Package: cellranger (via r-universe)

June 12, 2024

Title Translate Spreadsheet Cell Ranges to Rows and Columns

Version 1.1.0.9000

Description Helper functions to work with spreadsheets and the
``A1:D10'' style of cell range specification.

License MIT + file LICENSE

URL https://github.com/rsheets/cellranger

BugReports https://github.com/rsheets/cellranger/issues

Depends R (>= 3.1)

Imports rematch, rematch2, tibble

Suggests covr, knitr, rmarkdown, testthat (>= 1.0.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 6.0.1.9000

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

RemoteUrl https://github.com/rsheets/cellranger

RemoteRef HEAD

RemoteSha 7ecde54c70c9a66833e49bf2c874bb4a4fcc75f3

Contents

`A1_to_R1C1` *Convert cell reference strings from A1 to R1C1 format*

Description

Convert cell reference strings from A1 to R1C1 format. Strictly speaking, this only makes sense for absolute references, such as `"$B$4"`. Why? Because otherwise, we'd have to know the host cell of the reference. Set `strict = FALSE` to relax and treat pure relative references, like (`"B4"`), as if they are absolute. Mixed references, like (`"B$4"`), will always return `NA`, no matter the value of `strict`.

Usage

```
A1_to_R1C1(x, strict = TRUE)
```

Arguments

`x`	character vector of cell references in A1 format
`strict`	logical, affects reading and writing of A1 formatted cell references. When `strict = TRUE`, references must be declared absolute through the use of dollar signs, e.g., `$A$1`, for parsing. When making a string, `strict = TRUE` requests dollar signs for absolute reference. When `strict = FALSE`, pure relative reference strings will be interpreted as absolute, i.e. `A1` and `$A$1` are treated the same. When making a string, `strict = FALSE` will cause dollars signs to be omitted in the reference string.

Value

character vector of absolute cell references in R1C1 format

Examples

```
A1_to_R1C1("$A$1")
A1_to_R1C1("A1")                 ## raises a warning, returns NA
A1_to_R1C1("A1", strict = FALSE) ## unless strict = FALSE
A1_to_R1C1(c("A1", "B$4")) ## raises a warning, includes an NA, because
A1_to_R1C1(c("A1", "B$4"), strict = FALSE) ## mixed ref always returns NA
```

---

addr_col *Get column from cell location or reference*

Description

Get column from cell location or reference

Usage

```
addr_col(x, ...)

## S3 method for class 'cell_addr'
addr_col(x, ...)
```

Arguments

x	a suitable representation of cell(s) or a cell area reference
...	further arguments passed to or from other methods

Value

integer vector

Methods (by class)

- cell_addr: Method for cell_addr objects (ca <- cell_addr(1:4, 3)) addr_col(ca)

---

addr_row *Get row from cell location or reference*

Description

Get row from cell location or reference

Usage

```
addr_row(x, ...)

## S3 method for class 'cell_addr'
addr_row(x, ...)
```

Arguments

x	a suitable representation of cell(s) or a cell area reference
...	further arguments passed to or from other methods

Value

integer vector

Methods (by class)

- cell_addr: Method for cell_addr objects (ca <- cell_addr(1:4, 3)) addr_row(ca)

---

anchored *Specify cell limits via an anchor cell*

Description

Specify the targetted cell rectangle via an upper left anchor cell and the rectangle's row and column extent. The extent can be specified directly via dims or indirectly via the input object. Specification via input anticipates a write operation into the spreadsheet. If input is one-dimensional, the byrow argument controls whether the rectangle will extend down from the anchor or to the right. If input is two-dimensional, the col_names argument controls whether cells will be reserved for column or variable names. If col_names is unspecified, default behavior is to set it to TRUE if input has columns names and FALSE otherwise.

Usage

```
anchored(anchor = "A1", dim = c(1L, 1L), input = NULL, col_names = NULL,
  byrow = FALSE)
```

Arguments

anchor	character, specifying the upper left cell in "A1" or "R1C1" notation
dim	integer vector, of length two, holding the number of rows and columns of the targetted rectangle; ignored if input is provided
input	a one- or two-dimensioanl input object, used to determine the extent of the targetted rectangle
col_names	logical, indicating whether a row should be reserved for the column or variable names of a two-dimensional input; if omitted, will be determined by checking whether input has column names
byrow	logical, indicating whether a one-dimensional input should run down or to the right

Value

a cell_limits object

Examples

```
anchored()
as.range(anchored())
dim(anchored())

anchored("Q24")
as.range(anchored("Q24"))
dim(anchored("Q24"))

anchored(anchor = "R4C2", dim = c(8, 2))
as.range(anchored(anchor = "R4C2", dim = c(8, 2)))
as.range(anchored(anchor = "R4C2", dim = c(8, 2)), fo = "A1")
dim(anchored(anchor = "R4C2", dim = c(8, 2)))

(input <- head(iris))
anchored(input = input)
as.range(anchored(input = input))
dim(anchored(input = input))

anchored(input = input, col_names = FALSE)
as.range(anchored(input = input, col_names = FALSE))
dim(anchored(input = input, col_names = FALSE))

(input <- LETTERS[1:8])
anchored(input = input)
as.range(anchored(input = input))
dim(anchored(input = input))

anchored(input = input, byrow = TRUE)
as.range(anchored(input = input, byrow = TRUE))
dim(anchored(input = input, byrow = TRUE))
```

---

as.cell_addr *Convert to a cell_addr object*

Description

Convert various representations of a cell reference into an object of class `cell_addr`. Recall that `cell_addr` objects hold absolute row and column location, so `ra_ref` objects or cell reference strings with relative or mixed references will raise a warning and generate `NA`s.

Usage

```
as.cell_addr(x, ...)

as.cell_addr_v(x, ...)

## S3 method for class 'ra_ref'
```

```
as.cell_addr(x, ...)

## S3 method for class 'list'
as.cell_addr_v(x, ...)

## S3 method for class 'character'
as.cell_addr(x, fo = NULL, strict = TRUE, ...)

## S3 method for class 'character'
as.cell_addr_v(x, fo = NULL, strict = TRUE, ...)
```

Arguments

x	a cell reference
...	further arguments passed to or from other methods
fo	either "R1C1" (the default) or "A1" specifying the cell reference format; in many contexts, it can be inferred and is optional
strict	logical, affects reading and writing of A1 formatted cell references. When strict = TRUE, references must be declared absolute through the use of dollar signs, e.g., A1, for parsing. When making a string, strict = TRUE requests dollar signs for absolute reference. When strict = FALSE, pure relative reference strings will be interpreted as absolute, i.e. A1 and A1 are treated the same. When making a string, strict = FALSE will cause dollars signs to be omitted in the reference string.

Value

a cell_addr object

Examples

```
as.cell_addr(ra_ref())
rar <- ra_ref(2, TRUE, 5, TRUE)
as.cell_addr(rar)
## mixed reference
rar <- ra_ref(2, FALSE, 5, TRUE)
as.cell_addr(rar)
ra_ref_list <-
  list(ra_ref(), ra_ref(2, TRUE, 5, TRUE), ra_ref(2, FALSE, 5, TRUE))
as.cell_addr_v(ra_ref_list)
as.cell_addr("$D$12")
as.cell_addr("R4C3")
as.cell_addr(c("$C$4", "$D$12"))
as.cell_addr("$F2")
as.cell_addr("R[-4]C3")
as.cell_addr("F2", strict = FALSE)
```

as.range *Convert a cell_limits object to a cell range*

Description

Convert a cell_limits object to a cell range

Usage

```
as.range(x, fo = c("R1C1", "A1"), strict = FALSE, sheet = NULL)
```

Arguments

x	a cell_limits object
fo	either "R1C1" (the default) or "A1" specifying the cell reference format; in many contexts, it can be inferred and is optional
strict	logical, affects reading and writing of A1 formatted cell references. When strict = TRUE, references must be declared absolute through the use of dollar signs, e.g., A1, for parsing. When making a string, strict = TRUE requests dollar signs for absolute reference. When strict = FALSE, pure relative reference strings will be interpreted as absolute, i.e. A1 and A1 are treated the same. When making a string, strict = FALSE will cause dollars signs to be omitted in the reference string.
sheet	logical, indicating whether to include worksheet name; if NULL, worksheet is included if worksheet name is not NA

Value

length one character vector holding a cell range

Examples

```
rgCL <- cell_limits(ul = c(1, 2), lr = c(7, 6))
as.range(rgCL)
as.range(rgCL, fo = "A1")

rgCL_ws <- cell_limits(ul = c(1, 2), lr = c(7, 6), sheet = "A Sheet")
as.range(rgCL_ws)
as.range(rgCL_ws, fo = "A1")
```

as.ra_ref *Convert to a ra_ref object*

Description

Convert various representations of a cell reference into an object of class ra_ref.

- as.ra_ref is NOT vectorized and therefore requires the input to represent exactly one cell, i.e. be of length 1.
- as.ra_ref_v accepts input of length >= 1 and returns a list of ra_ref() objects.

Usage

```
as.ra_ref(x, ...)

as.ra_ref_v(x, ...)

## S3 method for class 'character'
as.ra_ref(x, fo = NULL, strict = TRUE, ...)

## S3 method for class 'character'
as.ra_ref_v(x, fo = NULL, strict = TRUE, ...)

## S3 method for class 'cell_addr'
as.ra_ref(x, ...)

## S3 method for class 'cell_addr'
as.ra_ref_v(x, ...)
```

Arguments

x	one or more cell references, as a character vector or cell_addr object
...	further arguments passed to or from other methods
fo	either "R1C1" (the default) or "A1" specifying the cell reference format; in many contexts, it can be inferred and is optional
strict	logical, affects reading and writing of A1 formatted cell references. When strict = TRUE, references must be declared absolute through the use of dollar signs, e.g., A1, for parsing. When making a string, strict = TRUE requests dollar signs for absolute reference. When strict = FALSE, pure relative reference strings will be interpreted as absolute, i.e. A1 and A1 are treated the same. When making a string, strict = FALSE will cause dollars signs to be omitted in the reference string.

Value

a ra_ref object, in the case of as.ra_ref, or a list of them, in the case of as.ra_ref_v

Examples

```
## as.ra_ref.character()
as.ra_ref("$F$2")
as.ra_ref("R[-4]C3")
as.ra_ref("B4")
as.ra_ref("B4", strict = FALSE)
as.ra_ref("B$4")

## this is actually ambiguous! is format A1 or R1C1 format?
as.ra_ref("RC2")
## format could be specified in this case
as.ra_ref("RC2", fo = "R1C1")
as.ra_ref("RC2", fo = "A1", strict = FALSE)

## as.ra_ref_v.character()
cs <- c("$A$1", "Sheet1!$F$14", "Sheet2!B$4", "D9")
## Not run:
## won't work because as.ra_ref requires length one input
as.ra_ref(cs)

## End(Not run)
## use as.ra_ref_v instead
as.ra_ref_v(cs, strict = FALSE)
## as.ra_ref.cell_addr
ca <- cell_addr(2, 5)
as.ra_ref(ca)
## as.ra_ref_v.cell_addr()

ca <- cell_addr(1:3, 1)
## Not run:
## won't work because as.ra_ref methods not natively vectorized
as.ra_ref(ca)

## End(Not run)
## use as.ra_ref_v instead
as.ra_ref_v(ca)
```

`cellranger` *cellranger*

Description

Helper functions to work with spreadsheets and the "A1:D10" style of cell range specification.

cell_addr *cell_addr class*

Description

The cell_addr class is used to hold the absolute row and column location for one or more cells. An object of class cell_addr is a list with two components of equal length, named row and col, consisting of integers greater than or equal to one or NA. This is in contrast to the ra_ref class, which holds a representation of a single absolute, relative, or mixed cell reference from, e.g., a formula.

Usage

```
cell_addr(row, col)
```

Arguments

row	integer. Must be the same length as col or of length one, which will be recycled to the length of col.
col	integer. Same deal as for row.

Value

a cell_addr object

Reference

Spreadsheet Implementation Technology: Basics and Extensions Peter Sestoft MIT Press 2014

Examples

```
cell_addr(4, 3)
(ca <- cell_addr(1:4, 3))
ca[2:3]
ca[[4]]
length(ca)
```

cell_cols *Specify cell limits only for columns*

Description

How does this differ from cell_limits? Two ways. First, the input can have length greater than 2, i.e. the columns can be specified as 1:n. If the length is greater than 2, both the min and max are taken with NA.rm = TRUE. Note it is not possible to request non-contiguous columns, i.e. columns 1, 2, and 5. In this case, the requested columns will run from the minimum of 1 to the maximum of 5. Second, the input can be given in the letter-based format spreadsheets use to label columns.

Usage

```
cell_cols(x)
```

Arguments

x	vector of column limits; if character, converted to numeric; if length greater than two, min and max will be taken with NA.rm = TRUE

Value

a cell_limits object

Examples

```
cell_cols(c(NA, 3))
cell_cols(c(7, NA))
cell_cols(4:16)
cell_cols(c(3, NA, 10))

cell_cols("C:G")
cell_cols(c("B", NA))
cell_cols(LETTERS)
```

---

cell_limits *Create a cell_limits object*

Description

A cell_limits object is a list with three components:

- ul vector specifying upper left cell of target rectangle, of the form c(ROW_MIN, COL_MIN)
- lr vector specifying lower right cell of target rectangle, of the form c(ROW_MAX, COL_MAX)
- sheet string specifying worksheet name, which may be NA, meaning it's unspecified

A value of NA in ul or lr means the corresponding limit is left unspecified. Therefore a verbose way to specify no limits at all would be cell_limits(c(NA, NA), c(NA, NA)). If the maximum row or column is specified but the associated minimum is not, then the minimum is set to 1. **NOTE**: I am reconsidering this behavior and might choose to use NA for the minimum in this case.

When specified via character, cell references can be given in A1 or R1C1 notation and must be interpretable as absolute references. For A1, this means either both row and column are annotated with a dollar sign $ or neither is. So, no mixed references, like B$4. For R1C1, this means no square brackets, like R[-3]C[3].

Usage

```
cell_limits(ul = c(NA_integer_, NA_integer_), lr = c(NA_integer_,
  NA_integer_), sheet = NA_character_)

## S3 method for class 'cell_limits'
dim(x)

as.cell_limits(x, ...)

## S3 method for class 'cell_limits'
as.cell_limits(x, ...)

## S3 method for class 'NULL'
as.cell_limits(x, ...)

## S3 method for class 'character'
as.cell_limits(x, fo = NULL, ...)
```

Arguments

ul	vector identifying upper left cell of target rectangle
lr	vector identifying lower right cell of target rectangle
sheet	string containing worksheet name, optional
x	input to convert into a cell_limits object
...	further arguments passed to or from other methods
fo	either "R1C1" (the default) or "A1" specifying the cell reference format; in many contexts, it can be inferred and is optional

Value

a cell_limits object

Examples

```
cell_limits(c(1, 3), c(1, 5))
cell_limits(c(NA, 7), c(3, NA))
cell_limits(c(NA, 7))
cell_limits(lr = c(3, 7))

cell_limits(c(1, 3), c(1, 5), "Sheet1")
cell_limits(c(1, 3), c(1, 5), "Spaces are evil")

dim(as.cell_limits("A1:F10"))

as.cell_limits("A1")
as.cell_limits("$Q$24")
as.cell_limits("A1:D8")
as.cell_limits("R5C11")
```

```
as.cell_limits("R2C3:R6C9")
as.cell_limits("Sheet1!R2C3:R6C9")
as.cell_limits("'Spaces are evil'!R2C3:R6C9")

## Not run:
## explicitly mixed A1 references won't work
as.cell_limits("A$2")
## mixed or relative R1C1 references won't work
as.cell_limits("RC[4]")

## End(Not run)
```

`cell_rows` *Specify cell limits only for rows*

Description

How does this differ from `cell_limits`? Here the input can have length greater than 2, i.e. the rows can be specified as `1:n`. If the length is greater than 2, both the min and max are taken with `NA.rm = TRUE`. Note it is not possible to request non-contiguous rows, i.e. rows 1, 2, and 5. In this case, the requested rows will run from the minimum of 1 to the maximum of 5.

Usage

```
cell_rows(x)
```

Arguments

`x`	numeric vector of row limits; if length greater than two, min and max will be taken with `NA.rm = TRUE`

Value

a `cell_limits` object

Examples

```
cell_rows(c(NA, 3))
cell_rows(c(7, NA))
cell_rows(4:16)
cell_rows(c(3, NA, 10))

dim(cell_rows(1:5))
```

guess_fo *Guess cell reference string format*

Description

Guess if cell references are in R1C1 or A1 format.

Usage

```
guess_fo(x, fo = c("R1C1", "A1"))
```

Arguments

x	character vector of cell reference strings
fo	default to assume if format is ambiguous

Value

character vector consisting of R1C1, A1, or NA

Examples

```
A1 <- c("A1", "$A1", "A$1", "$A$1", "a1")
guess_fo(A1)
R1C1 <- c("R1C1", "R1C[-1]", "R[-1]C1", "R[-1]C[9]")
guess_fo(R1C1)

guess_fo("RC2")
guess_fo("12")
guess_fo(12)
```

is_A1 *Test cell reference strings*

Description

Test cell reference strings for a specific format.

Usage

```
is_A1(x)

is_R1C1(x)
```

Arguments

x	character vector of cell reference strings

Value

a logical vector

Functions

- `is_A1`: A1 format, case insenstive; relative, absolute, or mixed
- `is_R1C1`: R1C1 format; relative, absolute, or mixed

Examples

```
is_A1("A1")
is_R1C1("A1")
is_R1C1("R4C12")

x <- c("A1", "$A4", "$b$12", "RC1", "R[-4]C9", "R5C3")
data.frame(x, is_A1(x), is_R1C1(x))
```

`letter-num-conversion` *Convert between letter and integer representations of column IDs*

Description

Convert "A1"-style column IDs from a letter representation to an integer, e.g. column A becomes 1, column D becomes 4, etc. Or go the other way around.

Usage

```
letter_to_num(x)

num_to_letter(y)
```

Arguments

`x`	a character vector of "A1" style column IDs (case insensitive)
`y`	a vector of integer column IDs

Details

- Google Sheets have up to 300 columns (column KN).
- Excel 2010 spreadsheets have up to 16,384 columns (column XFD).
- ZZ is column 702.
- ZZZ is column 18,278 (no known spreadsheet actually goes that high).

Value

a vector of column IDs, either character or integer

Examples

```
letter_to_num('Z')
letter_to_num(c('AA', 'ZZ', 'ABD', 'ZZZ'))
letter_to_num(c(NA, ''))
num_to_letter(28)
num_to_letter(900)
num_to_letter(18278)
num_to_letter(c(25, 52, 900, 18278))
num_to_letter(c(NA, 0, 4.8, -4))
```

`print.ra_ref` *Print ra_ref object*

Description

Print ra_ref object

Usage

```
## S3 method for class 'ra_ref'
print(x, fo = c("R1C1", "A1"), ...)
```

Arguments

`x`	an object of class `ra_ref`
`fo`	either `"R1C1"` (the default) or `"A1"` specifying the cell reference format; in many contexts, it can be inferred and is optional
`...`	further arguments passed to or from other methods

Examples

```
(rar <- ra_ref(3, TRUE, 1, TRUE))
print(ra_ref(), fo = "A1")
```

`R1C1_to_A1` *Convert R1C1 positioning notation to A1 notation*

Description

Convert cell reference strings from R1C1 to A1 format. This only makes sense for absolute references, such as `"R4C2"`. Why? Because otherwise, we'd have to know the host cell of the reference. Relative and mixed references, like (`"R[3]C[-1]"` and `"R[1]C5"`), will therefore return `NA`.

Usage

```
R1C1_to_A1(x, strict = TRUE)
```

Arguments

x	vector of cell positions in R1C1 notation
strict	logical, affects reading and writing of A1 formatted cell references. When strict = TRUE, references must be declared absolute through the use of dollar signs, e.g., A1, for parsing. When making a string, strict = TRUE requests dollar signs for absolute reference. When strict = FALSE, pure relative reference strings will be interpreted as absolute, i.e. A1 and A1 are treated the same. When making a string, strict = FALSE will cause dollars signs to be omitted in the reference string.

Value

character vector of absolute cell references in A1 notation

Examples

```
R1C1_to_A1("R1C1")
R1C1_to_A1("R10C52", strict = FALSE)
R1C1_to_A1(c("R1C1", "R10C52", "RC4", "R[-3]C[9]"))
```

---

ra_ref *ra_ref class*

Description

The ra_ref class is used to represent a single relative, absolute, or mixed cell reference, presumably found in a formula. When row_abs is TRUE, it means that row_ref identifies a specific row in an absolute sense. When row_abs is FALSE, it means that row_ref holds a positive, zero, or negative offset relative to the address of the cell containing the formula that contains the associated cell reference. Ditto for col_abs and col_ref.

Usage

```
ra_ref(row_ref = 1L, row_abs = TRUE, col_ref = 1L, col_abs = TRUE,
  sheet = NA_character_, file = NA_character_)
```

Arguments

row_ref	integer, row or row offset
row_abs	logical indicating whether row_ref is absolute or relative
col_ref	integer, column or column offset
col_abs	logical indicating whether col_ref is absolute or relative
sheet	the name of a sheet (a.k.a. worksheet or tab)
file	the name of a file (a.k.a. workbook)

Details

A `ra_ref` object can also store the name of a sheet and a file, though these will often be `NA`. A cell reference in a formula can potentially be qualified like this: `[my_workbook.xlxs]Sheet1!R2C3`. In Testoft (2014), he creates an entirely separate class for this, a `cell_ref`, which consists of a sheet- and file-ignorant `ra_ref` object and a sheet reference (he doesn't allow formulas to refer to other files). I hope I don't regret choosing a different path.

Value

a `ra_ref` object

Reference

Spreadsheet Implementation Technology: Basics and Extensions Peter Sestoft MIT Press 2014

Examples

```
ra_ref()
ra_ref(row_ref = 3, col_ref = 2)
ra_ref(row_ref = 10, row_abs = FALSE, col_ref = 3, col_abs = TRUE)
ra_ref(sheet = "a sheet")
```

---

to_string *Get string representation of cell references*

Description

Convert various representations of a cell reference to character

- `to_string` is not necessarily vectorized. For example, when the the input is of class `ra_ref`, it must of be of length one. However, to be honest, this will actually work for `cell_addr`, even when length > 1.
- `to_string_v` is guaranteed to be vectorized. In particular, input can be a `cell_addr` of length >= 1 or a list of `ra_ref` objects.

If either the row or column reference is relative, note that, in general, it's impossible to convert to an "A1" formatted string. We would have to know "relative to what?".

Usage

```
to_string(x, fo = c("R1C1", "A1"), strict = TRUE, sheet = NULL, ...)

to_string_v(x, fo = c("R1C1", "A1"), strict = TRUE, sheet = NULL, ...)

## S3 method for class 'ra_ref'
to_string(x, fo = c("R1C1", "A1"), strict = TRUE,
  sheet = NULL, ...)
```

```
## S3 method for class 'list'
to_string_v(x, fo = c("R1C1", "A1"), strict = TRUE,
  sheet = NULL, ...)

## S3 method for class 'cell_addr'
to_string(x, fo = c("R1C1", "A1"), strict = TRUE,
  sheet = FALSE, ...)

## S3 method for class 'cell_addr'
to_string_v(x, fo = c("R1C1", "A1"), strict = TRUE,
  sheet = FALSE, ...)
```

Arguments

x	a suitable representation of a cell or cell area reference: a single ra_ref object or a list of them or a cell_addr object
fo	either "R1C1" (the default) or "A1" specifying the cell reference format; in many contexts, it can be inferred and is optional
strict	logical, affects reading and writing of A1 formatted cell references. When strict = TRUE, references must be declared absolute through the use of dollar signs, e.g., A1, for parsing. When making a string, strict = TRUE requests dollar signs for absolute reference. When strict = FALSE, pure relative reference strings will be interpreted as absolute, i.e. A1 and A1 are treated the same. When making a string, strict = FALSE will cause dollars signs to be omitted in the reference string.
sheet	logical, indicating whether to include worksheet name; if NULL, worksheet is included if worksheet name is not NA
...	further arguments passed to or from other methods

Value

a character vector

Examples

```
## exactly one ra_ref --> string
to_string(ra_ref())
to_string(ra_ref(), fo = "A1")
to_string(ra_ref(), fo = "A1", strict = FALSE)
to_string(ra_ref(row_ref = 3, col_ref = 2))
to_string(ra_ref(row_ref = 3, col_ref = 2, sheet = "helloooo"))
(mixed_ref <- ra_ref(row_ref = 10, row_abs = FALSE, col_ref = 3))
to_string(mixed_ref)

## this will raise warning and generate NA, because row reference is
## relative and format is A1
to_string(mixed_ref, fo = "A1")

## a list of ra_ref's --> character vector
```

```
ra_ref_list <-
  list(ra_ref(), ra_ref(2, TRUE, 5, TRUE), ra_ref(2, FALSE, 5, TRUE))
to_string_v(ra_ref_list)

## cell_addr --> string
(ca <- cell_addr(3, 8))
to_string(ca)
to_string(ca, fo = "A1")

(ca <- cell_addr(1:4, 3))
to_string(ca)
to_string(ca, fo = "A1")
## explicitly go from cell_addr, length > 1 --> character vector
(ca <- cell_addr(1:4, 3))
to_string_v(ca)
to_string_v(ca, fo = "A1")
```

Index