

R パッケージをつくる

@Tokyo.R #119, Sep 20, 2025



Rパッケージを作りました！

- [日本アクチュアリー会](#)の研究活動の一環で作成した R パッケージ [midr](#) が CRAN に公開されました。

`midr: Learning from Black-Box Models by Maximum Interpretation Decomposition`

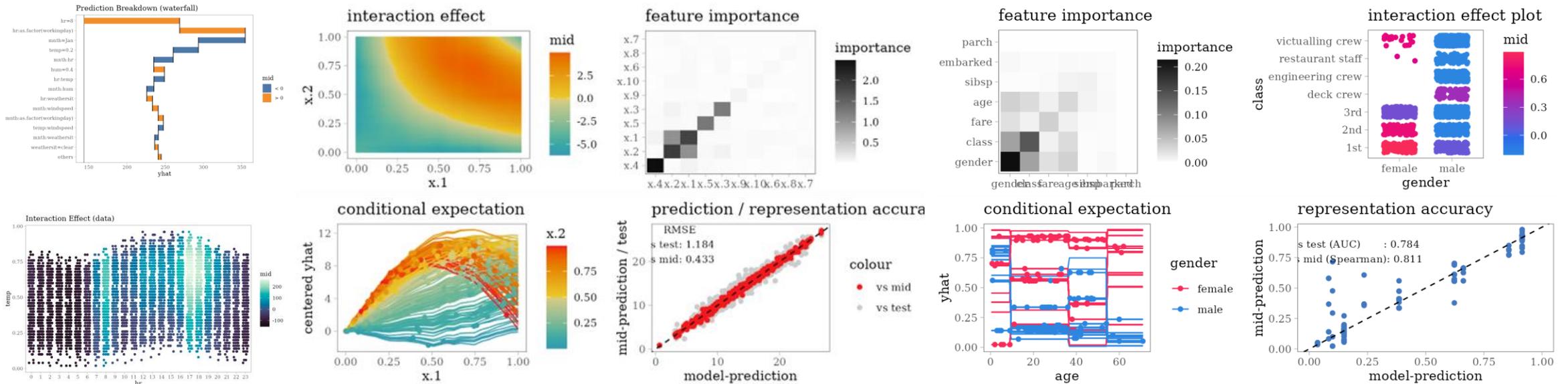
The goal of 'midr' is to provide a model-agnostic method for interpreting and explaining black-box predictive models by creating a globally interpretable surrogate model. The package implements 'Maximum Interpretation Decomposition' (MID), a functional decomposition technique that finds an optimal additive approximation of the original model. This approximation is achieved by minimizing the squared error between the predictions of the black-box model and the surrogate model. The theoretical foundations of MID are described in Iwasawa & Matsumori (2025) [Forthcoming], and the package itself is detailed in Asashiba et al. (2025) <[doi:10.48550/arXiv.2506.08338](https://doi.org/10.48550/arXiv.2506.08338)>.

Version: 0.5.2
Imports: graphics, grDevices, [Rcpp](#), [RcppEigen](#), [rlang](#), stats, utils
LinkingTo: [Rcpp](#), [RcppEigen](#)
Suggests: datasets, [ggplot2](#), [khroma](#), [knitr](#), [RColorBrewer](#), [rmarkdown](#), [scales](#), [shapviz](#), [testthat](#), [viridisLite](#)
Published: 2025-09-07
DOI: [10.32614/CRAN.package.midr](https://doi.org/10.32614/CRAN.package.midr)
Author: Ryoichi Asashiba [aut, cre], Hirokazu Iwasawa [aut], Reiji Kozuma [ctb]



本日共有させていただきたいこと

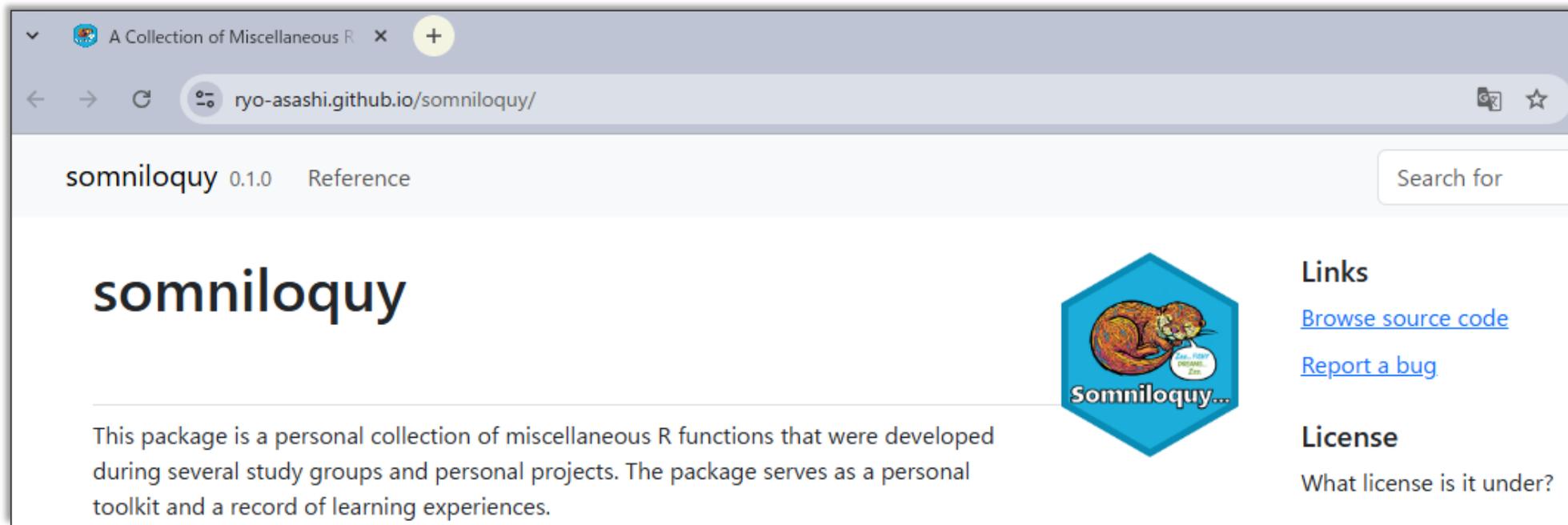
- **midr** は、機械学習モデルの解釈を目的とするパッケージです。[参考資料リンク]
 - 「ブラックボックスモデルの予測値を予測する代理モデルを構築する機能」と「その代理モデルを様々な切り口で可視化する機能」を実装しています（PDP、ALE、SHAP、PFI などと同様の用途）。



- 「パッケージを宣伝したい！」というのがこの応用セッションの本当の応募動機です。
...ですが、今日は R パッケージをつかった経験に焦点をあててお話をさせていただきます。

R パッケージをつくる

@Tokyo.R #119, Sep 20, 2025



The screenshot shows a web browser window with the following elements:

- Browser tabs: "A Collection of Miscellaneous R" and a plus sign for more tabs.
- Address bar: "ryo-asashi.github.io/somniloquy/"
- Page header: "somniloquy 0.1.0 Reference" and a search box labeled "Search for".
- Main content area:
 - Section title: "somniloquy"
 - Image: A blue hexagonal logo featuring a brown cat curled up and sleeping. A speech bubble next to the cat says "See... FROM ENGLISH...". Below the cat, the text "Somniloquy..." is visible.
 - Description: "This package is a personal collection of miscellaneous R functions that were developed during several study groups and personal projects. The package serves as a personal toolkit and a record of learning experiences."
- Right sidebar:
 - Section: "Links"
 - [Browse source code](#)
 - [Report a bug](#)
 - Section: "License"
 - Text: "What license is it under?"

R パッケージ、つくる必要なんてある？

パッケージをつくれれば...

- 便利な自作関数を他の R ユーザーに配布するのが簡単になります。
 - GitHub にあれば、`pak::pak()` や `devtools::install_github()` でインストール可能です。
- よく使う処理を自分用パッケージ (Personal R Package) として整理しておくこともできます。
 - Personal R Package の例 - <https://github.com/kbroman/broman>
 - ▶ `brocolors.R` の関数群は自分用のカラーパレットを返す。`pastedot.R` の関数群は文字列をドットなどでつなぐ、等々。
 - ▶ このパッケージでは、各関数のヘルプドキュメントがしっかり整備されています。
- 頻繁にコピー&ペーストして使っている処理はありませんか？

関数化して、自分用パッケージにまとめておきましょう！

- 過去に書いた関数を探す手間も、コピー&ペーストの手間も省けます。
- 関数にバグがあったときに、コードの修正がパッケージ内だけで済みます。
- **testthat** パッケージの機能を活用すれば、コードテストを導入できます。
- 関数名が衝突しても `pkg::fun()` で関数にアクセスすることができます。
- **roxygen2** で関数のヘルプドキュメントを残すことができ、`help(fun)` や `?fun` で読み直すことができます。



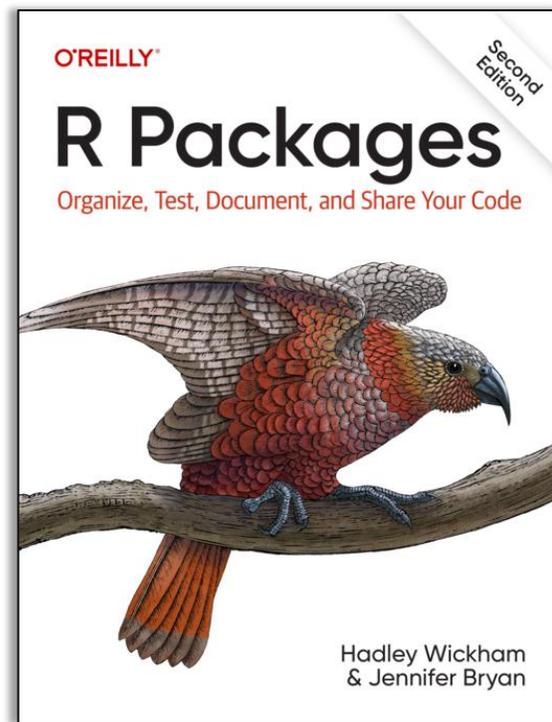
R Packages (2e) を読めば十分...？

- R パッケージを自作するために必要な情報はすべて [R Packages \(2e\)](#) に書かれています

...ですが、この本の説明はとてもしっかりした作法にのっとり、完璧に従うのはけっこう大変です。（第 1 章の最初のセクションだけで 20 項目もあります！）

まずは、もっと雑にやってみましょう！

- この資料では以下の 3 点を前提しています。
 - Git がインストールされていること
[git: 1.5 使い始める - Gitのインストール](#)
 - GitHub アカウントを持っていること
[GitHub Docs: GitHub でのアカウントの作成](#)
 - **devtools** がインストールされていること
[devtools: Tools to Make Developing R Packages Easier](#)
`install.packages("devtools")`

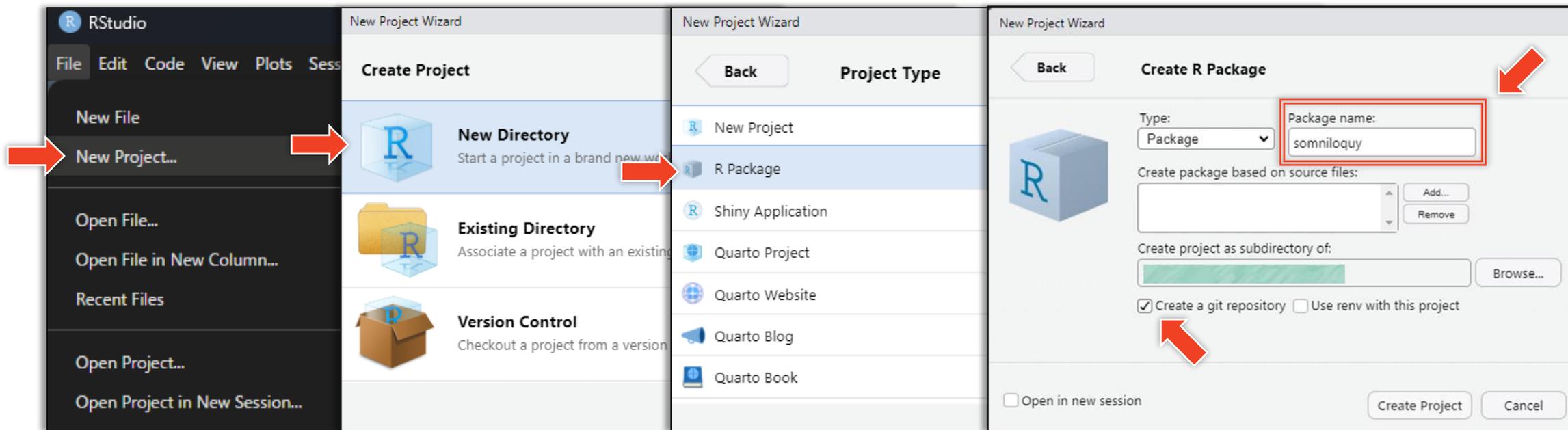


Getting started > 1. The Whole Game

- 1.1 Load devtools and friends
- 1.2 Toy package: regexcite
- 1.3 Preview the finished product
- 1.4 `create_package()`
- 1.5 `use_git()`
- 1.6 Write the first function
- 1.7 `use_r()`
- 1.8 `load_all()`
- 1.9 `check()`
- 1.10 Edit DESCRIPTION
- 1.11 `use_mit_license()`
- 1.12 `document()`
- 1.13 `check()` again
- 1.14 `install()`
- 1.15 `use_testthat()`
- 1.16 `use_package()`
- 1.17 `use_github()`
- 1.18 `use_readme_rmd()`
- 1.19 The end: `check()` and `install()`
- 1.20 Review

1. パッケージのプロジェクトを作成する

- RStudio のメニューから File > New Project... > New Directory > R Package を選択し、パッケージ名とプロジェクトフォルダの置き場所を指定します。
 - Create a git repository の欄にチェックを入れておきましょう。



- このデモでは **somniloquy** という名前のパッケージを作ります。
 - 勉強会などで自作した雑多な関数を放り込むためのパッケージにすることを想定しています。

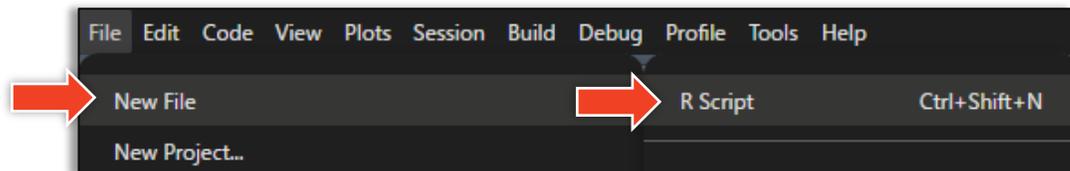
2. 作成されたフォルダを整理する

- パッケージと同名のプロジェクトフォルダが作成されたことを確認します。
- 自動的に開かれるスクリプト R/hello.R を閉じて、man/hello.Rd と一緒に削除します。

パス (プロジェクトフォルダ内)	概要
DESCRIPTION	パッケージのメタデータ (基本情報や依存関係など) を定義するファイル
NAMESPACE	関数の外部へのエクスポートや外部からのインポートを管理するファイル
R	パッケージの関数などを書いた R スクリプトの保存場所
! R/hello.R	削除 例示のための R スクリプト
man	関数などのドキュメントに対応する Rd ファイルの保存場所
! man/hello.Rd	削除 hello.R に対応する Rd ファイル
.Rbuildignore	パッケージのビルド時に無視するファイルを定めるファイル
.gitignore	Git のバージョン管理において無視するファイルを定めるファイル
somniloquy(パッケージ名).Rproj	R プロジェクトの設定ファイル

3. パッケージの関数を作成する

- File > New File > R Script で新しいファイルを作成し、File > Save as... で R/ に保存します。



- ファイル名は、そのファイルで定義する関数と同名にしておくと整理しやすいです。
 - ファイルの作成と保存は、`usethis::use_r("excess_plot")` のようにしても OK。
- 作成されたスクリプトの中で関数を定義し、ファイルを上書き保存します。
 - ここでは[平均／メディアン超過プロット](#)を描くための関数 `excess_plot()` を作成します。

excess_plot.R

```
1 excess_plot <- function(  
2   x, method = c("mean", "median"), mode = c("exact", "interpolate"),  
3   show_plot = TRUE, ...  
4 ) {  
5   method <- match.arg(method)  
6   mode <- match.arg(mode)  
  
32  if (is.null(args$type)) args$type <- "1"  
33  do.call(base::plot, c(list(y ~ u, data = res), args))  
34  invisible(res)  
35 } else {  
36   res  
37 }  
38 }
```

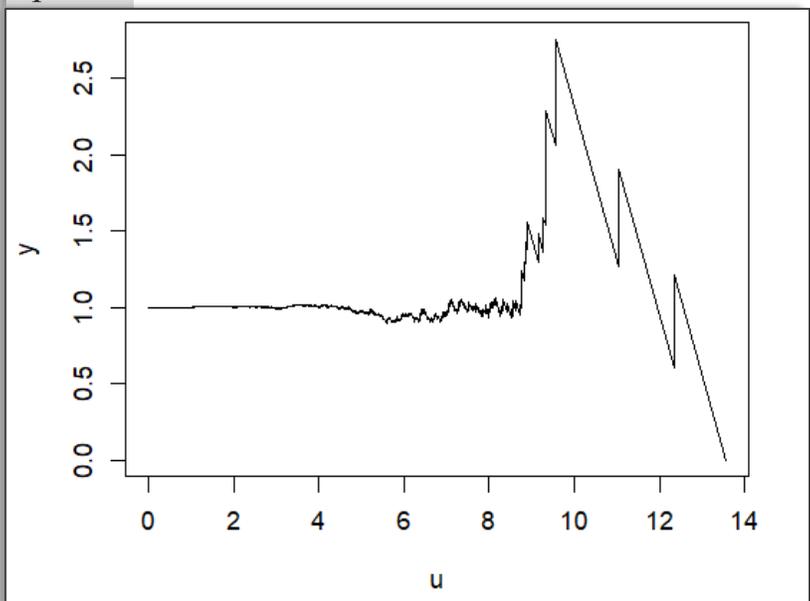
4. 関数が正しく機能することを確認する

- `devtools::load_all()` を使って作成した関数が正しく機能することを確認します。

console

```
> devtools::load_all()
i Loading somniloquy
> search()
[1] ".GlobalEnv"      "package:somniloquy" "devtools_shims"    "tools:rstudio"
[5] "package:stats"   "package:graphics"  "package:grDevices" "package:utils"
[9] "package:datasets" "package:methods"   "Autoloads"        "package:base"
> excess_plot(rexp(1e5))
```

plots



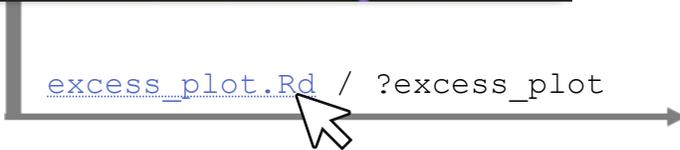
5. 関数がエクスポートされるように設定する

- 関数を作成した R スクリプトの冒頭に、その関数を端的に説明する「見出し」と `@export` タグを追加します。見出しやタグの行は `#'` で始めます。

```
excess_plot.R
1 #' Create Excess Plots
2 #'
3 #' @export
4 #'
5 excess_plot <- function(
```

- `devtools::document()` を実行して、`man/excess_plot.Rd` を作成します。

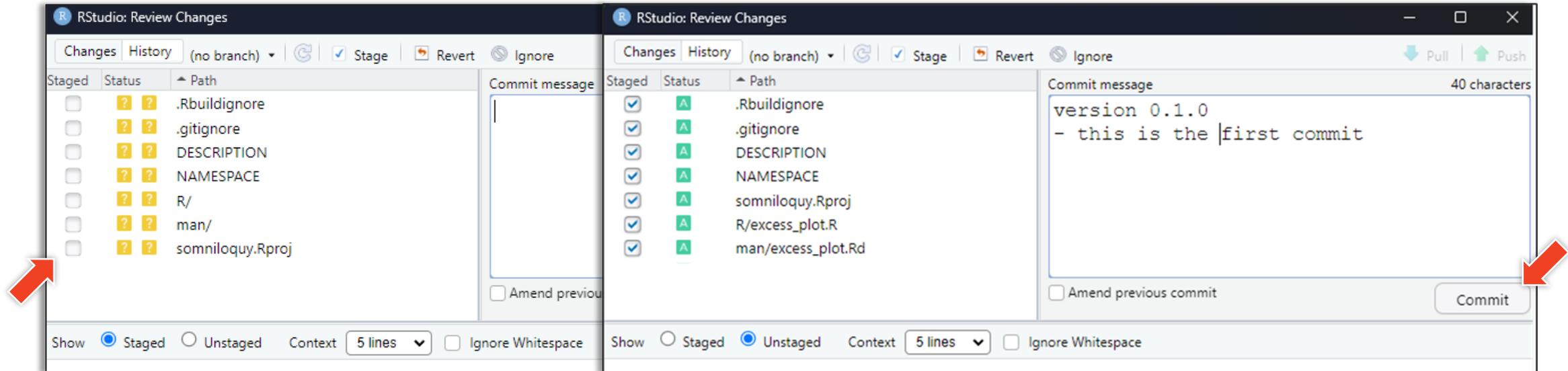
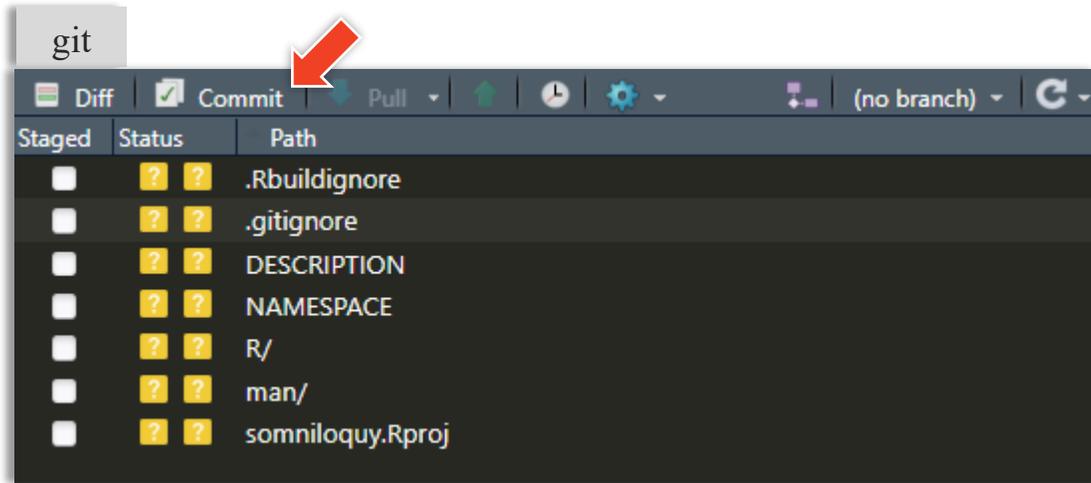
```
console
> devtools::document()
i Updating somniloquy documentation
i Loading somniloquy
Writing excess_plot.Rd
> ?excess_plot
i Rendering development documentation for "excess_plot"
```



```
help
excess_plot.Rd - Find in Topic
excess_plot {somniloquy} (preview) R Documentation
Create Excess Plots
Description
Create Excess Plots
Usage
excess_plot(
  x,
  method = c("mean", "median"),
  mode = c("exact", "interpolate"),
  show_plot = TRUE,
```

6. すべての変更をコミットする

- R Studio の git タブにある Commit から進み、すべての変更をコミットします。



7. GitHub にアップロードする

- `usethis::use_github()` でプロジェクトフォルダを GitHub にアップロードします。
 - 初めて `use_github()` を使うときは、ブラウザ経由で GitHub アカウントの認証を要求されます。

```
console
> usethis::use_github()
✓ Creating GitHub repository "ryo-asashi/somniloquy".
✓ Setting remote "origin" to
  "https://github.com/ryo-asashi/somniloquy.git".
✓ Adding "https://github.com/ryo-asashi/somniloquy" to URL.
✓ Adding "https://github.com/ryo-asashi/somniloquy/issues" to BugReports.
i There is 1 uncommitted file:
• DESCRIPTION

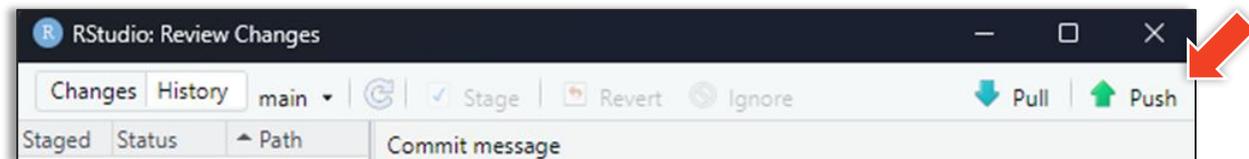
! Is it ok to commit it?

1: Nope
2: Negative
3: For sure

Selection: 3

✓ Adding files.
✓ Making a commit with message "Add GitHub links to DESCRIPTION".
✓ Pushing "main" branch to GitHub and setting "origin/main" as upstream
  branch.
✓ Opening URL <https://github.com/ryo-asashi/somniloquy>.
```

- 今後、ファイルの変更をコミットしたら、GitHub リポジトリにも変更をプッシュしましょう。



8. 完成！ GitHub からインストールする

- これで完成！
`pak::pak()` や `devtools::install_github()` でのインストールが可能です。

console

```
> pak::pak("ryo-asashi/somniloquy")

→ Will install 1 package.
→ Will download 1 package with unknown size.
+ somniloquy 0.1.0 [bld][cmp][dl] (GitHub: 123ef63)
i Getting 1 pkg with unknown size
✓ Got somniloquy 0.1.0 (source) (1.56 MB)
✓ Downloaded 1 package (1.56 MB) in 941ms
i Packaging somniloquy 0.1.0
✓ Packaged somniloquy 0.1.0 (4.3s)
i Building somniloquy 0.1.0
✓ Built somniloquy 0.1.0 (1.4s)
✓ Installed somniloquy 0.1.0 (github::ryo-asashi/somniloquy@123ef63) (119ms)
✓ 1 pkg: added 1, dld 1 (NA B) [13.6s]
```

console

```
> library(somniloquy)
> excess_plot(rnorm(1e5))
```

- なお、開発上は GitHub を経由せず `devtools::install()` でインストールしても問題ありません。

Optional. DESCRIPTION を更新する

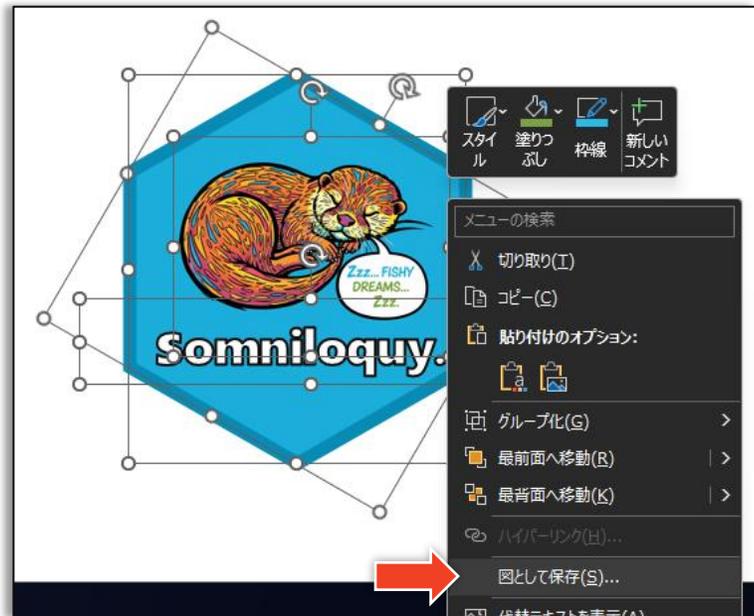
- DESCRIPTION は、パッケージのメタデータ（基本情報や依存先パッケージ）を記載するファイルです。ここでは、Title / Authors@R / Description などを書き換えておきます。
 - このうち、Title は GitHub リポジトリの About 欄にも反映させておくとよさそうです。

DESCRIPTION

```
1 Package: somniloquy
2 Type: Package
3 Title: A Collection of Miscellaneous R Functions
4 Version: 0.1.0
5 Authors@R: c(
6   person(
7     "Ryoichi", "Asashiba",
8     email = "ryoichi.asashiba@gmail.com",
9     role = c("aut", "cre")
10  )
11 )
12 Description: This package is a personal collection of miscellaneous R functions
13 License: What license is it under?
14 Encoding: UTF-8
15 LazyData: true
16 RoxygenNote: 7.3.2
17 URL: https://github.com/ryo-asashi/somniloquy
18 BugReports: https://github.com/ryo-asashi/somniloquy/issues
19
```

Optional. パッケージロゴを作成する

- PowerPoint で外枠・背景・タイトルを用意します。
 - 外枠と背景はそれぞれ「挿入>図形>六角形」で作成
「図形の書式」でサイズを 10.22 cm×11.82 cm に変更して 90 度回転
 - 外枠のみ、「図形の枠線>太さ」で枠の幅を 12pt に設定
- ロゴ用のイラストは生成AIに描いてもらいました。
- 作成したロゴは man/figures/logo.png に保存します。
 - パーツをまとめて選択して「右クリック>図として保存」



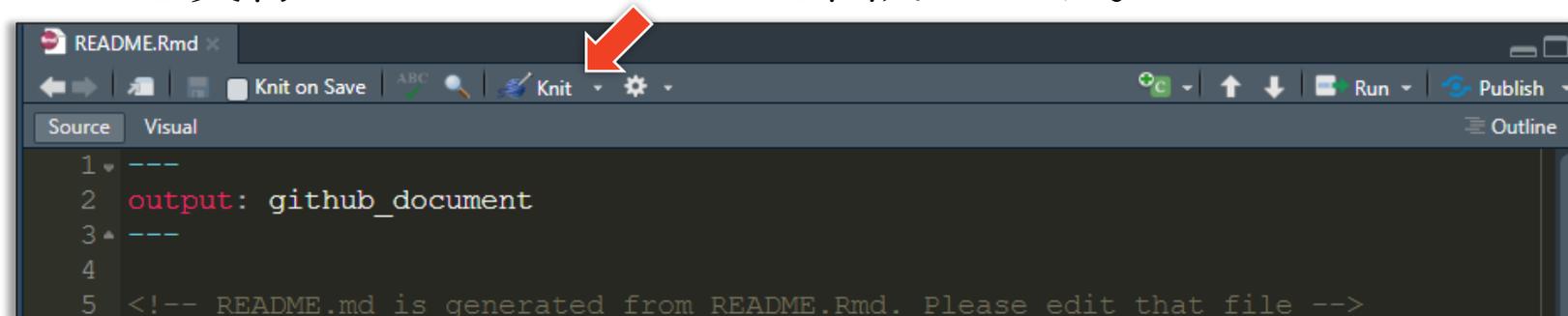
Optional. README を追加する

- `usethis::use_readme_rmd()` を実行して README.Rmd を作成します。

```
console
> usethis::use_readme_rmd()
✓ Setting active project to "C:/Users/daysb/Documents/RFiles/somniloquy".
✓ Writing README.Rmd.
✓ Adding "^README\\.Rmd$" to ..Rbuildignore.
❑ Modify README.Rmd.
✓ Writing ..git/hooks/pre-commit.
```

- README.Rmd の16行目を以下のように書き換えるとロゴを入れられます。
somniloquy
- 21行目は DESCRIPTION の Description をコピーしておけば OK です！
- 32行目以下は「パッケージの使用例」のサンプルです。とりあえず消しておきましょう。

- Knit を実行して README.md を出力します。



```
README.Rmd x
Knit on Save ABC Knit Run Publish
Source Visual Outline
1 ---
2 output: github_document
3 ---
4
5 <!-- README.md is generated from README.Rmd. Please edit that file -->
```

Optional. GitHub Pages で Web サイトを作る

- `usethis::use_pkgdown()` と `usethis::use_pkgdown_github_pages()` でパッケージを紹介するための [Web サイト](#) (GitHub Pages) を作れます。
 - ロゴを作っているなら、`pkgdown::build_favicons()` でファビコンも作成できます。

console

```
> usethis::use_pkgdown()
> usethis::use_pkgdown_github_pages()
! Overwrite pre-existing file .._pkgdown.yml?

1: Definitely
2: Nope
3: No

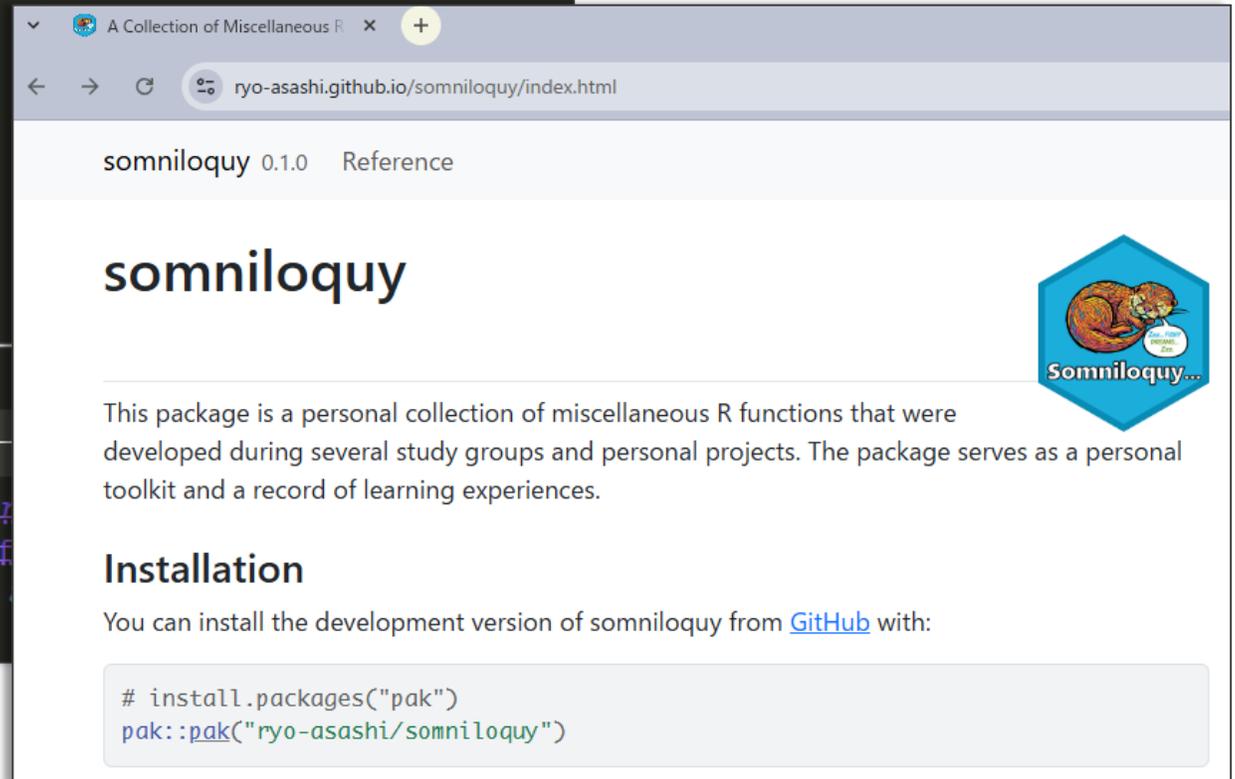
Selection: 1

> pkgdown::build_favicons()

— Building favicons —

i Building favicons with <https://realfavicons.com>
✓ Added apple-touch-icon.png, favicon-96x96.png, favicon-32x32.png, site.webmanifest, web-app-manifest-192x192.png, web-app-manifest-512x512.png.
```

Commit & Push



A Collection of Miscellaneous R

ryo-asashi.github.io/somniloquy/index.html

somniloquy 0.1.0 Reference

somniloquy



This package is a personal collection of miscellaneous R functions that were developed during several study groups and personal projects. The package serves as a personal toolkit and a record of learning experiences.

Installation

You can install the development version of somniloquy from [GitHub](#) with:

```
# install.packages("pak")
pak::pak("ryo-asashi/somniloquy")
```

この資料で説明しなかったこと (の一部)

- 他のパッケージの関数を利用する (**base**を除く)
 - DESCRIPTION の Imports: か Suggests: にパッケージ名を記載し、スクリプト内では `stats::median()` のようにパッケージ環境を指定して参照します。 `usethis::use_package("stats")` でも OK。
 - 特に、**ggplot2** を利用する場合は [vignette\("ggplot2-in-packages"\)](#) が参考になります。
- パッケージのライセンスを設定する
 - ライセンスは、パッケージのコードを著作権者でない人がどのように利用できるかを規定します。
 - `usethis::use_mit_license()` を実行すると MIT ライセンスを設定したものととして体裁が整います。
- パッケージの関数に対するコードテストを実行する
 - `usethis::use_testthat()` によって **testthat** パッケージのコードテスト機能を利用できます。
 - その後は `usethis::use_test()` 関数などを用いてテスト用スクリプトを追加していきます。
- パッケージの問題点をチェックする
 - `devtools::check()` を用いてパッケージが問題を抱えていないかを包括的にチェックできます。
 - 潜在的な問題があれば、**errors, warning, notes** という 3 種類の注意が表示されます。(CRAN への投稿を考えていないなら、**errors** 以外はそれほど深刻にとらえなくてもよいと思っています。)

CRAN にパッケージを公開するまでの苦難

- 第一の壁：`devtools::check()`
 - 気軽にいつものようにコードを書くと、`check()` に色々なことを指摘されます（外部パッケージの非エクSPORT関数を使うと警告、`ggplot2::aes()` や `do.call()` などの中で表現式 `expression` をそのまま使うと警告...などなど）。
 - 慎重に書き直せば大抵は回避できますが、全部対処するのはそれなりに骨が折れました...
- 第二の壁：[DavisVaughan/extrachecks](https://github.com/DavisVaughan/extrachecks)
 - 先人たちが、CRAN から受けたアドホックな指摘を受けた経験をリストアップして共有してくれています。

extrachecks

This is a place to dump extra ad-hoc checks that CRAN does that are not checked for by `devtools::check()`. Some of them are generally useful, some of them are highly specific, but all of them are reasons that an R package has been rejected by CRAN. The hope is that by making them public, we can lower the number of rejections by making package developers more informed.

First release:

- `usethis::use_cran_comments()`
- Update (aspirational) install instructions in README
- Proofread Title: and Description:
- [Check that all exported functions have @return and @examples](#)
- Check that Authors@R: includes a copyright holder (role 'cph')
- Check licensing of included files
- [Review https://github.com/DavisVaughan/extrachecks](https://github.com/DavisVaughan/extrachecks)

Prepare for release:

- `git pull`
- `urlchecker::url_check()`
- `devtools::build_readme()`
- `devtools::check(remote = TRUE, manual = TRUE)`
- `devtools::check_win_devel()`
- `git push`

Submit to CRAN:

- `usethis::use_version('minor')`: -> incremented to 0.5.0
- `devtools::submit_cran()`
- Approve email

Wait for CRAN...

- Accepted 🎉
- `usethis::use_github_release()`
- `usethis::use_dev_version(push = TRUE)`

最後にして最大の壁：References

- 第三の壁：[The CRAN Cookbook](#) - References

- 理想としては、各パッケージの説明文には、ユーザーがパッケージの背景理論を探求するのに役立つ参考文献を少なくとも1つ含めるべきです。文献は「GA」パッケージのような形式で記述してください。Ideally, each package should include at least one reference in the description text to help users further explore the theory behind the package. These references should be formatted as shown in the 'GA' package:



- 一応、Reference は必須ではないようですが、今回はパッケージの実装を説明する英語の論文を準備して [arXiv \[2506.08338\]](#) に投稿しました（これでだいたい半年くらいパッケージの公開が遅れました）。

midr: Learning from Black-Box Models by Maximum Interpretation Decomposition

by Ryoichi Asashiba, Reiji Kozuma, and Hirokazu Iwasawa

Abstract The use of appropriate methods of Interpretable Machine Learning (IML) and eXplainable Artificial Intelligence (XAI) is essential for adopting black-box predictive models in fields where model and prediction explainability is required. As a novel tool for interpreting black-box models, we introduce the R package **midr**, which implements Maximum Interpretation Decomposition (MID). MID is a functional decomposition approach that derives a low-order additive representation of a black-box model by minimizing the squared error between the model's prediction function and this additive representation. **midr** enables learning from black-box models by constructing a global surrogate model with advanced analytical capabilities. After reviewing related work and the theoretical foundation of MID, we demonstrate the package's usage and discuss some of its key features.

最後に少しだけ...

- **midr** パッケージの宣伝をさせてください！
- カラーテーマ (“color.theme”) 機能はもしかしたら便利なことがあるかも？
 - midr パッケージには、外部パッケージへの依存をできるだけ少なく抑えるという思想から、可視化にとって重要な“色”の取り扱いを自前で実装しています。
 - 以下のシンタックスに従う文字列で、カラーテーマをカスタマイズして呼び出すことができます。

```
"[(source)/] (name) [_r] [@ (type)] [?(query)]"
```

```
color.theme("tokyo")
```

Sequential Color Theme : "tokyo"



```
color.theme("tokyo_r")
```

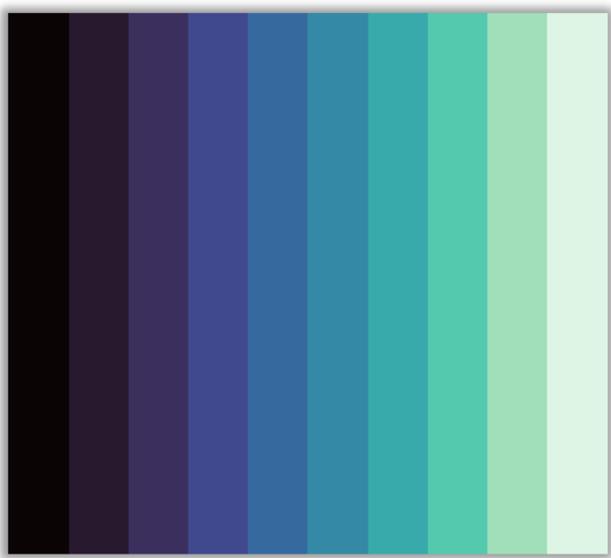
Sequential Color Theme : "tokyo"



カラーテーマは二つのメソッドを持ちます

- カラーテーマ（環境）は `...$palette()` と `...$ramp()` という関数を持ちます。それぞれ、正の整数 `n` と `[0, 1]` 区間に値を持つベクトル `x` に対して色名を返します。

```
n <- 10  
barplot(height = rep(1, n),  
        space = 0, border = NA, axes = FALSE,  
        col = color.theme("mako")$palette(n))
```



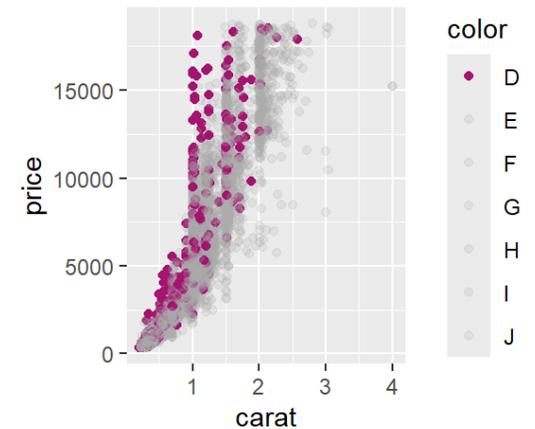
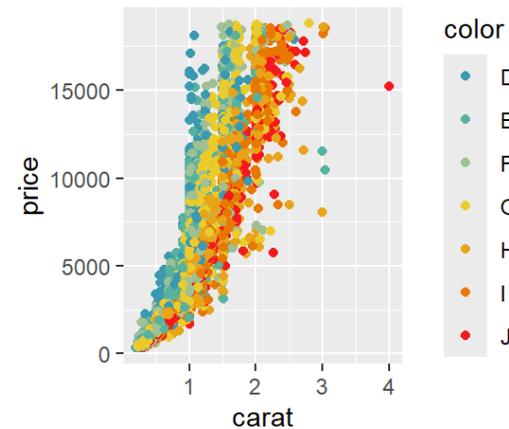
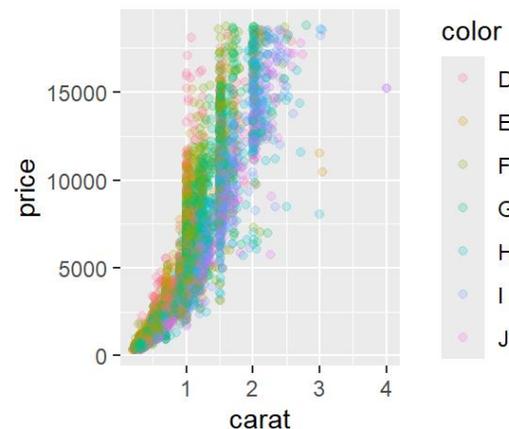
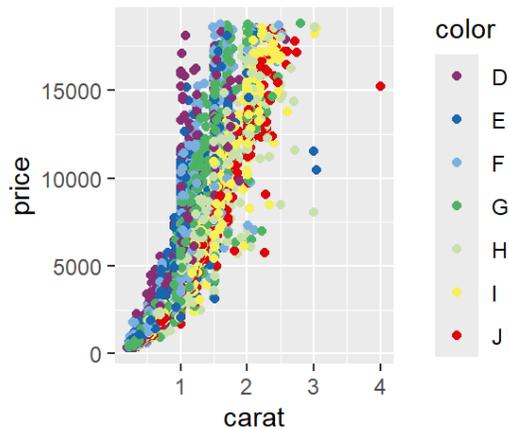
```
x <- seq(0, 1, length.out = 100)  
image(z = t(1:100),  
      axes = FALSE,  
      col = color.theme("mako")$ramp(x))
```



ggplot 用のインターフェースもあります

- **ggplot** 用に `scale_color_theme()` と `scale_fill_theme()` を実装しています。
 - テーマは最大 **234** 種類のプリセットが用意されており、さらに、テーマを自作することもできます。

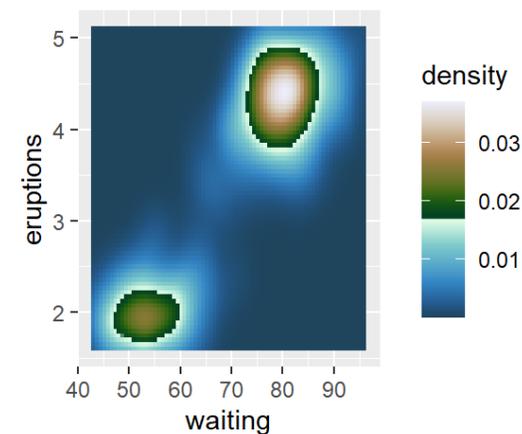
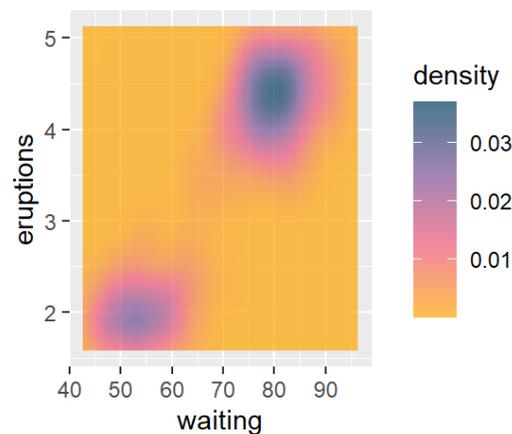
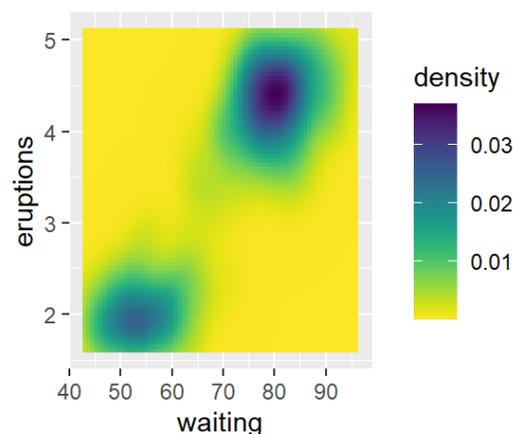
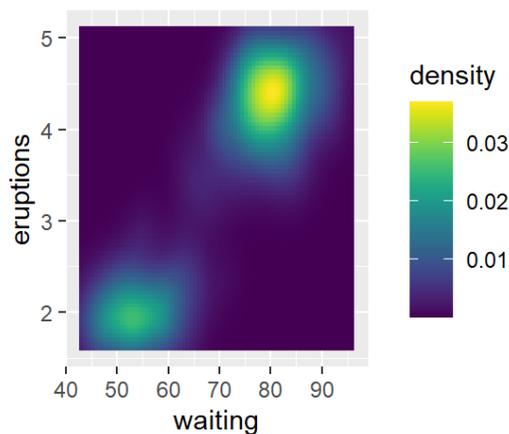
```
d <- diamonds[sample(nrow(diamonds), 1e4),]  
p <- ggplot(d) + geom_point(aes(carat, price, col = color))  
p + scale_color_theme("discreterainbow")  
p + scale_color_theme("HCL?alpha=.2")  
p + scale_color_theme("Zissou 1@q")  
p + scale_color_theme("highlight?base='#aaaaaa30'")
```



ggplot 用のインターフェースもあります

- **ggplot** 用に `scale_color_theme()` と `scale_fill_theme()` を実装しています。
 - テーマは最大 **234** 種類のプリセットが用意されており、さらに、テーマを自作することもできます。

```
v <- ggplot(faithfuld) + geom_tile(aes(waiting, eruptions, fill = density))
set_color_theme(c("#003f5c", "#7a5195", "#ef5675", "#ffa600"))
v + scale_fill_theme("viridis")
v + scale_fill_theme("viridis_r")
v + scale_fill_theme("newtheme_r?alpha=.7")
v + scale_fill_theme("bukavu", middle = 0.017)
```



Enjoy!