Package: bones (via r-universe)

October 16, 2026

Type Package

Title Content-Shaped Loading Placeholders for 'shiny' Outputs

Version 0.1.1

Description Shows a grey placeholder in the shape of the content while a 'shiny' output loads, in place of a spinner. The user can thus see what is coming, and not only that something happens. The shape comes from the type of the output. The placeholder keeps the space of the content, so the page does not move when the content arrives. When an output calculates again, the old content stays on the screen, dimmed, and a placeholder does not replace it.

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports htmltools (>= 0.5.4), utils

Suggests bslib, DT, echarts4r, gt, highcharter, plotly, reactable, rhandsontable, shiny (>= 1.7.0), shinytest2, testthat (>= 3.0.0), withr

URL https://github.com/tenmeh/bones, https://tenmeh.github.io/bones/

BugReports https://github.com/tenmeh/bones/issues

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Tanmay Chanda [aut, cre, cph]

Maintainer Tanmay Chanda <tanmaychanda96@gmail.com>

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

Date/Publication 2026-10-16 12:30:02 UTC

RemoteUrl https://github.com/cran/bones

RemoteRef HEAD

RemoteSha 6a67c978abb5cb195571f204551af57d1c7105a1

Contents

bones_auto 2
bones_block 3
bones_defaults 4
bones_dependency 6
bones_skeleton 6
withBones 8

Index **11**

`bones_auto` *Wrap every Shiny output in a UI in a skeleton placeholder*

Description

Gives skeletons to an app with no change to each output. `bones_auto()` looks through a UI and puts each Shiny output in `withBones()`. Each one gets the shape of its output, as with `withBones(output)`: a `plotOutput()` gets a chart, a `tableOutput()` gets rows, a DT table gets the shape of DT, and so on.

Usage

```
bones_auto(ui, ..., exclude = NULL)
```

Arguments

`ui`	A UI: a page, a tag, or a tag list.
`...`	Arguments for `withBones()`, used for every output that is wrapped: for example `stale = FALSE` or `animation = "pulse"`. Not `type`, because each output gets the shape of its own kind.
`exclude`	The ids of outputs to leave alone.

Details

Give it the whole page, or only a part of it:

```
ui <- bones_auto(page_sidebar(...))
ui <- page_sidebar(sidebar = ..., bones_auto(navset_card_tab(...)))
```

Value

`ui`, with each output in a placeholder container. A page keeps its class and its theme.

What it leaves alone

- An output that is already in `withBones()`. To give one output its own shape or options, wrap it yourself: `bones_auto()` keeps that wrapper.
- An output whose id is in `exclude`.
- An inline output, such as `textOutput("n", inline = TRUE)`. The wrapper is a block, so it would break the line of text.

A `uiOutput()` can hold anything, so it gets the text shape. Wrap it yourself to give it another shape, such as `type = "cards"`.

`bones_auto()` sees only the UI that you give it. Outputs that a `renderUI()` makes later are not in it. Call `bones_auto()` on the UI inside that `renderUI()` to give them skeletons too.

Examples

```
if (requireNamespace("shiny", quietly = TRUE)) {
  library(shiny)

  ui <- bones_auto(fluidPage(
    plotOutput("chart"),
    tableOutput("results"),
    textOutput("count", inline = TRUE),         # inline: left alone
    withBones(uiOutput("cards"), type = "cards") # wrapped already: kept
  ))
}
```

---

bones_block — *A grey block to build your own placeholder*

Description

The building block of a placeholder that you make yourself. Give the blocks to `withBones()` as `skeleton`, laid out with any tags. Each block takes the colours, the corner radius and the animation of bones, so your placeholder moves and looks like the built-in shapes.

Usage

```
bones_block(width = "100%", height = "0.75rem", shape = c("rect", "circle"))
```

Arguments

`width`	A CSS width, or a number of pixels. `"100%"` fills the width.
`height`	A CSS height, or a number of pixels. The default is the height of one line of text in the built-in shapes.
`shape`	`"rect"` for a block with the corner radius of bones, or `"circle"` for a round block, such as an avatar. Give a circle the same width and height.

Details

The tags at the top level of `skeleton` stack with a small gap. Inside a tag of your own, you set the layout: for example `display: flex` for a row, or `display: grid; gap: 0.5rem` for a column.

Value

An htmltools::tag.

See Also

`withBones()` and its section "Your own placeholder".

Examples

```
# A line of text, 60% wide.
bones_block("60%")

# An avatar.
bones_block(48, 48, shape = "circle")

if (requireNamespace("shiny", quietly = TRUE)) {
  library(shiny)

  # A list of three people: an avatar, a name and a line of text each.
  person <- div(
    style = "display: flex; gap: 0.75rem; align-items: center;",
    bones_block(40, 40, shape = "circle"),
    div(
      style = "flex: 1; display: grid; gap: 0.4rem;",
      bones_block("40%"), bones_block("80%", "0.6rem")
    )
  )
  withBones(uiOutput("people"), skeleton = tagList(person, person, person))
}
```

`bones_defaults` *Set the placeholder defaults for the session*

Description

Sets the defaults that `withBones()` uses for the rest of the session. An argument that is `NULL` does not change its default. An argument that you give to `withBones()` is stronger than a default from here.

Usage

```
bones_defaults(
  animation = NULL,
  color = NULL,
  highlight = NULL,
  radius = NULL,
  speed = NULL,
  stale = NULL,
  delay = NULL,
  min_time = NULL,
  remember = NULL
)
```

Arguments

`animation` How the placeholder moves:

- `"wave"`: a band of light moves along each bar.
- `"pulse"`: the whole placeholder fades out and in.
- `"cascade"`: each bar, cell or line lights up a little after the one before, so a ripple runs through the shape.
- `"sweep"`: one band of light crosses the whole placeholder, over every mark at once, the lines and dots of a chart too.
- `"none"`: no movement.

Users who ask their system to reduce motion always get `"none"`.

`color` The colour of the placeholder, as a CSS colour. By default it comes from the theme of the page: the text colour of the bslib theme, made faint. A colour given here is stronger than the theme.

`highlight` The colour of the band of light in the `"wave"` animation, and of the header row of a table. By default it comes from the theme, as `color` does.

`radius` The CSS corner radius of the placeholder bars. By default it follows the rounding of the theme.

`speed` The time of one animation cycle, in seconds.

`stale` The default for the `stale` argument of `withBones()`.

`delay, min_time` The defaults for the `delay` and `min_time` arguments of `withBones()`, in milliseconds.

`remember` The default for the `remember` argument of `withBones()`.

Value

The old values, invisibly, in the form that `options()` uses. Give them to `options()` to restore them. Do not give them to `bones_defaults()`: `NULL` there means "do not change", so an option that had no value before would keep the new value.

Examples

```
old <- bones_defaults(animation = "pulse", radius = "0.5rem")
bones_skeleton("text")

options(old)
```

---

bones_dependency *The bones HTML dependency*

Description

`withBones()` and `bones_skeleton()` attach this for you. It is exported for HTML that you make yourself with the `bones-*` classes, which needs the stylesheet and the script.

Usage

```
bones_dependency()
```

Value

An `htmltools::htmlDependency()`.

Examples

```
bones_dependency()
```

---

bones_skeleton *Make a skeleton placeholder*

Description

Makes the grey placeholder on its own, with no output in it. You do not need this for a Shiny output, because `withBones()` makes the placeholder for you. Use this function where there is no Shiny output: for example, inside a `renderUI()` while you get data, or to look at a shape while you design a page.

Usage

```
bones_skeleton(
  type = "text",
  rows = 6L,
  cols = 4L,
  lines = 3L,
  n = 3L,
  height = NULL
)
```

Arguments

type	The shape: "text", "table", "plot", "cards", "value", or one of the chart or table shapes above.
rows	The number of body rows, for type = "table" and the table package shapes.
cols	The number of columns, for type = "table" and the table package shapes.
lines	The number of lines, for type = "text".
n	The number of cards or values, for type = "cards" and "value".
height	A CSS height for the placeholder, or a number of pixels. NULL gives a default for the type, the same one that withBones() uses.

Details

The skeleton uses the animation and the colours from bones_defaults().

Value

An htmltools::tag, with the stylesheet attached.

Chart shapes

A chart has a shape for each common kind: "bar", "line", "scatter", "area", "histogram", "pie", "heatmap", "map", "network", "timeline" and "wordcloud". "plot" is the same as "bar". withBones() finds the shape of a map, network, timeline or word cloud widget from its output. plotOutput() cannot say which kind it will hold, because renderPlot() decides that later, on the server. So you name the kind yourself.

Table shapes

A table from a table package has a shape that copies that package: "dt", "reactable", "gt" and "rhandsontable". withBones() finds it from the output by itself.

Examples

```
bones_skeleton("text", lines = 4)
bones_skeleton("table", rows = 8, cols = 5)
bones_skeleton("plot", height = "300px")
bones_skeleton("scatter", height = "300px")
bones_skeleton("dt", rows = 10, cols = 5)
```

withBones *Wrap a Shiny output in a skeleton placeholder*

Description

Shows a grey placeholder in the shape of the content while the output loads for the first time. When the output calculates again, the old content stays on the screen and is dimmed. The user is possibly reading that content, and grey boxes in its place would remove information and add none.

Usage

```
withBones(
  ui,
  type = NULL,
  rows = 6L,
  cols = 4L,
  lines = 3L,
  n = 3L,
  height = NULL,
  animation = NULL,
  stale = NULL,
  delay = NULL,
  min_time = NULL,
  remember = NULL,
  skeleton = NULL
)
```

Arguments

`ui`	A Shiny output, such as `plotOutput("chart")`.
`type`	The shape of the skeleton: `"text"`, `"table"`, `"plot"`, `"cards"`, `"value"`, a chart shape (`"bar"`, `"line"`, `"scatter"`, `"area"`, `"histogram"`, `"pie"`, `"heatmap"`, `"map"`, `"network"`, `"timeline"`, `"wordcloud"`), or a table package shape (`"dt"`, `"reactable"`, `"gt"`, `"rhandsontable"`). `NULL` (the default) gets the shape from `ui`.
`rows, cols`	The number of body rows and columns, when `type` is `"table"` or a table package shape.
`lines`	The number of lines, when `type` is `"text"`.
`n`	The number of cards or values, when `type` is `"cards"` or `"value"`.
`height`	A CSS height for the placeholder, or a number of pixels, as in Shiny. `NULL` uses the height of the output. If the output has no height, `NULL` uses a default for the type.
`animation`	`"wave"`, `"pulse"`, `"cascade"`, `"sweep"` or `"none"`. See `bones_defaults()` for what each one does. `NULL` uses the value from `bones_defaults()`.

stale — TRUE keeps the old content on the screen, dimmed, while the output calculates again. FALSE shows the skeleton again. NULL uses the value from bones_defaults(), which is TRUE if you did not set it.

delay — The time in milliseconds before the skeleton, or the dimming, appears. A load that ends sooner shows nothing, so a fast output does not flicker. The space of the content is kept from the start. NULL uses the value from bones_defaults(), which is 300 if you did not set it.

min_time — The shortest time in milliseconds that the skeleton, or the dimming, stays on the screen once it has appeared. A load that ends a moment after delay thus does not flash a skeleton for one frame. NULL uses the value from bones_defaults(), which is 500 if you did not set it.

remember — TRUE stores the real height of the content in the browser, and keeps that height on the next visit in place of the estimate, so the page does not move when the content arrives. Only a height from an estimate is stored: a height from height, or from the output itself, is exact already. The height is kept per page and per output id, and it is used only when the output has almost the same width as when it was measured. For a plotly, echarts4r or highcharter output, remember also stores the kind of the chart (see "Shape"). NULL uses the value from bones_defaults(), which is TRUE if you did not set it.

skeleton — Your own placeholder, in place of a built-in shape: a tag or a tag list, made of bones_block() blocks and any layout around them. See "Your own placeholder". Give type or skeleton, not both.

Value

ui, in a placeholder container.

Shape

The shape comes from the output that you give: plotOutput() gets the shape of a chart, tableOutput() gets rows and columns, and textOutput() gets lines of text. Use type to set a different shape. You must do this for uiOutput(), because its content is not known before it arrives.

A plotOutput() gets the shape of a bar chart. If the chart is of another kind, give that kind: "line", "scatter", "area", "histogram", "pie" or "heatmap". The output cannot tell, because renderPlot() decides the kind later, on the server.

A table from a table package gets the shape of that package, found from its output: DT::DTOutput() gets "dt", reactable::reactableOutput() gets "reactable", gt::gt_output() gets "gt", and rhandsontable::rHandsontableOutput() gets "rhandsontable". Each shape copies the parts of its package, such as the search box of DT. Set rows to the number of rows on a page: DT and reactable show 10 by default. Before gt 1.0.0, gt_output() did not mark its output as a gt table, so with an older gt, give type = "gt".

A visualisation widget gets the shape of its kind, found from its output in the same way. A map (leaflet, tmap, mapview, mapdeck and others) gets "map". A network or a diagram (visNetwork, DiagrammeR, networkD3, collapsibleTree) gets "network". timevis gets "timeline", wordcloud2 gets "wordcloud", and dygraphs gets "line". ggmap and a ggplot2 map draw into a plotOutput(), so give them type = "map".

A plotly, echarts4r or highcharter output can draw any kind of chart. With no `type`, it finds its kind by itself, but only when its first value arrives in the browser: the spec of the chart names the kind of each series. The first skeleton thus has the bar shape. The next loads, with `stale = FALSE`, show the shape of the kind: `"bar"`, `"line"`, `"area"`, `"scatter"`, `"histogram"`, `"pie"`, `"heatmap"`, `"map"`, `"network"` or `"wordcloud"`. With `remember`, the kind is also stored in the browser, so on the next visit the first skeleton has that shape too. The first series that names a kind decides. A chart of a kind with no shape, such as a box plot, keeps the bar shape. A `type` that you give always wins. Other widgets that can draw any chart, such as ggiraph, keep the bar shape: give their kind with `type`.

Your own placeholder

When no built-in shape fits, give your own with `skeleton`. Build it from `bones_block()`, a grey block that takes the colours and the animation of bones, and lay the blocks out with any tags. The tags at the top level stack with a small gap. The placeholder behaves as a built-in shape does: it waits for `delay`, it stays for `min_time`, it keeps the space of the content, and the old content stays when `stale` is `TRUE`.

Layout

The wrapper keeps the space of the output until the content arrives, so the page does not move. If the output sets its own height, as `plotOutput()` does, the wrapper uses that height. If not, the height comes from `rows`, `lines` or `n`, or you can set it with `height`. When the content arrives, the content sets the height.

Examples

```
if (requireNamespace("shiny", quietly = TRUE)) {
  library(shiny)

  withBones(plotOutput("chart"))
  withBones(tableOutput("results"), rows = 8, cols = 5)
  withBones(uiOutput("cards"), type = "cards", n = 4)
  withBones(plotOutput("trend"), type = "line")

  # Your own placeholder: an avatar and two lines of text.
  withBones(
    uiOutput("profile"),
    skeleton = div(
      style = "display: flex; gap: 1rem; align-items: center;",
      bones_block("3rem", "3rem", shape = "circle"),
      div(
        style = "flex: 1; display: grid; gap: 0.5rem;",
        bones_block("60%"), bones_block("40%")
      )
    )
  )
}
```

Index

bones_auto, 2
bones_block, 3
bones_block(), *9, 10*
bones_defaults, 4
bones_defaults(), *7–9*
bones_dependency, 6
bones_skeleton, 6
bones_skeleton(), *6*

htmltools::htmlDependency(), *6*
htmltools::tag, *4*, *7*

options(), *5*

withBones, 8
withBones(), *2–7*