skimr

`skimr` provides a frictionless approach to summary statistics which conforms to the principle of least

surprise, displaying summary statistics the user can skim quickly to understand their data. It handles different data types and returns a `skim_df` object which can be included in a pipeline or displayed nicely for the human reader.

skimr

notes and vignettes carefully.

Installation

The current released version of `skimr` can be installed from CRAN. If you wish to install the current build of the next release you can do so using the following:

```
# install.packages("devtools")
devtools::install_github("ropensci/skimr")
```

The APIs for this branch should be considered reasonably stable but still subject to change if an issue is discovered.

To install the version with the most recent changes that have not yet been incorporated in the main branch (and may not be):

```
devtools::install_github("ropensci/skimr", ref = "develop")
```

Do not rely on APIs from the develop branch, as they are likely to change.

Skim statistics in the console

`skimr`:

- Provides a larger set of statistics than `summary()`, including missing, complete, n, and sd.
- reports each data types separately
- handles dates, logicals, and a variety of other types
- supports spark-bar and spark-line based on the pillar package.

Separates variables by class:

```
skim(chickwts)

## ── Data Summary ────────────────────────
##                              Values
## Name                         chickwts
## Number of rows               71
```

skimr

```
##   factor                   1
##   numeric                  1
## ________________________
## Group variables            None
##
## ── Variable type: factor ─────────────────────────────────────
##   skim_variable n_missing complete_rate ordered n_unique top_co
## 1 feed                  0             1 FALSE          6 soy: 1
##
## ── Variable type: numeric ────────────────────────────────────
##   skim_variable n_missing complete_rate mean   sd  p0  p25 p50
## 1 weight                0             1 261. 78.1 108 204. 258
```

Presentation is in a compact horizontal format:

```
skim(iris)

## ── Data Summary ────────────────────────
##                            Values
## Name                       iris
## Number of rows             150
## Number of columns          5
## _______________________
## Column type frequency:
##   factor                   1
##   numeric                  4
## ________________________
## Group variables            None
##
## ── Variable type: factor ─────────────────────────────────────
##   skim_variable n_missing complete_rate ordered n_unique top_co
## 1 Species               0             1 FALSE          3 set: 5
##
## ── Variable type: numeric ────────────────────────────────────
##   skim_variable n_missing complete_rate mean    sd  p0 p25  p50
## 1 Sepal.Length          0             1 5.84 0.828 4.3 5.1 5.8
## 2 Sepal.Width           0             1 3.06 0.436 2   2.8 3
## 3 Petal.Length          0             1 3.76 1.77  1   1.6 4.35
## 4 Petal.Width           0             1 1.20 0.762 0.1 0.3 1.3
```

Built in support for strings, lists and other column

skimr

```
skim(dplyr::starwars)

## ── Data Summary ────────────────────────
##                            Values
## Name                       dplyr::starwars
## Number of rows             87
## Number of columns          14
## _______________________
## Column type frequency:
##   character                8
##   list                     3
##   numeric                  3
## ________________________
## Group variables            None
##
## ── Variable type: character ──────────────────────────────────
##   skim_variable n_missing complete_rate min max empty n_unique
## 1 name                  0         1       3  21     0       87
## 2 hair_color            5         0.943   4  13     0       11
## 3 skin_color            0         1       3  19     0       31
## 4 eye_color             0         1       3  13     0       15
## 5 sex                   4         0.954   4  14     0        4
## 6 gender                4         0.954   8   9     0        2
## 7 homeworld            10         0.885   4  14     0       48
## 8 species               4         0.954   3  14     0       37
##
## ── Variable type: list ───────────────────────────────────────
##   skim_variable n_missing complete_rate n_unique min_length max
## 1 films                 0             1       24          1
## 2 vehicles              0             1       11          0
## 3 starships             0             1       16          0
##
## ── Variable type: numeric ────────────────────────────────────
##   skim_variable n_missing complete_rate  mean    sd p0   p25 p5
## 1 height                6         0.931 175.   34.8 66 167   18
## 2 mass                 28         0.678  97.3 169.  15  55.6  7
## 3 birth_year           44         0.494  87.6 155.   8  35    5
```

Has a useful summary function

```
skim(iris) |>
  summary()
```

skimr

```
##                            Values
## Name                       iris
## Number of rows             150
## Number of columns          5
## _______________________
## Column type frequency:
##   factor                   1
##   numeric                  4
## ________________________
## Group variables            None
```

Individual columns can be selected using tidyverse-style selectors

```
skim(iris, Sepal.Length, Petal.Length)

## ── Data Summary ────────────────────────
##                            Values
## Name                       iris
## Number of rows             150
## Number of columns          5
## _______________________
## Column type frequency:
##   numeric                  2
## ________________________
## Group variables            None
##
## ── Variable type: numeric ──────────────────────────────────────
##   skim_variable n_missing complete_rate mean    sd  p0 p25   p5
## 1 Sepal.Length          0             1 5.84 0.828 4.3 5.1 5.8
## 2 Petal.Length          0             1 3.76 1.77  1   1.6 4.3
```

Handles grouped data

skim() can handle data that has been grouped using dplyr::group_by().

```
iris |>
  dplyr::group_by(Species) |>
  skim()

## ── Data Summary ────────────────────────
```

skimr

```
## Number of rows             150
## Number of columns          5
## _______________________
## Column type frequency:
##   numeric                  4
## ________________________
## Group variables            Species
##
## ── Variable type: numeric ────────────────────────────────────
##    skim_variable Species    n_missing complete_rate  mean    sc
##  1 Sepal.Length  setosa             0             1 5.01  0.352
##  2 Sepal.Length  versicolor         0             1 5.94  0.516
##  3 Sepal.Length  virginica          0             1 6.59  0.636
##  4 Sepal.Width   setosa             0             1 3.43  0.379
##  5 Sepal.Width   versicolor         0             1 2.77  0.314
##  6 Sepal.Width   virginica          0             1 2.97  0.322
##  7 Petal.Length  setosa             0             1 1.46  0.174
##  8 Petal.Length  versicolor         0             1 4.26  0.470
##  9 Petal.Length  virginica          0             1 5.55  0.552
## 10 Petal.Width   setosa             0             1 0.246 0.105
## 11 Petal.Width   versicolor         0             1 1.33  0.198
## 12 Petal.Width   virginica          0             1 2.03  0.275
```

Behaves nicely in pipelines

```
iris |>
  skim() |>
  dplyr::filter(numeric.sd > 1)

## ── Data Summary ────────────────────────
##                            Values
## Name                       iris
## Number of rows             150
## Number of columns          5
## _______________________
## Column type frequency:
##   numeric                  1
## ________________________
## Group variables            None
##
## ── Variable type: numeric ────────────────────────────────────
##   skim_variable n_missing complete_rate mean    sd p0 p25   p50 p
```

skimr

Knitted results

Simply skimming a data frame will produce the horizontal print layout shown above. We provide a `knit_print` method for the types of objects in this package so that similar results are produced in documents. To use this, make sure the `skimmed` object is the last item in your code chunk.

```
faithful |>
  skim()
```

Name	faithful
Number of rows	272
Number of columns	2
_______________________	
Column type frequency:	
numeric	2
________________________	
Group variables	None

Data summary

Data summary

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p2
eruptions	0	1	3.49	1.14	1.6	2.
waiting	0	1	70.90	13.59	43.0	58.

Customizing skimr

skimr

create statistics for new classes and develop skimmers for data structures that are not data frames.

Specify your own statistics and classes

Users can specify their own statistics using a list combined with the `skim_with()` function factory. `skim_with()` returns a new `skim` function that can be called on your data. You can use this factory to produce summaries for any type of column within your data.

Assignment within a call to `skim_with()` relies on a helper function, `sfl` or `skimr` function list. By default, functions in the `sfl` call are appended to the default skimmers, and names are automatically generated as well.

```
my_skim <- skim_with(numeric = sfl(mad))
my_skim(iris, Sepal.Length)
```

But you can also helpers from the `tidyverse` to create new anonymous functions that set particular function arguments. The behavior is the same as in `purrr` or `dplyr`, with both `.` and `.x` as acceptable pronouns. Setting the `append = FALSE` argument uses only those functions that you've provided.

```
my_skim <- skim_with(
  numeric = sfl(
    iqr = IQR,
    p01 = ~ quantile(.x, probs = .01)
    p99 = ~ quantile(., probs = .99)
  ),
  append = FALSE
)
my_skim(iris, Sepal.Length)
```

And you can remove default skimmers by setting them to `NULL`.

```
my_skim <- skim_with(numeric = sfl(hist = NULL))
my_skim(iris, Sepal.Length)
```

Skimming other objects

skimr

- `character`
- `factor`
- `logical`
- `complex`
- `Date`
- `POSIXct`
- `ts`
- `AsIs`

`skimr` also provides a small API for writing packages that provide their own default summary functions for data types not covered above. It relies on R S3 methods for the `get_skimmers` function. This function should return a `sfl`, similar to customization within `skim_with()`, but you should also provide a value for the `class` argument. Here's an example.

```
get_skimmers.my_data_type <- function(column) {
  sfl(
    .class = "my_data_type",
    p99 = quantile(., probs = .99)
  )
}
```

Limitations of current version

We are aware that there are issues with rendering the inline histograms and line charts in various contexts, some of which are described below.

Support for spark histograms

With versions of R before 4.2.1, there are known issues with printing the spark-histogram characters when printing a data frame. For example, `"▂▅▇"` is printed as `"<U+2582><U+2585><U+2587>"`. This longstanding problem originates in the low-level code for printing dataframes. While some cases have been addressed, there are, for example, reports of this issue in Emacs ESS. While this is a deep issue, there is ongoing work to address it in base R. We recommend upgrading to at least R 4.2.1 to address this issue.

This means that while `skimr` can render the histograms to the console

- converting a `skimr` data frame to a vanilla R data frame, but tibbles render correctly
- in the context of rendering to a pdf using an engine that does not support utf-8.

One workaround for showing these characters in Windows is to set the CTYPE part of your locale to Chinese/Japanese/Korean with `Sys.setlocale("LC_CTYPE", "Chinese")`. The helper function `fix_windows_histograms()` does this for you.

And last but not least, we provide `skim_without_charts()` as a fallback. This makes it easy to still get summaries of your data, even if unicode issues continue.

Printing spark histograms and line graphs in knitted documents

Spark-bar and spark-line work in the console, but may not work when you knit them to a specific document format. The same session that produces a correctly rendered HTML document may produce an incorrectly rendered PDF, for example. This issue can generally be addressed by changing fonts to one with good building block (for histograms) and Braille support (for line graphs). For example, the open font “DejaVu Sans” from the `extrafont` package supports these. You may also want to try wrapping your results in `knitr::kable()`. Please see the vignette on using fonts for details.

Displays in documents of different types will vary. For example, one user found that the font “Yu Gothic UI Semilight” produced consistent results for Microsoft Word and Libre Office Write.

Inspirations

- TextPlots for use of Braille characters
- spark for use of block characters.

The earliest use of unicode characters to generate sparklines appears to be from 2009.

Exercising these ideas to their fullest requires a font with good support

skimr

Contributing

We welcome issue reports and pull requests, including potentially adding support for commonly used variable classes. However, in general, we encourage users to take advantage of skimr's flexibility to add their own customized classes. Please see the contributing and conduct documents.

Links

View on CRAN

Browse source code

Report a bug

License

GPL-3

Community

Contributing guide

Citation

Citing skimr

Developers

Elin Waring
Maintainer, author

Michael Quinn
Author

Amelia McNamara
Author

Eduardo Arino de la Rubia
Author

Hao Zhu
Author

Shannon Ellis
Author

More about authors...

skimr

Software Peer-Review

Peer Reviewed

ABOUT

About rOpenSci
Software Review
Our Team
Jobs
Donate
Contact Us

COMMUNITY

Our Community
Community calls
Events
Join the Discussion
Code of conduct

RESOURCES

Packages
Use Cases
Talks & Publications
Documentation
Newsletter
Cite rOpenSci