

# Package: encode (via r-universe)

September 5, 2024

**Type** Package

**Title** Represent Ordered Lists and Pairs as Strings

**Version** 0.3.6

**Author** Tim Bergsma

**Maintainer** Tim Bergsma <bergsmat@gmail.com>

**Description** Interconverts between ordered lists and compact string notation. Useful for capturing code lists, and pair-wise codes and decodes, for text storage. Analogous to factor levels and labels. Generics encode() and decode() perform interconversion, while codes() and decodes() extract components of an encoding. The function encoded() checks whether something is interpretable as an encoding. If a vector has an encoded 'guide' attribute, as\_factor() uses it to coerce to factor.

**License** GPL-3

**LazyData** TRUE

**Suggests** magrittr

**RoxygenNote** 6.0.1

**Repository** <https://bergsmat.r-universe.dev>

**RemoteUrl** <https://github.com/bergsmat/encode>

**RemoteRef** HEAD

**RemoteSha** eae3872986fe7bb58715374cff44a1d5e929850e

## Contents

as_factor . . . . .	2
codes . . . . .	3
codes.default . . . . .	3
decode . . . . .	4
decode.data.frame . . . . .	4
decode.default . . . . .	5
decodes . . . . .	6

decodes.default . . . . .	6
encode . . . . .	7
encode.character . . . . .	8
encode.default . . . . .	9
encode.list . . . . .	9
encoded . . . . .	10
encoded.default . . . . .	11

<b>Index</b>	<b>12</b>
--------------	-----------

---

as_factor	<i>Coerce to Factor using Encoding if Present</i>
-----------	---

---

## Description

Coerces to factor, blending levels with encoding, if present as a 'guide' attribute. Vectors without encodings (or with empty encodings) acquire levels equal to `unique(x)` (notice that storage order controls presentation order). Vectors with non-empty encodings are decoded after harmonizing the encoding and the actual data. Factors with encodings defer to order and display value of the encoding as much as possible. Missing levels are supplied. Unused levels are removed. Other attributes beside 'class' and 'levels' are preserved.

## Usage

```
as_factor(x)
```

## Arguments

x                    vector or factor

## Value

factor

## See Also

Other decode: [decode.data.frame](#), [decode.default](#), [decode](#)

## Examples

```
library(magrittr)
foo <- c(1, 2, NA, 4, 5)
as_factor(foo)
as_factor(factor(foo))
as_factor(as.factor(foo))
as_factor(structure(foo, guide = '...'))
as_factor(structure(foo, guide = '//5//'))
as_factor(structure(foo, guide = '//5/bar//'))
as_factor(structure(foo, guide = '//5/bar//6/baz//'))
as_factor(structure(factor(foo), guide = '//5/bar//'))
```

```
as_factor(structure(factor(foo), guide = '//5/bar//')) %>% sort
as_factor(structure(factor(foo), guide = '...'))
as_factor(structure(factor(foo), guide = '//1/bar//5/bar//'))
```

---

codes	<i>Extract Codes from an Object</i>
-------	-------------------------------------

---

### Description

Extracts Codes from an object. Default method is supplied.

### Usage

```
codes(x, ...)
```

### Arguments

x	object
...	passed arguments

### See Also

[codes.default](#)

Other codes: [codes.default](#)

---

codes.default	<i>Extract Codes by Default from an Object</i>
---------------	--

---

### Description

Extracts codes from an object using the default method.

### Usage

```
## Default S3 method:
codes(x, simplify = TRUE, ...)
```

### Arguments

x	object
simplify	whether to convert length one list to vector
...	passed arguments

**Value**

list, or vector if simplify = TRUE

**See Also**

Other codes: [codes](#)

---

decode	<i>Decode an Object</i>
--------	-------------------------

---

**Description**

Decodes an object. Default method supplied.

**Usage**

```
decode(x, ...)
```

**Arguments**

x	object
...	passed arguments

**See Also**

[decode.default](#)

Other decode: [as\\_factor](#), [decode.data.frame](#), [decode.default](#)

---

decode.data.frame	<i>Decode Data Frame.</i>
-------------------	---------------------------

---

**Description**

Decodes a data.frame. Calls as\_factor() for each column with an encoded guide attribute.

**Usage**

```
## S3 method for class 'data.frame'  
decode(x, ...)
```

**Arguments**

x	inherits data.frame
...	ignored

**Value**

same class as x

**See Also**

Other decode: [as\\_factor](#), [decode.default](#), [decode](#)

---

decode.default	<i>Decode an Object by Default</i>
----------------	------------------------------------

---

**Description**

Decodes an object using the default method. Typically x is a character vector containing codes that can be extracted from encoding. Corresponding decodes are returned as a factor with levels of unique decodes. If encoding is NULL, it is replaced with an encoding such that levels and labels are both unique(x). Duplicate codes are ignored. Duplicate decodes are collapsed (combined to a single level).

**Usage**

```
## Default S3 method:  
decode(x, encoding = NULL, ...)
```

**Arguments**

x	object
encoding	length one character that is itself encoded
...	passed arguments

**Value**

factor

**See Also**

Other decode: [as\\_factor](#), [decode.data.frame](#), [decode](#)

decodes *Extract Decodes from an Object*

---

**Description**

Extracts decodes from an object. Default method is supplied.

**Usage**

```
decodes(x, ...)
```

**Arguments**

x	object
...	passed arguments

**See Also**

[decodes.default](#)

Other decodes: [decodes.default](#)

---

decodes.default *Extract Decodes by Default from an Object*

---

**Description**

Extracts decodes from an object using the default method.

**Usage**

```
## Default S3 method:  
decodes(x, simplify = TRUE, ...)
```

**Arguments**

x	object
simplify	whether to convert length one list to vector
...	passed arguments

**Value**

list, or vector if simplify = TRUE

**See Also**

Other decodes: [decodes](#)

---

`encode`*Encode Factor-like Levels and Labels as a Simple String*

---

### Description

For compact storage, `encode` combines a set of levels and labels (codes and decodes) into a simple string. The default method converts its argument to character. The `list` method operates element-wise, expecting an equal number of label elements, each of which have the same length as the corresponding element of `x`.

### Usage

```
encode(x, ...)
```

### Arguments

<code>x</code>	object
<code>...</code>	passed arguments

### Details

An empty 'encoding' consists of four identical characters, e.g. `////`.

A non-empty encoding must be at least 5 characters long, beginning and ending with two instances of `sep` e.g. `//1//`. Levels are likewise separated from each other by double separators, e.g. `//1//2//`.

If a label (decode) is available for a level, it follows the corresponding level: the two are separated by a single instance of `sep`, e.g. `//1/a//2/b//`.

Encodings may be combined as elements of a character vector, i.e. an encoded vector. Choice of separator may vary among elements, but must be consistent within elements.

Labels (decodes) may be zero-length, but not levels (codes), e.g. `//1//` is valid but `///a//` is not. A zero-length decode is extracted as an empty string.

Duplicate levels (codes) result in a warning for `encode()`, and are otherwise silently ignored. Duplicate labels (decodes) result in case-collapsing.

### See Also

[encode.character](#) [encode.default](#) [encode.list](#) [codes](#) [decodes](#) [decode](#) [encoded](#)

Other encode: [encode.character](#), [encode.default](#), [encode.list](#)

### Examples

```
a <- encode(  
  x = list(  
    c('M', 'F'),  
    c(1:4)  
  ),
```

```

labels = list(
  c('male', 'female'),
  c('caucasian', 'asian', 'african', NA)
)
)
b <- encode(c(1:2), c('pediatric', 'adult'))
a
b
c <- c('a', NA, '##b##')
encoded(a)
encoded(b)
encoded(c)
encoded(' //4// ')
codes(a)
codes(b)
codes(b, simplify=FALSE)
codes(c)
codes('..1..')
decodes(a)
decodes(b)
decodes(c)
decode(1:4, '//1/a//2/b//3/c//')
decode(1:4, '//1/a//1/b//3/c//') # duplicate code: ignored
decode(1:4, '//1/a//2/a//3/c//') # duplicate decode: collapsed
# encode(c(1,1,2,3), c('a', 'b', 'c', 'd')) Warning: duplicate codes

```

---

encode.character

*Encode Character.*

---

## Description

Encodes character. If sep is NULL, it is replaced with the first of these that is not otherwise present in the result: /!\~!@#&

## Usage

```
## S3 method for class 'character'
encode(x, labels = NULL, sep = NULL, ...)
```

## Arguments

x	object
labels	same length as x if supplied
sep	a single character not present in x or labels
...	passed arguments



**Value**

character

**See Also**

Other encode: [encode.default](#), [encode.list](#), [encode](#)

---

encode.default	<i>Encode Default.</i>
----------------	------------------------

---

**Description**

Encodes using default method: coerces to character and and encodes the result.

**Usage**

```
## Default S3 method:  
encode(x, labels = NULL, ...)
```

**Arguments**

x	object
labels	same length as x if supplied
...	passed arguments

**Value**

character

**See Also**

Other encode: [encode.character](#), [encode.list](#), [encode](#)

---

encode.list	<i>Encode a List</i>
-------------	----------------------

---

**Description**

Encodes a list.

**Usage**

```
## S3 method for class 'list'  
encode(x, labels = NULL, ...)
```

**Arguments**

x	object
labels	same length as x if supplied
...	passed arguments

**Value**

list

**See Also**

Other encode: [encode.character](#), [encode.default](#), [encode](#)

---

encoded

*Check If Object is Encoded*

---

**Description**

Checks if object is encoded.

**Usage**

```
encoded(x, ...)
```

**Arguments**

x	object
...	passed arguments

**See Also**

[encoded.default](#)

Other encoded: [encoded.default](#)

---

encoded.default	<i>Check If Default Object is Encoded</i>
-----------------	---

---

**Description**

Checks if object is encoded, using default methodology. Always returns logical, telling whether the corresponding element represents an encoding of levels and labels. Objects with zero length give FALSE.

**Usage**

```
## Default S3 method:  
encoded(x, ...)
```

**Arguments**

x	object
...	passed arguments

**Value**

logical

**See Also**

Other encoded: [encoded](#)

# Index

`as_factor`, [2](#), [4](#), [5](#)

`codes`, [3](#), [4](#), [7](#)

`codes.default`, [3](#), [3](#)

`decode`, [2](#), [4](#), [5](#), [7](#)

`decode.data.frame`, [2](#), [4](#), [4](#), [5](#)

`decode.default`, [2](#), [4](#), [5](#), [5](#)

`decodes`, [6](#), [6](#), [7](#)

`decodes.default`, [6](#), [6](#)

`encode`, [7](#), [9](#), [10](#)

`encode.character`, [7](#), [8](#), [9](#), [10](#)

`encode.default`, [7](#), [9](#), [9](#), [10](#)

`encode.list`, [7](#), [9](#), [9](#)

`encoded`, [7](#), [10](#), [11](#)

`encoded.default`, [10](#), [11](#)