

# Package ‘SPB’

August 14, 2022

**Title** Simple Progress Bars for Procedural Coding

**Version** 1.0

**Maintainer** Fabio Ashtar Telarico <Fabio-Ashtar.Telarico@fdv.uni-lj.si>

**Description** Provides a simple progress bar to use for basic and advanced users that suits all those who prefer procedural programming. It is especially useful for integration into markdown files thanks to the progress bar's customisable appearance.

**Depends** methods, utils, crayon

**Imports** knitr

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**NeedsCompilation** no

**Author** Fabio Ashtar Telarico [aut, cre]  
(<<https://orcid.org/0000-0002-8740-7078>>)

**Repository** CRAN

**Date/Publication** 2022-08-14 08:50:07 UTC

## R topics documented:

create_pb . . . . .	2
print,simple.progress.bar-method . . . . .	4
simple.progress.bar . . . . .	5
summary,simple.progress.bar-method . . . . .	5
update_pb . . . . .	6

<b>Index</b>	<b>8</b>
--------------	----------

---

`create_pb`*Function to create a simple progress bar*

---

### Description

Function to create a simple progress bar

### Usage

```
create_pb(  
    length,  
    remaining = "-",  
    done = "=",  
    percentage = TRUE,  
    message = TRUE,  
    custom.message = NULL,  
    trim = 6L,  
    print = TRUE,  
    colour = "red",  
    bg.colour = "white"  
)
```

### Arguments

<code>length</code>	Length of the progress bar. For instance, in a loop it would be the number of iterations to complete.
<code>remaining</code>	The character (string of unitary length) used to represent operations still to do.
<code>done</code>	The character (string of unitary length) used to represent operations already completed
<code>percentage</code>	Logical   Whether to print the percentage progress after the bar. Defaults to TRUE
<code>message</code>	Logical   Whether to print a message before the progress bar. Defaults to TRUE
<code>custom.message</code>	Optional message to be printed before the progress bar. Defaults to NULL, which prints simply 'Progress status: ',
<code>trim</code>	How many points to trim from the maximum length of the terminal output. Default to 6L
<code>print</code>	Logical   Whether to print the progress bar after its initialisation. Defaults to TRUE
<code>colour</code>	Styling of the progress bar object. It can be a single string, in which case it applies to all the part of the progress bar (message, bar and percentage). It can also be a vector of length 3, in which case each string applies to one of the items (respectively message, bar and percentage). The strings must be taken amongst the following values: <ul style="list-style-type: none"><li>• black;</li><li>• blue;</li></ul>

	<ul style="list-style-type: none"> <li>• cyan;</li> <li>• green;</li> <li>• magenta;</li> <li>• red;</li> <li>• white;</li> <li>• yellow.</li> </ul>
bg.colour	<p>Styling of the progress bar object's background. It can be a single string, in which case it applies to all the part of the progress bar (message, bar and percentage). It can also be a vector of length 3, in which case each string applies to one of the items (respectively message, bar and percentage). The strings must be taken amongst the following values:</p> <ul style="list-style-type: none"> <li>• Black;</li> <li>• Blue;</li> <li>• Cyan;</li> <li>• Green;</li> <li>• Magenta;</li> <li>• Red;</li> <li>• White;</li> <li>• Grey;</li> <li>• Yellow.</li> </ul>

**Value**

An object of the S4 class `simple.progress.bar` containing:

message	A string representing the message to be printed together with the bar
bar	The progress bar
percentage	Progress in percentage
length	Length of the progress bar (integer)
remaining	Character representing the remaining operations
done	Character representing the operations already completed
style_msg	Styling of message using crayon
style_bar	Styling of bar using crayon
style_perc	Styling of percentage using crayon

**Author(s)**

Telarico, Fabio Ashtar

**Examples**

```
# Example without custom colours or custom message
pb<-create_pb(length=10, print=TRUE)

# Example without custom colours but with custom message
pb<-create_pb(length=10, custom.message = 'Custom progress message:',
print = TRUE)

# Example with custom colours and custom message
pb<-create_pb(length=10, custom.message = 'Custom progress message:',
print = TRUE,colour = c('magenta','red','blue'))
```

---

```
print,simple.progress.bar-method
Print method for simple progress bars
```

---

**Description**

Print method for simple progress bars

**Usage**

```
## S4 method for signature 'simple.progress.bar'
print(x)
```

**Arguments**

x                    A simple progress bar

**Value**

Nothing. Prints the progress bar to the console

**Author(s)**

Telarico, Fabio Ashtar

**Examples**

```
# Example without custom colours or custom message
pb<-create_pb(length=10, print=FALSE)
print(pb)
```

---

simple.progress.bar    *Class of a simple progress bar*

---

**Description**

Class of a simple progress bar

**Author(s)**

Telarico, Fabio Ashtar

**Examples**

```
pb<-create_pb(length=10, print=FALSE)
class(pb)
```

---

summary, simple.progress.bar-method  
*Summary method for simple progress bars*

---

**Description**

Summary method for simple progress bars

**Usage**

```
## S4 method for signature 'simple.progress.bar'
summary(object)
```

**Arguments**

object            A simple progress bar

**Value**

A summary including:

message	The message printed before the bar (if any)
status	The advancement status of the bar
max	The length of the bar
percentage	The advancement status of the bar in percentage points

The summary is also printed to the console as a kable (if knitr is installed). Otherwise it prints out as a normal table.

**Author(s)**

Telarico, Fabio Ashtar

**Examples**

```
# Example without custom message
pb<-create_pb(length=10, print=FALSE)
summary(pb)

# Example with a custom message
pb<-create_pb(length=43, print=FALSE,custom.message='Custom pb')
summary(pb)

# Example with a custom message and an updated value
pb<-create_pb(length=11, print=FALSE,custom.message='A new value:')
pb<-update_pb(pb,6)
summary(pb)
```

---

update\_pb

*Function to update the progress of a simple progress bar*

---

**Description**

Function to update the progress of a simple progress bar

**Usage**

```
update_pb(pb, value)
```

**Arguments**

pb	Name of the progress bar to update, previously initialised with create_pb()
value	Level of progress of the bar

**Value**

Prints the progress bar with the new value and returns an object of the S4 class `simple.progress.bar` containing:

message	A string representing the message to be printed together with the bar
bar	The progress bar
percentage	Progress in percentage
length	Length of the progress bar (integer)
remaining	Character representing the remaining operations
done	Character representing the operations already completed

style_msg	Styling of message using crayon
style_bar	Styling of bar using crayon
style_perc	Styling of percentage using crayon

**Author(s)**

Telarico, Fabio Ashtar

**Examples**

```
# A single bar in a loop
pb<-create_pb(length=10, print=TRUE)
for(i in 1:10){
  cat('This is the number',i,'\n')
  pb<-update_pb(pb,i)
  Sys.sleep(.3)
  cat(pb$value,'operation completed\n')
}

# Two progress bars in multiple loops
pb_for<-create_pb(length=3, print=TRUE,colour='red')
for(i in 1:3){
  cat('This is the number',i,'\n')
  pb_for<-update_pb(pb_for,i)
  Sys.sleep(1)

  pb_while<-create_pb(length=i, print=TRUE,colour='blue')
  while(pb_while$value<i){
    cat('This is a subtask \n')
    pb_while<-update_pb(pb_while,pb_while$value+1)
    Sys.sleep(1)
  }
}
```

# Index

`create_pb`, [2](#)

`print`, `simple.progress.bar-method`, [4](#)

`simple.progress.bar`, [5](#)

`summary`, `simple.progress.bar-method`, [5](#)

`update_pb`, [6](#)