Package: spaghetti (via r-universe)

May 16, 2026

Title Bidirectional Spreadsheet-Formula to OOXML Translator

Version 0.3.0

Description Translates spreadsheet formulas between the user-facing format (as displayed to the end user) and the OOXML storage format (as found in .xlsx XML source). Handles future function prefixes (_xlfn.), web-service namespaces (_xlfn._xlws.), LAMBDA parameter prefixes (_xlpm.), spilled range operators (ANCHORARRAY), implicit intersection (SINGLE), and optional localised function name translation across the locales in the Microsoft Terminology Collection (run setup_terminology() once to download and cache the translations).

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.0.0)

Imports utils

Suggests testthat (>= 3.0.0), openxlsx2, digest

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Date/Publication 2026-05-16 10:16:27 UTC

RemoteUrl https://github.com/JanMarvin/spaghetti

RemoteRef HEAD

RemoteSha b293ee7717dc26bb1ea007d2fcfbbd061e05aa04

Contents

`check_formula` *Check a formula for unknown function names*

Description

Tokenises a formula and reports any function names that are not in the function registry, with spelling suggestions for likely typos.

Usage

```
check_formula(formula, locale = NULL)
```

Arguments

`formula`	Character scalar or vector of formulas.
`locale`	Locale code or NULL. When set, localised names are translated before checking.

Details

Unlike `to_xml()`, this does not translate — it is a pure linting pass.

Value

A data frame with columns `formula`, `fn`, `suggestion`. `fn` is the unknown function name. `suggestion` is a comma-separated string of close matches, or `NA` if no suggestion was found. Returns an empty data frame (invisibly) if no issues are found.

Examples

```
check_formula("=SUIM(A1:A10)")          # typo: SUIM -> SUM
check_formula("=VLOKUP(A1,B:B,2,0)")    # typo: VLOKUP -> VLOOKUP
check_formula(c("=SUM(A1)", "=FLITER(A1:A10,B1:B10>0)"))
```

clear_terminology *Remove the cached terminology RDS.*

Description

Remove the cached terminology RDS.

Usage

```
clear_terminology()
```

Value

Invisibly TRUE if a cache was removed, FALSE if none existed.

from_xml *Convert an OOXML storage formula to user-facing format*

Description

Convert an OOXML storage formula to user-facing format

Usage

```
from_xml(formula, locale = NULL)
```

Arguments

formula	Character scalar or vector. OOXML formula(s), with or without =.
locale	Two-letter locale code or NULL. When set, function names are translated to the target locale and the locale separator is used in output.

Value

Character scalar or vector: user-facing formula(s) starting with =.

Examples

```
from_xml("=_xlfn.SEQUENCE(10)")
from_xml("=_xlfn.LAMBDA(_xlpm.temp, (5/9) * (_xlpm.temp-32))(100)")
from_xml("=_xlfn._xlws.FILTER(A1:A10,B1:B10>5)")
from_xml("=SUM(_xlfn.ANCHORARRAY(A1))")
## Not run: from_xml("=_xlfn.SEQUENCE(10)", locale = "de")
from_xml(c("=_xlfn.SEQUENCE(5)", "=SUM(_xlfn.ANCHORARRAY(A1))"))
```

function_description *Look up the English description for a function.*

Description

Reads from the cached Microsoft Terminology Collection table; available only after `setup_terminology()` has been run. Returns NA for every input if the cache is missing.

Usage

```
function_description(fn)
```

Arguments

`fn`	Character vector of function names (English, any case).

Value

Character vector of descriptions (NA where not found / available).

Examples

```
function_description(c("SUM", "XLOOKUP", "LAMBDA"))
```

function_prefix *Identify what prefix a function will receive*

Description

Useful for inspecting the registry without running a full translation.

Usage

```
function_prefix(fn)
```

Arguments

`fn`	Character vector of function names.

Value

Character vector: `"legacy"`, `"xlfn"`, or `"xlws"`.

Examples

```
function_prefix(c("SUM", "SEQUENCE", "FILTER", "LAMBDA", "XLOOKUP"))
```

function_table *Return the full function translation table.*

Description

Columns: `fn` (English name), `description` (if available), and one column per supported locale with the localised function name (NA if not translated for that locale).

Usage

```
function_table()
```

Details

If `setup_terminology()` hasn't been run, returns a single-column data frame with just `fn`.

Value

data.frame

See Also

`setup_terminology()`, `supported_locales()`

Examples

```
head(function_table())
```

has_terminology *Check whether locale terminology has been loaded.*

Description

Check whether locale terminology has been loaded.

Usage

```
has_terminology()
```

Value

Logical scalar.

See Also

`setup_terminology()`, `clear_terminology()`

Examples

```
has_terminology()
```

`is_ooxml` *Check whether a formula is already in OOXML storage format*

Description

A formula is considered "already OOXML" if it contains at least one `_xlfn.`, `_xlws.`, or `_xlpm.` token.

Usage

```
is_ooxml(formula)
```

Arguments

`formula`	Character scalar.

Value

Logical.

Examples

```
is_ooxml("=_xlfn.SEQUENCE(10)")   # TRUE
is_ooxml("=SEQUENCE(10)")          # FALSE
```

`round_trip` *Round-trip a formula through OOXML and back*

Description

Converts to OOXML then back to user-facing form. Useful for testing idempotency.

Usage

```
round_trip(formula, locale = NULL, out_locale = NULL)
```

Arguments

`formula`	Character scalar.
`locale`	Locale for input formula (passed to `to_xml()`).
`out_locale`	Locale for output (passed to `from_xml()`).

Value

Named list with `xml` (OOXML form) and `formula` (round-tripped user-facing form).

Examples

```
round_trip("=LAMBDA(x, x * 2)(5)")
```

---

setup_terminology *Download and parse the Microsoft Terminology Collection.*

Description

Source()s the parser script bundled in `inst/extdata/parse_locales.R`, then calls its `download_and_parse_tbx()` function to fetch the zip from Microsoft's public download URL, validate the SHA-256 (if you supply one), unzip, parse, and write the resulting translation table to a per-user cache directory.

Usage

```
setup_terminology(
  expected_sha256 = .MTC_EXPECTED_SHA256,
  force = FALSE,
  workers = max(1L, parallel::detectCores() - 1L),
  quiet = FALSE
)
```

Arguments

`expected_sha256`	SHA-256 hex digest expected for the downloaded zip. Defaults to the digest of the version of the zip known to this release of spaghetti. A mismatch produces a warning (not an error), since Microsoft may republish the file. Pass `NULL` to skip the check entirely.
`force`	If TRUE, re-download even if a cache exists.
`workers`	Number of parallel TBX-parsing workers. Default detects cores - 1, capped to 8.
`quiet`	If TRUE, suppress progress messages.

Details

Subsequent R sessions load the cache automatically on package attach.

Value

Invisibly, the path to the cached RDS.

Dependencies

The parser uses the `openxlsx2` package (for XML parsing) and `digest` (for SHA-256 verification). Both are declared in `Suggests:` and installed only if you call this function.

Licensing

Microsoft has not published an explicit license for the contents of the Terminology Collection. The data is downloaded directly from Microsoft; this package does not redistribute it.

See Also

`has_terminology()`, `clear_terminology()`, `terminology_info()`

Examples

```
## Not run:
setup_terminology()
# Skip verification entirely:
setup_terminology(expected_sha256 = NULL)

## End(Not run)
```

`supported_locales` *List supported locale codes.*

Description

Returns the locale columns present in the cached function table, i.e. the locales for which at least partial translation data was loaded. Returns `character(0)` if `setup_terminology()` hasn't been run.

Usage

```
supported_locales()
```

Value

Character vector of locale codes (e.g. `c("de", "fr", "es", ...)`).

See Also

`setup_terminology()`, `has_terminology()`

Examples

```
supported_locales()
```

terminology_info *Metadata about the currently loaded terminology cache.*

Description

Returns the provenance attributes that were attached to the cached RDS at download time: source URL, observed SHA-256, download timestamp, and the spaghetti version that produced the cache. Returns NULL if no terminology is currently loaded.

Usage

```
terminology_info()
```

Value

A named list, or NULL.

Examples

```
terminology_info()
```

to_xml *Convert a user-facing formula to OOXML storage format*

Description

Convert a user-facing formula to OOXML storage format

Usage

```
to_xml(formula, locale = NULL, warn_unknown = TRUE)
```

Arguments

formula	Character scalar or vector. Formula(s), with or without =.
locale	Two-letter locale code ("de", "fr", ...) or NULL. When set, localised function names are translated to English and the locale argument separator (; for many European locales) is accepted.
warn_unknown	Logical; warn for unknown function names (default TRUE).

Value

Character scalar or vector: OOXML formula(s) starting with =.

Examples

```
to_xml("=SEQUENCE(10)")
to_xml("=LAMBDA(temp, (5/9) * (temp-32))(100)")
to_xml("=FILTER(A1:A10, B1:B10 > 5)")
to_xml("=SUM(A1#)")
to_xml("=LET(tc,(B2-32)*5/9,rh,0.6,tc*ATAN(0.151977*(rh*100+8.313659)^0.5))")
## Not run: to_xml("=SUMMEWENN(A1:A10;\"x\";B1:B10)", locale = "de")
to_xml(c("=SUM(A1:A10)", "=SEQUENCE(5)", "=FILTER(A1:A10, B1:B10 > 0)"))
```

---

xlex *Tokenise and display a formula as an ASCII tree*

Description

Parses a formula using spaghetti's lexer and renders the token tree in the style of `tidyxl::xlex()`. Nesting follows the call structure: tokens inside a function's parentheses appear as children of that function node.

Usage

```
xlex(formula, locale = NULL, print = TRUE)
```

Arguments

`formula`	Character scalar. Formula with or without leading =.
`locale`	Two-letter locale code or NULL. Used to select the correct argument separator (; for German etc.) and to translate localised function names in the display label.
`print`	Logical. Print the tree to the console (default TRUE). Set FALSE to get the data frame silently.

Details

The formula is displayed as-is (no OOXML translation). Pass the result of `to_xml()` if you want to inspect the storage form.

Value

A data frame with columns `depth`, `val`, `label`, invisibly. `depth` is the nesting level (0 = root, 1 = top-level arguments, ...). Printed as a side-effect when `print = TRUE`.

Examples

```
xlex("=SUM(A1:A10)")
xlex("=IF(A1>0, VLOOKUP(A1, B:C, 2, 0), NA())")
xlex("=LAMBDA(x, x * 2)(5)")
xlex("=SUMMEWENNS(C2:C10; A2:A10; \"Berlin\")", locale = "de")
# Inspect OOXML form
xlex(to_xml("=FILTER(A1:A10, B1:B10 > 0)"))
```

Index