Package ‘wk’

March 16, 2021

Title Lightweight Well-Known Geometry Parsing
Version 0.4.1
Maintainer Dewey Dunnington <dewey@fishandwhistle.net>
Description Provides a minimal R and C++ API for parsing well-known binary and well-known text representation of geometries to and from R-native formats. Well-known binary is compact and fast to parse; well-known text is human-readable and is useful for writing tests. These formats are only useful in R if the information they contain can be accessed in R, for which high-performance functions are provided here.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
LinkingTo cpp11
SystemRequirements C++11
Suggests testthat (&gt;= 3.0.0), vctrs (&gt;= 0.3.0), wkutils, sf, tibble
URL https://paleolimbot.github.io/wk/,
https://github.com/paleolimbot/wk
BugReports https://github.com/paleolimbot/wk/issues
Config/testthat/edition 3
NeedsCompilation yes
Author Dewey Dunnington [aut, cre] (&lt;https://orcid.org/0000-0002-9415-4582&gt;),
Edzer Pebesma [aut] (&lt;https://orcid.org/0000-0001-8049-7069&gt;)
Repository CRAN
Date/Publication 2021-03-16 13:50:03 UTC
R topics documented:

```
crc ................................................................. 2
handle_wkt_without_vector_size ................................ 3
new_wk_crc ....................................................... 4
new_wk_rct ......................................................... 4
new_wk_wkb .......................................................... 5
new_wk_wkt .......................................................... 5
new_wk_xy ........................................................... 6
plot.wk_wkt ........................................................ 6
rct ................................................................. 8
vctrs-methods ....................................................... 9
wkb ................................................................. 10
wkb_format .......................................................... 11
wkt ................................................................. 12
wk_bbox ............................................................ 13
wk_count ........................................................... 13
wk_crs ............................................................. 14
wk_crs_equal ......................................................... 15
wk_crs_inherit ...................................................... 15
wk_debug ........................................................... 16
wk_format .......................................................... 17
wk_handle.data.frame ............................................. 18
wk_handle.sfg ...................................................... 19
wk_identity ........................................................ 20
wk_meta ........................................................... 21
wk_problems ......................................................... 23
wk_translate.sfc ................................................... 23
wk_void ............................................................ 24
wk_writer.sfc ...................................................... 25
xy ................................................................. 26
```

Index 28

```
crc 2D Circle Vectors
```

Description

2D Circle Vectors

Usage

```
crc(x = double(), y = double(), r = double(), crs = wk_crs_auto())

as_crc(x, ...)
```

## S3 method for class 'wk_crc'
**as_crc(x, ...)**

```r
## S3 method for class 'matrix'
as_crc(x, ..., crs = NULL)

## S3 method for class 'data.frame'
as_crc(x, ..., crs = NULL)
```

**Arguments**

- `x, y` Coordinates of the center
- `r` Circle radius
- `crs` A value to be propagated as the CRS for this vector.
- `...` Extra arguments passed to `as_crc()`.

**Value**

A vector along the recycled length of bounds.

**Examples**

```r
crc(1, 2, 3)
```

---

**handle_wkt_without_vector_size**

`Test handlers for handling of unknown size vectors`

**Description**

Test handlers for handling of unknown size vectors

**Usage**

```r
handle_wkt_without_vector_size(handleable, handler)
```

**Arguments**

- `handleable` A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.
- `handler` A `wk_handler` object.

**Examples**

```r
handle_wkt_without_vector_size(wkt(), wk_vector_meta_handler())
```
### new_wk_crc

S3 details for crc objects

**Description**

S3 details for crc objects

**Usage**

```r
new_wk_crc(x = list(x = double(), y = double(), r = double()), crs = NULL)
```

**Arguments**

- **x**: A `crc()`
- **crs**: A value to be propagated as the CRS for this vector.

### new_wk_rct

S3 details for rct objects

**Description**

S3 details for rct objects

**Usage**

```r
new_wk_rct(
  x = list(xmin = double(), ymin = double(), xmax = double(), ymax = double()),
  crs = NULL
)
```

**Arguments**

- **x**: A `rct()`
- **crs**: A value to be propagated as the CRS for this vector.
new_wk_wkb

S3 Details for wk_wkb

Description

S3 Details for wk_wkb

Usage

new_wk_wkb(x = list(), crs = NULL)
validate_wk_wkb(x)
is_wk_wkb(x)

Arguments

x A (possibly) wkb() vector

new_wk_wkt

S3 Details for wk_wkt

Description

S3 Details for wk_wkt

Usage

new_wk_wkt(x = character(), crs = NULL)
is_wk_wkt(x)
validate_wk_wkt(x)

Arguments

x A (possibly) wkt() vector
crs A value to be propagated as the CRS for this vector.
new_wk_xy  
S3 details for xy objects

Description
S3 details for xy objects

Usage
new_wk_xy(x = list(x = double(), y = double()), crs = NULL)
new_wk_xyz(x = list(x = double(), y = double(), z = double()), crs = NULL)
new_wk_xym(x = list(x = double(), y = double(), m = double()), crs = NULL)
new_wk_xyzm(
  x = list(x = double(), y = double(), z = double(), m = double()),
  crs = NULL
)
validate_wk_xy(x)
validate_wk_xyz(x)
validate_wk_xym(x)
validate_wk_xyzm(x)

Arguments

x  A xy() object.
crs  A value to be propagated as the CRS for this vector.

plot.wk_wkt  Plot well-known geometry vectors

Description
Plot well-known geometry vectors
plot.wk_wkt

Usage

## S3 method for class 'wk_wkt'
plot(
  x,
  ..., 
  asp = 1,
  bbox = NULL,
  xlab = "",
  ylab = "",
  rule = "evenodd",
  add = FALSE
)

## S3 method for class 'wk_wkb'
plot(
  x,
  ..., 
  asp = 1,
  bbox = NULL,
  xlab = "",
  ylab = "",
  rule = "evenodd",
  add = FALSE
)

## S3 method for class 'wk_xy'
plot(x, ..., asp = 1, bbox = NULL, xlab = ",", ylab = "", add = FALSE)

## S3 method for class 'wk_rct'
plot(x, ..., asp = 1, bbox = NULL, xlab = ",", ylab = "", add = FALSE)

## S3 method for class 'wk_crc'
plot(x, ..., asp = 1, bbox = NULL, xlab = ",", ylab = "", add = FALSE)

Arguments

x

A wk() or wkb() vector.

... Passed to plotting functions for features: graphics::points() for point and
multipoint geometries, graphics::lines() for linestring and multilinestring
geometries, and graphics::polypath() for polygon and multipolygon geometries.

asp Passed to graphics::plot()

bbox The limits of the plot in the form returned by wkt_ranges().

xlab Passed to graphics::plot()

ylab Passed to graphics::plot()

rule The rule to use for filling polygons (see graphics::polypath())

add Should a new plot be created, or should x be added to the existing plot?
Value
The input, invisibly.

Examples

```r
# requires the wkutils package
if (requireNamespace("wkutils")) {
  plot(as_wkt("LINESTRING (0 0, 1 1)"))
  plot(as_wkb("LINESTRING (0 0, 1 1)"))
}
```

---

rct 2D rectangle vectors

Description

2D rectangle vectors

Usage

```r
rct(
  xmin = double(),
  ymin = double(),
  xmax = double(),
  ymax = double(),
  crs = wk_crs_auto()
)
```

```r
as_rct(x, ...)
```

```r
## S3 method for class 'wk_rct'
as_rct(x, ...)
```

```r
## S3 method for class 'matrix'
as_rct(x, ..., crs = NULL)
```

```r
## S3 method for class 'data.frame'
as_rct(x, ..., crs = NULL)
```

Arguments

- `xmin`, `ymin`, `xmax`, `ymax`
  - Rectangle bounds.
- `crs`
  - A value to be propagated as the CRS for this vector.
- `x`
  - An object to be converted to a `rct()`.
- `...`
  - Extra arguments passed to `as_rct()`.
Value

A vector along the recycled length of bounds.

Examples

rct(1, 2, 3, 4)

<table>
<thead>
<tr>
<th>vctrs-methods</th>
<th>Vctrs methods</th>
</tr>
</thead>
</table>

Description

Vctrs methods

Usage

vec_cast.wk_wkb(x, to, ...)
vec_ptype2.wk_wkb(x, y, ...)
vec_cast.wk_wkt(x, to, ...)
vec_ptype2.wk_wkt(x, y, ...)
vec_cast.wk_xy(x, to, ...)
vec_ptype2.wk_xy(x, y, ...)
vec_cast.wk_xyz(x, to, ...)
vec_ptype2.wk_xyz(x, y, ...)
vec_cast.wk_xym(x, to, ...)
vec_ptype2.wk_xym(x, y, ...)
vec_cast.wk_xyzm(x, to, ...)
vec_ptype2.wk_xyzm(x, y, ...)
vec_cast.wk_rct(x, to, ...)
vec_ptype2.wk_rct(x, y, ...)
vec_cast.wk_crc(x, to, ...)
vec_ptype2.wk_crc(x, y, ...)
wkb

Mark lists of raw vectors as well-known binary

Description
Mark lists of raw vectors as well-known binary

Usage
wkb(x = list(), crs = wk_crs_auto())
parse_wkb(x, crs = wk_crs_auto())
wk_platform_endian()
as_wkb(x, ...)

## Default S3 method:
as_wkb(x, ...)

## S3 method for class 'character'
as_wkb(x, ..., crs = NULL)

## S3 method for class 'wk_wkb'
as_wkb(x, ...)

## S3 method for class 'blob'
as_wkb(x, ..., crs = NULL)

## S3 method for class 'WKB'
as_wkb(x, ..., crs = NULL)

Arguments

x A list() of raw() vectors or NULL.

Argument
to

... A value to be propagated as the CRS for this vector.

Value
A new_wk_wkb()
Examples

wkb(wkt_translate_wkb("POINT (20 10)"))

---

### Deprecated functions

**Description**

These functions are deprecated and will be removed in a future version.

**Usage**

- `wkb_format(wkb, max_coords = 3, precision = 6, trim = TRUE)`
- `wkt_format(wkt, max_coords = 3, precision = 6, trim = TRUE)`
- `wkb_problems(wkb)`
- `wkt_problems(wkt)`
- `wkb_translate_wkt(wkb, ..., precision = 16, trim = TRUE)`
- `wkb_translate_wkb(wkb, ...)`
- `wkt_translate_wkt(wkt, ..., precision = 16, trim = TRUE)`
- `wkt_translate_wkb(wkt, ...)`

**Arguments**

- **wkb**
  - A list() of raw() vectors, such as that returned by sf::st_as_binary().
- **max_coords**
  - The maximum number of coordinates to include in the output.
- **precision**
  - The rounding precision to use when writing (number of decimal places).
- **trim**
  - Trim unnecessary zeroes in the output?
- **wkt**
  - A character vector containing well-known text.
- **...**
  - Used to keep backward compatibility with previous versions of these functions.
Mark character vectors as well-known text

Description

Mark character vectors as well-known text

Usage

wkt(x = character(), crs = wk_crs_auto())
parse_wkt(x, crs = wk_crs_auto())
as_wkt(x, ...)

## Default S3 method:
as_wkt(x, ...)

## S3 method for class 'character'
as_wkt(x, ..., crs = NULL)

## S3 method for class 'wk_wkt'
as_wkt(x, ...)

Arguments

x A character() vector containing well-known text.
crs A value to be propagated as the CRS for this vector.
... Unused

Value

A new_wk_wkt()

Examples

wkt("POINT (20 10)")
Description
2D bounding rectangles

Usage
wk_bbox(handleable, ...)

## Default S3 method:
wk_bbox(handleable, ...)

wk_bbox_handler()

Arguments
handleable A geometry vector (e.g., wkb(), wkt(), xy(), rct(), or sf::st_sfc()) for which wk_handle() is defined.
...
Passed to the wk_handle() method.

Value
A rct() of length 1.

Examples
wk_bbox(wkt("LINESTRING (1 2, 3 5)"))

---

wk_count

Count geometry components

Description
Counts the number of geometries, rings, and coordinates found within each feature. As opposed to wk_meta(), this handler will iterate over the entire geometry.

Usage
wk_count(handleable, ...)

## Default S3 method:
wk_count(handleable, ...)

wk_count_handler()
Arguments

handleable  A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.

...  Passed to the `wk_handle()` method.

Value

A data.frame with one row for every feature encountered and columns:

- `n_geom`: The number of geometries encountered, including the root geometry. Will be zero for a null feature.
- `n_ring`: The number of rings encountered. Will be zero for a null feature.
- `n_coord`: The number of coordinates encountered. Will be zero for a null feature.

Examples

```r
wk_count(as_wkt("LINESTRING (0 0, 1 1)"))
wk_count(as_wkb("LINESTRING (0 0, 1 1)"))
```

---

`wk_crs`  
*Set and get vector CRS*

Description

The `wk` package doesn’t operate on CRS objects, but does propagate them through subsetting and concatenation. A CRS object can be any R object, and `x` can be any object whose `crs` attribute carries a CRS. These functions are S3 generics to keep them from being used on objects that do not use this system of CRS propagation.

Usage

```r
wk_crs(x)

## S3 method for class 'wk_vctr'
wk_crs(x)

## S3 method for class 'wk_rcrd'
wk_crs(x)

wk_crs(x) <- value

wk_set_crs(x, crs)

wk_crs_output(x, y)
```
**Arguments**

- **x, y**
  - An objects whose "crs" attribute is used to carry a CRS.
- **crs, value**
  - An object that can be interpreted as a CRS

---

**wk_crs_equal**

*Compare CRS objects*

**Description**

The `wk_crs_equal()` function uses special S3 dispatch on `wk_crs_equal_generic()` to evaluate whether or not two CRS values can be considered equal. When implementing `wk_crs_equal_generic()`, every attempt should be made to make `wk_crs_equal(x, y)` and `wk_crs_equal(y, x)` return identically.

**Usage**

```r
wk_crs_equal(x, y)
```

```r
wk_crs_equal_generic(x, y, ...)
```

**Arguments**

- **x, y**
  - Objects stored in the `crs` attribute of a vector.
- **...**
  - Unused

**Value**

- `TRUE` if `x` and `y` can be considered equal, `FALSE` otherwise.

---

**wk_crs_inherit**

*Special CRS values*

**Description**

The CRS handling in the wk package requires two sentinel CRS values. The first, `wk_crs_inherit()`, signals that the vector should inherit a CRS of another vector if combined. This is useful for empty, `NULL`, and/or zero-length geometries. The second, `wk_crs_auto()`, is used as the default argument of `crs` for constructors so that zero-length geometries are assigned a CRS of `wk_crs_inherit()` by default.

**Usage**

```r
wk_crs_inherit()
```

```r
wk_crs_auto()
```

```r
wk_crs_auto_value(x, crs)
```
Arguments

**x**
A raw input to a constructor whose length and crs attribute is used to determine the default CRS returned by `wk_crs_auto()`.

**crs**
A value for the coordinate reference system supplied by the user.

Examples

```r
wk_crs_auto_value(list(), wk_crs_auto())
wk_crs_auto_value(list(), 1234)
wk_crs_auto_value(list(NULL), wk_crs_auto())
```

---

**wk_debug**

*Debug filters and handlers*

Description

Debug filters and handlers

Usage

```r
wk_debug(handleable, handler = wk_void_handler(), ...)
wk_debug_filter(handler = wk_void_handler())
```

Arguments

**handleable**
A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.

**handler**
A `wk_handler` object.

**...**
Passed to the `wk_handle()` method.

Value

The result of the handler.

Examples

```r
wk_debug(wkt("POINT (1 1)"))
wk_handle(wkt("POINT (1 1)") , wk_debug_filter())
```
**Description**

Provides an abbreviated version of the well-known text representation of a geometry. This returns a constant number of coordinates for each geometry, so is safe to use for geometry vectors with many (potentially large) features. Parse errors are passed on to the format string and do not cause this handler to error.

**Usage**

```r
wk_format(handleable, precision = 7, trim = TRUE, max_coords = 6, ...)
```

```r
wkt_format_handler(precision = 7, trim = TRUE, max_coords = 6)
```

**Arguments**

- `handleable`: A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.
- `precision`: If `trim` is `TRUE`, the total number of significant digits to keep for each result or the number of digits after the decimal place otherwise.
- `trim`: Use `FALSE` to keep trailing zeroes after the decimal place.
- `max_coords`: The maximum number of coordinates to include in the output.
- `...`: Passed to the `wk_handle()` method.

**Value**

A character vector of abbreviated well-known text.

**Examples**

```r
wk_format(wkt("MULTIPOLYGON (((0 0, 10 0, 0 10, 0 0)))"))
wkt_format(new_wk_wkt("POINT ENTPY"))
wk_handle(
    wkt("MULTIPOLYGON (((0 0, 10 0, 0 10, 0 0)))"),
    wkt_format_handler()
)
Use data.frame with wk

Description

Use data.frame with wk

Usage

## S3 method for class 'data.frame'
wk_handle(handleable, handler, ..., .env = parent.frame())

## S3 method for class 'data.frame'
wk_writer(handleable, ...)

## S3 method for class 'data.frame'
wk_crs(x)

## S3 method for class 'data.frame'
wk_set_crs(x, crs)

## S3 method for class 'data.frame'
wk_restore(handleable, result, ...)

## S3 method for class 'tbl_df'
wk_restore(handleable, result, ...)

## S3 method for class 'data.frame'
wk_translate(handleable, to, ..., .env = parent.frame())

## S3 method for class 'tbl_df'
wk_translate(handleable, to, ..., .env = parent.frame())

## S3 method for class 'sf'
wk_translate(handleable, to, ...)

## S3 method for class 'sf'
wk_restore(handleable, result, ...)

Arguments

handleable    A geometry vector (e.g., \texttt{wkb()}, \texttt{wkt()}, \texttt{xy()}, \texttt{rct()}, or \texttt{sf::st_sfc()}) for which \texttt{wk_handle()} is defined.

handler       A \texttt{wk_handler} object.

...           Passed to the \texttt{wk_handle()} method.
wk_handle.sfg

Passed to `getS3method()`, which is used to find the column in a `data.frame()` for which a `wk_handle()` method is defined.

An objects whose "crs" attribute is used to carry a CRS.

An object that can be interpreted as a CRS

The result of a filter operation intended to be a transformation.

A prototype object.

Examples

```r
wk_handle(data.frame(a = wkt("POINT (0 1)")), wkb_writer())
wk_translate(wkt("POINT (0 1)"), data.frame(col_name = wkb()))
wk_translate(data.frame(a = wkt("POINT (0 1)")), data.frame(wkb()))
```

---

**Description**

The handler is the basic building block of the wk package. In particular, the `wk_handle()` generic allows operations written as handlers to "just work" with many different input types. The wk package provides the `wk_void()` handler, the `wk_format()` handler, the `wk_debug()` handler, the `wk_problems()` handler, and `wk_writer()` for `wkb()`, `wkt()`, `xy()`, and `sf::st_sfc()` vectors.

**Usage**

```r
## S3 method for class 'sfg'
wk_handle(handleable, handler, ...)

## S3 method for class 'sf'
wk_handle(handleable, handler, ...)

## S3 method for class 'bbox'
wk_handle(handleable, handler, ...)

## S3 method for class 'wk_crc'
wk_handle(
  handleable,
  handler,
  ...
  n_segments = getOption("wk.crc_n_segments", NULL),
  resolution = getOption("wk.crc_resolution", NULL)
)

## S3 method for class 'wk_rct'
wk_handle(handleable, handler, ...)
```
## S3 method for class 'sfc'
wk_handle(handleable, handler, ...)

## S3 method for class 'wk_wkb'
wk_handle(handleable, handler, ...)

## S3 method for class 'wk_wkt'
wk_handle(handleable, handler, ...)

## S3 method for class 'wk_xy'
wk_handle(handleable, handler, ...)

wk_handle(handleable, handler, ...)

new_wk_handler(handler_ptr, subclass = character())

is_wk_handler(handler)

as_wk_handler(handler, ...)

**Arguments**

- **handleable**: A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.
- **handler**: A `wk_handler` object.
- **...**: Passed to the `wk_handle()` method.
- **n_segments, resolution**: The number of segments to use when approximating a circle. The default uses `getOption("wk.crc_n_segments")` so that this value can be set for implicit conversions (e.g., `as_wkb()`). Alternatively, set the minimum distance between points on the circle (used to estimate `n_segments`). The default is obtained using `getOption("wk.crc_resolution")`.
- **handler_ptr**: An external pointer to a newly created WK handler
- **subclass**: The handler subclass

**Value**

A WK handler.

---

### wk_identity

*Copy a geometry vector*

**Description**

Copy a geometry vector
Usage

wk_identity(handleable, ...)
wk_identity_filter(handler)
wk_restore(handleable, result, ...)

## Default S3 method:
wk_restore(handleable, result, ...)

Arguments

handleable A geometry vector (e.g., wkb(), wkt(), xy(), rct(), or sf::st_sfc()) for which wk_handle() is defined.
...
Passed to the wk_handle() method.
handler A wk_handler object.
result The result of a filter operation intended to be a transformation.

Value

A copy of handleable.

Examples

wk_identity(wkt("POINT (1 2)"))

---

wk_meta Extract feature-level meta

Description

These functions return the non-coordinate information of a geometry and/or vector. They do not parse an entire geometry/vector and are intended to be very fast even for large vectors.

Usage

wk_meta(handleable, ...)

## Default S3 method:
wk_meta(handleable, ...)

wk_vector_meta(handleable, ...)

## Default S3 method:
wk_vector_meta(handleable, ...)

wk_meta_handler()

wk_vector_meta_handler()

wk_geometry_type_label(geometry_type)

wk_geometry_type(geometry_type_label)

Arguments

handleable A geometry vector (e.g., wkb(), wkt(), xy(), rct(), or sf::st_sfc()) for which wk_handle() is defined.

... Passed to the wk_handle() method.

geometry_type An integer code for the geometry type. These integers follow the WKB specification (e.g., 1 for point, 7 for geometrycollection).

geometry_type_label A character vector of (lowercase) geometry type labels as would be found in WKT (e.g., point, geometrycollection).

Value

A data.frame with columns:

- geometry_type: An integer identifying the geometry type. A value of 0 indicates that the types of geometry in the vector are not known without parsing the entire vector.
- size: For points and linestrings, the number of coordinates; for polygons, the number of rings; for collections, the number of child geometries. A value of zero indicates an EMPTY geometry. A value of NA means this value is unknown without parsing the entire geometry.
- has_z: TRUE if coordinates contain a Z value. A value of NA means this value is unknown without parsing the entire vector.
- has_m: TRUE if coordinates contain an M value. A value of NA means this value is unknown without parsing the entire vector.
- srid: An integer identifying a CRS or NA if this value was not provided.
- precision: A grid size or 0.0 if a grid size was not provided. Note that coordinate values may not have been rounded; the grid size only refers to the level of detail with which they should be interpreted.

Examples

wk_vector_meta(as_wkt("LINESTRING (0 0, 1 1)"))
wk_meta(as_wkt("LINESTRING (0 0, 1 1)"))
wk_meta(as_wkb("LINESTRING (0 0, 1 1)"))

wk_geometry_type_label(1:7)
wk_geometry_type(c("point", "geometrycollection"))
**Description**

The problems handler returns a character vector of parse errors and can be used to validate input of any type for which `wk_handle()` is defined.

**Usage**

```r
wk_problems(handleable, ...)

wk_problems_handler()
```

**Arguments**

- `handleable`: A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.
- `...`: Passed to the `wk_handle()` method.

**Value**

A character vector of parsing errors. `NA` signifies that there was no parsing error.

**Examples**

```r
wk_problems(new_wk_wkt(c("POINT EMTPY", "POINT (20 30)")))
wk_handle(
  new_wk_wkt(c("POINT EMTPY", "POINT (20 30)")),
  wk_problems_handler()
)
```

---

**wk_translate.sfc**

*Translate geometry vectors*

**Description**

Translate geometry vectors
Usage

```r
## S3 method for class 'sfc'
wk_translate(handleable, to, ...)

wk_translate(handleable, to, ...)

## Default S3 method:
wk_translate(handleable, to, ...)
```

Arguments

- `handleable`: A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.
- `to`: A prototype object.
- `...`: Passed to the `wk_handle()` method.

Value

NULL

Examples

```r
wk_void(wkt("POINT (1 4)"))
wk_handle(wkt("POINT (1 4)"), wk_void_handler())
```
**wk_writer.sfc**

*Write geometry vectors*

---

**Description**

Write geometry vectors

**Usage**

```r
## S3 method for class 'sfc'
wk_writer(handleable, ...)

## S3 method for class 'sf'
wk_writer(handleable, ...)

sfc_writer()

wkb_writer(buffer_size = 2048L, endian = NA_integer_)

wkf_writer(precision = 16L, trim = TRUE)

wk_writer(handleable, ...)

## S3 method for class 'wk_wkt'
wk_writer(handleable, ..., precision = 16, trim = TRUE)

## S3 method for class 'wk_wkb'
wk_writer(handleable, ...)

## S3 method for class 'wk_xy'
wk_writer(handleable, ...)

xy_writer()
```

**Arguments**

- `handleable`: A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.
- `...`: Passed to the writer constructor.
- `buffer_size`: Control the initial buffer size used when writing WKB.
- `endian`: Use 1 for little endian, 0 for big endian, or NA for system endian.
- `precision`: If `trim` is TRUE, the total number of significant digits to keep for each result or the number of digits after the decimal place otherwise.
- `trim`: Use FALSE to keep trailing zeroes after the decimal place.
Value

A wk_handler.

Description

Efficient point vectors

Usage

```r
efficient_point_vectors
```
... Passed to methods.

dims A set containing one or more of c("x","y","z","m").

Value

A vector of coordinate values.

Examples

xy(1:5, 1:5)
xyz(1:5, 1:5, 10)
xym(1:5, 1:5, 10)
xyzm(1:5, 1:5, 10, 12)
Index

as_crc (crc), 2
as_rct (rct), 8
as_wk_handler (wk_handler.sfg), 19
as_wkb (wkb), 10
as_wkt (wkt), 12
as_xy (xy), 26

character(), 12
crc, 2
crc(), 4
data.frame(), 19
getS3method(), 19
graphics::lines(), 7
graphics::plot(), 7
graphics::points(), 7
graphics::polyath(), 7

handle_wkt_without_vector_size, 3

is_wk_handler (wk_handler.sfg), 19
is_wk_wkb (new_wk_wkb), 5
is_wk_wkt (new_wk_wkt), 5

list(), 10

new_wk_crc, 4
new_wk_handler (wk_handler.sfg), 19
new_wk_rct, 4
new_wk_wkb, 5
new_wk_wkb(), 10
new_wk_wkt, 5
new_wk_wkt(), 12
new_wk_xy, 6
new_wk_xym (new_wk_xy), 6
new_wk_xyz (new_wk_xy), 6
new_wk_xyzm (new_wk_xy), 6

parse_wkb (wkb), 10
parse_wkt (wkt), 12

plot.wk_crc (plot.wk_wkt), 6
plot.wk_rct (plot.wk_wkt), 6
plot.wk_wkb (plot.wk_wkt), 6
plot.wk_wkt, 6
plot.wk_xy (plot.wk_wkt), 6

raw(), 10, 11
rct, 8
rct(), 3, 4, 8, 13, 14, 16–18, 20–25

sf::st_sfc(), 3, 13, 14, 16–25
sfc_writer (wk_writer.sfc), 25

validate_wk_wkb (new_wk_wkb), 5
validate_wk_wkt (new_wk_wkt), 5
validate_wk_xy (new_wk_xy), 6
validate_wk_xym (new_wk_xy), 6
validate_wk_xyz (new_wk_xy), 6
validate_wk_xyzm (new_wk_xy), 6

vctrs::vec_cast(), 10
vctrs::vec_ptype2(), 10
vec_cast.wk_crc (vctrs-methods), 9
vec_cast.wk_rct (vctrs-methods), 9
vec_cast.wk_wkb (vctrs-methods), 9
vec_cast.wk_wkt (vctrs-methods), 9
vec_cast.wk_xy (vctrs-methods), 9
vec_cast.wk_xym (vctrs-methods), 9
vec_cast.wk_xyz (vctrs-methods), 9
vec_cast.wk_xyzm (vctrs-methods), 9

wk_bbox, 13