Package 'wk'

January 3, 2022

Title Lightweight Well-Known Geometry Parsing

Version 0.6.0

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

RoxygenNote 7.1.2

SystemRequirements C++11

Suggests testthat (>= 3.0.0), vctrs (>= 0.3.0), sf, tibble, readr

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] (<<https://orcid.org/0000-0002-9415-4582>>),
Edzer Pebesma [aut] (<<https://orcid.org/0000-0001-8049-7069>>)

Repository CRAN

Date/Publication 2022-01-03 13:20:02 UTC

R **topics documented:**

---

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, ...)

## 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

```
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

```
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

```
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

```
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

```
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, geodesic = NULL)

validate_wk_wkb(x)

is_wk_wkb(x)
```

Arguments

x   A (possibly) [wkb()](#) vector

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

geodesic   TRUE if edges must be interpolated as geodesics when coordinates are spherical,
                FALSE otherwise.

---

new_wk_wkt   *S3 Details for wk_wkt*

Description

S3 Details for wk_wkt

Usage

```
new_wk_wkt(x = character(), crs = NULL, geodesic = 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.
geodesic	TRUE if edges must be interpolated as geodesics when coordinates are spherical, FALSE otherwise.

---

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.

---

rct	*2D rectangle vectors*

Description

2D rectangle vectors

Usage

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

as_rct(x, ...)

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

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

## 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)
```

---

vctrs-methods   *Vctrs methods*

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, ...)
```

Arguments

x, y, to, ...   See `vctrs::vec_cast()` and `vctrs::vec_ptype2()`.

---

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(), geodesic = FALSE)

parse_wkb(x, crs = wk_crs_auto(), geodesic = FALSE)

wk_platform_endian()

as_wkb(x, ...)

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

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

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

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

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

Arguments

x	A `list()` of `raw()` vectors or NULL.
crs	A value to be propagated as the CRS for this vector.
geodesic	TRUE if edges must be interpolated as geodesics when coordinates are spherical, FALSE otherwise.
...	Unused

Value

A `new_wk_wkb()`

Examples

```
as_wkb("POINT (20 10)")
```

---

wkb_translate_wkt   *Deprecated functions*

Description

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

Usage

```
wkb_translate_wkt(wkb, ..., precision = 16, trim = TRUE)

wkb_translate_wkb(wkb, ..., endian = NA_integer_)

wkt_translate_wkt(wkt, ..., precision = 16, trim = TRUE)

wkt_translate_wkb(wkt, ..., endian = NA_integer_)
```

Arguments

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

---

wkt   *Mark character vectors as well-known text*

Description

Mark character vectors as well-known text

Usage

```
wkt(x = character(), crs = wk_crs_auto(), geodesic = FALSE)

parse_wkt(x, crs = wk_crs_auto(), geodesic = FALSE)

as_wkt(x, ...)

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

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

## 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.
geodesic	TRUE if edges must be interpolated as geodesics when coordinates are spherical, FALSE otherwise.
...	Unused

Value

A [new_wk_wkt()](#)

Examples

```
wkt("POINT (20 10)")
```

---

wk_bbox	*2D bounding rectangles*

Description

2D bounding rectangles

Usage

```
wk_bbox(handleable, ...)

wk_envelope(handleable, ...)
```

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

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

## S3 method for class 'wk_rct'
wk_envelope(handleable, ...)

## S3 method for class 'wk_crc'
wk_envelope(handleable, ...)

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

wk_bbox_handler()

wk_envelope_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_chunk_map_feature   *Operate on handleables by chunk*

Description

It is often impractical, inefficient, or impossible to perform an operation on a vector of geometries with all the geometries loaded into memory at the same time. These functions generalize the pattern of split-apply-combine to one or more handlers recycled along a common length. These functions are designed for developers rather than users and should be considered experimental.

Usage

```
wk_chunk_map_feature(
  handleables,
  fun,
  vector_args = NULL,
  args = NULL,
  input_handler_factory = wk_writer,
  output_template = NULL,
  strategy = wk_chunk_strategy_feature(chunk_size = 10000)
)
```

Arguments

handleables	A single handleable or a list() of handleables recycleable along a common length.
fun	A function called like fun(!!! transformed_handleables,!!! vector_args,!!! args) for each chunk. For wk_chunk_map_feature() this must be length-stable (i.e., return a value whose size is the recycled length of handleables and vector_args for that chunk).
vector_args	Vectorized arguments to fun.
args	Non-vectorized arguments to fun.
input_handler_factory	
	A function of handleable applied to handleable inputs. The default, wk_writer(), will result in fun getting called with a clone of the handleables for each chunk. Another useful pattern is to return a single type of handler so that all handleables have a common type.
output_template	
	A vector whose subset-assign method will get called for every chunk or NULL to ignore the output of fun.
strategy	A function of handleables and n_features such as that returned by wk_chunk_strategy_feature().

Value

output_template of the recycled common length of handleables and vector_args filled with values generated by fun.

Examples

```
# apply a transformation or calculate a value using the data frame version
# of the geometries (but without resolving all of them at once)
wk_chunk_map_feature(
  wk_linestring(xy(1:10, 1:10), rep(1:5, each = 2)),
  function(features) {
    coords <- wk_coords(features)
    vapply(split(coords, coords$feature_id), nrow, integer(1))
  },
  output_template = integer()
)
```

---

wk_chunk_strategy_single

Chunking strategies

Description

Chunking strategies

Usage

```
wk_chunk_strategy_single()

wk_chunk_strategy_feature(n_chunks = NULL, chunk_size = NULL)

wk_chunk_strategy_coordinates(n_chunks = NULL, chunk_size = NULL, reduce = "*")
```

Arguments

n_chunks, chunk_size

> Exactly one of the number of chunks or the chunk size. For wk_chunk_strategy_feature()
> the chunk size refers to the number of features; for wk_chunk_strategy_coordinates()
> this refers to the number of coordinates as calculated from multiple handleables
> using reduce.

reduce

> For wk_chunk_strategy_coordinates() this refers to the function used with
> Reduce() to combine coordinate counts from more than one handleable.

Value

A function that returns a data.frame with columns from and to when called with a handleable
and the feature count.

Examples

```
feat <- c(as_wkt(xy(1:4, 1:4)), wkt("LINESTRING (1 1, 2 2)"))
wk_chunk_strategy_single()(list(feat), 5)
wk_chunk_strategy_feature(chunk_size = 2)(list(feat), 5)
wk_chunk_strategy_coordinates(chunk_size = 2)(list(feat), 5)
```

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

```
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

```
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(...)

wk_is_geodesic_output(...)
```

Arguments

x, ...	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

```
wk_crs_equal(x, y)

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

```
wk_crs_inherit()

wk_crs_longlat(crs = NULL)

wk_crs_auto()

wk_crs_auto_value(x, crs)
```

Arguments

crs	A value for the coordinate reference system supplied by the user.
x	A raw input to a construuctor whose length and crs attributte is used to determine the default CRS returned by `wk_crs_auto()`.

Examples

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

---

wk_crs_proj_definition

CRS object generic methods

Description

CRS object generic methods

Usage

```
wk_crs_proj_definition(crs, proj_version = NULL, verbose = FALSE)

## S3 method for class '`NULL`'
wk_crs_proj_definition(crs, proj_version = NULL, verbose = FALSE)

## S3 method for class 'character'
wk_crs_proj_definition(crs, proj_version = NULL, verbose = FALSE)

## S3 method for class 'double'
wk_crs_proj_definition(crs, proj_version = NULL, verbose = FALSE)

## S3 method for class 'integer'
wk_crs_proj_definition(crs, proj_version = NULL, verbose = FALSE)
```

Arguments

crs	An arbitrary R object
proj_version	A [package_version()](#) of the PROJ version, or NULL if the PROJ version is unknown.
verbose	Use TRUE to request a more verbose version of the PROJ definition (e.g., WKT2). The default of FALSE should return the most compact version that completely describes the CRS. An authority:code string (e.g., "OGC:CRS84") is the recommended way to represent a CRS when verbose is FALSE, if possible, falling back to the most recent version of WKT2.

Value

- wk_crs_proj_definition() Returns a string used to represent the CRS in PROJ. For recent PROJ version you'll want to return WKT2; however you should check proj_version if you want this to work with older versions of PROJ.

Examples

```
wk_crs_proj_definition("EPSG:4326")
```

---

wk_debug   *Debug filters and handlers*

Description

Debug filters and handlers

Usage

```
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

```
wk_debug(wkt("POINT (1 1)"))
wk_handle(wkt("POINT (1 1)"), wk_debug_filter())
```

---

wk_flatten   *Extract simple geometries*

Description

Extract simple geometries

Usage

```
wk_flatten(handleable, ..., max_depth = 1)

wk_flatten_filter(handler, max_depth = 1L, add_details = FALSE)
```

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.
max_depth	The maximum (outer) depth to remove.
handler	A wk_handler object.
add_details	Use TRUE to add a "wk_details" attribute, which contains columns feature_id, part_id, and ring_id.

Value

handleable transformed such that collections have been expanded and only simple geometries (point, linestring, polygon) remain.

Examples

```
wk_flatten(wkt("MULTIPOINT (1 1, 2 2, 3 3)"))
wk_flatten(
  wkt("GEOMETRYCOLLECTION (GEOMETRYCOLLECTION (GEOMETRYCOLLECTION (POINT (0 1))))"),
  max_depth = 2
)
```

---

wk_format   *Format well-known geometry for printing*

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

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

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

```
wk_format(wkt("MULTIPOLYGON (((0 0, 10 0, 0 10, 0 0)))"))
wk_format(new_wk_wkt("POINT ENTPY"))
wk_handle(
  wkt("MULTIPOLYGON (((0 0, 10 0, 0 10, 0 0)))"),
  wkt_format_handler()
)
```

---

wk_handle.data.frame   *Use data.frame with wk*

Description

Use data.frame with wk

Usage

```
## S3 method for class 'data.frame'
wk_handle(handleable, handler, ...)

## 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_is_geodesic(x)

## S3 method for class 'data.frame'
wk_set_geodesic(x, geodesic)

## 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, ...)

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

## 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., 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.
x	Objects whose "crs" attribute is used to carry a CRS.
crs	An object that can be interpreted as a CRS
geodesic	TRUE if edges must be interpolated as geodesics when coordinates are spherical, FALSE otherwise.
result	The result of a filter operation intended to be a transformation.
to	A prototype object.

Examples

```
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()))
```

---

wk_handle.wk_crc   *Read geometry vectors*

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()s for wkb(), wkt(), xy(), and sf::st_sfc()) vectors.

Usage

```
## 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, ...)

is_handleable(handleable)

new_wk_handler(handler_ptr, subclass = character())

is_wk_handler(handler)

as_wk_handler(handler, ...)

## 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, ...)
```

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()](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_handle_slice.data.frame

Handle specific regions of objects

Description

Handle specific regions of objects

Usage

```
## S3 method for class 'data.frame'
wk_handle_slice(handleable, handler, from = NULL, to = NULL, ...)

wk_handle_slice(
  handleable,
  handler = wk_writer(handleable),
  from = NULL,
  to = NULL,
  ...
)

## Default S3 method:
wk_handle_slice(
  handleable,
  handler = wk_writer(handleable),
  from = NULL,
  to = NULL,
  ...
)
```

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.
from	1-based index of the feature to start from
to	1-based index of the feature to end at
...	Passed to the wk_handle() method.

Value

A subset of `handleable`

Examples

```
wk_handle_slice(xy(1:5, 1:5), wkt_writer(), from = 3, to = 5)
wk_handle_slice(
  data.frame(let = letters[1:5], geom = xy(1:5, 1:5)),
  wkt_writer(),
  from = 3, to = 5
)
```

---

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_is_geodesic`   *Set and get vector geodesic edge interpolation*

Description

Set and get vector geodesic edge interpolation

Usage

```
wk_is_geodesic(x)

wk_set_geodesic(x, geodesic)

wk_is_geodesic(x) <- value

wk_geodesic_inherit()
```

Arguments

x   An R object that contains edges

geodesic, value

                    TRUE if edges must be interpolated as geodesics when coordinates are spherical,
                    FALSE otherwise.

Value

TRUE if edges must be interpolated as geodesics when coordinates are spherical, FALSE otherwise.

---

wk_linestring *Create lines, polygons, and collections*

Description

Create lines, polygons, and collections

Usage

```
wk_linestring(handleable, feature_id = 1L, ..., geodesic = NULL)

wk_polygon(handleable, feature_id = 1L, ring_id = 1L, ..., geodesic = NULL)

wk_collection(
  handleable,
  geometry_type = wk_geometry_type("geometrycollection"),
  feature_id = 1L,
  ...
)

wk_linestring_filter(handler, feature_id = 1L)

wk_polygon_filter(handler, feature_id = 1L, ring_id = 1L)

wk_collection_filter(
  handler,
  geometry_type = wk_geometry_type("geometrycollection"),
  feature_id = 1L
)
```

Arguments

handleable	A geometry vector (e.g., wkb(), wkt(), xy(), rct(), or sf::st_sfc()) for which wk_handle() is defined.
feature_id	An identifier where changes in sequential values indicate a new feature. This is recycled silently as needed.
...	Passed to the wk_handle() method.
geodesic	Use TRUE or FALSE to explicitly force the geodesic-ness of the output.
ring_id	An identifier where changes in sequential values indicate a new ring. Rings are automatically closed. This is recycled silently as needed.
geometry_type	The collection type to create.
handler	A wk_handler object.

Value

An object of the same class as handleable with whose coordinates have been assembled into the given type.

Examples

```
wk_linestring(xy(c(1, 1), c(2, 3)))
wk_polygon(xy(c(0, 1, 0), c(0, 0, 1)))
wk_collection(xy(c(1, 1), c(2, 3)))
```

---

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"))
```

---

wk_plot   *Plot well-known geometry vectors*

Description

Plot well-known geometry vectors

Usage

```
wk_plot(
  handleable,
  ...,
  asp = 1,
  bbox = NULL,
  xlab = "",
  ylab = "",
  rule = "evenodd",
  add = FALSE
)
```

```
## Default S3 method:
wk_plot(
  handleable,
  ...,
  asp = 1,
  bbox = NULL,
  xlab = "",
  ylab = "",
  rule = "evenodd",
  add = FALSE
)

## 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

`handleable`	A geometry vector (e.g., `wkb()`, `wkt()`, `xy()`, `rct()`, or `sf::st_sfc()`) for which `wk_handle()` is defined.
`...`	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, xlab, ylab`	Passed to `graphics::plot()`
`bbox`	The limits of the plot as a `rct()` or compatible object
`rule`	The rule to use for filling polygons (see `graphics::polypath()`)
`add`	Should a new plot be created, or should `handleable` be added to the existing plot?
`x`	A `wkb()` or `wkt()`

Value

The input, invisibly.

Examples

```
plot(as_wkt("LINESTRING (0 0, 1 1)"))
plot(as_wkb("LINESTRING (0 0, 1 1)"))
```

`wk_problems`	*Validate well-known binary and well-known text*

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

```
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

```
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_set_z   *Set coordinate values*

Description

Set coordinate values

Usage

```
wk_set_z(handleable, z, ...)

wk_set_m(handleable, m, ...)

wk_drop_z(handleable, ...)

wk_drop_m(handleable, ...)

wk_trans_set(value, use_z = NA, use_m = NA)
```

Arguments

handleable	A geometry vector (e.g., wkb(), wkt(), xy(), rct(), or sf::st_sfc()) for which wk_handle() is defined.
z, m	A vector of Z or M values applied feature-wise and recycled along handleable. Use NA to keep the existing value of a given feature.
...	Passed to the wk_handle() method.
value	An xy(), xyz(), xym(), or xyzm() of coordinates used to replace values in the input. Use NA to keep the existing value.
use_z, use_m	Used to declare the output type. Use TRUE to ensure the output has that dimension, FALSE to ensure it does not, and NA to leave the dimension unchanged.

Examples

```
wk_set_z(wkt("POINT (0 1)"), 2)
wk_set_m(wkt("POINT (0 1)"), 2)
wk_drop_z(wkt("POINT ZM (0 1 2 3)"))
wk_drop_m(wkt("POINT ZM (0 1 2 3)"))
```

---

wk_transform   *Apply coordinate transformations*

Description

Apply coordinate transformations

Usage

```
wk_transform(handleable, trans, ...)

wk_transform_filter(handler, trans)
```

Arguments

handleable	A geometry vector (e.g., wkb(), wkt(), xy(), rct(), or sf::st_sfc()) for which wk_handle() is defined.
trans	An external pointer to a wk_trans object
...	Passed to the wk_handle() method.
handler	A wk_handler object.

Examples

```
wk_transform(xy(0, 0), wk_affine_translate(2, 3))
```

---

wk_translate.sfc   *Translate geometry vectors*

Description

Translate geometry vectors

Usage

```
## 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.

---

wk_trans_affine   *Affine transformer*

Description

Affine transformer

Usage

```
wk_trans_affine(trans_matrix)

wk_affine_identity()

wk_affine_rotate(rotation_deg)

wk_affine_scale(scale_x = 1, scale_y = 1)

wk_affine_translate(dx = 0, dy = 0)

wk_affine_fit(src, dst)

wk_affine_rescale(rct_in, rct_out)

wk_affine_compose(...)

wk_affine_invert(x)
```

Arguments

`trans_matrix`	A 3x3 transformation matrix
`rotation_deg`	A rotation to apply in degrees counterclockwise.
`scale_x, scale_y`	
	Scale factor to apply in the x and y directions, respectively
`dx, dy`	Coordinate offsets in the x and y direction
`src, dst`	Point vectors of control points used to estimate the affine mapping (using `base::qr.solve()`).
`rct_in, rct_out`	
	The input and output bounds
`...`	Zero or more transforms in the order they should be applied.
`x`	A `wk_trans_affine()`

---

wk_trans_inverse *Generic transform class*

Description

Generic transform class

Usage

```
wk_trans_inverse(trans, ...)

as_wk_trans(x, ...)

## S3 method for class 'wk_trans'
as_wk_trans(x, ...)

new_wk_trans(trans_ptr, subclass = character())
```

Arguments

`trans`	An external pointer to a wk_trans object
`...`	Passed to S3 methods
`x`	An object to be converted to a transform.
`trans_ptr`	An external pointer to a wk_trans_t transform struct.
`subclass`	An optional subclass to apply to the pointer

---

wk_vertices   *Extract vertices*

Description

These functions provide ways to extract individual coordinate values. Whereas wk_vertices()
returns a vector of coordinates as in the same format as the input, wk_coords() returns a data
frame with coordinates as columns.

Usage

```
wk_vertices(handleable, ...)

wk_coords(handleable, ...)

wk_vertex_filter(handler, add_details = FALSE)
```

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.
add_details	Use TRUE to add a "wk_details" attribute, which contains columns feature_id, part_id, and ring_id.

Value

- wk_vertices() extracts vertices and returns the in the same format as the handler

- wk_coords() returns a data frame with columns columns feature_id (the index of the feature from whence it came), part_id (an arbitrary integer identifying the point, line, or polygon from whence it came), ring_id (an arbitrary integer identifying individual rings within polygons), and one column per coordinate (x, y, and/or z and/or m).

Examples

```
wk_vertices(wkt("LINESTRING (0 0, 1 1)"))
wk_coords(wkt("LINESTRING (0 0, 1 1)"))
```

---

wk_void   *Do nothing*

Description

This handler does nothing and returns NULL. It is useful for benchmarking readers and handlers and when using filters that have side-effects (e.g., `wk_debug()`). Note that this handler stops on the first parse error; to see a list of parse errors see the `wk_problems()` handler.

Usage

```
wk_void(handleable, ...)

wk_void_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

NULL

Examples

```
wk_void(wkt("POINT (1 4)"))
wk_handle(wkt("POINT (1 4)"), wk_void_handler())
```

---

wk_writer.sfc   *Write geometry vectors*

Description

When writing transformation functions, it is often useful to know which handler should be used to create a (potentially modified) version of an object. Some transformers (e.g., `wk_vertices()`) modify the geometry type of an object, in which case a generic writer is needed. This defaults to `wkb_writer()` because it is fast and can handle all geometry types.

Usage

```
## 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_)

wkt_writer(precision = 16L, trim = TRUE)

wk_writer(handleable, ..., generic = FALSE)

## Default S3 method:
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, ..., generic = FALSE)

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.
generic	Use TRUE to obtain a writer that can write all geometry types.

Value

A wk_handler.

---

xy   *Efficient point vectors*

Description

Efficient point vectors

Usage

```
xy(x = double(), y = double(), crs = wk_crs_auto())

xyz(x = double(), y = double(), z = double(), crs = wk_crs_auto())

xym(x = double(), y = double(), m = double(), crs = wk_crs_auto())

xyzm(
  x = double(),
  y = double(),
  z = double(),
  m = double(),
  crs = wk_crs_auto()
)

xy_dims(x)

as_xy(x, ...)

## Default S3 method:
as_xy(x, ..., dims = NULL)

## S3 method for class 'wk_xy'
as_xy(x, ..., dims = NULL)

## S3 method for class 'matrix'
as_xy(x, ..., crs = NULL)

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

Arguments

x, y, z, m	Coordinate values.
crs	A value to be propagated as the CRS for this vector.
...	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