

# Package ‘incadata’

May 5, 2019

**Type** Package

**Title** Recognize and Handle Data in Formats Used by Swedish Cancer Centers

**Version** 0.8.2

**Description** Handle data in formats used by cancer centers in Sweden, both from 'INCA' (<<https://rcc.incanet.se>>) and by the older register platform 'Rockan'. All variables are coerced to suitable classes based on their format. Dates (from various formats such as with missing month or day, with or without century prefix or with just a week number) are all recognized as dates and coerced to the ISO 8601 standard (Y-m-d). Boolean variables (internally stored either as 0/1 or ``True"/`False"/blanks when exported) are coerced to logical. Variable names ending in '\_Beskrivning' and '\_Varde' will be character, and 'PERSNR' will be coerced (if possible) to a valid personal identification number 'pin' (by the 'sweidnumbr' package). The package also allow the user to interactively choose if a variable should be coerced into a potential format even though not all of its values might conform to the recognized pattern. It also contain a caching mechanism in order to temporarily store data sets with its newly decided formats in order to not rerun the identification process each time. The package also include a mechanism to aid the documentation process connected to projects build on data from 'INCA'. From version 0.7, some general help functions are also included, as previously found in the 'rccmisc' package.

**License** GPL-2

**Depends** R (>= 2.10)

**Imports** backports, decoder, dplyr, rvest, sweidnumbr, xml2

**Suggests** testthat, knitr, rmarkdown, R.rsp

**VignetteBuilder** R.rsp

**URL** <https://cancercentrum.bitbucket.io/incadata>

**BugReports** <https://www.bitbucket.org/cancercentrum/incadata/issues>

**LazyData** true

**RoxygenNote** 6.1.1

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Erik Bulow [aut, cre] (<<https://orcid.org/0000-0002-9973-456X>>)

**Maintainer** Erik Bulow <[erik.bulow@rccvast.se](mailto:erik.bulow@rccvast.se)>

**Repository** CRAN

**Date/Publication** 2019-05-05 20:30:04 UTC

## R topics documented:

as.Dates . . . . .	3
as.incadata . . . . .	4
as_numeric . . . . .	6
best_match . . . . .	7
clean_text . . . . .	8
create_s3_method . . . . .	9
cut.integer . . . . .	10
documents . . . . .	10
exceed_threshold . . . . .	11
exportr . . . . .	12
ex_data . . . . .	13
find_documents . . . . .	13
find_links . . . . .	14
find_register . . . . .	15
id . . . . .	15
is.inca . . . . .	16
is.incalogical . . . . .	16
is.scalar_in . . . . .	17
is.wholenumber . . . . .	18
lownames . . . . .	18
lt . . . . .	19
next_method . . . . .	19
psum . . . . .	20
specify_missing . . . . .	20
use_incadata . . . . .	21
width . . . . .	22

**Index**

**23**

---

`as.Dates`*Converting potential date to Date vector*

---

**Description**

The function recognizes dates in formats used by INCA and Rockan.

**Usage**

```
as.Dates(x)
```

**Arguments**

x                    atomic vector

**Details**

Regular expressions are used to match any of the following date formats:

- Y-m-d: The ISO 8601 standard such as "2017-02-16" as used by INCA.
- Ymd: such as "20160216" as used by the Rockan registers
- Any of the above with missing day such as "2017-02-00" or "20170200" as used if the exact date is unknown.
- Any of the above with missing month such as "2017-00-00" or "20170000" as sometimes used if the exact date is unknown.
- Dates between 1950 and 1980 can have missing century prefix, such as "67-01-01", "670101", "670100", "670000" etc as earlier used for some dates in the Rockan registers.
- Dates from the 20th century can also have month and day changed to week number such as "6723" or "196723" as sometimes used for death dates in the cancer register (originating from the population register).
- The special INCA variable SKAPAD\_DATUM is also recognized as data but is originally a date and time object ([POSIXct](#))

All dates are coerced to Y-m-d (ISO 8601):

- a missing day is set to 15
- a missing month is set to July
- a week number is translated to the "median day" of that week
- SKAPAD\_DATUM has its time stamp dropped

An alternative would be to use random assignments of dates within specified periods. This would have some benefits but does not conform to behavior used elsewhere by INCA.

**Value**

vector of class "Date"

**Possible date range**

All potential dates are accepted as such. RCC data should however only contain historic data. Dates from the future does therefore raise warnings. The same is true for dates before 1830. The Swedish cancer register was initiated in 1958. The earliest possible dates found in the register should therefore originate from birth date of really old people diagnosed with cancer during that year.

**See Also**

[as.Date](#)

**Examples**

```
as.Dates(c(1212121212, "20000101", "2014-10-15", 5806))

## Not run:
# Note that the as.Date (as oppose to as.Dates)
# does not handle missing dates as empty strings
as.Date(c("", "2017-02-16")) # Error
as.Dates(c("", "2017-02-16")) # NA "2017-02-16"

## End(Not run)
```

---

as.incadata

*Identify data formats used by INCA and Rockan*

---

**Description**

Coerce data of any form to its relevant type as identified either by column/vector names or by variable content and convert all variable names to lower case.

**Usage**

```
as.incadata(x, ...)

is.incadata(x)

## S3 method for class 'data.frame'
as.incadata(x, decode = TRUE, id = TRUE,
  ask = TRUE, ...)

## Default S3 method:
as.incadata(x, n_i = NULL, ...)
```

**Arguments**

x	data
...	arguments passed to <code>exceed_threshold</code> (of most use is probably "threshold" and "force", see the "interactive use" section below)
decode	Should <code>decode</code> be applied to variables with identified variable names? (TRUE by default).
id	Should an id-column be added (see <code>id</code> )?
ask	ask for input if unsure how to coerce variables (see the "interactive use" section below)
n_i	used internally between methods (should not be set by the user)

**Details**

Vectors are coerced to identified formats in the following order:

- vectors recognized as Boolean by `is.incalogical` are coerced to logical (this is a strict format than can not be contaminated with any unwanted values, section "interactive use" below does therefore not apply to these values)
- vectors with an already specified class attribute (except the common "factor" class) remains as members of that class
- columns or vectors names 'persnr' or 'pnr' will be coerced to the 'pin' class by `as.pin`
- columns or vectors with names ending in '\_Beskrivning', '\_Varde', '\_Gruppenamn' or '\_id' are always treated as character (not factors; see section "factors" below)
- column or vectors named "PAT\_ID", "KON\_VALUE" and "LAN\_VALUE" are also always treated as character. These could also be thought of as numerics but are treated as character internally by INCA. To stay with that format ensures the assumption of a stable format.
- If all values of a vector are NA, it is coerced from logical to character. This might be a faulty assumption but it is in fact more likely that an empty vector is a character variable (since most INCA variables are of type character) than that it is a Boolean vector (that has its own format in INCA).
- Dates in formats recognized by `as.Dates` are coerced to such.
- Integers (even if stored as characters or factors) without leading zeros (except when the zero is the only digit) are coerced to integers
- Numerics (even if stored as characters or factors) containing either a Swedish decimal comma or an English decimal point are coerced to numeric (with possible commas changed to points).
- all other formats are coerced to character. This includes integers with leading zeroes (since these might be unit codes where a leading zero might bear meaning).

**Value**

`as.incadata.data.frame` object of class `incadata` based on the "tibble"-class used within the "tidyverse" with all variables possibly coerced as described above.

`as.incadata.default` input vector coerced to relevant class

`is.incadata` TRUE for objects of class `incadata`, otherwise FALSE

## factors

Note that the `incadata` format does not include factors. Factors can be really useful for some applications but our philosophy is that they should be explicitly stated as such when needed. It is otherwise common that factor levels are created just by the responses present in a certain data set. These might or might not contain a complete list of possible alternatives from a INCA variable with a fixed value set.

## interactive use

Some vectors can be undoubtedly recognized according to specifications above. It is however possible that a vector of an intended format might have been "contaminated" with data of some other form. This might happen for example when a numeric variable is technically a character in INCA. For example a hospital unit code like `c(111, 123, "?")` might suddenly occur (if someone use a question mark as placeholder for an unknown code). Ordinary coercing rules of R would treat this vector as a character (see `c`), although it might be more correct to treat it as a numeric with "?" set to NA.

The `as.incadata` function relies on `exceed_threshold` to ignore such contaminated values if they represent only a (preferably small) proportion of the values.

By default, if contaminated values exist but only to a proportion of less than 10 percent, the function will stop and ask the user for input on how to handle this variable. If the proportion exceeds 10 percent, ordinary coercing principles will apply.

The 10 percent limit can be modified by argument `threshold` and it is possible to force vectors with contaminated values to the otherwise potential format (without the need of individual confirmation) by setting argument `force = TRUE` (passed to `exceed_threshold`).

---

as\_numeric

*Test object for, or coerce to, numeric*

---

## Description

`as_numeric` is essentially a wrapper to `as.numeric` except that objects of class `factor` are first coerced to character and then to numeric. `is_numeric` test if `x` is "somehow numeric" (see examples).

## Usage

```
as_numeric(x)
```

```
is_numeric(x)
```

## Arguments

`x` object to be coerced or tested (and return a logical vector of the same length) or should it test the whole vector as one object and return a logical vector of length one. (TRUE by default).

**Examples**

```
df <- data.frame(v = c("46513", "45"))
class(df$v) # factor

# Note that
as.numeric(df$v) # 2 1
# but
as_numeric(df$v) # 46513 45

is_numeric(1) # TRUE
is_numeric("1") # TRUE
is_numeric(as.factor(1)) # TRUE
is_numeric(as.factor("khh")) # FALSE
```

---

best\_match

*Tries to correct misspelling of character string*


---

**Description**

This function uses fuzzy string matching to replace one possibly misspelled (or in other way not fully correct) character string with a correct version of the same string.

**Usage**

```
best_match(x, key, clean_text = TRUE, no_match = NA, all = FALSE)
```

**Arguments**

x	is a character string (or a character vector) that should be matched to the key
key	is a vector containing the correct spellings of the character strings.
clean_text	(boolean of length one) should arguments x and key be passed to <a href="#">clean_text</a> before matched (to ignore special characters)?
no_match	Output value if there is no match. Default is NA. The input is returned unchanged if not matched and no_match = NULL.
all	is a boolean indicator to specify what happens if there is more than one match. Default is FALSE resulting in a warning message and that only the first match is used. The result can then be returned as a vector. If TRUE, all possible matches are returned and the result must therefore be a list.

**Value**

The function returns a character vector of the same length as x if all = FALSE but with each element substituted to its best match in the key-vector. Strings that could not be matched are NA if (no\_match = TRUE) or unchanged if no\_match = FALSE. If all = TRUE, one input character string could result in more than one output character string. The output might therefore be longer than the input.

**See Also**[clean\\_text](#)**Examples**

```
best_match(c("Hej_apa!", "erik", "babian"),
c("hej apa", "hej bepa", "kungen", "Erik"))
best_match(c("Hej_apa", "erik", "babian"),
c("hej apa", "hej bepa", "kungen", "Erik"), no_match = FALSE)
```

---

`clean_text`*Clean/standardize text*

---

**Description**

Removes punctuation and spaces from character string. Also makes it lower case.

**Usage**

```
clean_text(x)
```

**Arguments**

x                    a character string to "clean"

**Value**

the cleaned character string (no punctuation, spaces or capital letters)

**See Also**[best\\_match](#)**Examples**

```
clean_text("HELLO_World!!!")
```



---

create\_s3\_method      *Template functions to generate basic S3 methods for new classes*

---

## Description

create\_s3\_method creates a method that applies NextMethod but that also keeps additional attributes (such as class). create\_s3\_print creates a print method.

## Usage

```
create_s3_method(generic = NULL, object = NULL)
```

```
create_s3_print(fun, ...)
```

## Arguments

generic, object	as described for <a href="#">NextMethod</a>
fun	Function to transform object before print (probably <a href="#">as.character</a> , <a href="#">as.numeric</a> or similar).
...	additional arguments passed to print method

## Details

Don't forget to also create for example a data.frame method by  
`as.data.frame.xxx <- as.data.frame.vector`

## Value

S3-method.

## Examples

```
a <- structure(1:10, class = c("b", "numeric"))
a[3] # Normal subsetting makes a loose its attributes
`.b` <- create_s3_method("[")
print.b <- create_s3_print(as.numeric)
a[3] # attributes preserved even if we can't see them
str(a[3])
```

---

cut.integer	<i>Convert integer vector to Factor</i>
-------------	---

---

**Description**

S3-method for cut applied to integer vectors where all outcome factors are integer intervals.

**Usage**

```
## S3 method for class 'integer'
cut(x, ...)
```

**Arguments**

x	integer vector
...	further arguments passed to or from other methods

**Value**

If cut.default(x, ...) returns only integer intervals, these are formatted in a more natural way and returned as an ordered factor. If non integer interval limits occur, the output of cut.default(x, ...) is returned as is.

**Examples**

```
cut.default(1:100, seq(0, 100, 20)) # Gives a quite unnatural output
cut(1:100, seq(0, 100, 20)) # Gives nicer and ordered output
cut(1:10, 3) # no integer intervals and therefor same as cut.default
```

---

documents	<i>Download and possibly open INCA documentation</i>
-----------	--

---

**Description**

Download and possibly open INCA documentation

**Usage**

```
documents(reg, doc = NULL, dir = ".", max_open = 3)
```

**Arguments**

reg	name of register to look for
doc	(part of) document name to look for
dir	directory where to save files
max_open	maximum number of files to open automatically (only on Mac OS X). Set to 0 to avoid any opening of files.

**Value**

Nothing. The function is called for its side effects.

**Examples**

```
## Not run:
documents("lunga", "uppfoljning")

## End(Not run)
```

---

exceed\_threshold      *Check if transformation/coercing of a vector is good enough*

---

**Description**

This function is primarily aimed to check if the transformation of a vector was successful enough to return the transformed value instead of the original.

**Usage**

```
exceed_threshold(original, transformed, threshold = 0.9, force = FALSE,
  ask = FALSE, var_name = "the input vector")
```

**Arguments**

original	the original vector
transformed	the transformed vector with NA-values for non transformed values
threshold	is a numeric value in [0,1] specifying the proportion of cells in transformed that should be recognised as correctly coerced to accept the new class. This does not effect the function output (except when force = TRUE) but will have some diagnostic benefits.
force	Should a candidate vector (candidate according to threshold) be forced to its suggested class (with non-coercable elements set to NA). FALSE by default but if the function is called interactively, the user will also have the option to set force = TRUE on the fly.
ask	this argument gives you the chance to interactively inspect your data and specify if a column is a date or not, on the fly. This is FALSE by default for as.Dates.default but TRUE for as.Dates.dataframe. It only applies when the function is runned interactively and only when force == FALSE.
var_name	a name for the object to be used in messages (you could probably just leave this as default, NULL; it is mostly used for internal purposes!).

**Value**

Either original or transformed.

## Examples

```
x <- c(rep("2012-01-01", 9), "foo")
exceed_threshold(x, as.Date(x))
exceed_threshold(x, as.Date(x), force = TRUE)
exceed_threshold(x, as.Date(x), ask = TRUE)
exceed_threshold(x, as.Date(x), threshold = 1)
exceed_threshold(x, as.Date(x), var_name = "bar", force = TRUE)

x <- c(1:9, "baz")
exceed_threshold(x, suppressWarnings(as.numeric(x)))
```

---

exportr

*Dump script together with functions from required packages*

---

## Description

If a package is not installed on the computer/server intended to run a final script, this function can take the script and export it together with all objects (functions, methods etcetera) from specified R packages. It might thereafter be possible to transfer the script and to run it even if all packages are not installed by the host.

## Usage

```
exportr(script = NULL, packages, recursive = TRUE,
        outfile = "./generated_r_script.R", force = FALSE)
```

## Arguments

script	connection with script (file) to append to function definitions
packages	name of packages (as character) to be explicitly included.
recursive	argument passed to <a href="#">package_dependencies</a>
outfile	filename for dump file
force	this function works only in interactive mode by default but output can be forced by this argument set to TRUE

## Details

Some packages use external dependencies and/or compiled code. This is not handled by the function. Hence, there is no guarantee that the script will actually work!

## Value

nothing (function called for its side effects)

---

ex_data	<i>Synthetic example data from INCA</i>
---------	---

---

**Description**

A data set resembling the typical form of INCA data. Variable names are real but all data has been carefully anonymized!

**Usage**

```
ex_data
```

**Format**

A data frame (not an object of class `incadata` with 497 rows and 433 variables)

**Details**

All data is random! There is no logical relation between any variables, not even between `x_Beskrivning` and `x_Varde`!

**Examples**

```
# Inspect the data
dplyr::glimpse(ex_data)

# Coerce to incadata
as.incadata(ex_data)
```

---

find_documents	<i>List URLs to documents for a register</i>
----------------	--

---

**Description**

List URLs to documents for a register

**Usage**

```
find_documents(url, doc = NULL)
```

**Arguments**

url	url to web page where to look for documents
doc	(part of) document name to look for

**Value**

names character vector with urls to documents

**Examples**

```
## Not run:
find_documents(find_register("all"))
find_documents(find_register("peniscancer"), "uppfoljning")

## End(Not run)
```

---

find_links	<i>Find links from web page</i>
------------	---------------------------------

---

**Description**

Find links from web page

**Usage**

```
find_links(url, select = NULL)
```

**Arguments**

url	URL to web page with links (must be under 'www.cancercentrum.se')
select	select only links matching specified pattern

**Value**

Named character vector with absolute URLs to links found on 'www.cancercentrum.se'

**Examples**

```
## Not run:
# Find e-mailaddresses to spam
find_links(
  "https://cancercentrum.se/vast/om-oss/kontakta-oss/",
  "mailto:"
)

## End(Not run)
```

---

find_register	<i>Find register by name</i>
---------------	------------------------------

---

**Description**

The specified name does not need to be exact since a search algorithm is applied to match existing registers. Names of the registers

**Usage**

```
find_register(reg = NULL)
```

**Arguments**

reg	name of register to look for
-----	------------------------------

**Value**

Named character vector with URL to specified register

**Examples**

```
find_register("all")
## Not run:
find_register("kronisk") # More than one possible alternative

## End(Not run)
```

---

id	<i>Add id variables to data frame</i>
----	---------------------------------------

---

**Description**

Construct id variable for patient data.

**Usage**

```
id(x, id = c("persnr", "pnr", "pat_id", "pn", "id"),
   ignore.case = TRUE)
```

**Arguments**

x	data frame
id	names of a possible id variable found in x
ignore.case	should name matching be done regardless of character case?

**Value**

Character variable with either the first name from `id` found in `x` or `rownames(x)` if no named column found.

---

<code>is.inca</code>	<i>Check if R is running from INCA</i>
----------------------	--

---

**Description**

Check if R is running from INCA

**Usage**

```
is.inca(logical = TRUE)
```

**Arguments**

<code>logical</code>	Should the return value be a simple boolean whether we are running from INCA or not?
----------------------	--

**Value**

Either TRUE/FALSE if `logical = TRUE` or one of "PROD", "TEST" or "LOCAL" depending on where R is running (if `logical = FALSE`)

**Examples**

```
is.inca()
```

---

<code>is.incallogical</code>	<i>Coerce to logical if value is logical according to INCA</i>
------------------------------	--

---

**Description**

Boolean vectors in INCA are stored internally as 0/1 and are changed to "True"/blank when exported. These functions identify such a variable as Boolean and can coerce it to such.

**Usage**

```
is.incallogical(x)
```

```
incallogical2logical(x)
```

**Arguments**

<code>x</code>	vector (potentially logical)
----------------	------------------------------



**Details**

It is common that check boxes are blanks by default but that this should be interpreted as TRUE. There are however some uncommon cases where the boxes are marked with "False" for FALSE. We can therefore not be certain of the meaning of a blank value. These will therefore be treated as NA.

**Value**

is.incalogical returns TRUE if the vector is logical according to INCA:s internal rules, FALSE otherwise. incalogical2logical returns a logical vector if x can be coerced to such.

**Examples**

```
is.incalogical(c("", "", "True", "")) # TRUE
is.incalogical(c("", "False", "", "")) # TRUE
is.incalogical(c("", "FALSE", "", "")) # FALSE
is.incalogical(logical(2)) # will be recognised as well
```

---

is.scalar_in	<i>Test if scalar is in interval</i>
--------------	--------------------------------------

---

**Description**

Test if scalar is in interval

**Usage**

```
is.scalar_in(left, right)

is.scalar_in01(x)
```

**Arguments**

left, right	arguments passed to <a href="#">between</a>
x	R object to be tested, most likely a numeric vector of length one (other formats are allowed but will always return FALSE).

**Value**

is.scalar\_in01 returns TRUE if x is an atomic vector of length one and  $0 \leq \text{as\_numeric}(x) \leq 1$ .  
is.scalar\_in return a function similar to is.scalar\_in01 but with specified boundaries.

**Examples**

```
is.scalar_in01(.5) # TRUE
is.scalar_in01(5) # FALSE

is_scalar_in09 <- is.scalar_in(0,9)
is_scalar_in09(5) # TRUE
```

is.wholenumber      *Test if a numeric vector consists of whole numbers*

---

**Description**

Function borrowed from the example section for [integer](#).

**Usage**

```
is.wholenumber(x, tol = .Machine$double.eps^0.5)
```

**Arguments**

x                    a numeric vector  
tol                  How much is x allowed to deviate from round(x) to be a whole number.

**Value**

Logical vector with same length as x.

**Examples**

```
is.wholenumber(1) # is TRUE  
(x <- seq(1, 5, by = 0.5) )  
is.wholenumber( x ) #--> TRUE FALSE TRUE ...
```

---

lownames              *Make all names in data.frame lower case*

---

**Description**

Tests are also performed so that all column names will stay unique!

**Usage**

```
lownames(df)
```

**Arguments**

df                    A data.frame, possibly with some names with capital letters

**Value**

df is returned unchanged, except that capital letters in names are changed to lower case.

**Examples**

```
df <- data.frame>Hello = 1:10, World = 1:10)  
lownames(df)
```

---

lt *Lead time from one date to another*

---

**Description**

Lead time from one date to another

**Usage**

```
lt(from, to, neg = FALSE)
```

**Arguments**

from, to start and stop dates (in formats that can be recognized as RCC dates).  
neg except negative lead times (set to NA if neg = FALSE)?

**Value**

Numeric vector

**Examples**

```
lt("2017-02-10", "2017-02-16") # 6  
lt("2017-02-16", "2017-02-10") # negative lead times ignored by default  
lt("2017-02-16", "2017-02-10", TRUE) # -6
```

---

next\_method *Function to create methods for generics*

---

**Description**

Function to create methods for generics

**Usage**

```
next_method()
```

---

psum *Parallel sum*

---

### Description

This function is to [sum](#), what [pmin](#) and [pmax](#) is to [min](#) and [max](#).

### Usage

```
psum(..., na.rm = FALSE)
```

### Arguments

... numeric vectors  
na.rm a logical indicating whether missing values should be removed.

### Examples

```
psum(1:10, 1:10, 1:10)
```

---

specify\_missing *Specify missing values for a vector*

---

### Description

Change specified values to NA

### Usage

```
specify_missing(x, ..., default_missing = c("", NA, "blanks"))
```

### Arguments

x vector  
... values that should be changed to NA if found in x  
default\_missing a vector with additional default values to change to NA. These are treated the same as ... but are added by default if not removed. A special value "blank" can be used to indicate all empty strings (all characters matching `[:blank:]`, see [regex](#)).

### Value

x itself but with specified values set to NA.

**Examples**

```
x <- sample(100)
x[sample(100, 10)] <- 999
specify_missing(x, 999)
```

---

use_incadata	<i>Use incadata from file or dataframe df</i>
--------------	---

---

**Description**

Read in a file (locally) or use global object named df (on INCA) and coerce to incadata-object.

**Usage**

```
use_incadata(file, cache = TRUE, sep = ";", dec = ",", ...)
```

**Arguments**

file	file name as character (ignored if called from INCA)
cache	use cache to speed up the loading (see section: "Cache" below)
sep, dec	arguments passed to <a href="#">read.csv2</a>
...	arguments passed to <a href="#">as.incadata</a> .

**Value**

object returned by [as.incadata](#)

**Cache**

To process all data through [as.incadata](#) can be time consuming for large data sets. It is therefore advised to use caching (argument `cache = TRUE`) to avoid unnecessary processing of already formatted data. If `cache = TRUE`, the function will read and process the data only the first time (or if the original data is later changed). A processed and cached version of the data is saved with suffix ".rds". The cached version is always compared to the original file by its MD5 sum and is always updated if needed.

**Examples**

```
## Not run:
# Create a csv file with example data in a temporary directory
fl <- tempfile("ex_data", fileext = ".csv2")
write.csv2(incadata::ex_data, fl)

# First time the file is read from csv2
use_incadata(fl)
dir(tempdir) # a cache file is saved along the original csv2-file
use_incadata(fl) # Next time file loaded from cache

## End(Not run)
```

---

`width`*Calculate the width of the range of x*

---

**Description**

Calculate the width of the range of x

**Usage**

```
width(x)
```

**Arguments**

x                    object to calculate range for

**Value**

The width of the range of x as integer.

**Examples**

```
width(1:10)
width(c(6748, 234, 2456, 5678))
width(sample(345))
```

# Index

## \*Topic **datasets**

- ex\_data, 13
  
- as.character, 9
- as.Date, 4
- as.Dates, 3, 5
- as.incadata, 4, 21
- as.numeric, 9
- as.pin, 5
- as\_numeric, 6
  
- best\_match, 7, 8
- between, 17
  
- c, 6
- clean\_text, 7, 8, 8
- create\_s3\_method, 9
- create\_s3\_print (create\_s3\_method), 9
- cut.integer, 10
  
- decode, 5
- documents, 10
  
- ex\_data, 13
- exceed\_threshold, 5, 6, 11
- exportr, 12
  
- find\_documents, 13
- find\_links, 14
- find\_register, 15
  
- id, 5, 15
- incallogical2logical (is.incallogical), 16
- integer, 18
- is.inca, 16
- is.incadata (as.incadata), 4
- is.incallogical, 5, 16
- is.scalar\_in, 17
- is.scalar\_in01 (is.scalar\_in), 17
- is.wholenumber, 18
- is\_numeric (as\_numeric), 6

- lownames, 18
- lt, 19
  
- max, 20
- min, 20
  
- next\_method, 19
- NextMethod, 9
  
- package\_dependencies, 12
- pmax, 20
- pmin, 20
- POSIXct, 3
- psum, 20
  
- read.csv2, 21
- regex, 20
  
- specify\_missing, 20
- sum, 20
  
- use\_incadata, 21
  
- width, 22