Package: tidyged (via r-universe)

August 24, 2024

Title Handle GEDCOM Files Using Tidyverse Principles

Version 0.16.0.9000

Description Create and summarise family tree GEDCOM files using tidy dataframes.

URL https://jl5000.github.io/tidyged/, https://github.com/jl5000/tidyged

License GPL (>= 2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE, roclets = c(``namespace", ``rd", ``roxytest::testthat_roclet"))

RoxygenNote 7.2.0

Depends R (>= 2.10)

Imports dplyr, stringr, purrr, tibble, tidyr, tidyged.internals

Suggests testthat, roxytest, rmarkdown, knitr, waldo, tidyged.utils, tidyged.io, visged

Remotes jl5000/tidyged.internals, jl5000/tidyged.utils, jl5000/tidyged.io, jl5000/visged

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://jl5000.r-universe.dev

RemoteUrl https://github.com/jl5000/tidyged

RemoteRef HEAD

RemoteSha 5ef13a2943b4f0d1bc6b7d23c758f8ec96a355ed

Contents

activate_indi 3
active_record 4

address 4
add_children 5
add_famg 6
add_famg_event 7
add_indi 9
add_indi_association 10
add_indi_fact 10
add_indi_family_link_as_child 12
add_indi_family_link_as_spouse 13
add_indi_links_to_families 14
add_indi_names 15
add_indi_names_var 16
add_media 17
add_note 18
add_parents 19
add_repo 19
add_siblings 20
add_sour 21
add_sour_repo_citation 22
add_spouse 23
date_approximated 23
date_calendar 23
date_current 24
date_exact 24
date_period 24
date_range 24
describe_famg 25
describe_records 26
df_indi 26
df_indi_facts 27
find_indi_refn 28
find_xref 29
gedcom 30
get_ancestors 32
get_descendants 33
get_famg_children 34
get_famg_partners 34
get_families_as_child 35
get_families_as_partner 35
get_indi_children 36
get_indi_cousins 37
get_indi_parents 37
get_indi_partners 38
get_indi_siblings 39
get_supporting_records 39
get_valid_xref 40
insert_explicit_marr_types 41
is_famg_birth_child 41

is_indi 42
mutate_tag_namespace 43
name_pieces 43
num_indi 44
order_famg_children 45
place 45
primary_indi_name 46
remove_dates_for_tests 47
remove_famg 48
remove_indi 48
remove_indi_name 49
remove_indi_name_var 50
remove_media 50
remove_note 51
remove_records 51
remove_repo 52
remove_sour 52
sample555 53
set_active_record 53
source_citation 54
str.tidyged 55
subm 56
summary.tidyged 56
temporarily_remove_name_slashes 57
update_change_date 57
xrefs_indi 58

Index **59**

`activate_indi` *Activate a record*

Description

Set a specific record to be the active record.

Usage

```
activate_indi(gedcom, record)

activate_famg(gedcom, record)

activate_subm(gedcom, record)

activate_media(gedcom, record)

activate_note(gedcom, record)
```

```
activate_sour(gedcom, record)

activate_repo(gedcom, record)
```

Arguments

`gedcom`	A tidyged object.
`record`	The xref of the record to be activated.

Value

The same tidyged object with "active_record" attribute set to the specific record to allow easy editing.

`active_record` *Get the active record in a tidyged object*

Description

Get the active record in a tidyged object

Usage

```
active_record(gedcom)
```

Arguments

`gedcom`	A tidyged object.

Value

The xref of the active record.

`address` *Define an address*

Description

Define an address

Usage

```
address(
  local_address_lines = character(),
  city = character(),
  state = character(),
  postal_code = character(),
  country = character(),
  phone_number = character(),
  email = character(),
  fax = character(),
  web_page = character()
)
```

Arguments

`local_address_lines`
A character vector containing up to three local address lines.

`city` The city of the address.

`state` The state/county of the address.

`postal_code` The postal code of the address.

`country` The country of the address.

`phone_number` A character vector containing up to three phone numbers.

`email` A character vector containing up to three email addresses.

`fax` A character vector containing up to three fax numbers.

`web_page` A character vector containing up to three web pages.

Value

A tibble describing an address.

`add_children` *Create multiple children for a Family Group*

Description

Create multiple children for a Family Group

Usage

```
add_children(gedcom, xref = character(), sexes = NULL)
```

Arguments

`gedcom` A tidyged object.

`xref` The xref of a Family Group record to act on if one is not activated (will override active record).

`sexes` A character string giving the sexes of each child. For example, "FFM" to add two daughters and one son. See the help for `add_indi()` for possible codes.

Value

A tidyged object with additional child records.

`add_famg` *Add a Family Group record to a tidyged object*

Description

Add a Family Group record to a tidyged object

Usage

```
add_famg(
  gedcom,
  husband = character(),
  wife = character(),
  children = character(),
  number_of_children = character(),
  user_reference_numbers = character(),
  family_notes = character(),
  multimedia_links = character()
)
```

Arguments

`gedcom` A tidyged object.

`husband` An xref identifying the husband of this family.

`wife` An xref identifying the wife of this family.

`children` A character vector of xrefs identifying the children of this family. These are assumed to be biological children ("birth"). If non-biological children are to be defined, use a named vector to define the relationships using a name of either "adopted" or "foster". For example: children = c("@I4@", adopted = "@I7@", adopted = "@I10@") defines a single biological child and two adopted children.

`number_of_children`
The reported number of children known to belong to this family, regardless of whether the associated children are represented here.

`user_reference_numbers`
A unique user-defined number or text that the submitter uses to identify this record. You can supply more than one in a vector. You can also define a user reference type by using a named vector (e.g c(id1 = "123A", id2 = "456B")).

`family_notes` A character vector of notes accompanying this Family group record. These could be xrefs to existing Note records.

`multimedia_links`
A character vector of Multimedia record xrefs accompanying this record.

Details

If you need to add further information about this family (i.e. events), use the `add_famg_event()` function.

The function will automatically add links to this family to the respective Individual records of the wife, husband, and children.

Value

An updated tidyged object including the Family group record.

`add_famg_event` *Add a family event to a Family Group record*

Description

Add a family event to a Family Group record

Usage

```
add_famg_event(
  gedcom,
  type,
  descriptor = "",
  classification = character(),
  date = character(),
  husband_age = character(),
  wife_age = character(),
  cause = character(),
  event_place = place(),
  event_address = address(),
  notes = character(),
  responsible_agency = character(),
  religious_affiliation = character(),
  multimedia_links = character(),
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom` A tidyged object.

`type` A (case-insensitive) three-letter code giving the type of event. See Details.

`descriptor` A short description of an 'other' event.

`classification` A descriptive word or phrase used to further classify this event. This should be used whenever the 'other' event is used (but can also be used with others). Recommended values for non-marriage relationships can be found in Details.

`date` A date_calendar(), date_approximated(), date_period(), or date_range() object giving the timing of the event..

`husband_age` A character string that indicates the age in years, months, and days that the husband was at the time of the event. Any combination of these is permitted. Any labels must come after their corresponding number, for example; "4y 8m 10d".

`wife_age` A character string that indicates the age in years, months, and days that the wife was at the time of the event. Any combination of these is permitted. Any labels must come after their corresponding number, for example; "4y 8m 10d".

`cause` Used in special cases to record the reasons which precipitated an event.

`event_place` A place() object giving the place associated with this event.

`event_address` An address() object giving the address associated with this event.

`notes` A character vector of notes accompanying the event. These could be xrefs to existing Note records.

`responsible_agency`
The organisation, institution, corporation, person, or other entity that has responsibility for the event.

`religious_affiliation`
A name of the religion with which this event was affiliated.

`multimedia_links`
A character vector of Multimedia record xrefs accompanying this record.

`xref` The xref of a record to act on if one is not activated (will override active record).

`update_date_changed`
Whether to add/update the change date for the record.

Details

The three-letter codes used for the type parameter are:

ann(ulment), cen(sus), div(orce), dif (divorce filed), eng(agement), mab (marriage banns), mac (marriage contract), mal (marriage license), rel(ationship), mas (marriage settlement), res(idence).

Alternatively eve (for any other event).

Example classifications of non-marriage relationships are:

not married, civil, living together, living apart together. See page 59 of the GEDCOM Specification for more examples.

Value

An updated tidyged object with an expanded Family group record including this event.

add_indi *Add an Individual record to a tidyged object*

Description

Add an Individual record to a tidyged object

Usage

```
add_indi(
  gedcom,
  sex = "U",
  user_reference_numbers = character(),
  indi_notes = character(),
  multimedia_links = character(),
  qn = character()
)
```

Arguments

`gedcom` A tidyged object.

`sex` The sex of the individual. Either "M" (male), "F" (female), "U" (undetermined), "X" (intersex), or "N" (not recorded).

`user_reference_numbers` A unique user-defined number or text that the submitter uses to identify this record. You can supply more than one in a vector. You can also define a user reference type by using a named vector (e.g c(id1 = "123A", id2 = "456B")).

`indi_notes` A character vector of notes accompanying this Individual record. These could be xrefs to existing Note records.

`multimedia_links` A character vector of Multimedia record xrefs accompanying this record.

`qn` A shortcut to quickly define a name for this individual. This is a shortcut for the add_indi_names() function (which you should really use instead), but this is useful for quick demonstrations or tests.

Details

If you need to add further information about this individual (e.g. names), use the add_indi_* functions.

Value

An updated tidyged object including the Individual record.

add_indi_association *Add an association with another individual*

Description

Add an association with another individual

Usage

```
add_indi_association(
  gedcom,
  associated_with,
  association,
  association_notes = character(),
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom`	A tidyged object.
`associated_with`	An xref identifying the associated individual.
`association`	A word or phrase stating the nature of the association.
`association_notes`	A character vector of notes accompanying this association. These could be xrefs to existing Note records.
`xref`	The xref of a record to act on if one is not activated (will override active record).
`update_date_changed`	Whether to add/update the change date for the record.

Value

An updated tidyged object with an expanded Individual record including this association.

add_indi_fact *Add a fact associated with an individual*

Description

Add a fact associated with an individual

Usage

```
add_indi_fact(
  gedcom,
  type,
  descriptor = "",
  classification = character(),
  date = character(),
  age = character(),
  cause = character(),
  user_reference_type = character(),
  fact_place = place(),
  fact_address = address(),
  notes = character(),
  responsible_agency = character(),
  religious_affiliation = character(),
  adoptive_family_xref = character(),
  adopting_parent = character(),
  multimedia_links = character(),
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

gedcom A tidyged object.

type A (case-insensitive) three-letter code giving the type of event or attribute. See Details.

descriptor A short description of the attribute (which is not a residence) or 'other' event.

classification A descriptive word or phrase used to further classify this fact. This should be used whenever the 'other' event/attribute is used (but can also be used with others).

date A date_calendar(), date_approximated(), date_period(), or date_range() object giving the timing of the fact.

age A character string that indicates the age in years, months, and days that the individual was at the time of the fact. Any combination of these is permitted. Any labels must come after their corresponding number, for example; "4y 8m 10d".

cause Used in special cases to record the reasons which precipitated an event. Normally this will be used for a death event to show cause of death, such as might be listed on a death certificate.

user_reference_type

A user-defined type to associate with an attribute. This argument is mandatory for type = nid and type = att.

fact_place A place() object giving the place associated with this fact.

fact_address An address() object giving the address associated with this fact.

`notes` A character vector of notes accompanying the fact. These could be xrefs to existing Note records.

`responsible_agency` The organisation, institution, corporation, person, or other entity that has responsibility for the fact.

`religious_affiliation` A name of the religion with which this fact was affiliated.

`adoptive_family_xref` The xref of the family that adopted this individual. Only used for adoption events.

`adopting_parent` A code which shows which parent in the associated family adopted this individual. Either "HUSB", "WIFE", or "BOTH".

`multimedia_links` A character vector of Multimedia record xrefs accompanying this record.

`xref` The xref of a record to act on if one is not activated (will override active record).

`update_date_changed` Whether to add/update the change date for the record.

Details

The three-letter codes used for the type parameter are:

Attributes: res(idence), occ(upation), edu(cation), pos(sessions), cas(te), phy(sical description), rel(igion), cit(izenship or nationality), nob(ility title), nid (national ID number), nur (number of relationships), nuc (number of children),

Events: bir(th), dea(th), cen(sus), ado(ption), bap(tism), chr(istening), bur(ial), adu(lt christening), wil(l), gra(duation), pro(bate), ret(irement), cre(mation), bar(-mitvah), bas(-mitzvah), emi(gration), imm(igration), con(firmation), fir(st communion), nat(uralization).

Alternatively eve or att (for any other event or attribute).

If attributes (except residence) or 'other' events/attributes are used then the descriptor argument must be provided.

Value

An updated tidyged object with an expanded Individual record including this fact.

`add_indi_family_link_as_child` *Add a family link as a child*

Description

Add a family link as a child

Usage

```
add_indi_family_link_as_child(
  gedcom,
  family_xref,
  linkage_type = "birth",
  linkage_notes = character(),
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom`	A tidyged object.
`family_xref`	The xref of the family associated of which this individual is a child.
`linkage_type`	A code used to indicate the child to family relationship. Either, "birth" (default), "foster", or "adopted".
`linkage_notes`	A character vector of notes accompanying this linkage. These could be xrefs to existing Note records.
`xref`	The xref of a record to act on if one is not activated (will override active record).
`update_date_changed`	Whether to add/update the change date for the record.

Details

These are only to be used by the add_famg function. The child is added separately.

Value

An updated tidyged object with an expanded Individual record including this family link.

`add_indi_family_link_as_spouse` *Add a family link as a spouse*

Description

Add a family link as a spouse

Usage

```
add_indi_family_link_as_spouse(
  gedcom,
  family_xref,
  linkage_notes = character(),
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom`	A tidyged object.
`family_xref`	The xref of the family associated of which this individual is a spouse.
`linkage_notes`	A character vector of notes accompanying this linkage. These could be xrefs to existing Note records.
`xref`	The xref of a record to act on if one is not activated (will override active record).
`update_date_changed`	Whether to add/update the change date for the record.

Details

These are only to be used by the add_famg function. The spouse is added separately.

Value

An updated tidyged object with an expanded Individual record including this family link.

add_indi_links_to_families

Add family links as a child or spouse

Description

This function adds links connecting an Individual record to existing Family Group records. Family links will be added to the Individual record, and the Family Group records will be updated to link to this individual.

Usage

```
add_indi_links_to_families(
  gedcom,
  parents = character(),
  child_linkage_type = "birth",
  child_linkage_notes = character(),
  spouse = character(),
  children = character(),
  spouse_linkage_notes = character(),
  xref = character(),
  famg_xref_chil = character(),
  famg_xref_spou = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom` A tidyged object.

`parents` A character vector of parent xrefs so that this person can be linked to a Family Group record as a child.

`child_linkage_type` A code used to indicate the relationship with the parent(s). Either, "birth" (default), "foster", or "adopted".

`child_linkage_notes` A character vector of notes accompanying the family linkage as a child. These could be xrefs to existing Note records.

`spouse` The xref of the spouse so that this person can be linked to a Family Group record as a spouse.

`children` A character vector of children xrefs so that this person can be linked to a Family Group record as a spouse.

`spouse_linkage_notes` A character vector of notes accompanying the family linkage as a spouse These could be xrefs to existing Note records.

`xref` The xref of a record to act on if one is not activated (will override active record).

`famg_xref_chil` The xref of a Family Group record to be added to as a child. This is not required if any parents are given, but if it is provided, it will override any parents given.

`famg_xref_spou` The xref of a Family Group record to be added to as a spouse. This is not required if the spouse or children are given, but if it is provided, it will override any spouses or children given.

`update_date_changed` Whether to add/update the change date for the records.

Details

The function will only add one link to a Family Group record as a child, and one link to a Family Group record as a spouse.

Value

An updated tidyged object with an expanded Individual record including the family link(s) and expanded Family Group record(s) linking to this individual.

`add_indi_names` *Add a personal name (and components) to an Individual record*

Description

This function can be applied to an Individual record several times to record personal names. The personal name is constructed automatically from the components.

Usage

```
add_indi_names(
  gedcom,
  names = name_pieces(),
  type = character(),
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom`	A tidyged object.
`names`	A name_pieces() object defining the components of the personal name.
`type`	The name type, e.g. "birth", "aka", "maiden".
`xref`	The xref of a record to act on if one is not activated (will override active record).
`update_date_changed`	Whether to add/update the change date for the record.

Value

An updated tidyged object with an expanded Individual record including these names.

add_indi_names_var *Add a variation of a personal name to an Individual record*

Description

Add a variation of a personal name to an Individual record

Usage

```
add_indi_names_var(
  gedcom,
  primary_name,
  method,
  names_var = name_pieces(),
  phonetic_variation = TRUE,
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom` A tidyged object.

`primary_name` The name for which this is a variation. This must be given exactly (without forward slashes).

`method` Indicates the method used in transforming the text to the variation.

`names_var` A name_pieces() object defining the components of the name variation.

`phonetic_variation` Whether the name variation is a phonetic variation (TRUE, default) or a romanised variation (FALSE).

`xref` The xref of a record to act on if one is not activated (will override active record).

`update_date_changed` Whether to add/update the change date for the record.

Value

An updated tidyged object with an expanded Individual record including these name variants.

`add_media` *Add a Multimedia record to a tidyged object*

Description

Add a Multimedia record to a tidyged object

Usage

```
add_media(
  gedcom,
  file_reference,
  format,
  source_media = character(),
  title = character(),
  user_reference_numbers = character(),
  media_notes = character()
)
```

Arguments

`gedcom` A tidyged object.

`file_reference` A reference for the file, typically a filepath or URL. It is strongly recommended you do not use backslashes, only use forward slashes.

`format` A string indicating the format of the multimedia file. See Details.

`source_media` A code that indicates the type of material in which the referenced source is stored. See Details.

`title` The title of the multimedia file.

`user_reference_numbers`
A unique user-defined number or text that the submitter uses to identify this record. You can supply more than one in a vector. You can also define a user reference type by using a named vector (e.g c(id1 = "123A", id2 = "456B")).

`media_notes` A character vector of notes accompanying this Multimedia record. These could be xrefs to existing Note records.

Details

The formats allowed are currently limited to:

"AAC", "AVI", "BMP", "ePub", "FLAC", "GIF", "JPEG", "MKV", "mobi", "MP3", "PCX", "PDF", "PNG", "TIFF", "WAV".

The source media must be one of:

audio, book, card, electronic, fiche, film, magazine, manuscript, map, newspaper, photo, tombstone, video.

Value

An updated tidyged object including the Multimedia record.

add_note *Add a Note record to a tidyged object*

Description

Add a Note record to a tidyged object

Usage

```
add_note(gedcom, text, user_reference_numbers = character())
```

Arguments

`gedcom` A tidyged object.

`text` A character string containing the text of the note.

`user_reference_numbers`
A unique user-defined number or text that the submitter uses to identify this record. You can supply more than one in a vector. You can also define a user reference type by using a named vector (e.g c(id1 = "123A", id2 = "456B")).

Value

An updated tidyged object including the Note record.

add_parents *Add parent records for an individual*

Description

This function adds placeholder records for an individual's parents.

Usage

```
add_parents(gedcom, xref = character(), inc_sex = TRUE)
```

Arguments

`gedcom`	A tidyged object.
`xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`inc_sex`	Whether to populate the sex of the parents. This will ensure that there is one male and one female parent. Otherwise the sex will be assigned as "U" (undetermined).

Details

This function may also create a Family Group record and will not modify existing parents.

Value

A tidyged object with additional parent records.

add_repo *Add a Repository record to a tidyged object*

Description

Add a Repository record to a tidyged object

Usage

```
add_repo(
  gedcom,
  name,
  repo_address = address(),
  user_reference_numbers = character(),
  repo_notes = character()
)
```

Arguments

`gedcom`	A tidyged object.
`name`	The name of the repository.
`repo_address`	An address() object giving the address of the repository.
`user_reference_numbers`	A unique user-defined number or text that the submitter uses to identify this record. You can supply more than one in a vector. You can also define a user reference type by using a named vector (e.g c(id1 = "123A", id2 = "456B")).
`repo_notes`	A character vector of notes accompanying this Repository record. These could be xrefs to existing Note records.

Value

An updated tidyged object including the Repository record.

add_siblings *Create multiple siblings for an Individual*

Description

Create multiple siblings for an Individual

Usage

```
add_siblings(gedcom, xref = character(), sexes = NULL)
```

Arguments

`gedcom`	A tidyged object.
`xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`sexes`	A character string giving the sexes of each sibling. For example, "FFM" to add two sisters and one brother. See the help for `add_indi()` for possible codes.

Value

A tidyged object with additional sibling records.

add_sour *Add a Source record to a tidyged object*

Description

Add a Source record to a tidyged object

Usage

```
add_sour(
  gedcom,
  events_recorded = character(),
  date_period_covered = date_period(),
  jurisdiction = character(),
  responsible_agency = character(),
  originator = character(),
  title = character(),
  short_title = character(),
  publication_detail = character(),
  source_text = character(),
  user_reference_numbers = character(),
  data_notes = character(),
  sour_notes = character(),
  multimedia_links = character()
)
```

Arguments

`gedcom` A tidyged object.

`events_recorded` An enumeration of the different kinds of events that were recorded in this source. Each enumeration is separated by a comma. See the Gedcom 5.5.5 Specification for more details.

`date_period_covered` A date_period() object indicating the time period of events covered by this source.

`jurisdiction` A character string indicating the lowest level jurisdiction encompassing all places named in this source. See the Gedcom 5.5.5 Specification for more details.

`responsible_agency` The organisation, institution, corporation, person, or other entity that has responsibility for the source data.

`originator` The person, agency, or entity who created the record.

`title` The title of the source work, record, or item and, when appropriate, the title of the larger work or series of which it is a part.

`short_title` An abbreviated or shortened version of the title (if required).

`publication_detail`
: When and where the source record was created. For published works, this includes information such as the city of publication, name of the publisher, and year of publication.

`source_text`
: A verbatim copy of relevant text contained within the source. This indicates notes or text that are actually contained in the source document, not the submitter's opinion about the source.

`user_reference_numbers`
: A unique user-defined number or text that the submitter uses to identify this record. You can supply more than one in a vector. You can also define a user reference type by using a named vector (e.g c(id1 = "123A", id2 = "456B")).

`data_notes`
: A character vector of notes associated with the data in this Source record. These could be xrefs to existing Note records.

`sour_notes`
: A character vector of notes accompanying this Source record. These could be xrefs to existing Note records.

`multimedia_links`
: A character vector of Multimedia record xrefs accompanying this Source record.

Value

An updated tidyged object including the Source record.

`add_sour_repo_citation` *Add a source repository citation to a Source record*

Description

This structure is used within a source record to point to a name and address record of the holder of the source document.

Usage

```
add_sour_repo_citation(
  gedcom,
  repository,
  call_number = character(),
  xref = character(),
  update_date_changed = TRUE
)
```

Arguments

`gedcom`
: A tidyged object.

`repository`
: A character string identifying the repository xref.

`call_number`	An identification or reference description used to file and retrieve items from the holdings of a repository.
`xref`	The xref of a record to act on if one is not activated (will override active record).
`update_date_changed`	Whether to add/update the change date for the record.

Value

An updated tidyged object with an expanded Source record including this repository citation.

`add_spouse` *Add a spouse for an individual*

Description

This creates a record for a spouse and their Family Group record.

Usage

```
add_spouse(gedcom, xref = character(), sex = "U")
```

Arguments

`gedcom`	A tidyged object.
`xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`sex`	The sex of the spouse.

Value

A tidyged object with additional spouse and Family Group records.

`date_approximated` *Construct a DATE_APPROXIMATED string*

Description

See `tidyged.internals::date_approximated` for details.

`date_calendar` *Construct a DATE_CALENDAR string*

Description

See `tidyged.internals::date_calendar` for details.

date_current *Return the current date in DATE_EXACT format*

Description

See tidyged.internals::date_current for details.

date_exact *Construct a DATE_EXACT string*

Description

See tidyged.internals::date_exact for details.

date_period *Construct a DATE_PERIOD string*

Description

See tidyged.internals::date_period for details.

date_range *Construct a DATE_RANGE string*

Description

See tidyged.internals::date_range for details.

describe_famg *Get a description of a record*

Description

Get descriptions of a record at various degrees of detail.

Usage

```
describe_famg(gedcom, xref, short_desc = FALSE)

describe_indi(gedcom, xref, name_only = FALSE, short_desc = FALSE)

describe_media(gedcom, xref, file_ref_only = FALSE, short_desc = FALSE)

describe_sour(gedcom, xref, title_only = FALSE, short_desc = FALSE)

describe_repo(gedcom, xref, name_only = FALSE, short_desc = FALSE)

describe_note(gedcom, xref, short_desc = FALSE)

describe_subm(gedcom, xref, name_only = FALSE, short_desc = FALSE)
```

Arguments

`gedcom`	A tidyged object.
`xref`	An xref of a record.
`short_desc`	Whether to return a shorter description.
`name_only`	Whether to return the individual/repository name only. If none is found, the xref is returned.
`file_ref_only`	Whether to return the multimedia file reference only. If none is found, the xref is returned.
`title_only`	Whether to return the source title only. If none is found, the xref is returned.

Details

This function offers three levels of detail. For example, individual records can be:

"Joe Bloggs" (name_only = TRUE) "Individual @I1@, Joe Bloggs" (short_desc = TRUE) "Individual @I1@, Joe Bloggs, child of X and Y, born on x/x/x in place, died on x/x/x in place" (short_desc = FALSE)

Value

A character string describing the record.

Examples

```
describe_famg(sample555, "@F1@")
describe_famg(sample555, "@F1@", short_desc  = TRUE)
describe_indi(sample555, "@I1@")
describe_indi(sample555, "@I1@", short_desc = TRUE)
describe_indi(sample555, "@I1@", name_only = TRUE)
```

describe_records *Get descriptions for records*

Description

Get descriptions for records

Usage

```
describe_records(gedcom, xrefs, short_desc = FALSE)
```

Arguments

gedcom	A tidyged object.
xrefs	A vector of record xrefs. Only unique records are used. Header and trailer records are ignored.
short_desc	Whether to return a shorter description.

Value

A vector of record descriptions.

Examples

```
describe_records(sample555, sample555$record)
```

df_indi *Summarise records in a tidyged object*

Description

These functions give a summary of key information of individuals/families/notes etc.

Usage

```
df_indi(gedcom)

df_famg(gedcom)

df_media(gedcom)

df_sour(gedcom)

df_repo(gedcom)

df_note(gedcom)
```

Arguments

`gedcom`	A tidyged object.

Value

A tibble summarising records where every row is a record.

Examples

```
df_indi(sample555)
df_famg(sample555)
df_sour(sample555)
df_repo(sample555)
```

`df_indi_facts` *Create a table summarising all individual/family facts*

Description

This function creates a tidy table making it easy to extract fact details for an individual or family group.

Usage

```
df_indi_facts(gedcom, xref)

df_famg_facts(gedcom, xref)
```

Arguments

`gedcom`	A tidyged object.
`xref`	The xref of the Individual or Family Group record.

Details

Notes and source citations are not included in the summary, as well as other more obscure fields.

Value

A tibble containing a row for each fact.

Examples

```
df_indi_facts(sample555, "@I1@")
df_famg_facts(sample555, "@F1@")
```

find_indi_refn *Helper functions to locate record xrefs*

Description

These functions act as wrappers to the `find_xref` function to find one or more record xrefs.

Usage

```
find_indi_refn(gedcom, pattern, ignore_case = FALSE)

find_indi_name(gedcom, pattern, ignore_case = FALSE)

find_indi_name_all(gedcom, pattern, ignore_case = FALSE)

find_repo_refn(gedcom, pattern, ignore_case = FALSE)

find_repo_name(gedcom, pattern, ignore_case = FALSE)

find_repo_name_all(gedcom, pattern, ignore_case = FALSE)

find_note_refn(gedcom, pattern, ignore_case = FALSE)

find_note_text(gedcom, pattern, ignore_case = FALSE)

find_note_text_all(gedcom, pattern, ignore_case = FALSE)

find_media_refn(gedcom, pattern, ignore_case = FALSE)

find_media_fileref(gedcom, pattern, ignore_case = FALSE)

find_media_fileref_all(gedcom, pattern, ignore_case = FALSE)

find_sour_refn(gedcom, pattern, ignore_case = FALSE)
```

```
find_sour_titl(gedcom, pattern, ignore_case = FALSE)

find_sour_titl_all(gedcom, pattern, ignore_case = FALSE)

find_famg_refn(gedcom, pattern, ignore_case = FALSE)
```

Arguments

gedcom	A tidyged object.
pattern	The search pattern to use (regular expression).
ignore_case	Should case differences be ignored in the match?

Details

If you have your own specific use cases to identify records, it's easy to write your own wrapper. It's best to name your function `find_recordtype_*` and end it with `_all` if it can return multiple xrefs. If you provide more than one search pattern, you should also include the mode argument.

Value

A character vector of xref(s).

Examples

```
find_indi_name(sample555, "Mary")
find_indi_name_all(sample555, "Williams")
find_repo_name(sample555, "library", ignore_case = TRUE)
find_sour_titl(sample555, "Madison.+Records")
```

find_xref *Find an xref of a record given a set of search terms*

Description

Find an xref of a record given a set of search terms

Usage

```
find_xref(
  gedcom,
  search_patterns,
  mode = "strict",
  multiple = FALSE,
  ignore_case = FALSE
)
```

Arguments

`gedcom` A tidyged object.

`search_patterns` A named vector of terms to search for (see Details).

`mode` Whether to only return an xref if all patterns are matched ("strict"). A value of "best" will return the xref with the most matches. If either of these still result in more than one xref it will return an error unless `multiple` is TRUE.

`multiple` If more than one xref is found (according to `mode`), whether to return all xrefs or throw an error.

`ignore_case` Should case differences be ignored in the match?

Details

This is a helper function to identify the xref of a record given information such as a name or reference number. You provide a named `search_patterns` vector of namespaced tag-pattern pairs, such as:

c(INDI.NAME = "Homer", INDI.SEX = "M", INDI.BIRT.DATE = "JAN 1974")

If you're not sure what namespace to use, use the `mutate_tag_namespace` function.

The search patterns will be treated as regular expressions, so they will match a value if it contains the pattern provided. You can anchor your search pattern if you want an exact match, e.g. "^JAN 1974$". If you're not familiar with regular expressions, you may need to escape certain characters such as brackets and a full-stop/period (i.e. `\\.`).

Value

A vector of one or more xrefs.

Examples

```
find_xref(sample555, c(INDI.BURI.PLAC = "Spring Hill"), multiple = FALSE)
find_xref(sample555, c(INDI.SEX = "M"), multiple = TRUE)
find_xref(sample555, c(FAM.MARR.DATE = "1859"), multiple = FALSE)
find_xref(sample555, c(REPO.ADDR.CITY = "Salt Lake"), multiple = TRUE)
find_xref(sample555, c(INDI.NAME.SURN = "Williams", INDI.ADOP.DATE = "Never"),
mode = "best", multiple = TRUE)
```

`gedcom` *Create a base tidyged object*

Description

This function creates a minimal tidyged object with header and footer sections and a single submitter record.

Usage

```
gedcom(
  submitter_details = subm(),
  gedcom_description = character(),
  gedcom_copyright = character(),
  source_data_name = character(),
  source_data_date = date_exact(),
  source_data_copyright = character(),
  receiving_system = "gedcompendium",
  language = "English"
)
```

Arguments

`submitter_details`
: Details of the submitter of the file (you?) using the subm() function. If no submitter name is provided, the username is used.

`gedcom_description`
: A note to describe the contents of the file in terms of "ancestors or descendants of" so that the person receiving the data knows what genealogical information the file contains.

`gedcom_copyright`
: A copyright statement needed to protect the copyrights of the submitter of this GEDCOM file.

`source_data_name`
: The name of the electronic data source that was used to obtain the data in this file.

`source_data_date`
: The date this source was created or published. Ensure you create this date with the date_exact() function.

`source_data_copyright`
: A copyright statement required by the owner of data from which this information was obtained.

`receiving_system`
: The name of the system expected to process the GEDCOM-compatible file.

`language`
: The human language in which the data in the file is normally read or written.

Value

A minimal tidyged object.

get_ancestors *Identify all ancestors for an individual*

Description

This function identifies records in an entire branch of the family tree above a certain individual.

Usage

```
get_ancestors(
  gedcom,
  indi_xref = character(),
  inc_indi = FALSE,
  inc_sibs = FALSE,
  inc_famg = FALSE,
  inc_supp = FALSE,
  birth_only = TRUE
)
```

Arguments

`gedcom` A tidyged object.

`indi_xref` The xref of an Individual record to act on if one is not activated (will override active record).

`inc_indi` Whether to also include the individual themselves.

`inc_sibs` Whether to also include all siblings of ancestors (siblings of this individual will only be included if the individual is included).

`inc_famg` Whether to also include all Family Group records where this individual is a child (and all ancestors' Family Group records).

`inc_supp` Whether to also include all supporting records (Note, Source, Repository, Multimedia).

`birth_only` Whether to only include biological ancestors.

Value

A vector of xrefs of ancestors.

Examples

```
get_ancestors(sample555, "@I3@")
get_ancestors(sample555, "@I3@", inc_indi = TRUE)
get_ancestors(sample555, "@I3@", inc_indi = TRUE, inc_famg = TRUE)
```

get_descendants *Identify all descendants for an individual*

Description

This function identifies records in an entire branch of the family tree below a certain individual.

Usage

```
get_descendants(
  gedcom,
  indi_xref = character(),
  inc_indi = FALSE,
  inc_part = FALSE,
  inc_famg = FALSE,
  inc_supp = FALSE,
  birth_only = TRUE
)
```

Arguments

gedcom	A tidyged object.
indi_xref	The xref of an Individual record to act on if one is not activated (will override active record).
inc_indi	Whether to also include the individual themselves.
inc_part	Whether to also include all partners of this individual (and their descendants and descendants' partners).
inc_famg	Whether to also include all Family Group records where this individual is a partner (and all descendants' Family Group records).
inc_supp	Whether to also include all supporting records (Note, Source, Repository, Multimedia).
birth_only	Whether to only include biological descendants.

Value

A vector of xrefs of descendants.

`get_famg_children` *Identify all children in a Family Group*

Description

Identify all children in a Family Group

Usage

```
get_famg_children(gedcom, famg_xref, birth_only = FALSE, return_name = FALSE)
```

Arguments

`gedcom`	A tidyged object.
`famg_xref`	The xref of a Family Group record to act on if one is not activated (will override active record).
`birth_only`	Whether to only return biological children.
`return_name`	Whether to return the child name(s) instead of the xref(s).

Value

A character vector of partner xrefs or names.

`get_famg_partners` *Identify all partners in a Family Group*

Description

Identify all partners in a Family Group

Usage

```
get_famg_partners(gedcom, famg_xref, return_name = FALSE)
```

Arguments

`gedcom`	A tidyged object.
`famg_xref`	The xref of a Family Group record to act on if one is not activated (will override active record).
`return_name`	Whether to return the partners name(s) instead of the xref(s).

Value

A character vector of partner xrefs or names.

`get_families_as_child` *Identify all families for an individual where they are a child*

Description

Identify all families for an individual where they are a child

Usage

```
get_families_as_child(gedcom, indi_xref = character(), birth_only = FALSE)
```

Arguments

`gedcom`	A tidyged object.
`indi_xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`birth_only`	Whether to only return the family containing the biological parents.

Value

A character vector of family xrefs.

Examples

```
get_families_as_child(sample555, "@I3@")
```

`get_families_as_partner` *Identify all families for an individual where they are a partner*

Description

Identify all families for an individual where they are a partner

Usage

```
get_families_as_partner(gedcom, indi_xref = character())
```

Arguments

`gedcom`	A tidyged object.
`indi_xref`	The xref of an Individual record to act on if one is not activated (will override active record).

Value

A character vector of family xrefs.

Examples

```
get_families_as_partner(sample555, "@I2@")
```

get_indi_children *Identify all children for an individual*

Description

Identify all children for an individual

Usage

```
get_indi_children(
  gedcom,
  indi_xref = character(),
  birth_only = FALSE,
  return_name = FALSE
)
```

Arguments

gedcom	A tidyged object.
indi_xref	The xref of an Individual record to act on if one is not activated (will override active record).
birth_only	Whether to only return biological children.
return_name	Whether to return the childrens name(s) instead of the xref(s).

Value

A character vector of children xrefs or names.

Examples

```
get_indi_children(sample555, "@I2@")
get_indi_children(sample555, "@I2@", return_name = TRUE)
```

`get_indi_cousins` *Identify all cousins for an individual*

Description

Identify all cousins for an individual

Usage

```
get_indi_cousins(
  gedcom,
  indi_xref = character(),
  degree = 1,
  inc_half_cous = FALSE,
  return_name = FALSE
)
```

Arguments

`gedcom`	A tidyged object.
`indi_xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`degree`	Whether to return first cousins (degree = 1), second cousins (degree = 2), etc.
`inc_half_cous`	Whether to include half cousins.
`return_name`	Whether to return the parents name(s) instead of the xref(s).

Value

A character vector of cousin xrefs or names.

`get_indi_parents` *Identify all parents for an individual*

Description

Identify all parents for an individual

Usage

```
get_indi_parents(
  gedcom,
  indi_xref = character(),
  birth_only = FALSE,
  return_name = FALSE
)
```

Arguments

`gedcom`	A tidyged object.
`indi_xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`birth_only`	Whether to only return biological parents.
`return_name`	Whether to return the parents name(s) instead of the xref(s).

Value

A character vector of parent xrefs or names.

Examples

```
get_indi_parents(sample555, "@I2@")
get_indi_parents(sample555, "@I3@")
get_indi_parents(sample555, "@I3@", return_name = TRUE)
```

`get_indi_partners` *Identify all partners for an individual*

Description

Identify all partners for an individual

Usage

```
get_indi_partners(gedcom, indi_xref = character(), return_name = FALSE)
```

Arguments

`gedcom`	A tidyged object.
`indi_xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`return_name`	Whether to return the partner's name(s) instead of the xref(s).

Value

A character vector of partner xrefs or names.

Examples

```
get_indi_partners(sample555, "@I1@")
get_indi_partners(sample555, "@I1@", return_name = TRUE)
get_indi_partners(sample555, "@I3@")
```

`get_indi_siblings` *Identify all siblings for an individual*

Description

Identify all siblings for an individual

Usage

```
get_indi_siblings(
  gedcom,
  indi_xref = character(),
  birth_only = FALSE,
  inc_half_sibs = FALSE,
  return_name = FALSE
)
```

Arguments

`gedcom`	A tidyged object.
`indi_xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`birth_only`	Whether to only return biological siblings.
`inc_half_sibs`	Whether to include siblings that only share one parent.
`return_name`	Whether to return the parents name(s) instead of the xref(s).

Value

A character vector of sibling xrefs or names.

`get_supporting_records` *Identify all supporting records for a set of records*

Description

This function gets all supporting records (and onwards dependencies) for a set of records. Supporting records are note, multimedia, source, and repository records, i.e. those providing supporting evidence and comments.

Usage

```
get_supporting_records(
  gedcom,
  xrefs,
  inc_note = TRUE,
  inc_media = TRUE,
  inc_sour = TRUE,
  inc_repo = TRUE
)
```

Arguments

gedcom A tidyged object.

xrefs The xrefs of records to get supporting records for.

inc_note Whether to include Note records.

inc_media Whether to include Multimedia records.

inc_sour Whether to include Source records.

inc_repo Whether to include Repository records.

Value

A character vector of supporting record xrefs.

Examples

```
get_supporting_records(sample555, "@I1@")
get_supporting_records(sample555, "@F1@")
```

get_valid_xref *Derive a valid cross-reference identifier*

Description

Validate an xref provided explicitly or implicitly (through the active record).

Usage

```
get_valid_xref(gedcom, xref, record_type, record_type_fn)
```

Arguments

gedcom A tidyged object.

xref A record xref.

record_type A character string describing the record type. Generally one of the global record_string_* values.

record_type_fn A function to check the record type. Generally one of the is_* functions.

Details

This helper function is designed to derive and run validation checks on an xref provided explicitly or implicitly. An xref is provided implicitly through the active record of the tidyged object.

Once found, the xref is checked to ensure it is of the appropriate type.

Value

A valid xref identifier.

insert_explicit_marr_types *Insert explicit marriage subrecords for a Family Group record*

Description

Insert explicit marriage subrecords for a Family Group record

Usage

```
insert_explicit_marr_types(gedcom, xref)
```

Arguments

gedcom	A tidyged object.
xref	The xref of a Family Group record.

Value

The same tidyged object with explicit marriage subrecords in the Family Group record.

is_famg_birth_child *Determine whether an individual is a child of a family by birth*

Description

Determine whether an individual is a child of a family by birth

Usage

```
is_famg_birth_child(gedcom, indi_xref, famg_xref)
```

Arguments

`gedcom`	A tidyged object.
`indi_xref`	The xref of an Individual record to act on if one is not activated (will override active record).
`famg_xref`	The xref of the Family Group record.

Value

A logical value indicating whether the individual is a child of a family by birth.

`is_indi` *Check whether a given record is a particular type*

Description

Check whether a given record is a particular type

Usage

```
is_indi(gedcom, xref)

is_famg(gedcom, xref)

is_subm(gedcom, xref)

is_repo(gedcom, xref)

is_media(gedcom, xref)

is_note(gedcom, xref)

is_sour(gedcom, xref)
```

Arguments

`gedcom`	A tidyged object.
`xref`	The xref of the record.

Value

A logical indicating whether the record is of a particular type.

Examples

```
is_indi(sample555, "@I1@")
is_note(sample555, "@I1@")
is_famg(sample555, "@F2@")
is_repo(sample555, "@R1@")
```

mutate_tag_namespace *Add a tag namespace column to a tidyged object*

Description

Add a tag namespace column to a tidyged object

Usage

```
mutate_tag_namespace(gedcom)
```

Arguments

gedcom	A tidyged object.

Details

This function is useful if you want to find the namespace of a particular value for the `find_xref` function.

Value

A tidyged object with an additional 'tag_ns' column containing the full namespace of the tag.

name_pieces *Define a personal name's components*

Description

Define a personal name's components

Usage

```
name_pieces(
  prefix = character(),
  given = character(),
  nickname = character(),
  surname_prefix = character(),
  surname = character(),
  suffix = character(),
  notes = character()
)
```

Arguments

`prefix` The name prefix, e.g. Cmdr.

`given` The given name or earned name. Different given names are separated by a comma.

`nickname` A descriptive or familiar name used in connection with one's proper name.

`surname_prefix` Surname prefix or article used in a family name. For example in the name "de la Cruz", this value would be "de la".

`surname` Surname or family name. Different surnames are separated by a comma.

`suffix` Non-indexing name piece that appears after the given name and surname parts, e.g. Jr. Different name suffix parts are separated by a comma.

`notes` A character vector of notes accompanying this name. These could be xrefs to existing Note records.

Value

A tibble describing a personal name's components.

num_indi *Get the number of records in a tidyged object*

Description

These functions return the number of records of a particular type in a tidyged object.

Usage

```
num_indi(gedcom)

num_famg(gedcom)

num_subm(gedcom)

num_media(gedcom)

num_note(gedcom)

num_repo(gedcom)

num_sour(gedcom)
```

Arguments

`gedcom` A tidyged object.

Value

The number of records of the relevant type.

Examples

```
num_indi(sample555)
num_famg(sample555)
num_repo(sample555)
```

order_famg_children *Order children in a Family Group record by birth date*

Description

Order children in a Family Group record by birth date

Usage

```
order_famg_children(gedcom, xref)
```

Arguments

gedcom	A tidyged object.
xref	The xref of a Family Group record.

Details

Any children without a date a birth are placed last.

Value

The same tidyged object with rearranged children rows in the Family Group record.

place *Define a place associated with a fact*

Description

Define a place associated with a fact

Usage

```
place(
  name = character(),
  phonetic_var = character(),
  romanised_var = character(),
  latitude = character(),
  longitude = character(),
  notes = character()
)
```

Arguments

`name` The jurisdictional name of the place. Jurisdictions are separated by commas, for example, "Cove, Cache, Utah, USA."

`phonetic_var` A named character vector of phonetic variations of the place name. Element names must give the phonetisation method used in transforming the text to the corresponding phonetic variation. i.e. c(method1 = "var1", method2 = "var2")

`romanised_var` A named character vector of romanised variations of the place name. Element names must give the romanisation method used in transforming the text to the corresponding romanised variation. i.e. c(method1 = "var1", method2 = "var2")

`latitude` The value specifying the latitudinal coordinate of the place. The latitude coordinate is the direction North or South from the equator in degrees and fraction of degrees carried out to give the desired accuracy. For example: 18 degrees, 9 minutes, and 3.4 seconds North would be formatted as "N18.150944"

`longitude` The value specifying the longitudinal coordinate of the place. The longitude coordinate is Degrees and fraction of degrees east or west of the zero or base meridian coordinate. For example: 168 degrees, 9 minutes, and 3.4 seconds East would be formatted as "E168.150944".

`notes` A character vector of notes accompanying the place. These could be xrefs to existing Note records.

Value

A tibble describing a place.

`primary_indi_name` *Make an Individual name appear first in the Individual record*

Description

Make an Individual name appear first in the Individual record

Usage

```
primary_indi_name(gedcom, name, xref = character())
```

Arguments

`gedcom`	A tidyged object.
`name`	The personal name to move.
`xref`	The xref of a record to act on if one is not activated (will override active record).

Value

An updated tidyged object with the promoted name in the Individual record

---

`remove_dates_for_tests` *Remove all creation dates from a tidyged object*

Description

Remove all creation dates from a tidyged object

Usage

```
remove_dates_for_tests(gedcom)
```

Arguments

`gedcom`	A tidyged object.

Details

This is a function used in tests so that the objects created do not change every time.

Value

The tidyged object with creation dates removed.

remove_famg *Remove a Family group record from a tidyged object*

Description

This function removes a record containing details of a family group.

Usage

```
remove_famg(gedcom, family_xref = character(), remove_individuals = FALSE)
```

Arguments

`gedcom`	A tidyged object.
`family_xref`	The xref of a Family Group record to act on if one is not activated (will override active record).
`remove_individuals`	Whether to also remove the records for all Individuals in the family.

Details

This function will also automatically remove references to this record in other individual records. If remove_individuals is set to TRUE, it will also remove all records for individuals in this family (including associations).

Value

An updated tidyged object excluding the selected Family group record (and potentially the individuals within it).

remove_indi *Remove an Individual record from a tidyged object*

Description

This function removes an active Individual record from the tidyged object.

Usage

```
remove_indi(gedcom, individual = character(), remove_associations = TRUE)
```

Arguments

`gedcom`	A tidyged object.
`individual`	The xref or name of an Individual record to act on if one is not activated (will override active record).
`remove_associations`	Whether to also remove associations with this individual in other individual records. Defaults to TRUE. You shouldn't really leave dead links to individual records that no longer exist.

Details

At a minimum it will also remove references to this individual in Family group records. If remove_associations is TRUE (default) it will remove associations with this individual in other Individual records.

Value

An updated tidyged object excluding the selected Individual record.

`remove_indi_name` *Remove a personal name (and components) from an Individual record*

Description

Remove a personal name (and components) from an Individual record

Usage

```
remove_indi_name(gedcom, name)
```

Arguments

`gedcom`	A tidyged object.
`name`	The personal name to remove.

Value

An updated tidyged object with an Individual record excluding this personal name (and components).

remove_indi_name_var *Remove a variation of a personal name from an Individual record*

Description

Remove a variation of a personal name from an Individual record

Usage

```
remove_indi_name_var(gedcom, variation_name, phonetic_variation = TRUE)
```

Arguments

`gedcom`	A tidyged object.
`variation_name`	The personal name variation to remove.
`phonetic_variation`	Whether the name variation is a phonetic variation (TRUE, default) or a romanised variation (FALSE).

Value

An updated tidyged object with an Individual record excluding this personal name variation (and components).

remove_media *Remove a Multimedia record from a tidyged object*

Description

Remove a Multimedia record from a tidyged object

Usage

```
remove_media(gedcom, multimedia = character())
```

Arguments

`gedcom`	A tidyged object.
`multimedia`	The xref of a Multimedia record to act on if one is not activated (will override active record).

Value

An updated tidyged object excluding the selected Multimedia record.

`remove_note` *Remove a Note record from a tidyged object*

Description

Remove a Note record from a tidyged object

Usage

```
remove_note(gedcom, note = character())
```

Arguments

`gedcom`	A tidyged object.
`note`	The xref or excerpt of a Note record to act on if one is not activated (will override active record).

Value

An updated tidyged object excluding the selected Note record.

`remove_records` *Remove multiple records at once*

Description

Remove multiple records at once

Usage

```
remove_records(gedcom, xrefs)
```

Arguments

`gedcom`	A tidyged object.
`xrefs`	A vector of xrefs to remove.

Value

An updated tidyged object with the records removed.

`remove_repo` *Remove a Repository record from a tidyged object*

Description

Remove a Repository record from a tidyged object

Usage

```
remove_repo(gedcom, repository = character())
```

Arguments

`gedcom` A tidyged object.

`repository` The xref of a Repository record to act on if one is not activated (will override active record).

Value

An updated tidyged object excluding the selected Repository record.

`remove_sour` *Remove a Source record from a tidyged object*

Description

Remove a Source record from a tidyged object

Usage

```
remove_sour(gedcom, sour = character())
```

Arguments

`gedcom` A tidyged object.

`sour` The xref or title of a Source record to act on if one is not activated (will override active record).

Value

An updated tidyged object excluding the selected Source record.

`sample555` *A sample GEDCOM file in tidyged format*

Description

This is the official GEDCOM 5.5.5 sample file.

Usage

```
sample555
```

Format

A tibble with 97 rows and 4 variables:

level The level in the GEDCOM structural hierarchy

record The unique cross reference for the record

tag The tag describing the nature of the line value given

value The GEDCOM line value

Source

https://www.gedcom.org/samples.html

`set_active_record` *Flag a record as being active*

Description

This allows an easy mechanism to edit particular records without specifying them each time.

Usage

```
set_active_record(gedcom, xref)
```

Arguments

gedcom	A tidyged object.
xref	The xref of the record to activate.

Value

The same tidyged object with an "active_record" attribute set to the xref of the record

source_citation *Create a citation of a Source record*

Description

Create a citation of a Source record

Usage

```
source_citation(
  gedcom,
  source,
  where = character(),
  event = character(),
  role = character(),
  entry_date = character(),
  source_text = character(),
  certainty = character(),
  notes = character(),
  multimedia_links = character()
)
```

Arguments

gedcom A tidyged object.

source A character string identifying the source. This can either be an xref or term(s) to match to a source title.

where Specific location within the information referenced. For a published work, this could include the volume of a multi-volume work and the page number(s). For a newspaper, it could include a column number and page number. A census record might have an enumerating district, page number, line number, dwelling number, and family number. The data in this field should be in the form of a label and value pair, such as Label1: value, Label2: value, with each pair being separated by a comma. For example, Film: 1234567, Frame: 344, Line: 28.

event A code that indicates the type of event which was responsible for the source entry being recorded. For example, if the entry was created to record a birth of a child, then the type would be BIRT regardless of the assertions made from that record, such as the mother's name or mother's birth date. This will allow a prioritised best view choice and a determination of the certainty associated with the source used in asserting the cited fact.

role Indicates what role this person played in the event that is being cited in this context.

entry_date A date_calendar(), date_period(), date_range(), or date_approximated() value giving the date that this data was entered into the original source document.

`source_text`	A verbatim copy of any description contained within the source. This indicates notes or text that are actually contained in the source document, not the submitter's opinion about the source.
`certainty`	An evaluation of the credibility of a piece of information, based upon its supporting evidence. Some systems use this feature to rank multiple conflicting opinions for display of most likely information first. It is not intended to eliminate the receiver's need to evaluate the evidence for themselves. Values allowed: "unreliable", "subjective", "secondary", "primary".
`notes`	A character vector of notes accompanying the citation. These could be xrefs to existing Note records.
`multimedia_links`	A character vector of Multimedia record xrefs accompanying this record.

Value

A tibble describing a source citation.

`str.tidyged` *Get the structure of a tidyged object*

Description

This function gives a breakdown of record counts in the GEDCOM file.

Usage

```
## S3 method for class 'tidyged'
str(object, ...)
```

Arguments

`object`	A tidyged object.
`...`	Not used.

Value

A printed summary of records in the tidyged object.

Examples

```
str(sample555)
```

subm *Define a Submitter record for a new tidyged object*

Description

Define a Submitter record for a new tidyged object

Usage

```
subm(
  name = unname(Sys.info()["user"]),
  subm_address = address(),
  subm_notes = character()
)
```

Arguments

name	The name of the submitter.
subm_address	An address() object containing the submitter address.
subm_notes	A character vector of notes accompanying this Submitter record.

Details

This function is supposed to be used in the gedcom() function to define a new tidyged object.

This submitter record identifies the individual or organization that contributed information contained in the GEDCOM file.

This function deviates from the specification slightly by omitting multimedia links. The logic for this is that if the file is just being created, then there will be no multimedia records to link to.

Value

A Submitter record to be incorporated into a new tidyged object.

summary.tidyged *Get a summary of a tidyged object*

Description

This function shows key information from the header of a tidyged object, including submitter and description.

Usage

```
## S3 method for class 'tidyged'
summary(object, ...)
```

Arguments

`object`	A tidyged object.
`...`	Not used.

Value

A printed summary of the tidyged object.

Examples

```
summary(sample555)
```

`temporarily_remove_name_slashes` *Temporarily remove forward slashes from surnames*

Description

Temporarily remove forward slashes from surnames

Usage

```
temporarily_remove_name_slashes(gedcom)
```

Arguments

`gedcom`	A tidyged object.

Value

A tidyged object with all forward slashes removed from surnames.

`update_change_date` *Update a record's change date*

Description

Update a record's change date

Usage

```
update_change_date(gedcom, xref)
```

Arguments

gedcom	A tidyged object.
xref	The xref of a record.

Value

An updated tidyged object where the specified record has a change date of today.

xrefs_indi *Get the xrefs of particular record types*

Description

These functions return the xrefs of all records of a particular type in a tidyged object.

Usage

```
xrefs_indi(gedcom)

xrefs_famg(gedcom)

xrefs_subm(gedcom)

xrefs_sour(gedcom)

xrefs_repo(gedcom)

xrefs_note(gedcom)

xrefs_media(gedcom)
```

Arguments

gedcom	A tidyged object.

Value

A vector of xrefs of records of the relevant type.

Examples

```
xrefs_indi(sample555)
xrefs_famg(sample555)
xrefs_note(sample555)
xrefs_repo(sample555)
xrefs_sour(sample555)
```

Index

∗ **datasets**
sample555, 53

activate_famg *(*activate_indi*)*, 3
activate_indi, 3
activate_media *(*activate_indi*)*, 3
activate_note *(*activate_indi*)*, 3
activate_repo *(*activate_indi*)*, 3
activate_sour *(*activate_indi*)*, 3
activate_subm *(*activate_indi*)*, 3
active_record, 4
add_children, 5
add_famg, 6
add_famg_event, 7
add_indi, 9
add_indi_association, 10
add_indi_fact, 10
add_indi_family_link_as_child, 12
add_indi_family_link_as_spouse, 13
add_indi_links_to_families, 14
add_indi_names, 15
add_indi_names_var, 16
add_media, 17
add_note, 18
add_parents, 19
add_repo, 19
add_siblings, 20
add_sour, 21
add_sour_repo_citation, 22
add_spouse, 23
address, 4

date_approximated, *23*, 23
date_calendar, *23*, 23
date_current, *24*, 24
date_exact, *24*, 24
date_period, *24*, 24
date_range, *24*, 24
describe_famg, 25
describe_indi *(*describe_famg*)*, 25
describe_media *(*describe_famg*)*, 25
describe_note *(*describe_famg*)*, 25
describe_records, 26
describe_repo *(*describe_famg*)*, 25
describe_sour *(*describe_famg*)*, 25
describe_subm *(*describe_famg*)*, 25
df_famg *(*df_indi*)*, 26
df_famg_facts *(*df_indi_facts*)*, 27
df_indi, 26
df_indi_facts, 27
df_media *(*df_indi*)*, 26
df_note *(*df_indi*)*, 26
df_repo *(*df_indi*)*, 26
df_sour *(*df_indi*)*, 26

find_famg_refn *(*find_indi_refn*)*, 28
find_indi_name *(*find_indi_refn*)*, 28
find_indi_name_all *(*find_indi_refn*)*, 28
find_indi_refn, 28
find_media_fileref *(*find_indi_refn*)*, 28
find_media_fileref_all
*(*find_indi_refn*)*, 28
find_media_refn *(*find_indi_refn*)*, 28
find_note_refn *(*find_indi_refn*)*, 28
find_note_text *(*find_indi_refn*)*, 28
find_note_text_all *(*find_indi_refn*)*, 28
find_repo_name *(*find_indi_refn*)*, 28
find_repo_name_all *(*find_indi_refn*)*, 28
find_repo_refn *(*find_indi_refn*)*, 28
find_sour_refn *(*find_indi_refn*)*, 28
find_sour_titl *(*find_indi_refn*)*, 28
find_sour_titl_all *(*find_indi_refn*)*, 28
find_xref, 29

gedcom, 30
get_ancestors, 32
get_descendants, 33
get_famg_children, 34
get_famg_partners, 34
get_families_as_child, 35

get_families_as_partner, 35
get_indi_children, 36
get_indi_cousins, 37
get_indi_parents, 37
get_indi_partners, 38
get_indi_siblings, 39
get_supporting_records, 39
get_valid_xref, 40

insert_explicit_marr_types, 41
is_famg *(is_indi)*, 42
is_famg_birth_child, 41
is_indi, 42
is_media *(is_indi)*, 42
is_note *(is_indi)*, 42
is_repo *(is_indi)*, 42
is_sour *(is_indi)*, 42
is_subm *(is_indi)*, 42

mutate_tag_namespace, 43

name_pieces, 43
num_famg *(num_indi)*, 44
num_indi, 44
num_media *(num_indi)*, 44
num_note *(num_indi)*, 44
num_repo *(num_indi)*, 44
num_sour *(num_indi)*, 44
num_subm *(num_indi)*, 44

order_famg_children, 45

place, 45
primary_indi_name, 46

remove_dates_for_tests, 47
remove_famg, 48
remove_indi, 48
remove_indi_name, 49
remove_indi_name_var, 50
remove_media, 50
remove_note, 51
remove_records, 51
remove_repo, 52
remove_sour, 52

sample555, 53
set_active_record, 53
source_citation, 54
str.tidyged, 55
subm, 56
summary.tidyged, 56

temporarily_remove_name_slashes, 57

update_change_date, 57

xrefs_famg *(xrefs_indi)*, 58
xrefs_indi, 58
xrefs_media *(xrefs_indi)*, 58
xrefs_note *(xrefs_indi)*, 58
xrefs_repo *(xrefs_indi)*, 58
xrefs_sour *(xrefs_indi)*, 58
xrefs_subm *(xrefs_indi)*, 58