string vector of address(es) with clean place name(s).

`clean_address_possessives`*Standardise possessives in address entry(/ies)*

Description

Attempts to standardise possessives in provided address entry(/ies).

Usage

```
clean_address_possessives(addresses)
```

Arguments

`addresses` A character string vector of address(es).

Value

A character string vector of address(es) with clean possessive(s).

`clean_address_post_clean`*Post-cleaning operation for address entry(/ies)*

Description

Performs post-cleaning operations on provided address entry(/ies).

Usage

```
clean_address_post_clean(addresses)
```

Arguments

`addresses` A character string vector of address(es).

Value

A character string vector with address(es) cleaner than the one provided in `addresses`.

`clean_address_pre_clean`*Pre-cleaning operation for address entry(/ies)*

Description

Performs pre-cleaning operations on provided address entry(/ies).

Usage

```
clean_address_pre_clean(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A character string vector with address(es) cleaner than the one provided in addresses.

`clean_address_saints` *Clean "Saint" prefix in address entry(/ies)*

Description

Attempts to clean "Saint" prefix in provided address entry(/ies).

Usage

```
clean_address_saints(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A character string vector of address(es) with clean "Saint" prefix(es).

`clean_address_suffixes`*Clean unwanted suffixes in address entry(/ies)*

Description

Attempts to clean unwanted suffixes in provided address entry(/ies).

Usage

```
clean_address_suffixes(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A character string vector of address(es) with unwanted suffix(es) removed.

`clean_address_worksites`*Clean worksites in address entry(/ies)*

Description

Attempts to clean worksites in provided address entry(/ies).

Usage

```
clean_address_worksites(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A character string vector of address(es) with clean worksite name(s).

clean_forename	<i>Clean entry(/ies) forename</i>
----------------	-----------------------------------

Description

Attempts to clean provided forename.

Usage

```
clean_forename(names)
```

Arguments

names	A character string vector of forename(s).
-------	---

Value

A character string vector of cleaned forename(s).

Details

Single letter forenames are standardised to the forename starting with that letter occurring the most frequently in the dataset. i.e A. -> Alexander, B. -> Bernard, C. -> Colin, D. -> David, etc.

clean_forename_punctuation	<i>Standardise punctuation in forename(s)</i>
----------------------------	---

Description

Attempts to standardise punctuation in provided forename entry(/ies).

Usage

```
clean_forename_punctuation(forenames)
```

Arguments

forenames	A character string vector of forename(s).
-----------	---

Value

A character string vector of forename(s) with clean punctuation.

`clean_forename_separate_words`*Separate double-barrelled forename(s)*

Description

Attempts to separate double-barrelled forename(s) in provided forename entry(/ies).

Usage

```
clean_forename_separate_words(forenames)
```

Arguments

forenames A character string vector of forename(s).

Value

A character string vector of forename(s) with clean double-barrelled forename(s).

`clean_forename_spelling`*Clean forename(s) spelling*

Description

Attempts to clean spelling in provided forename entry(/ies).

Usage

```
clean_forename_spelling(forenames)
```

Arguments

forenames A character string vector of forename(s).

Value

A character string vector of forename(s) with clean forename(s) spelling.

clean_mac	<i>Standardise "Mac" prefix in people's name</i>
-----------	--

Description

Attempts to standardise "Mac" prefix in provided name entry(/ies).

Usage

```
clean_mac(names)
```

Arguments

names	A character string vector of name(s).
-------	---------------------------------------

Value

A character string vector of name(s) with clean "Mac" prefix(es).

clean_name_ends	<i>Clean ends in entry(/ies) names</i>
-----------------	--

Description

Attempts to clean ends in provided name entry(/ies).

Usage

```
clean_name_ends(names)
```

Arguments

names	A character string vector of names
-------	------------------------------------

Value

A character string vector of names with clean ends.

clean_occupation	<i>Clean entry(/ies) occupation</i>
------------------	-------------------------------------

Description

Attempts to clean provided occupation.

Usage

```
clean_occupation(occupations)
```

Arguments

occupations A character string vector of occupation(s).

Value

A character string vector of cleaned occupation(s).

clean_parentheses	<i>Clean entry(/ies) of in brackets information</i>
-------------------	---

Description

Attempts to clean entry(/ies) of unwanted information displayed in brackets.

Usage

```
clean_parentheses(x)
```

Arguments

x A character string vector.

Value

A character string vector with within brackets content removed.

clean_specials	<i>Clean entry(/ies) special characters</i>
----------------	---

Description

Attempts to clean entry(/ies) of unwanted special character(s).

Usage

```
clean_specials(x)
```

Arguments

x	A character string vector.
---	----------------------------

Value

A character string vector with special character(s) removed.

clean_string_ends	<i>Clean string ends</i>
-------------------	--------------------------

Description

Attempts to clean ends of strings provided.

Usage

```
clean_string_ends(strings)
```

Arguments

strings	A character string vector.
---------	----------------------------

Value

A character string vector with clean entry(/ies) ends.

clean_surname	<i>Clean entry(/ies) surname</i>
---------------	----------------------------------

Description

Attempts to clean provided surname.

Usage

```
clean_surname(names)
```

Arguments

names	A character string vector of surname(s).
-------	--

Value

A character string vector of cleaned surname(s).

Details

Multiple spelling names are standardised to that of the capital letter header in the general directory. i.e. Abercrombie, Abercromby -> Abercromby; Bayne, Baynes -> Bayne; Beattie, Beatty -> Beatty; etc.

clean_surname_punctuation	<i>Standardise punctuation in surname(s)</i>
---------------------------	--

Description

Attempts to standardise punctuation in provided surname entry(/ies).

Usage

```
clean_surname_punctuation(surnames)
```

Arguments

surnames	A character string vector of surname(s).
----------	--

Value

A character string vector of surname(s) with clean punctuation.

clean_surname_spelling	<i>Clean surname(s) spelling</i>
------------------------	----------------------------------

Description

Attempts to clean spelling in provided surname entry(/ies).

Usage

```
clean_surname_spelling(surnames)
```

Arguments

surnames	A character string vector of surnames.
----------	--

Value

A character string vector of surnames with clean spelling.

clean_title	<i>Clean entry(/ies) name title</i>
-------------	-------------------------------------

Description

Attempts to clean titles attached to names provided: Captain, Major, etc.

Usage

```
clean_title(names)
```

Arguments

names	A character string vector of name(s).
-------	---------------------------------------

Value

A character string vector of name(s) with cleaned title(s).

combine_get_address_house_type
Get house address column type

Description

Identifies the type of the house address column provided: number or body.

Usage

```
combine_get_address_house_type(column)
```

Arguments

column A Character string: ends in "house.number" or "house.body".

Value

A Character string: "number" or "body".

combine_has_match_failed
Check for failed matches

Description

Provided with two equal length vectors, returns TRUE for indexes where both entries are "NA" and FALSE otherwise.

Usage

```
combine_has_match_failed(number, body)
```

Arguments

number A vector of address number(s). Integer or character string.
 body A character string vector of address body(/ies).

Value

A boolean vector: TRUE for indexes where both number and body are "NA", FALSE otherwise.

```
combine_label_failed_matches
```

Label failed matches

Description

Labels failed matches as such in the provided Scottish post office directory data.frame.

Usage

```
combine_label_failed_matches(directory)
```

Arguments

directory	A Scottish post office directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include address.house.number, address.house.body.
-----------	---

Value

A data.frame of the same class as the one provided in directory. Columns include address.house.number, address.house.body. For entries for which both address.house.number and address.house.body are NA, address.house.number and address.house.body are labelled as "" and "Failed to match with general directory" respectively.

```
combine_label_if_match_failed
```

Label failed matches

Description

Labels failed matches as such.

Usage

```
combine_label_if_match_failed(type = c("number", "body"), ...)
```

Arguments

type	A Character string, one of: "number" or "body". Type of column to label.
...	Further arguments to be passed down to combine_has_match_failed

Value

A character string vector: address(es) "number" or "body" as specified in type if match succeeded, "" (type = "number") or "Failed to match with general directory" (type = "body") otherwise.

```
combine_make_match_string
```

Mutate operation(s) in directory data.frame trade address column

Description

Creates a 'match.string' column in the provided Scottish post office directory data.frame composed of entry(/ies) full name and trade address pasted together. Missing trade address entry(/ies) are replaced with a random generated string.

Usage

```
combine_make_match_string(directory)
```

Arguments

directory	A Scottish post office directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, address.trade.number, address.trade.body.
-----------	--

Value

A data.frame of the same class as the one provided in directory; columns include at least forename, surname, address.trade.number, address.trade.body, match.string.

Details

The purpose of the 'match.string' column is to facilitates the matching of the general to trades directory down the line. It allows to calculate a string distance metric between each pair of entries and match those falling below a specified threshold.

See Also

[combine_match_general_to_trades](#) for the matching of the general to trades directory.

```
combine_match_general_to_trades
```

Match general to trades directory records

Description

Attempts to complement Scottish post office trades directory data.frame with house address information from the Scottish post office general directory data.frame provided by matching records from the two datasets using the distance metric specified.

Usage

```
combine_match_general_to_trades(
  trades_directory,
  general_directory,
  progress = TRUE,
  verbose = FALSE,
  distance = TRUE,
  matches = TRUE,
  ...
)
```

Arguments

trades_directory	A Scottish post office trades directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include surname, forename, address.trade.number, address.trade.body.
general_directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include surname, forename, address.trade.number, address.trade.body, address.house.number, address.house.body.
progress	Whether progress should be shown (TRUE) or not (FALSE).
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).
distance	Whether (TRUE) or not (FALSE) a column 'distance' showing the string distance between records used for their matching and calculated using the method specified below should be added to the output dataset.
matches	Whether (TRUE) or not (FALSE) a column 'match' showing general directory matches' name and address(es) should be added to the output dataset.
...	Further arguments to be passed down to stringdist_left_join .

Value

A [tibble](#); columns include at least surname, forename, address.trade.number, address.trade.body, address.house.number, address.house.body.

Examples

```
trades_directory <- tibble::tibble(
  page = rep("71", 3L),
  rank = c("135", "326", "586"),
  surname = c("Abbott", "Abercromby", "Blair"),
  forename = c("William", "Alexander", "John Hugh"),
  occupation = c("Wine and spirit merchant", "Baker", "Victualler"),
  type = rep("OWN ACCOUNT", 3L),
  address.trade.number = c("18, 20", "12", "280"),
  address.trade.body = c("London Road", "Dixon Place", "High Street")
)
```

```

general_directory <- tibble::tibble(
  page = rep("71", 2L),
  surname = c("Abbott", "Abercromby"), forename = c("William", "Alexander"),
  occupation = c("Wine and spirit merchant", "Baker"),
  address.trade.number = c("18, 20", ""),
  address.house.number = c("136", "29"),
  address.trade.body = c("London Road", "Dixon Place"),
  address.house.body = c("Queen Square", "Anderston Quay")
)
combine_match_general_to_trades(
  trades_directory, general_directory, progress = TRUE, verbose = FALSE,
  distance = TRUE, method = "osa", max_dist = 5
)

```

```
combine_match_general_to_trades_plain
```

Match general to trades directory records

Description

Attempts to complement Scottish post office trades directory data.frame with house address information from the Scottish post office general directory data.frame provided by matching records from the two datasets using the distance metric specified.

Usage

```

combine_match_general_to_trades_plain(
  trades_directory,
  general_directory,
  verbose,
  matches,
  ...
)

```

Arguments

trades_directory

A Scottish post office trades directory in the form of a data.frame or other object that inherits from the data.frame class such as a [tibble](#). Columns must at least include surname, forename, address.trade.number, address.trade.body.

general_directory

A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a [tibble](#). Columns must at least include surname, forename, address.trade.number, address.trade.body, address.house.number, address.house.body.

verbose

Whether the function should be executed silently (FALSE) or not (TRUE).

matches	Whether (TRUE) or not (FALSE) a column 'match' showing general directory matches' name and address(es) should be added to the output dataset.
...	Further arguments to be passed down to <code>stringdist_left_join</code> .

Value

A data.frame of the same class as that of the one provided in `trades_directory` and/or `general_directory`. Should `trades_directory` and `general_directory` be provided as objects of different classes, the class of the return data.frame will be that of the parent class. i.e. if `trades_directory` and `general_directory` are provided as a pure data.frame and a `tibble` respectively, a pure data.frame is returned. Columns include at least surname, forename, address.trade.number, address.trade.body, address.house.number, address.house.body.

See Also

`combine_match_general_to_trades`.

combine_match_general_to_trades_progress

Match general to trades directory records

Description

Attempts to complement Scottish post office trades directory data.frame with house address information from the Scottish post office general directory data.frame provided by matching records from the two datasets using the distance metric specified. Shows a progress bar indicating function progression.

Usage

```
combine_match_general_to_trades_progress(
  trades_directory,
  general_directory,
  verbose,
  matches,
  ...
)
```

Arguments

`trades_directory`

A Scottish post office trades directory in the form of a data.frame or other object that inherits from the data.frame class such as a `tibble`. Columns must at least include surname, forename, address.trade.number, address.trade.body.

general_directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include surname, forename, address.trade.number, address.trade.body, address.house.number, address.house.body.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).
matches	Whether (TRUE) or not (FALSE) a column 'match' showing general directory matches' name and address(es) should be added to the output dataset.
...	Further arguments to be passed down to stringdist_left_join .

Value

A data.frame of the same class as that of the one provided in trades_directory and/or general_directory. Should trades_directory and general_directory be provided as objects of different classes, the class of the return data.frame will be that of the parent class. i.e. if trades_directory and general_directory are provided as a pure data.frame and a [tibble](#) respectively, a pure data.frame is returned. Columns include at least surname, forename, address.trade.number, address.trade.body, address.house.number, address.house.body.

See Also

[combine_match_general_to_trades](#).

combine_no_trade_address_to_random_string

Mutate operation(s) in directory data.frame address.trade column.

Description

Replaces missing trade address(es) in the provided Scottish post office directory data.frame with random string(s). Random string(s) only show(s) in body of trade address entry(/ies).

Usage

```
combine_no_trade_address_to_random_string(directory)
```

Arguments

directory	A Scottish post office directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include address.trade.
-----------	--

Value

A data.frame of the same class as the one provided in directory; columns include at least address.trade.

Details

Prevents unwarranted matches when matching general to trades directory. Unrelated records with similar name and trade address entry labelled as missing would be otherwise matched.

`combine_random_string_if_no_address`

Conditionally return a random string

Description

Returns a 22 character long random string if address provided is labelled as missing ("No trade/house address found").

Usage

```
combine_random_string_if_no_address(address)
```

Arguments

address	A character string.
---------	---------------------

Value

A length 1 character string vector: 22 character long random string if address labelled as missing ("No trade/house address found"), address otherwise.

`combine_random_string_if_pattern`

Conditionally return a random string

Description

Search for specified pattern in provided string; if found returns a 22 character long random string otherwise return original string.

Usage

```
combine_random_string_if_pattern(string, regex)
```

Arguments

string	A character string.
regex	Character string regex specifying the pattern to look for in string.

Value

A length 1 character string vector: 22 character long random string if regex found in string, string otherwise.

```
general_clean_directory
```

Mutate operation(s) in Scottish post office general directory data.frame column(s)

Description

Attempts to clean the provided Scottish post office general directory data.frame.

Usage

```
general_clean_directory(directory, progress = TRUE, verbose = FALSE)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation and addresses.
progress	Whether progress should be shown (TRUE) or not (FALSE).
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A [tibble](#); columns include at least forename, surname, occupation, address.trade.number, address.trade.body, address.house.number and address.house.body. "house" suffix in occupation column is move to addresses, occupation information is repatriated from addresses to occupation column; addresses is split into trade and house address columns; additional records are created for each extra trade address identified. Entries are further cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

Examples

```
pages <- rep("71", 2L)
surnames <- c("ABOT", "ABRCROMBIE")
forenames <- c("Wm.", "Alex")
occupations <- c("Wine and spirit mercht - See Advertisement in Appendix.", "")
addresses = c(
  "1S20 Londn rd; ho. 13<J Queun sq",
  "Bkr; I2 Dixon Street, & 29 Auderstn Qu.; res 2G5 Argul st."
)
directory <- tibble::tibble(
  page = pages, surname = surnames, forename = forenames,
  occupation = occupations, addresses = addresses
)
general_clean_directory(directory, progress = TRUE, verbose = FALSE)
```

`general_clean_directory_plain`*Mutate operation(s) in Scottish post office general directory data.frame column(s)*

Description

Attempts to clean the provided Scottish post office general directory data.frame.

Usage

```
general_clean_directory_plain(directory, verbose)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation and addresses.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include at least forename, surname, occupation, address.trade.number, address.trade.body, address.house.number and address.house.body. "house" suffix in occupation column is move to addresses, occupation information is repatriated from addresses to occupation column; addresses is split into trade and house address column; additional records are created for each extra trade address identified. Entries are further cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

`general_clean_directory_progress`*Mutate operation(s) in Scottish post office general directory data.frame column(s)*

Description

Attempts to clean the provided Scottish post office general directory data.frame. Shows a progress bar indication the progression of the function.

Usage

```
general_clean_directory_progress(directory, verbose)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation and addresses.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include at least forename, surname, occupation, address.trade.number, address.trade.body, address.house.number and address.house.body. "house" suffix in occupation column is move to addresses, occupation information is repatriated from addresses to occupation column; addresses is split into trade and house address column; additional records are created for each extra trade address identified. Entries are further cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

`general_clean_entries` *Mutate operation(s) in Scottish post office general directory data.frame column(s)*

Description

Attempts to clean entries of the provided Scottish post office general directory data.frame provided.

Usage

```
general_clean_entries(directory, verbose)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation, address.trade.number, address.trade.body and/or address.house.number, address.house.body.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include the same as those in the data.frame provided in directory. Entries are cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

`general_fix_structure` *Mutate operation(s) in Scottish post office general directory data.frame column(s)*

Description

Attempts to fix the structure of the raw Scottish post office general directory data.frame provided. For each entry, `general_fix_structure` attempts to fix parsing errors by moving pieces of information provided to the right columns; further attempts to separate trade from house address, separate multiple trade addresses as well as separate number from address body.

Usage

```
general_fix_structure(directory, verbose)
```

Arguments

<code>directory</code>	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include occupation, addresses.
<code>verbose</code>	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in `directory`; columns include at least `occupation`, `address.trade.number`, `address.trade.body`, `address.house.number` and `address.house.body`. "house" suffix in `occupation` column is move to `addresses`, occupation information is repatriated from `addresses` to `occupation` column; `addresses` is split into trade and house address columns; additional records are created for each extra trade address identified.

`general_move_house_to_address` *Mutate operation(s) in Scottish post office general directory data.frame column(s)*

Description

For some raw Scottish post office general directory entries, the word "house" referring to address type lives in the `occupation` column as a result of parsing errors. `general_move_house_to_address` attempts to move this information to the appropriate destination: the `addresses` column.

Usage

```
general_move_house_to_address(directory, regex)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include occupation and addresses.
regex	Regex to use for the task provided as a character string.

Value

A data.frame of the same class as the one provided in directory; columns include at least occupation and addresses. Entries in the occupation column are cleaned of "house" suffix; entries showing "house" suffix in occupation column see "house, " pasted as prefix to corresponding addresses column content.

general_repatriate_occupation_from_address

Mutate operation(s) in Scottish post office general directory data.frame column(s)

Description

For some raw Scottish post office general directory entries occupation information lives in the addresses column as a result of parsing errors. general_repatriate_occupation_from_address attempts to move this information to the appropriate destination: the occupation column.

Usage

```
general_repatriate_occupation_from_address(directory, regex)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include occupation and addresses.
regex	Regex to use for the task provided as a character string.

Value

A data.frame of the same class as the one provided in directory; columns include at least occupation and addresses.

```
general_split_address_numbers_bodies
```

Mutate operation(s) in Scottish post office general directory data.frame column(s)

Description

Attempts to separate number from body of address entries in the Scottish post office general directory data.frame provided

Usage

```
general_split_address_numbers_bodies(
  directory,
  regex_split_address_numbers,
  regex_split_address_body,
  regex_split_address_empty,
  ignore_case_filter,
  ignore_case_match
)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include address.trade and address.house.
regex_split_address_numbers	Regex to use to match address number(s).
regex_split_address_body	Regex to use to match address body(/ies).
regex_split_address_empty	Regex to use to match empty address entries.
ignore_case_filter	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) for using one of the regexes above as filtering regex in utils_regmatches_if_found .
ignore_case_match	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) for using one of the regexes above as matching regex in utils_regmatches_if_found .

Value

A data.frame of the same class as the one provided in directory; columns include at least address.trade.number, address.trade.body, address.house.number and address.house.body.

```
general_split_trade_addresses
```

Mutate operation(s) in Scottish post office general directory data.frame column(s)

Description

Attempts to separate multiple trade addresses in the Scottish post office general directory data.frame provided for entries for which more than one are provided.

Usage

```
general_split_trade_addresses(
  directory,
  regex_split,
  ignore_case_split,
  regex_filter,
  ignore_case_filter,
  regex_match,
  ignore_case_match
)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include addresses.trade.
regex_split	Regex to use to split addresses.
ignore_case_split	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) for regex_split above.
regex_filter	Regex to use to search for address entries with post-split undesired leftovers.
ignore_case_filter	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) for regex_filter above.
regex_match	Regex to use to clear address entries from post-split undesired leftovers.
ignore_case_match	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) for regex_match above.

Value

A data.frame of the same class as the one provided in directory; columns include at least address.trade. Multiple trade addresses are separated for entries for which more than one are provided. Each trade address identified lives on an individual row with information in the other columns duplicated.

```
general_split_trade_house_addresses
```

Mutate operation(s) in Scottish post office general directory data.frame column(s)

Description

Attempts to separate house address from trade address(es) in the Scottish post office general directory data.frame provided for entries for which a house address is provided along trade address(es).

Usage

```
general_split_trade_house_addresses(directory, regex, verbose)
```

Arguments

directory	A Scottish post office general directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include addresses.
regex	Regex to use for the task provided as a character string.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include at least addresses.trade and address.house. Trade addresses are separated from house address for entries for which a house address is provided along trade address(es).

```
globals_address_names
```

Place names in address entries

Description

A dataset containing regular expression meant to match commonly (OCR) misread place names in directory address entries. For each place name a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

```
globals_address_names
```

Format

A data frame with 3 variables:

pattern regex for place name matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_ampersand	<i>Ampersand in directory entries</i>
-------------------	---------------------------------------

Description

A dataset containing regular expression meant to match common (OCR) errors in reading the ampersand character: "&" in directory entries. For each error pattern a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

globals_ampersand

Format

A data frame with 3 variables:

pattern regex for ampersand reading error matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_ampersand_vector	<i>Ampersand in directory entries</i>
--------------------------	---------------------------------------

Description

A character vector of regular expressions to match common (OCR) errors in reading the ampersand character: "&" in directory entries.

Usage

globals_ampersand_vector

Format

A character string vector.

`globals_and_double_quote`*Ampersand in directory entries*

Description

A character vector of regular expressions to match common (OCR) errors in reading the ampersand character: "&" in directory entries.

Usage`globals_and_double_quote`**Format**

A character string vector.

Details

Some regexes contain the double quote character: '"'.

`globals_and_single_quote`*Ampersand in directory entries*

Description

A character vector of regular expressions to match common (OCR) errors in reading the ampersand character: "&" in directory entries.

Usage`globals_and_single_quote`**Format**

A character string vector.

Details

Some regexes contain the single quote character: "'".

globals_forenames	<i>Forenames in directory records</i>
-------------------	---------------------------------------

Description

A dataset containing regular expression meant to match commonly (OCR) misread forenames in directory name entries. For each forename a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

globals_forenames

Format

A data frame with 3 variables:

pattern regex for forename matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_general_colnames	<i>General directory column names</i>
--------------------------	---------------------------------------

Description

A character vector of column names for general directories.

Usage

globals_general_colnames

Format

A character string vector.

globals_macs	<i>"Mac" pre-fixes in name entries</i>
--------------	--

Description

A dataset containing regular expression meant to match commonly (OCR) misread "Mac" pre-fixes in directory name entries. For each "Mac" pre-fix a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

globals_macs

Format

A data frame with 3 variables:

pattern regex for "Mac" pre-fix matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_numbers	<i>Numbers in address entries</i>
-----------------	-----------------------------------

Description

A dataset containing regular expression meant to match commonly (OCR) misread numbers in directory address entries. For each number a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

globals_numbers

Format

A data frame with 3 variables:

pattern regex for number matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_occupations	<i>Occupations in directory records</i>
---------------------	---

Description

A dataset containing regular expression meant to match commonly (OCR) misread occupations in directory entries. For each occupation a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

globals_occupations

Format

A data frame with 3 variables:

pattern regex for occupation matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_places_raw	<i>Place types in address entries</i>
--------------------	---------------------------------------

Description

A character vector of common place types found in directory address entries

Usage

globals_places_raw

Format

A character string vector.

globals_places_regex	<i>Place types in address entries</i>
----------------------	---------------------------------------

Description

A dataset containing regular expression meant to match commonly (OCR) misread place types in directory address entries. For each place type a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

```
globals_places_regex
```

Format

A data frame with 3 variables:

pattern regex for place type matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_regex_address_house_body_number	<i>Regular expression for mutate operations in directory datasets</i>
---	---

Description

Regular expression used in the making of the match.string that eventually enables the matching of general and trades directory records.

Usage

```
globals_regex_address_house_body_number
```

Format

A character string vector.

See Also

[combine_label_failed_matches](#)

`globals_regex_address_prefix`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to remove undesired pre-fixes in general directory address records.

Usage`globals_regex_address_prefix`**Format**

A character string vector.

See Also

[utils_remove_address_prefix](#)

`globals_regex_and_filter`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to the word "and" in a filtering operation part of a mutate operation in the general directory provided.

Usage`globals_regex_and_filter`**Format**

A character string vector.

See Also

[general_split_trade_addresses](#)

`globals_regex_and_match`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to match the word "and" in a filtering operation part of a mutate operation in the general directory provided.

Usage`globals_regex_and_match`**Format**

A character string vector.

See Also

[general_split_trade_addresses](#)

`globals_regex_get_address_house_type`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used in the making of the match.string that eventually enables the matching of general and trades directory records.

Usage`globals_regex_get_address_house_type`**Format**

A character string vector.

See Also

[combine_get_address_house_type](#)

`globals_regex_house_split_trade`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to separate trades from house addresses in general directory.

Usage`globals_regex_house_split_trade`**Format**

A character string vector.

See Also

[general_split_trade_house_addresses](#)

`globals_regex_house_to_address`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to move the word "house" from the occupation column to the addresses column in general directory.

Usage`globals_regex_house_to_address`**Format**

A character string vector.

See Also

[general_move_house_to_address](#)

`globals_regex_irrelevants`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to match irrelevant information in the directory dataset provided.

Usage`globals_regex_irrelevants`**Format**

A character string vector.

See Also

[utils_clear_irrelevants](#)

`globals_regex_occupation_from_address`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to repatriate occupation from address column in general directory.

Usage`globals_regex_occupation_from_address`**Format**

A character string vector.

See Also

[general_repatriate_occupation_from_address](#)

`globals_regex_split_address_body`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to separate numbers from body in provided general directory address entries.

Usage`globals_regex_split_address_body`**Format**

A character string vector.

See Also

[general_split_address_numbers_bodies](#)

`globals_regex_split_address_empty`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to separate numbers from body in provided general directory address entries.

Usage`globals_regex_split_address_empty`**Format**

A character string vector.

See Also

[general_split_address_numbers_bodies](#)

`globals_regex_split_address_numbers`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to separate numbers from body in provided general directory address entries.

Usage`globals_regex_split_address_numbers`**Format**

A character string vector.

See Also

[general_split_address_numbers_bodies](#)

`globals_regex_split_trade_addresses`*Regular expression for mutate operations in directory datasets*

Description

Regular expression used to split multiple trade addresses when more than one are provided.

Usage`globals_regex_split_trade_addresses`**Format**

A character string vector.

See Also

[utils_remove_address_prefix](#)

globals_regex_titles	<i>Regular expression for mutate operations in directory datasets</i>
----------------------	---

Description

Regular expression used to match title in provided directory name entries.

Usage

```
globals_regex_titles
```

Format

A character string vector.

globals_saints	<i>Saints in address names</i>
----------------	--------------------------------

Description

A dataset containing regular expression meant to match commonly (OCR) misread name of Saints in directory address names. For each Saint a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

```
globals_saints
```

Format

A data frame with 3 variables:

pattern regex for Saint name matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_suffixes	<i>Address suffixes</i>
------------------	-------------------------

Description

A dataset containing regular expression meant to match commonly (OCR) misread suffixes in directory address entries. For each suffix a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

globals_suffixes

Format

A data frame with 3 variables:

pattern regex for suffix matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_surnames	<i>Surnames in directory records</i>
------------------	--------------------------------------

Description

A dataset containing regular expression meant to match commonly (OCR) misread surnames in directory name entries. For each surname a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

globals_surnames

Format

A data frame with 3 variables:

pattern regex for surname matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_titles	<i>Titles in directory name records</i>
----------------	---

Description

A dataset containing regular expression meant to match commonly (OCR) misread titles in directory name records. For each title a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage

```
globals_titles
```

Format

A data frame with 3 variables:

pattern regex for title matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

globals_trades_colnames	<i>Trades directory column names</i>
-------------------------	--------------------------------------

Description

A character vector of column names for trades directories.

Usage

```
globals_trades_colnames
```

Format

A character string vector.

`globals_union_colnames`*Combined directories column names*

Description

A character vector of column names for the dataset where general directory records are matched to trades directory records.

Usage`globals_union_colnames`**Format**

A character string vector.

`globals_worksites`*Worksites in address entries*

Description

A dataset containing regular expression meant to match commonly (OCR) misread worksite names in directory address entries. For each worksite a replacement pattern is provided for used in substitution operations as well as a boolean operator indicating whether the corresponding regex is case sensitive or not.

Usage`globals_worksites`**Format**

A data frame with 3 variables:

pattern regex for worksite name matching

replacement replacement pattern for substitution operations

ignore_case boolean operator indicating whether the corresponding regex is case sensitive or not.

trades_clean_directory

Mutate operation(s) in Scottish post office trades directory data.frame column(s)

Description

Attempts to clean the provided Scottish post office trades directory data.frame.

Usage

```
trades_clean_directory(directory, progress = TRUE, verbose = FALSE)
```

Arguments

directory	A Scottish post office trades directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation, address.trade.number and address.trade.body.
progress	Whether progress should be shown (TRUE) or not (FALSE).
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include at least forename, surname, occupation, address.trade.number and address.trade.body. Entries are cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

Examples

```
pages <- rep("71", 2L)
ranks <- c("135", "326")
surnames <- c("ABOT", "ABRCROMBIE")
forenames <- c("Wm.", "Alex")
occupations <- c(
  "Wine and spirit mercht - See Advertisement in Appendix.", "Bkr"
)
types <- rep("OWN ACCOUNT", 2L)
numbers <- c("1S20", "I2")
bodies <- c("Londn rd.", "Dixen pl")
directory <- tibble::tibble(
  page = pages, rank = ranks, surname = surnames, forename = forenames,
  occupation = occupations, type = types,
  address.trade.number = numbers, address.trade.body = bodies
)
trades_clean_directory(directory, progress = TRUE, verbose = FALSE)
```

`trades_clean_directory_plain`*Mutate operation(s) in Scottish post office trades directory data.frame
column(s)*

Description

Attempts to clean the provided Scottish post office trades directory data.frame.

Usage

```
trades_clean_directory_plain(directory, verbose)
```

Arguments

directory	A Scottish post office trades directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation, address.trade.number and address.trade.body.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include at least forename, surname, occupation, address.trade.number and address.trade.body. Entries are cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

`trades_clean_directory_progress`*Mutate operation(s) in Scottish post office trades directory data.frame
column(s)*

Description

Attempts to clean the provided Scottish post office trades directory data.frame. Shows a progress bar indicating function progression.

Usage

```
trades_clean_directory_progress(directory, verbose)
```

Arguments

directory	A Scottish post office trades directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation, address.trade.number and address.trade.body.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include at least forename, surname, occupation, address.trade.number and address.trade.body. Entries are cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

trades_clean_entries	<i>Mutate operation(s) in Scottish post office trades directory data.frame column(s)</i>
----------------------	--

Description

Attempts to clean entries of the provided Scottish post office trades directory data.frame.

Usage

```
trades_clean_entries(directory, verbose)
```

Arguments

directory	A Scottish post office trades directory in the form of a data.frame or other object that inherits from the data.frame class such as a tibble . Columns must at least include forename, surname, occupation, address.trade.number, address.trade.body.
verbose	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A data.frame of the same class as the one provided in directory; columns include the same as those in the data.frame provided in directory. Entries are cleaned of optical character recognition (OCR) errors and subject to a number of standardisation operations.

utils_clean_address	<i>Clean directory address entries</i>
---------------------	--

Description

Clean address entries in the provided directory dataframe.

Usage

```
utils_clean_address(directory, type = c("body", "number", "ends"))
```

Arguments

directory	A directory dataframe.
type	A character string: "body", "number" or "ends". Specifies the type of address cleaning to be performed. For "body", "number" and "ends" clean_address_body , clean_address_number and clean_address_ends are called respectively

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71"),
  surname = c("ABOT", "ABRCROMBIE"), forename = c("Wm.", "Alex"),
  occupation = c("Wine and spirit merchant", "Baker"),
  address.number = c("-", "1820", "", "12"),
  address.body = c(
    "London st. ; house, Mary hill.*",
    "&Dixon st.; residence, Craigrownie, Cove.$"
  ),
  stringsAsFactors = FALSE
)
utils_clean_address(directory, "body")
utils_clean_address(directory, "number")

## End(Not run)
```

utils_clean_addresses *Clean directory addresses*

Description

Clean all address records in provided directory dataframe.

Usage

```
utils_clean_addresses(directory)
```

Arguments

directory A directory dataframe. Columns must include address.house.number, address.house.number and/or address.trade.number, address.trade.number.

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71", "71"),
  surname = c("ABOT", "ABRCROMBIE", "BLAI"), forename = c("Wm.", "Alex", "Jn Huh"),
  occupation = c("Wine and spirit merchant", "Baker", "Victualer"),
  address.trade.number = c("-", 1820, "", "280"),
  address.trade.body = c("London st. ; house, Mary hill.*", "", "High stret"),
  stringsAsFactors = FALSE
)
utils_clean_addresses(directory)

## End(Not run)
```

utils_clean_address_body
Clean address(es) body

Description

Clean body record of provided address(es).

Usage

```
utils_clean_address_body(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A vector of character strings.

Examples

```
## Not run:
  utils_clean_address_body(
    c("London st.", "Mary hill.*", "&Dixon st.", "Craigrownie, Cove.$")
  )

## End(Not run)
```

utils_clean_address_ends

Clean address entry ends

Description

Clean beginning and end of the provided address entries.

Usage

```
utils_clean_address_ends(addresses)
```

Arguments

addresses A character string vector of address(es).

Value

A vector of character strings.

Examples

```
## Not run:
  utils_clean_address_ends(
    c(
      " -; 18, 20 London st.; house, Mary hill.*",
      ",,12 &Dixon st.; residence, Craigrownie, Cove.$"
    )
  )

## End(Not run)
```

utils_clean_address_number	<i>Clean address(es) number</i>
----------------------------	---------------------------------

Description

Clean number record of provided address(es).

Usage

utils_clean_address_number(addresses)

Arguments

addresses A character string vector of address(es).

Value

A vector of character strings.

Examples

```
## Not run:
  utils_clean_address_number(c(" -; 1820", ", ,12"))

## End(Not run)
```

utils_clean_ends	<i>Clean entry ends</i>
------------------	-------------------------

Description

Clean entry ends for the specified columns in the directory dataframe provided

Usage

utils_clean_ends(directory, ...)

Arguments

directory A directory dataframe.
... Columns to clean provided as expressions.

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71", "71"),
  surname = c("ABOT", "ABRCROMBIE", "BLAI"), forename = c("Wm.", "Alex", "Jn Huh"),
  occupation = c("Wine and spirit merchant", "Baker", "Victualer"),
  address.trade.number = c("-", 1820, "", "280"),
  address.trade.body = c("London st. ; house, Mary hill.*", "", "High stret"),
  stringsAsFactors = FALSE
)
utils_clean_ends(directory, address.trade.number, address.trade.body)

## End(Not run)
```

utils_clean_names	<i>Clean entries name records</i>
-------------------	-----------------------------------

Description

Clean name columns (forename & surname) of provided directory dataframe.

Usage

```
utils_clean_names(directory)
```

Arguments

directory A directory dataframe.

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71"),
  surname = c("ABOT", "ABRCROMBIE"), forename = c("Wm.", "Alex"),
  occupation = c("Wine and spirit merchant", "Baker"),
  address.number = c("-", 1820, "", 12),
  address.body = c(
    "London st. ; house, Mary hill.*",
    "&Dixon st.; residence, Craigrownie, Cove.$"
  ),
  stringsAsFactors = FALSE
)
utils_clean_names(directory)

## End(Not run)
```

`utils_clean_occupations`*Clean entries occupation record*

Description

Clean "occupation" column of provided directory dataframe.

Usage

```
utils_clean_occupations(directory)
```

Arguments

`directory` A directory dataframe.

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71"),
  surname = c("ABOT", "ABRCROMBIE"), forename = c("Wm.", "Alex"),
  occupation = c("wine and spirit mercht", "bkr"),
  address.number = c("-; 1820", ",,12"),
  address.body = c(
    "London st. ; house, Mary hill.*",
    "&;Dixon st.; residence, Craigrownie, Cove.$"
  ),
  stringsAsFactors = FALSE
)
utils_clean_occupations(directory)

## End(Not run)
```

`utils_clear_content` *Clear string of matched content*

Description

Clears the provided string of the content specified as a regex.

Usage

```
utils_clear_content(string_search, regex_content, ignore_case)
```

Arguments

string_search Character string to search for match(es).
 regex_content PCRE type regex provided as a character string of match(es) to search for.
 ignore_case Boolean specifying whether case should be ignored (TRUE) or not (FALSE).

Value

A character string.

Examples

```
## Not run:
  utils_clear_content("glasgow-entrepreneurs", "^.+-", TRUE)

## End(Not run)
```

 utils_clear_irrelevants

Mutate operation(s) in directory dataframe column(s)

Description

Attempts to get rid of irrelevant information in all columns of the provided directory dataframe provided

Usage

```
utils_clear_irrelevants(directory, ...)
```

Arguments

directory A directory dataframe.
 ... Further arguments to be passed down to [utils_clear_content](#).

Value

A dataframe.

Examples

```
## Not run:
  directory <- data.frame(
    page = c("71", "71"),
    surname = c("ABOT", "ABRCROMBIE"), forename = c("Wm.", "Alex"),
    occupation = c("Wine and spirit merchant – See Advertisement in Appendix.", "Baker"),
    address.trade.number = c("18, 20", "12"),
    address.house.number = c("136", "265"),
```

```

        address.trade.body = c("London Street.", "Dixon Street."),
        address.house.body = c("Queen Street.", "Argyle Street"),
        stringsAsFactors = FALSE
    )
    utils_clear_irrelevants(directory, globals_regex_irrelevants, ignore_case = TRUE)

## End(Not run)

```

utils_execute	<i>Execute function</i>
---------------	-------------------------

Description

Executes the function provided. Execution can be silenced via the verbose parameter.

Usage

```
utils_execute(verbose, fun, ...)
```

Arguments

verbose	Boolean specifying whether to silence the function execution (FALSE) or not (TRUE).
fun	Function to execute provided as an expression.
...	Argument(s) to be passed to the function above for execution.

Value

Whatever the provided function returns.

Examples

```

## Not run:
  utils_execute(TRUE, message, "I'm showing in console")

## End(Not run)

```

utils_format_directory_raw

Format raw directory for further processing

Description

Takes a raw directory dataframe (just loaded), adds a column with the corresponding directory name, replaces all NA entries with an empty string, clear all entries of unwanted blank characters, format page number as integer, returns the output with the directory name column in first position.

Usage

```
utils_format_directory_raw(df, name)
```

Arguments

df	A raw directory dataframe as output by utils_load_directories_csv .
name	Directory name provided as a character string.

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71"),
  surname = c("ABOT", "ABRCROMBIE"), forename = c("Wm.", "Alex"),
  occupation = c("wine and spirit mercht", "bkr"),
  addresses = c(
    "depot -; 1820 London st. ; house, Mary hill.*",
    "workshop,,12 &Dixon st.; residence, Craigrownie, Cove.$ "
  ),
  stringsAsFactors = FALSE
)
utils_format_directory_raw(directory, "1861-1862")

## End(Not run)
```

utils_gsub_if_found *Conditionally amend character string vector.*

Description

Searches for specified pattern in provided character string vector. If found, substitutes all occurrences of an alternative pattern in an alternative character string and returns the output. If not return the default character string provided.

Usage

```
utils_gsub_if_found(
  regex_filter,
  string_filter,
  regex_search,
  string_replace,
  string_search,
  default,
  ignore_case_filter,
  ignore_case_search
)
```

Arguments

regex_filter	Pattern to look for provided as a character string regex.
string_filter	Character string vector to search into for the pattern provided in regex_filter above.
regex_search	Alternative pattern provided as a character string regex to look in the alternative character string provided in string_search below.
string_replace	Substitution character string for matches of regex_search above in string_search below.
string_search	Alternative character string to search into for the pattern provided in regex_search above.
default	Character string returned if pattern provided in regex_filter not found.
ignore_case_filter	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) in search for regex_filter in string_filter.
ignore_case_search	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) in search for regex_search in string_search.

Value

A character string vector.

Examples

```
## Not run:
  utils_gsub_if_found(
    "glasgow", c("glasgow-entrepreneurs", "aberdeen-entrepreneurs"),
    "(?<=-).+$", "merchant", "edinburgh-entrepreneurs", "pattern not found",
    TRUE, TRUE
  )

## End(Not run)
```

utils_IO_load	<i>Load object into memory</i>
---------------	--------------------------------

Description

Load saved object as .rds file back into memory.

Usage

```
utils_IO_load(...)
```

Arguments

... Destination parameters to be passed to [utils_IO_path](#).

Value

R object from destination .rds file.

Examples

```
## Not run:
  utils_IO_load("home/projects", "glasgow-entrepreneurs")

## End(Not run)
```

utils_IO_path	<i>Make path for input/output operations</i>
---------------	--

Description

Paste provided path to directory and file name provided using '/' as separator.

Usage

```
utils_IO_path(directory_path, ..., extension)
```

Arguments

directory_path Path to directory where file_name lives as character string.
 ... File name components provided as character strings to be passed down to [utils_make_file](#).
 extension File extension as character string

Value

Path to destination file as a character string.

Examples

```
## Not run:
  utils_IO_path("home/projects", "glasgow-entrepreneurs", "csv")

## End(Not run)
```

utils_IO_write	<i>Write object to long term memory</i>
----------------	---

Description

Save the object provided to specified path as .rds file.

Usage

```
utils_IO_write(data, ...)
```

Arguments

data R object to save.
 ... Destination parameters to be passed to [utils_IO_path](#).

Value

No return value, called for side effects.

Examples

```
## Not run:
  utils_IO_write(mtcars, "home/projects", "mtcars")

## End(Not run)
```

`utils_is_address_missing`*Check is address entry not missing*

Description

Checks whether or not for each address in the evaluation environment, body and number are filled/not empty.

Usage

```
utils_is_address_missing(type)
```

Arguments

type A character string: "house" or "trade", specifying the type of address to check.

Value

A Boolean vector: TRUE if both number and body are empty.

Details

The function is for primarily use in the [utils_label_address_if_missing](#) function called by [utils_label_missing_addresses](#) where it provides a filtering vector used for labelling missing addresses. `utils_is_address_missing` creates an expression and further evaluates it two levels up in the environment tree, in other words in the directory dataframe eventually passed down to [utils_label_missing_addresses](#).

`utils_label_address_if_missing`*Label addresses if missing*

Description

If address is empty label body accordingly: "no house/trade address found".

Usage

```
utils_label_address_if_missing()
```

Value

A character string vector of address bodies, unchanged if provided, labelled as missing otherwise.

Details

The function is for primarily use in the `utils_label_missing_addresses` function where it provides a vector of address bodies `utils_label_address_if_missing` creates an expression and further evaluates it one level up in the environment tree, in other words in the directory dataframe eventually passed down to `utils_label_missing_addresses`.

```
utils_label_missing_addresses
```

Label empty addresses as missing

Description

Labels empty address bodies as "not house/trade address found" in the provided directory dataframe.

Usage

```
utils_label_missing_addresses(directory)
```

Arguments

<code>directory</code>	A directory dataframe. Columns must include <code>address.house.number</code> , <code>address.house.number</code> and/or <code>address.trade.number</code> , <code>address.trade.number</code> .
------------------------	--

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71"),
  surname = c("ABOT", "ABRCROMBIE"), forename = c("Wm.", "Alex"),
  occupation = c("Wine and spirit merchant", "Baker"),
  address.number = c("-; 1820", ""),
  address.body = c(
    "London st. ; house, Mary hill.*",
    ""
  ),
  stringsAsFactors = FALSE
)
utils_label_missing_addresses(directory)

## End(Not run)
```

`utils_load_directories_csv`*Load directory "csv" file(s) into memory*

Description

Loads specified directory "csv" file(s) into memory. Stacks individual directories into a single dataframe and further passes the output down to [utils_format_directory_raw](#) for initial formatting.

Usage

```
utils_load_directories_csv(  
  type = c("general", "trades"),  
  directories,  
  path,  
  verbose  
)
```

Arguments

<code>type</code>	A character string: "general" or "trades". Refers to the type of directory to shall be loaded.
<code>directories</code>	A character string vector providing the name(s) of the directory(/ies) to load.
<code>path</code>	A character string specifying the path to the folder where the directory(/ies) live as ".csv" file(s).
<code>verbose</code>	Whether the function should be executed silently (FALSE) or not (TRUE).

Value

A dataframe.

Examples

```
## Not run:  
utils_load_directories_csv(  
  "general", "1861-1862",  
  "home/projects/glasgow-entrepreneurs/data/general-directories", FALSE  
)  
  
## End(Not run)
```

utils_make_file	<i>Make file name</i>
-----------------	-----------------------

Description

Pastes the arguments provided together using '-'. Appends result string with the extension provided.

Usage

```
utils_make_file(..., extension)
```

Arguments

...	File name component(s) as character string(s).
extension	File extension as character string

Value

File name as a character string.

Examples

```
utils_make_file("glasgow", "entrepreneurs", extension = "csv")
```

utils_make_path	<i>Make destination path</i>
-----------------	------------------------------

Description

Pastes the arguments provided together using '/' as separator.

Usage

```
utils_make_path(...)
```

Arguments

...	Path components as character string(s).
-----	---

Value

Path to last element provided as a character string.

Examples

```
utils_make_path("home", "projects", "glasgow-entrepreneurs.csv")
```

utils_mutate_across	<i>Mutate operation(s) in dataframe column(s)</i>
---------------------	---

Description

Applies provided function across specified column(s) in provided dataframe.

Usage

```
utils_mutate_across(df, columns, fun, ...)
```

Arguments

df	A dataframe.
columns	Vector of expression(s) or character string(s) specifying the columns to apply the function below to in the provided dataframe.
fun	Function to execute provided as an expression.
...	Argument(s) to be passed to the function above for execution.

Value

A dataframe.

Examples

```
## Not run:
df <- data.frame(
  location = "glasgow", occupation = "wine merchant",
  stringsAsFactors = FALSE
)
utils_mutate_across(df, c("location", "occupation"), paste0, "!")

## End(Not run)
```

utils_mute	<i>Mute a function call execution</i>
------------	---------------------------------------

Description

Executes the function provided while silencing the potential messages related to its execution

Usage

```
utils_mute(fun, ...)
```

Arguments

fun	Function to execute as an expression.
...	Argument(s) to be passed to the function above for execution.

Value

Whatever the provided function in fun returns.

Examples

```
## Not run:
  utils_mute(message, "I'm not showing in console")

## End(Not run)
```

utils_paste_if_found *Conditionally amend character string vector.*

Description

Searches for specified pattern in provided character string. Return pasted provided character string(s) if found or provided default character string if not.

Usage

```
utils_paste_if_found(regex_filter, string_filter, default, ignore_case, ...)
```

Arguments

regex_filter	Pattern to look for provided as a character string regex.
string_filter	Character string vector to search into for the pattern provided in regex_filter above.
default	Character string returned if pattern provided in regex_filter not found.
ignore_case	Boolean specifying whether case should be ignored (TRUE) or not (FALSE).
...	Character string(s) to be paste together using a space as separator and returned if pattern provided in regex_filter found.

Value

A character string vector.

Examples

```
## Not run:
utils_paste_if_found(
  "^glasgow", c("glasgow-entrepreneurs", "aberdeen-entrepreneurs"),
  "pattern not found", TRUE, "pattern", "found"
)

## End(Not run)
```

utils_regmatches_if_found

Conditionally amend character string vector.

Description

Searches for specified pattern in provided character string vector. If found, searches for alternative pattern in an alternative character string and returns any match or an empty string if none. If original pattern not found, returns the default character string provided.

Usage

```
utils_regmatches_if_found(
  string_filter,
  regex_filter,
  string_search,
  regex_search,
  default,
  ignore_case_filter,
  ignore_case_match,
  not
)
```

Arguments

string_filter	Character string vector to search into for the pattern provided in regex_filter above.
regex_filter	Pattern to look for provided as a character string regex.
string_search	Alternative character string to search into for the pattern provided in regex_search above.
regex_search	Alternative pattern provided as a character string regex to look for in the alternative character string provided in string_search below.
default	Character string returned if pattern provided in regex_filter not found.
ignore_case_filter	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) in search for regex_filter in string_filter.

ignore_case_match	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) in search for regex_search in string_search.
not	Boolean specifying whether to negate the regex_filter search pattern (TRUE) or not (FALSE).

Value

A character string vector.

Examples

```
## Not run:
utils_regmatches_if_found(
  c("glasgow-entrepreneurs", "aberdeen-entrepreneurs"), "^glasgow",
  "edinburgh-entrepreneurs", "^.(?=-)", "merchant", TRUE, TRUE, FALSE
)

## End(Not run)
```

utils_regmatches_if_not_empty

Conditionally amend character string vector.

Description

Searches for non-empty string in provided character string vector. If found searches for alternative pattern in an alternative character string and returns any match or an empty string if none.

Usage

```
utils_regmatches_if_not_empty(
  string_filter,
  string_search,
  regex_search,
  ignore_case_search
)
```

Arguments

string_filter	A Character string vector.
string_search	Alternative character string to search into for the pattern provided in regex_search below
regex_search	Alternative pattern provided as a character string regex to look for in the alternative character string provided in string_search above.
ignore_case_search	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) in search for regex_search in string_search.

Value

A list of character string vectors.

Examples

```
## Not run:
utils_regmatches_if_not_empty(
  c("glasgow-entrepreneurs", "", "aberdeen-entrepreneurs"),
  "edinburgh-entrepreneurs" , "^edinburgh", TRUE
)

## End(Not run)
```

```
utils_remove_address_prefix
      Clear undesired address prefixes
```

Description

Clear address entries in the provided directory dataframe of undesired prefixes such as "depot", "office", "store", "works" or "workshops".

Usage

```
utils_remove_address_prefix(directory, regex, ignore_case)
```

Arguments

directory	A directory dataframe with an addresses column.
regex	Regex character string to be use for matching.
ignore_case	Boolean specifying whether case should be ignored (TRUE) or not (FALSE) in search for regex in addresses column entries of directory.

Value

A dataframe.

Examples

```
## Not run:
directory <- data.frame(
  page = c("71", "71"),
  surname = c("ABOT", "ABRCROMBIE"), forename = c("Wm.", "Alex"),
  occupation = c("Wine and spirit merchant", "Baker"),
  addresses = c(
    "depot -; 1820 London st. ; house, Mary hill.*",
    "workshop,,12 &Dixon st.; residence, Craigrownie, Cove.$ "
  ),
)
```

```
    stringsAsFactors = FALSE
  )
  regex <- globals_regex_address_prefix
  utils_remove_address_prefix(directory, regex, TRUE)

## End(Not run)
```

utils_split_and_name *Split string into tibble*

Description

Split provided string according to specified pattern. Organise output as a [tibble](#).

Usage

```
utils_split_and_name(string, pattern, num_col, colnames)
```

Arguments

string	Character string to be split.
pattern	Pattern to split on as character string (can be a regex).
num_col	Number of parts to split the string into as integer.
colnames	Column names for the output tibble.

Value

A [tibble](#)

Examples

```
## Not run:
  utils_split_and_name("glasgow-entrepreneurs", "-", 2, c("location", "occupation"))

## End(Not run)
```

`utils_squish_all_columns`*Clear extra white spaces in dataframe*

Description

Removes blanks (white spaces and tabs) at the beginning and end of all entries of the provided dataframe. Converts all series of white space and/or tab(s) in the body of all dataframe entries into a single white space.

Removes blanks (white spaces and tabs) at the beginning and end of all entries of the provided dataframe. Converts all series of white space and/or tab(s) in the body of all dataframe entries into a single white space.

Usage

```
utils_squish_all_columns(df)
```

```
utils_squish_all_columns(df)
```

Arguments

`df` A dataframe.

Value

A dataframe.

A dataframe.

Examples

```
## Not run:
df <- data.frame(
  location = " glasgow ", occupation = "wine   merchant",
  stringsAsFactors = FALSE
)
df <- utils_squish_all_columns(df)

## End(Not run)
## Not run:
df <- data.frame(
  location = " glasgow ", occupation = "wine   merchant",
  stringsAsFactors = FALSE
)
df <- utils_squish_all_columns(df)

## End(Not run)
```

Index

* datasets

- globals_address_names, [33](#)
- globals_ampersand, [34](#)
- globals_ampersand_vector, [34](#)
- globals_and_double_quote, [35](#)
- globals_and_single_quote, [35](#)
- globals_forenames, [36](#)
- globals_general_colnames, [36](#)
- globals_macs, [37](#)
- globals_numbers, [37](#)
- globals_occupations, [38](#)
- globals_places_raw, [38](#)
- globals_places_regex, [39](#)
- globals_regex_address_house_body_number, [39](#)
- globals_regex_address_prefix, [40](#)
- globals_regex_and_filter, [40](#)
- globals_regex_and_match, [41](#)
- globals_regex_get_address_house_type, [41](#)
- globals_regex_house_split_trade, [42](#)
- globals_regex_house_to_address, [42](#)
- globals_regex_irrelevants, [43](#)
- globals_regex_occupation_from_address, [43](#)
- globals_regex_split_address_body, [44](#)
- globals_regex_split_address_empty, [44](#)
- globals_regex_split_address_numbers, [45](#)
- globals_regex_split_trade_addresses, [45](#)
- globals_regex_titles, [46](#)
- globals_saints, [46](#)
- globals_suffixes, [47](#)
- globals_surnames, [47](#)
- globals_titles, [48](#)
- globals_trades_colnames, [48](#)
- globals_union_colnames, [49](#)
- globals_worksites, [49](#)
- clean_address_attached_words, [4](#)
- clean_address_body, [4](#), [53](#)
- clean_address_ends, [5](#), [53](#)
- clean_address_mac, [5](#)
- clean_address_names, [6](#)
- clean_address_number, [6](#), [53](#)
- clean_address_others, [7](#)
- clean_address_places, [7](#)
- clean_address_possessives, [8](#)
- clean_address_post_clean, [8](#)
- clean_address_pre_clean, [9](#)
- clean_address_saints, [9](#)
- clean_address_suffixes, [10](#)
- clean_address_worksites, [10](#)
- clean_forename, [11](#)
- clean_forename_punctuation, [11](#)
- clean_forename_separate_words, [12](#)
- clean_forename_spelling, [12](#)
- clean_mac, [13](#)
- clean_name_ends, [13](#)
- clean_occupation, [14](#)
- clean_parentheses, [14](#)
- clean_specials, [15](#)
- clean_string_ends, [15](#)
- clean_surname, [16](#)
- clean_surname_punctuation, [16](#)
- clean_surname_spelling, [17](#)
- clean_title, [17](#)
- combine_get_address_house_type, [18](#), [41](#)
- combine_has_match_failed, [18](#), [19](#)
- combine_label_failed_matches, [19](#), [39](#)
- combine_label_if_match_failed, [19](#)
- combine_make_match_string, [20](#)
- combine_match_general_to_trades, [20](#), [20](#), [23](#), [24](#)

- combine_match_general_to_trades_plain, [22](#)
- combine_match_general_to_trades_progress, [23](#)
- combine_no_trade_address_to_random_string, [24](#)
- combine_random_string_if_no_address, [25](#)
- combine_random_string_if_pattern, [25](#)
- general_clean_directory, [26](#)
- general_clean_directory_plain, [27](#)
- general_clean_directory_progress, [27](#)
- general_clean_entries, [28](#)
- general_fix_structure, [29](#)
- general_move_house_to_address, [29](#), [42](#)
- general_repatriate_occupation_from_address, [30](#), [43](#)
- general_split_address_numbers_bodies, [31](#), [44](#), [45](#)
- general_split_trade_addresses, [32](#), [40](#), [41](#)
- general_split_trade_house_addresses, [33](#), [42](#)
- globals_address_names, [33](#)
- globals_ampersand, [34](#)
- globals_ampersand_vector, [34](#)
- globals_and_double_quote, [35](#)
- globals_and_single_quote, [35](#)
- globals_forenames, [36](#)
- globals_general_colnames, [36](#)
- globals_macs, [37](#)
- globals_numbers, [37](#)
- globals_occupations, [38](#)
- globals_places_raw, [38](#)
- globals_places_regex, [39](#)
- globals_regex_address_house_body_number, [39](#)
- globals_regex_address_prefix, [40](#)
- globals_regex_and_filter, [40](#)
- globals_regex_and_match, [41](#)
- globals_regex_get_address_house_type, [41](#)
- globals_regex_house_split_trade, [42](#)
- globals_regex_house_to_address, [42](#)
- globals_regex_irrelevants, [43](#)
- globals_regex_occupation_from_address, [43](#)
- globals_regex_split_address_body, [44](#)
- globals_regex_split_address_empty, [44](#)
- globals_regex_split_address_numbers, [45](#)
- globals_regex_split_trade_addresses, [45](#)
- globals_regex_titles, [46](#)
- globals_saints, [46](#)
- globals_suffixes, [47](#)
- globals_surnames, [47](#)
- globals_titles, [48](#)
- globals_trades_colnames, [48](#)
- globals_union_colnames, [49](#)
- globals_worksites, [49](#)
- stringdist_left_join, [21](#), [23](#), [24](#)
- tibble, [19–24](#), [26–33](#), [50–52](#), [74](#)
- trades_clean_directory, [50](#)
- trades_clean_directory_plain, [51](#)
- trades_clean_directory_progress, [51](#)
- trades_clean_entries, [52](#)
- utils_clean_address, [53](#)
- utils_clean_address_body, [54](#)
- utils_clean_address_ends, [55](#)
- utils_clean_address_number, [56](#)
- utils_clean_addresses, [54](#)
- utils_clean_ends, [56](#)
- utils_clean_names, [57](#)
- utils_clean_occupations, [58](#)
- utils_clear_content, [58](#), [59](#)
- utils_clear_irrelevants, [43](#), [59](#)
- utils_execute, [60](#)
- utils_format_directory_raw, [61](#), [67](#)
- utils_gsub_if_found, [62](#)
- utils_IO_load, [63](#)
- utils_IO_path, [63](#), [63](#), [64](#)
- utils_IO_write, [64](#)
- utils_is_address_missing, [65](#)
- utils_label_address_if_missing, [65](#), [65](#)
- utils_label_missing_addresses, [65](#), [66](#), [66](#)
- utils_load_directories_csv, [61](#), [67](#)
- utils_make_file, [64](#), [68](#)
- utils_make_path, [68](#)
- utils_mutate_across, [69](#)
- utils_mute, [69](#)
- utils_paste_if_found, [70](#)
- utils_regmatches_if_found, [31](#), [71](#)

`utils_regmatches_if_not_empty`, [72](#)
`utils_remove_address_prefix`, [40](#), [45](#), [73](#)
`utils_split_and_name`, [74](#)
`utils_squish_all_columns`, [75](#)