# **knitr**: R による美麗で柔軟そして高速な動的レポート生成

著者: Xie, Yihui (谢益辉)　　翻訳者: Katagiri, Satoshi[*1] (片桐 智志)

ver. 1.7 (2024/03/17 02:20:50 JST に更新されました. 本家の 2023/12/06 17:03:20 JST の版に対応しています)

[*1] twitter@ill_identified

# 目次

# knitr



**訳注**: Yihui 氏のサイトの一連のドキュメントはかなり以前から氏のブログ投稿として断続的に更新されてきたものを編纂しているため, 内容の重複した記述がいくつかありますし, RStudio および R Markdown が普及している現在では, やや out-of-date な記述も見られます (現在では Sweave はあまり使いません). そのため, **サイドバーの目次から辿れるページ, およびそれらでリンクされているページ以外は重要度が低いと見なし, 翻訳していません**. 同様の理由から「用例」も一部を除き翻訳していません (**共同翻訳者は常に歓迎します**: https://github.com/Gedevan-Aleksizde/knitr-doc-ja).
しかしながら翻訳時の **knitr** および R Markdown に関する日本語の情報流通を鑑みるに, 非常に有用なドキュメントであると翻訳者は確信しています. それぞれのトピックがどの環境を想定したものなのか, よく確認してご覧ください.

## 概要

**knitr** パッケージは R を使用した透明な動的なレポート作成エンジンとしてデザインされ, 長年にわたって存在した Sweave の問題を解決するとともに他のアドオンパッケージの機能を統合しています (**knitr**　Sweave + cacheSweave + pgfSweave + weaver + `animation::saveLatex` + `R2HTML::RweaveHTML` + `highlight::HighlightWeaveLatex` + 0.2 * brew + 0.1 * SweaveListingUtils + その他).

- 透明性とはユーザーが入力と出力に完全にアクセスできることを意味します. 例えば R ターミナル上では `1 + 2` は `[1] 3` を出力しますが, **knitr** によってユーザーは `1 + 2` を LaTeX の `\begin{verbatim}`

と `\end{verbatim}` や, HTML タグ`<div class="rsource">` と`</div>` の間に出力したりできますし, `[1] 3` を `\begin{Routput}` と `\end{Routput}` の間に出力できたりします. 詳細は 2 章『フック』を参照してください.

- **knitr** は R コードを R ターミナル上で実行した場合でもユーザーの意図する結果と一致することを目指しています. 例えば `qplot(x, y)` でそのままグラフを出力できます (`print()` は不要です) し, デフォルトでコードチャンク内の**全ての**プロットが出力されます.
- **pgfSweave** や **cacheSweave** といったパッケージは有用な機能を Sweave に提供しました (高品質な tikz のグラフィックとキャッシュ) が, **knitr** はこの実装をより簡単にしました.
- **knitr** のデザインによってあらゆる言語 (例: R, Python, awk) が入力可能となり, あらゆるマークアップ言語の形式で出力が可能となりました (例: LaTeX, HTML, Markdown, AsciiDoc, reStructuredText).

このパッケージは GitHub 上で開発されており, インストール手順と FAQ もここで参照できます. パッケージの README も確認してください. このサイトは **knitr** の完全なドキュメントとして提供されており, メインマニュアル, グラフィックス, 用例, knitr-examples を閲覧することができます. さらに統合的な参考資料としては, knitr book (邦訳未刊行) をご覧ください.

## モチベーション

Sweave の拡張の難しいところは **utils** パッケージから大量のコードをコピーする必要があることでした (`SweaveDrivers.R` のコードは 700 行を超えています). そしてこれは上述の 2 つのパッケージ両方での作業です. コードをコピーペーストしたら, パッケージ開発者は R の公式バージョンの変更にとても慎重に対応せねばなりません — うんざりする仕事だとは思いませんか. **knitr** パッケージでは sweave の処理の全行程を小規模な管理しやすい関数群へとモジュール化したため, 喜ばしいことにメンテナンスも機能拡張も容易になりました (例えば簡単に HTML への対応もできるようになりました). その一方で **knitr** は多くのビルトイン機能を持ちますが, パッケージの機能のコア部分をハックする必要のないようになっています. そして, Sweave のマニュアルの FAQ の項目にあったいくつかの問題は, **knitr** は直接解決することができます.

> 旧態依然とした態度をプログラム構造へと変えましょう.「我々の主な仕事は何をすべきかコンピュータに教えることである」と考えるのをやめ, むしろ「人間に対してコンピュータにさせたかったことを説明する」ように注力することです
>
> — ドナルド E. クヌース『文芸的プログラミング』, 1984

## 特徴

本パッケージのアイディアは他のパッケージから借用し, いくつか (キャッシュなど) は異なる形で再実装されています. 以下にパッケージの特徴をいくつか挙げます:

- **信頼性の高い**出力: バックエンドで `evaluate` を使用することで, **knitr** においてあらゆるコードは, 結果のプリント, グラフのプロットそして警告・メッセージ・エラーでさえ, R ターミナルでするのと同様に書き出されます. (厳密にプログラミングする場合, これらは無視するべきではありません. 特に警告は).
  - **ggplot2** や **lattice** のようなグリッドベースのグラフィックパッケージでは, ユーザーはしばしば `print()` を書くのを忘れてしまう, これは些細な問題です. 実際に R ターミナルで出力する時に `print()` は不要ですから. **knitr** ではあなたが意図したとおりの結果を得ることができます.
- キャッシュ機能の組み込み: **cacheSweave** のようですが, しかし **knitr** ではキャッシュの保存や遅延読み込みのため base R の関数を直接使い, そして最も顕著な違いはキャッシュされたチャンクは依然として出力もできることです (**cacheSweave** ではキャッシュされたチャンクは. `print()` を明示的に書いたとしても何も出力しませんが, **knitr** はキャッシュされても通常通り出力します).
- R コードの整形: R コードの自動整形に **formatR** パッケージが使用されます (長い行の折り返し, スペースとインデントの追加, など). これは `keep.source=FALSE` としてもコメントが犠牲になることはありません.
- 20 を超えるグラフィックデバイスを直接サポートしています. チャンクオプションに `dev='CairoPNG'` と書けば, すぐさま **Cairo** パッケージの `CairoPNG()` に切り替えることができますし, `dev='tikz'` なら **tikzDevice** パッケージの `tikz()` に切り替えられます. これ以上簡単にすることなどできないでしょう? これらの組み込みのデバイス (正確に言うならラッパー) は単位にインチを使用しています. ビットマップでもです (ピクセルは `dpi` オプションに基づいて変換されます. デフォルトは 72 です).
- グラフィックスにおけるさらなる柔軟性:
  - 出力の幅と高さをさらに指定することが出来ます (`fig.width` はグラフィックデバイスのオプションで, `out.width` は文書に出力する際のものです. たとえば LaTeX なら `out.width='.8\\textwidth'` とします.)
  - グラフの位置も再調整できます. グラフが生成された位置に正確に掲載することも, チャンクとまとめて表示することもできます (`fig.show='hold'` オプションを使用してください).
  - 最後のグラフだけ欲しいというのでない限り (`fig.keep='last'` オプションを使用してください), コードチャンクごとに複数のグラフが保存されます.
- R コードはコードチャンクに直接書くだけでなく, 他の R スクリプトファイルから読み込むこともでき, 文書を書きながらコードを実行するのが簡単になりました (特に LyX で有益です).
- パワーユーザーのために, さらなるカスタマイズが可能になっています.
  - R コードをパースする際に正規表現は拒否されます. つまり, `<<>>=`, `@`, `\Sexpr{}` などを使う必要はありません. もしやりたいのなら任意のパターンを使用できます. 例: `%% begin.rcode`, `%% end.rcode`.
  - フック (hook) は出力の制御を拒否します. 例えばエラーメッセージを赤い太字で表示したいとしたら, ソースコードをイタリック体で表示したいとしたら, フック機能をコードの実行前および実行語に定義することが出来ます. フック機能はこのパッケージの力を無限に拡大する可能性を持ちます. (例えば, アニメーションとか, rgl 3D プロットとか...)

多くの取り組みが, デフォルトでの美しい出力と可読性の向上をもたらしています. 例えばコードチャンクはシンタックスハイライトされて表示され, LaTeX ではさらに (**framed** パッケージを使うことで) 薄灰色

の背景色がつきます．そのため他のテキストよりやや目立って表示されます．読書体験はきっと `verbatim` や `Verbatim` 環境を使うよりも良いものとなります．プロンプトで表示される先頭の`>` や `+` はデフォルトでは**出力されません** (`prompt=TRUE` で表示することもできます)．このような記号はコードに割り込んでくるので，コードをコピーして自分で実行する際に非常に不便であり，ドキュメントを読む時にまったくもって迷惑していました (訳注: **とてもよくわかる**)．

## 謝辞

Sweave, **pgfSweave**, **cacheSweave**, **brew**, **decumar**, **R2HTML**, **tikzDevice**, **highlight**, **digest**, **evaluate**, **roxygen2**, そしてもちろん，R の開発者に対し，多くのひらめきとツールをもたらしてくれたことに感謝します．多くのベータテスターによるフィードバックに心から感謝します．本パッケージは **documar** のデザインに基づいて始まりました．

## FOAS

**knitr** が Foundation for Open Access Statistics (FOAS) の協力のもとで開発されたことを誇りに思います．FOAS はフリーウェア，オープンアクセス，統計学における再現可能な研究を推進するという世界規模の課題を持った非営利の公益法人です．

# 第 1 章

# オプション

チャンクオプションとパッケージオプションについて

オリジナルのページ: https://yihui.org/knitr/options/

オリジナルの更新日: 2020-06-30

---

このドキュメントでは **knitr** 全般の機能を紹介しており, `Rnw` や `Rhtml` の仕様についても言及しています. R Markdown でも **knitr** の機能は使われますが, R Markdown 独自の中間処理によって, 最終的な出力がここで説明されているものと異なる可能性がある点に注意してお読みください.

**knitr** パッケージはソースコード, テキスト, グラフ, チャンクで使用するプログラミング言語といった, コードチャンクのコンポネントのほとんど全部をカスタマイズするための多くのオプションを提供します. `knit` 処理のカスタマイズをパッケージレベルでカスタマイズするオプションもあります. この章では **knitr** で使用できる全てのチャンクオプションとパッケージオプションを解説します. 以下のリスト中で, オプションのデフォルト値になっているものはカッコ内に表記しています.

## 1.1 チャンクオプション一覧

チャンクオプションはチャンクのヘッダに書きます. チャンクヘッダの構文は文書フォーマットがなんであるかに依存します. 例えば.`Rnw` ファイル (R + LaTeX) であれば, `<<` `>>=` という記号の中に書きます. `.Rmd` ならば, チャンクヘッダは ```` ```{r} ```` 内に書きます. 以下の例は主に `.Rmd` (R Markdown) の場合ですが, ほとんどのチャンクオプションはどのフォーマットでも使用可能です.

チャンクオプションは以下のように **タグ名=値**という形式で書きます.

```
```{r, my-chunk, echo=FALSE, fig.height=4, dev='jpeg'}`r ''`
```
```

**チャンクラベル**は特殊なチャンクオプションです (例: 先ほどの例の `my-chunk` がそれにあたります). これは唯一のタグが不要なチャンクオプションです (つまり, 値のみ書くことになります). もし **タグ名=値**の形式で書きたいのならば, チャンクオプション名の `label` を明示的に使うこともできます.

```
```{r label="my-chunk"}`r ''`
```
```

各チャンクのラベルは文書内において一意であることが前提です. 特にキャッシュとグラフのファイル名はチャンクラベルで紐付けているため重要です. ラベルのないチャンクは `unnamed-chunk-i` という形式でラベル名が割り当てられます. `i` は順に整数が割り当てられます.

文書全体のチャンクオプションのデフォルト値を変更するために `knitr::opts_chunk$set()` を使うことができます. 例えば以下のようなチャンクを文書の冒頭に書きます.

```
```{r, setup, include=FALSE}`r ''`
knitr::opts_chunk$set(
  comment = '', fig.width = 6, fig.height = 6
)
```
```

チャンクオプションの豆知識をいくつか掲載します.

1. チャンクヘッダは 1 行で書かねばなりません. 改行してはいけません.
2. チャンクラベルとファイルパスにスペース, ピリオド `.`, アンダースコア `_` を使用するのは避けましょう. セパレータが必要ならば, ハイフン `-` の使用を推奨します. 例えば `setup-options` はラベル名として望ましいですが `setup.options` や `chunk 1` は良くありません. `fig.path = 'figures/mcmc-'` はパス名として良いですが, `fig.path = 'markov chain/monte carlo'` は良くありません.
3. 全てのオプションの値は **R の構文として適切でなければなりません**. チャンクオプションを関数の引数のように考えると良いでしょう.

- 例えば **character** 型をとるオプションは引用符で囲まなければなりません. 例: `results = 'asis'` や `out.width = '\\textwidth'`. ただしリテラルのバックスラッシュは二重のバックスラッシュが必要なことを忘れないでください.

- 理論上はチャンクラベルもまた引用符で囲む必要がありますが, 利便性のため書かなくとも自動で引用符が追加されます (例: ```{r, 2a}``` は ```{r, label='2a'}``` として扱われます).
- R のコードとして有効なものである限り, いくらでも複雑な構文を書くことができます.

コードチャンクの本文に #| の後に続けてチャンクオプションを書くという, 代替的な構文も使用できます.

````
```{r}`r '''`
#| my-chunk, echo = FALSE, fig.width = 10,
#| fig.cap = "This is a long long
#|   long long caption."
plot(cars)
```
````

この構文では, チャンクオプションはチャンク本文の冒頭で, 連続した行で書く必要があります. つまり, 全てのオプションは, 行頭の特殊なコメント記号 #| から書き始める必要があります. チャンクオプションとコードの間を 1 行開けるかどうかは任意です. この記法はオプションの改行が許容されます. 好きなだけ改行してオプションを書くことができます. 同じオプションが本文とチャンクヘッダ (```{} の内側) の両方で指定された場合, 前者が後者を上書きします. あるいは, チャンク内では<**タグ**>: <**値**> のような YAML 式の記法でオプションを書くこともできます. 通常は, この記法では 1 行に 1 つづつオプションを書かねばなりません. 以下がその例です.

````
```{r}`r '''`
#| echo: false
#| fig.width: 10
```
````

YAML 記法を選択した場合, 生の R の式ではなく YAML の値として有効なものを書かねばなりません.

以下では **オプション**: (**デフォルト値**; 値の型) という形式で, **knitr** で使えるチャンクオプションのリストを掲載します.

### 1.1.1 コード評価関連

- **eval** (TRUE; logical または numeric): コードチャンクを評価するかどうか. どの R の評価式を評価するかを選ぶために numeric のベクトルを使用することもできます. 例: eval=c(1, 3, 4) ならば 1 つ目, 3 つ目, そして 4 つ目の評価式を評価し, eval = -(4:5) は 4, 5 つ目の式以外の全てを評価します.

### 1.1.2 テキストの出力関連

- **echo** (TRUE; logical または numeric): 出力される文書にソースコードを表示するかどうか.「表示」「隠す」に対応する TRUE/FALSE に加え, どの R の評価式を表示するかを選ぶために numeric のベクトルを使用することもできます. 例: echo=2:3 は 2, 3 番めの評価式を表示し, echo = -4 は 4 番目だけを非表示にします.
- **results** ('markup'; character): 実行結果のテキストの部分をどう表示するかを制御します. このオプションは通常のテキスト出力にのみ影響することに注意してください (警告・メッセージ・エラーは適用範囲外です). 取りうる値は次のとおりです.
  - markup: 出力の文書フォーマットに応じて適切な環境でテキスト出力をマークアップします. 例えば R Markdown ならば "[1] 1 2 3" が **knitr** によって以下のように加工されます. この場合, results='markup' は囲み (```) 付きのコードブロックとして出力されることを意味します.

```
[1] 1 2 3
```

  - asis: テキスト出力を「そのまま」書き出します. つまり, 生の結果テキストをマークアップ一切なしでそのまま文書に書き出します.

```{r, results='asis'}
cat("I'm raw **Markdown** content.\n")
```

  - **hold** チャンクと flush の全てのテキスト出力をチャンクの末尾に固定します.
  - **hide** (TRUE または FALSE): テキスト出力を表示しません.
- **collapse** (FALSE; logical): 可能であれば, ソースと出力をつなげて 1 つのブロックにするかどうかです (デフォルトではソースと出力はそれぞれ独立したブロックです). このオプションは Markdown 文書でのみ有効です.
- **warning** (TRUE; logical): warning() で出力される警告文を保存するかどうかです. FALSE の場合, 全ての警告文は文書に出力されず, 代わりにコンソールに出力されます. 警告文の一部を選ぶインデックスとして, numeric 型のベクトルを指定することもできます. このインデックスは何番目も警告文を表示するかどうかを意味していることに注意してください. 例えば 3 はこのチャンクから投げられた 3 番目の警告文を意味するものであって,「何番目の R コードの警告文の出力を許可するか」ではないことに注意してください.
- **error** (TRUE; logical): stop() で出力されるエラー文を保持するかどうかです. デフォルトの TRUE では, **コード評価はエラーが出ても停止しません!** エラー時点で停止させたい場合はこのオプションを FALSE に指定してください. **R Markdown ではこのデフォルト値は FALSE に変更されていることに注意**してください. チャンクオプションに include=FALSE がある場合, 起こりうるエラーを見落とさないように, **knitr** はエラー時点で停止するようになります. これらの注意書きを理解した上で, 起こりうる

エラーをなおも無視したい場合は, error に evaluate::evaluate() で定義されている数値を指定してください.
  - 0 ならば, ターミナル上にコードをペーストしたときのように, エラーが起こった後も評価されます.
  - 1 ならば, エラーが発生した時点で評価を停止しますが, 正常終了扱いとなります. よって, エラーに対処するには the error フック を使用します.
  - 2 ならば, エラーは通常の信号を発します. つまり, R の実行が中止されます.
- **message** (TRUE; logical): message() が出力するメッセージ文を (warning オプションと同様に) 表示するかどうかです.
- **include** (TRUE; logical): 出力する文書にチャンクの出力を含めるかどうかです. FALSE ならばなにも書き出されませんが, コードの評価はなされ, チャンク内にプロット命令があるのならグラフのファイルも生成されます. よってこの図をそれ以降で任意に挿入することもできます.
- **tab.cap** (NULL; character): コードチャンク内の kable() 関数に対してキャプションを与えます. このオプションを動作させるには, チャンク内で 1 度だけ kable() を呼び出す必要があります.
- **strip.white** (TRUE; logical): 出力時にソースコードの冒頭と末尾から空白行を除去するかどうかです.
- **class.output** (NULL; character): テキストの出力ブロックに追加するクラス名のベクトル. このオプションは R Markdown で HTML を出力する際にのみ機能します. 例えば class.output = c('foo', 'bar') はテキスト出力が<pre class="foo bar"></pre> で囲まれたブロックとして書き出されます.
- **class.message**/**class.warning**/**class.error** (NULL; character): class.output と同様に, R Markdown においてそれぞれ メッセージ文, 警告文, エラー文のブロックに対してクラス名を与えます. class.source もまた同様にソースコードのブロックに対して適用されます. “コードの装飾” のセクションを参照してください.
- **attr.output**/**attr.message**/**attr.warning**/**attr.error** (NULL; character): 上記の class.* オプション群と同様に, Pandoc に対してコードブロックの属性を指定します. つまり class.* は attr.* の特殊ケースです. 例: class.source = 'numberLines' は attr.source = '.numberLines' と等価ですが, attr.source は任意の値を取ることができます. 例えば, attr.source = c('.numberLines', 'startFrom="11"').
- **render** (knitr::knit_print; function(x, options, ...)): チャンクで表示される値に対して適用する関数です. 関数の第 1 引数には (x) はチャンクの各評価式が評価された結果が与えられます. このチャンクのチャンクオプションがリストとして第二引数 opstions に与えられます. この関数は文字列を返すことを想定しています. 詳細は package vignette (vignette('knit_print', package = 'knitr')) を参照してください.
- **split** (FALSE; logical): 出力ブロックを分割したファイルに書き出すかどうか. LaTeX ならば \input{} で読み込み, HTML ならば<iframe></iframe> タグで読み込まれます. このオプションは .Rnw, .Rtex そして .Rhtml でのみ機能します.

### 1.1.3 コードの装飾関連

- **tidy** (FALSE): R コードを整形するかどうかです. 他の有効な値は次のとおりです.
  - TRUE (tidy = 'formatR' と等価です): 整形のために formatR::tidy_source() を呼び出します.

  - 'styler': コード整形のために styler::style_text() を呼び出します.
  - 整形されたコードを返す, function(code, ...) {} という形式の任意の関数.
  - 整形が失敗した場合, 元の R コードは変更されません (警告は表示されます).
- tidy.opts: (NULL; list) tidy オプションで指定した関数へのオプション引数のリストです. 例えば tidy.opts = list(blank = FALSE, width.cutoff = 60) は tidy = 'formatR' に対して空白行を削除し各行が 60 文字におさまるように改行しようとします.
- **prompt** (FALSE; logical): R コードにプロンプト記号 (> など) を付けるかどうかです. ?base::options ヘルプページの prompt と continue を参照してください. プロンプト記号の追加は, 読者がコードをコピーするのを難しくさせるため, prompt=FALSE のほうが良い選択であることに留意してください. エンジンが R 以外の場合, このオプションはうまく機能しないことがあります (issue #1274).
- **comment** ('##'; character): テキスト出力の各行の先頭文字です. デフォルトでは, コメントアウトできるよう ## となっているので, 読者は文書から任意の範囲をそのままコピーしても出力部分は無視されるのでそのまま実行することができます. comment = '' を指定することで, デフォルトの ## は除去されます.
- **highlight** (TRUE; logical): ソースコードをシンタックスハイライトするかどうかです[*1].
- **class.source** (NULL; character): 出力された文書のソースコードブロックのクラス名です. 出力ブロックに対して機能する class.output をはじめとする class.* シリーズと同様です.
- **attr.source** (NULL; character): ソースコードブロックの属性です. attr.output などの attr.* シリーズと同様です.
- 
  - lang (NULL; character): コードチャンクの言語名です. デフォルトでは言語名はエンジン名と同じです. 例: r. このオプションは主に Markdown ベースの文書出力でシンタックスハイライトするためのものです.
- **size** ('normalsize'; character): .Rnw 使用時のチャンクサイズのフォントサイズです. 指定可能なサイズは overleaf のヘルプページ (英語) を参照してください[*2].
- **background** ('#F7F7F7'; character): .Rnw 使用時のチャンクブロックの背景色です[*3].
- **indent** (character): チャンクの出力で各行に追加する文字です. 典型的には空白と同義です. このオプションは読み込み専用を想定しており, 値は **knitr** が文書を読み込む際に設定されます. 例えば以下のチャンクでは, indent は空白文字 2 個です[*4].

````
```{r indent-example, echo=2}`r ''`
set.seed(42)
rnorm(10)
```
````

[*1] 訳注: R Markdown ではさらに, YAML フロントマターで適用するハイライトのテーマ名を指定できます
[*2] 訳注: \normalsize, \Large, \LARGE など LaTeX で指定できるフォントサイズを表すマクロのことを指しています
[*3] 訳注: R Markdown では背景色は CSS や class.output などで設定する必要があります. 詳細は R Markdown Cookbook などを参照してください
[*4] 訳注: R Markdown の場合は **knitr** 以外の中間処理があるため, 必ずしもこのルールを守りません

### 1.1.4 キャッシュ関連

- **cache** (FALSE; logical): コードチャンクのキャッシュを取るかどうかです. 初回の実行またはキャッシュが存在しない場合は通常通り実行され, 結果がデータセットが保存され (.rdb, .rdx ファイルなど), それ以降でコードチャンクが評価されることがあれば, 以前保存されたこれらのファイルからこのチャンクの結果を読み出します. ファイル名がチャンクラベルと R コードの MD5 ハッシュ値で一致する必要があることに注意してください. つまりチャンクになんらかの変更がある度に異なる MD5 ハッシュ値が生成されるため, キャッシュはその度に無効になります. 詳細はキャッシュの解説 を参考にしてください.
- **cache.path** ('cache/'; character): 生成したキャッシュファイルの保存場所を指定します. R Markdown ではデフォルトでは入力ファイルの名前に基づきます. 例えば INPUT.Rmd の FOO というラベルのチャンクのキャッシュは INPUT_cache/FOO_*.* というファイルパスに保存されます.
- **cache.vars** (NULL; character): キャッシュデータベースに保存される変数名のベクトルを指定します. デフォルトではチャンクで作られた全ての変数が識別され保存されますが, 変数名の自動検出はロバストではないかもしれませんし, 保存したい変数を選別したい場合もあるかもしれないので, 保存したい変数を手動選択することもできます.
- **cache.globals** (NULL; character): このチャンクで作成されない変数の名前のベクトルを指定します. このオプションは主に autodep = TRUE オプションをより正確に動作させたいときに使います. チャンク B で使われているグローバル変数が チャンク A のローカル変数として使われているときなど. グローバル変数の自動検出に失敗した際に使う場合, こにオプションを使って手動でグローバル変数の名前を指定してください (具体例として issue #1403 を参照してください). さらに, cache.globals = FALSE は, 変数がグローバルかローカルかにかかわらず, コードチャンク内のすべての変数を検出することを意味します.
- **cache.lazy** (TRUE; logical): 遅延読み込み lazyLoad() を使うか, 直接 load() でオブジェクトを読み込むかを指定します. 非常に大きなオブジェクトに対しては, 遅延読み込みは機能しないかもしれません. よってこの場合は cache.lazy = FALSE が望ましいかもしれません (issue #572 を参照してください).
- **cache.comments** (NULL; logical): FALSE の場合, R コードチャンク内のコメントを書き換えてもキャッシュが無効になりません.
- **cache.rebuild** (FALSE; logical): TRUE の場合, キャッシュが有効であってもチャンクのコードの再評価を行います. このオプションはキャッシュの無効化の条件を指定したいときに有用です. 例えば cache.rebuild = !file.exists("some-file") とすれば some-file が存在しないときにチャンクが評価されキャッシュが再構成されます (issue #238 を参照).
- **dependson** (NULL; character または numeric): このチャンクが依存しているチャンクのラベル名を文字ベクトルで指定します. このオプションはキャッシュされたチャンクでのみ適用されます. キャッシュされたチャンク内のオブジェクトは, 他のキャッシュされたチャンクに依存しているかもしれず, 他のチャンクの変更に合わせてこのチャンクも更新する必要があるかもしれません.
  - dependson に numeric ベクトルを与えた場合, それはチャンクラベルのインデックスを意味します. 例えば dependson = 1 ならばこの文書の 1 番目のチャンクに依存することを意味し, dependson = c(-1, -2) は直前の 2 つのチャンクを意味します (負のインデックスは現在のチャンクからの相対

的な位置を表します).
  - opts_chunk$set() によってグローバルにチャンクオプションを設定した場合, このオプションは機能しません. ローカルなチャンクオプションとして設定しなければなりません.
- **autodep** (FALSE; logical): グローバル変数を検出することでチャンク間の依存関係を分析するかどうかを指定します (あまり信頼できません). よって, dependson を明示的に指定する必要はありません.

### 1.1.5 グラフ関連

- **fig.path** ('figure/'; character): 図の保存ファイルパスを生成する際の接尾語. fig.path とチャンクラベルを連結したものがフルパスになります. figure/prefix- のようにディレクトリ名が含まれて, それが存在しない場合はディレクトリが作成されます.
- **fig.keep** ('high'; character): グラフをどのように保存するかです. 可能な値は次のとおりです.
  - high: 高水準プロットのみ保存 (低水準の変更は全て高水準プロットに統合されます).
  - none: 全て破棄します.
  - all: 全てのプロットを保存します (低水準プロットでの変更は新しいグラフとして保存されます).
  - first: 最初のプロットのみ保存します.
  - last: 最後のプロットのみ保存します.
  - 数値ベクトルを指定した場合, その値は保存する低水準プロットのインデックスとなります. 低水準プロットとは lines() や points() などの関数によるグラフ描画のことです. fig.keep について より理解するには次のようなチャンクを考えてください. 通常はこれで 2 つのグラフを出力します (fig.keep = 'high' を指定したので). fig.keep = 'none' としたなら, いかなるグラフも保存されません. fig.keep = 'all' ならば, 4 つのグラフとして保存されます. fig.keep = 'first' ならば plot(1) によって作成されたグラフが保存されます. fig.keep = 'last', なら, 最後の 10 本の垂線を描画したグラフが保存されます.

```
plot(1) # 高水準プロット
abline(0, 1) # 低水準の作図
plot(rnorm(10)) # 高水準プロット
# ループ内での複数の低水準作図 (R 評価式としては 1 つ)
for (i in 1:10) {
  abline(v = i, lty = 2)
}
```

- **fig.show** ('asis'; character): グラフをどのように表示し, 配置するかです. 可能な値は次のとおりです.
  - asis: グラフが生成された場所にそのまま出力します (R ターミナルで実行した場合とおなじように).
  - hold: 全てのグラフをまとめてチャンクの最後に出力します.

  - animate: チャンクに複数のグラフがある場合, 連結して 1 つのアニメーションにします.
  - hide: グラフをファイルに保存しますが, 出力時は隠します.
- **dev** (LaTeX の場合は'pdf'[*5], HTML/Markdown の場合は'png'; character): グラフをファイルに保存する際のグラフィックデバイスです. base R および, **Cairo**, **cairoDevice**, **svglite**, **ragg**, **tikzDevice** パッケージの提供するデバイスに対応しています. たとえば, pdf, png, svg, jpeg, tiff, cairo_pdf, CairoJPEG, CairoPNG, Cairo_pdf, Cairo_png, svglite, gridSVG, ragg_png, tikz, などが使用できます. 有効なデバイスの一覧は names(knitr:::auto_exts) を参照してください. また, function(filename, width, height) という引数を定義した関数名を文字列で与えることでも指定できます. 画像サイズの単位は **常にインチ**です. ビットマップであってもインチで指定したものがピクセルに変換されます.

チャンクオプション dev, fig.ext, fig.width, fig.height, dpi はベクトルを与えることが可能で, 1 つのプロットに対して複数のファイル形式で保存できます. 例えば dev = c('pdf', 'png') は 1 つのグラフに対して 1 つづつ PDF と PNG ファイルを作成します. ベクトルの長さが足りない場合は再利用されます. ファイルが同じ拡張子で作成された場合は, fig.ext でファイルの接尾辞を変更して指定してください. そうでない場合は, 新しく生成されたファイルで上書きされます. 例えば, dev = 'png' と fig.width = c(10, 6) と指定したときに, さらに fig.ext = c('1.png', '2.png') を指定すると, 幅の異なる 2 つのファイルをそれぞれ異なる名前で保存できます.

- **dev.args** (NULL; list): グラフィックデバイスに与える追加の引数です. 例えば dev.args = list(bg = 'yellow', pointsize = 10) を dev = 'png' に与えられます. 特定のデバイスに依存するオプション (詳細はそれぞれのデバイスのドキュメントを確認してくださいさい). dev に複数のデバイスが指定されている場合は dev.args を引数のリストをさらにリストでくくることになるでしょう. それぞれの引数リストが対応するデバイスに与えられます. 例: dev = c('pdf', 'tiff'), dev.args = list(pdf = list(colormodel = 'cmyk', useDingats = TRUE), tiff = list(compression = 'lzw')).
- **fig.ext** (NULL; character): 出力するグラフのファイル拡張子です. NULL ならばグラフィックデバイスに応じて自動決定されます. 詳細は knitr:::auto_exts を確認してください.
- **dpi** (72; numeric). ビットマップデバイスに対する DPI (インチ毎ドット, dpi * inches = pixels) です.
- **fig.width**, **fig.height** (いずれも 7; numeric): グラフの幅と高さです. 単位はインチです. グラフィックデバイスに与えられます.
- **fig.asp** (NULL; numeric): グラフのアスペクト比, つまり 高さ / 幅 の比です. fig.asp が指定された場合, 高さ (fig.height) は fig.width * fig.asp によって自動設定されます.
- **fig.dim** (NULL; numeric): fig.width と fig.height を指定する長さ 2 の数値のベクトルです. 例: fig.dim = c(5, 7) は fig.width = 5, fig.height = 7 の省略形です. fig.asp と fig.dim が指定された場合, fig.asp は無視されます (警告文が出力されます).
- **out.width**, **out.height** (NULL; character): 出力時の画像の幅と高さです. 実体としての幅と高さである fig.width と fig.height とは異なります. つまりグラフは文書に表示される際にスケールが調整されます. 出力フォーマットに応じて, これら 2 つのオプションはそれぞれ特殊な値を取ることができます. 例えば LaTeX ならば .8\\linewidth, 3in, 8cm などと指定でき, HTML ならば 300px と指定でき

[*5] 訳注: pdf は日本語表示に向いていないため, cairo_pdf などを利用することをおすすめします

ます．.Rnw ならば out.width のデフォルト値は \\maxwidth に変更され，その値は framed のページで書いたように定義されます．例えば'40%' のようにパーセンテージで指定もでき，これは LaTeX では 0.4\linewidth に置き換えられます．

- **out.extra** (NULL; character): 図の表示に関するその他のオプションです．LaTeX で出力する場合は \includegraphics[] に挿入される任意の文字に対応し (例: out.extra = 'angle=90' ならば図の 90 度回転)，HTML なら<img /> に挿入されます (例: out.extra = 'style="border:5px solid orange;"')．
- **fig.retina** (1; numeric): このオプションは HTML で出力する際にのみ適用されます．Retina ディスプレイ に対して画像サイズを調整する比率 (多くの場合は 2 を指定します) です．チャンクオプションの dpi を dpi * fig.retina で，out.width を fig.width * dpi / fig.retina で計算します．例えば fig.retina = 2 なら，画像の物理サイズが 2 倍となり，その表示サイズは半分になります．
- **resize.width**, **resize.height** (NULL; character): LaTeX で出力する際に \resizebox{}{} コマンドで使われます．これら 2 つのオプションは Tikz グラフィックスをリサイズしたい場合のみ必要になります．それ以外に通常使うことはありません．しかし **tikzDevice** の開発者によれば，他の箇所のテキストとの一貫性のため，Tikz グラフィックスはリサイズを想定していません．値の 1 つでも NULL ならば，! が使用されます (この意味がわからない方は **graphicx** のドキュメントを読んでください)．
- **fig.align** ('default'; character): 出力時の画像の位置揃え (アラインメント) です． 可能な値は default, left, right, center です．default は位置について特に何も調整しません．
- **fig.link** (NULL; character) 画像に与えるリンク．
- **fig.env** ('figure'; character): 画像に使われる LaTeX 環境．例えば fig.env = 'marginfigure' ならば \begin{marginfigure} で囲まれます．このオプションの使用は fig.cap が指定されいることが条件です．
- **fig.cap** (NULL; character): 図のキャプションです．
- **fig.alt** (NULL; character) HTML 出力時の図の <img> タグの alt 属性に使う代替テキストです．デフォルトでは，代替テキストが与えられた場合チャンクオプション fig.cap には代替テキストが使われます．
- **fig.scap** (NULL; character): 図の短縮キャプションです．出力が LaTeX の場合のみ意味をなします．短縮キャプションは \caption[] コマンドに挿入され，大抵の場合は PDF 出力時の「図一覧」で表示される見出しとして使われます．
- **fig.lp** ('fig:'; character).; 図の相互参照に使われるラベル[*6]の接頭語で，\label{} コマンドに挿入されます．実際のラベルはこの接頭語とチャンクラベルを連結して作られます．例えば図のラベルが ```{r, foo-plot} ならば，デフォルトでは図のラベルは fig:foo-plot になります．\label{} への挿入は，LaTeX としてレンダリングされるチャンクに依存することに注意してください．詳細はこの issue を見てください．[*7]
- **fig.id** (NULL; logical): TRUE を指定すると，コードチャンクから生成された画像に，自動生成された ID が割り当てられます．つまり，HTML 出力の場合は，画像が<img id="..." /> と書かれます．デフォルトでは，ID は fig.lp の値・チャンクラベル・fig.cur の値を連結したものになります．ラテン文字以

[*6] 訳注: チャンクラベルと混同しないでください

[*7] 訳注: この注釈の意図は，**knitr** や **rmarkdown** は LaTeX 文書の相互参照をサポートしていないことに注意を促すものです．例えば R Markdown 文書で相互参照を使用するには，**bookdown** パッケージが必要になるため，このオプションを使っただけで相互参照が使用できるわけではありません．

外の全ての文字は, ダッシュに置き換えられます. 例えば, 'fig:hello world 1' は'fig-hello-world-1' になります. ID を生成する関数を定義して与えることも可能です. この関数は, 現在のチャンクのオプションをリスト型の引数として受け取り, 1 要素の character 型を返すようにしてください. 例えばこのように書きます. fig.id = function(options) { paste0('img-', options$label, options$fig.cur) }.

- **fig.pos** ('': character): LaTeX の \begin{figure}[] に使われる, 画像の位置調整オプション[*8]を指定します.
- **fig.subcap** (NULL): subfigures のためのキャプションです. 複数のグラフが 1 つのチャンクにあり, かつ fig.subcap も fig.cap is NULL である場合, \subfloat{} が個別の画像の表示に使われます (この場合はプリアンブルに \usepackage{subfig} と書く必要があります). 具体例は 067-graphics-options.Rnw を参照してください.
- **fig.ncol** (NULL; integer). subfigure の数です. 例えばこの issue を見てください (fig.ncol も fig.sep も LaTeX でのみ機能します).
- **fig.sep** (NULL; character): subfigures どうしの間に挿入されるセパレータを指定する文字ベクトルです. fig.ncol が指定された場合, デフォルトでは fig.sep に N 個ごとに \newline が挿入されます (N は列の数です). 例えば fig.ncol = 2 ならばデフォルトは fig.sep = c('', '', '\\newline', '', '', '\\newline', '', ...) となります. fig.sep の長さがサブ画像の数より大きい場合を除いて, $i$ 番目のセパレータは $i$ 番目のサブ画像の後に追加されます. この例外の場合は, fig.sep の 1 番目の要素が最初のサブ画像の前に追加され, $(i+1)$ 番目の要素が $i$ 番目の画像の後に追加されます.
- **fig.process** (NULL; function): 画像ファイルに対する後処理の関数です. 関数は画像のファイルパスを引数として, 挿入したい新しい画像のファイルを返すものであるべきです. 関数に options 引数がある場合, この引数にチャンクオプションのリストが与えられます.
- **fig.showtext** (NULL; logical): TRUE ならばグラフの描画前に showtext::showtext_begin() が呼ばれます. 詳細は **showtext** パッケージのドキュメントを参照してください[*9].
- **external** (TRUE; logical): tikz グラフィックの処理 (PDF 生成時のコンパイル前の処理) を外部化するかどうかです. **tikzDevice** パッケージの tikz() デバイスを使う場合 (つまり dev='tikz' を指定したとき) のみ使用され, コンパイル時間を短縮することが可能です.
- **sanitize** (FALSE; character). tikz グラフィックでサニタイズ (ここでは, LaTeX で特殊な意味を持つ文字のエスケープ処理) するかどうかです. 詳細は **tikzDevice** パッケージのドキュメントを参照してください.

さらにこの他に, ユーザーが使用することを想定していない隠しオプションが 2 つあります. fig.cur (複数の図表がある場合の, 現在の図番号 / インデックス) と fig.num (チャンク内の図の合計数) です. これら 2 つのオプションは **knitr** が複数の図そしてアニメーションを処理するためにあります. 場合によっては手動で保存した画像ファイルを使ってアニメーションを書き出す場合などに役に立つかもしれません (使用例として graphics manual を参照してください).

[*8] 訳注: LaTeX では通常は図の位置は調整されますが, fig.pos='H' ならばその位置で固定されます

[*9] 訳注: showtext は手っ取り早く日本語を表示できますが, いくつかの制約があります. 詳細は『おまえはもう R のグラフの日本語表示に悩まない (各 OS 対応)』『R でのフォントの扱い』などを見てください.

### 1.1.6 アニメーション関連

- **interval** (1; numeric): アニメーションの 1 フレームごとの時間 (単位は秒) です.
- **animation.hook** (knitr::hook_ffmpeg_html; function または character). HTML 出力時のアニメーション作成用のフック関数を指定します. デフォルトでは FFmpeg を使って WebM 動画ファイルに変換します.
  - 別のフック関数として **gifski** パッケージの knitr::hook_gifski 関数は GIF アニメーションを作ることができます.
  - このオプションは 'ffmpeg' や 'gifski' といった文字列を指定することもできます. これらは対応するフック関数の省略形です. 例: animation.hook = 'gifski' は animation.hook = knitr::hook_gifski を意味します.
- **aniopts** ('controls,loop'; character): アニメーションに対する追加のオプションです. 詳細は LaTeX の **animate** パッケージのドキュメントを参照してください.
- **ffmpeg.bitrate** (1M; character): WebM 動画の質を制御するための FFmpeg の引数 -b:v に対応する値を指定できます.
- **ffmpeg.format** (webm; character): FFmpeg の出力する動画フォーマットです. つまり, 動画ファイルの拡張子名です.

### 1.1.7 コードチャンク関連

- **code** (NULL; character): 指定された場合, そのチャンクのコードを上書きします. この機能によって, プログラミング的にコード挿入が可能になります. 例えば code = readLines('test.R') とすれば test.R の内容を現在のチャンクで実行します.
- **file** (NULL; character): これが指定された場合, code オプションが, チャンクとして読み込まれた外部ファイルの内容で上書きされます. file = "test.R" というチャンクオプションは code = xfun::read_all("test.R") を指定しているのと同じことを意味します.
- **ref.label** (NULL; character): 現在のチャンクのコードに引き継ぐ, 別のチャンクのラベルの文字列ベクトルを指定します (動作例はチャンク参照 を確認してください).

### 1.1.8 子文書関連

- **child** (NULL; character): 親文書に挿入する子文書のファイルパスを示す文字ベクトルを指定します.

### 1.1.9 言語エンジン関連

- **engine** ('R'; character): コードチャンクの言語名です．指定可能な名前は names(knitr::knit_engines$get()) で確認できます．例: python, sql, julia, bash, c, など．knitr::knit_engines で他の言語を使うためのセットアップが可能です．
- **engine.path** (NULL; character): 実行可能なエンジンのパスを指定します．あなたのお使いのシステムの別の実行ファイルを使用するためのオプションです．例えば python はデフォルトでは /usr/bin/python を参照しますが，他のバージョンを使うため engine.path = '~/anaconda/bin/python' などと指定することもできます[*10]．engine.path もまたパスのリストを与えられます．これによってエンジンごとにそれぞれパスを指定することができます．以下のコードが例です．リストの名前はエンジン名と一致する必要があります．

```
knitr::opts_chunk$set(engine.path = list(
  python = "~/anaconda/bin/python",
  ruby = "/usr/local/bin/ruby"
))
```

- **engine.opts** (NULL; character): 言語エンジンに与える追加引数です．チャンクの段階ではオプションを文字列またはリストで指定することができます．

````
```{bash, engine.opts='-l'}`r ''`
echo $PATH
```
````

````
```{cat, engine.opts = list(file = "my_custom.css")}`r ''`
h2 {
  color: blue;
}
```
````

グローバルレベルでは，要素名に言語名を与えた文字列のリストが使用できます．engine.path と同様に，knitr::opts_chunk$set() で引数のテンプレートを作ると便利です．

[*10] 訳注: R Markdown の場合，Python のバージョンは reticulate パッケージでも制御できます．むしろそちらをつかったほうが便利だと思われます．

```
knitr::opts_chunk$set(engine.opts = list(
  perl = "-Mstrict -Mwarnings",
  bash = "-o errexit"
))
```

各エンジンはそれぞれ自身の `engine.opts` を持ち, 固有のオプションを定義します. 言語エンジンのドキュメントを調べるべきでしょう. R Markdown クックブックには cat[*11], sass/scss[*12]エンジンの例が掲載されています.

### 1.1.10 オプションテンプレート関連

- **`opts.label`** (`NULL`; `character`): このオプションは, チャンクオプションを他のオプションのテンプレート knitr::opts_template **や, コードチャンクから引き継ぐ機能があります. テンプレートの詳細は**?knitr::opts_template **を参照してください. このオプションは, ラベル名の代入された** character **型のベクトルを受け取ります. このベクトル内の各要素に対して, `**knitr**` は最初に** knitr::opts_template‘ にあるラベルを探し出して, 見つかれば, 現在のチャンクにそのオプションテンプレートを適用しようとします. その後で, 文書内の他のチャンクラベルと名前の一致するものを探し出し, 見つかれば, 現在のチャンクに一致したチャンクのオプションを適用します. ここで参照されたコードチャンクは,「被参照コードチャンク (refferenced code chunk)」と呼ばれます.
  チャンクオプションの優先順位は, (1) ローカルチャンク (2) 被参照コードチャンク, (3) `knitr::opts_template` のテンプレート, (4) `knitr::opts_chunk` となります.
  チャンクオプション `opts.label` に対し, `opts.label = TRUE` は特殊な値で, `opts.label = ref.label` と同じ意味になります. つまり, `ref.label` オプションで指定したコードチャンクからオプションを参照します. `ref.label` と `opts.label` のいろいろな使用例は #121 in the knitr-examplesrepository を見てください.

### 1.1.11 ソースコードの抽出関連

- **`purl`** (`TRUE`; `logical`): ソースドキュメントから `knitr::purl()` でソースコードを取りだす時, このチャンクを含めるか除外するかどうかです.

[*11] 翻訳版: https://gedevan-aleksizde.github.io/rmarkdown-cookbook/eng-cat.html
[*12] 翻訳版: https://gedevan-aleksizde.github.io/rmarkdown-cookbook/eng-sass.html

### 1.1.12 その他のチャンクオプション

- **R.options** (NULL; list): コードチャンク内でのローカルな R オプションを指定します. これらは options() によってこのコードチャンクの直前に一時的に設定され, 実行後に戻されます.

## 1.2 パッケージオプション一覧

パッケージオプションは knitr::opts_knit を使用することで変更できます. **knitr::opts_chunk と混同しないでください**. 使用例は以下のとおりです.

```
01 knitr::opts_knit$set(progress = TRUE, verbose = TRUE)
```

別の方法として, R の基本関数である options() を使ってパッケージオプションを設定する場合は ?knitr::opts_knit を参照してください.

可能なオプションは次のとおりです.

- **aliases** (NULL; character): チャンクオプションのエイリアスを指定する名前付きベクトルです. 例えば c(h = 'fig.height', w = 'fig.width') は **knitr** に h は fig.height w は fig.width と同じ意味だと認識させます. このオプションは名前の長いチャンクオプションのタイピング労力を削減できます.
- **base.dir** (NULL; character): グラフを生成する際のディレクトリの絶対パスです.
- **base.url** (NULL; character): HTML ページに掲載する画像のベース URL です.
- **concordance** (FALSE; logical): この機能は RStudio によって実装されている機能で, .Rnw でのみ有効です. 入力ファイルの行番号に対応した行番号を出力ファイルに書き出すかどうかを指定します. これにより, 出力から入力の誘導が可能になり, 特に LaTeX のエラー発生時に役に立ちます.
- **eval.after** (c('fig.cap', 'fig.scap', 'fig.alt';character): **オプション名の文字ベクトルを指定します. このオプションはチャンクが評価された**後で**評価され, 他の全てのオプションはチャンクが評価される前に評価されます. 例えば** eval.after = 'fig.cap' **が指定されているときに** fig.cap = paste('p-value is', t.test(x)$p.value) **とすると**,eval.after **にはチャンクの評価後の** x' の値が使用されます.
- **global.par** (FALSE; logical): TRUE にすると, それ以前のコードチャンクでの par() での設定が引き継がれます (もちろん, この設定は R グラフィックスのみで有効です). デフォルトでは **knitr** はグラフの記録のために新規のグラフィックデバイスを開き, コードの評価後に閉じるため, par() による設定はその都度破棄されます.
- **header** (NULL; character): 文書の開始前に挿入するテキストを指定します. (例えば, LaTeX ならば \documentclass{article} の直後, HTML ならば<head> タグの直後). このオプションは LaTeX プリアンブルや HTML ヘッダでコマンドやスタイルの定義をするのに有用です. ドキュメントの開始地点は

knitr::knit_patterns$get('document.begin') で知ることができます. このオプションは .Rnw と .Rhtml 限定の機能です[*13].

- label.prefix: (c(table = 'tab:'); character) ラベルの接頭語を指定します. 現時点では kable::kable() によって生成される表のラベルに対する接頭語のみサポートしています.
- **latex.options.color**, **latex.options.graphicx** (NULL): それぞれ LaTeX パッケージの **color** と **graphicx** に対するオプションを指定します. これらのオプションは .Rnw 限定の機能です[*14].
- **latex.tilde** (NULL): .Rnw 文書のシンタックスハイライトされたチャンク出力内でのチルダ文字を表すための, LaTeX コマンドの文字列です (使用例は issue #1992 を見てください).
- **out.format** (NULL; character): 可能な値は latex, sweave, html, markdown, jekyll です. このオプションは入力ファイル名に応じて自動で決定され, 自動設定されるフック関数に影響します. 例えば?knitr::render_latex を参考にしてください. このオプションは knitr::knit() が実行される**前に**設定する必要があります (文書内で設定しても機能しません).
- **progress** (TRUE; logical): knitr::knit() の実行中にプログレスバーを表示するかどうかを指定します.
- **root.dir** (NULL; character): コードチャンク評価時のルートディレクトリを指定します. NULL の場合, 入力ファイルと同じ場所が指定されます.
- **self.contained** (TRUE; logical): 出力する文書が自己完結的であるべきかどうかを指定します (.tex ファイルにスタイルを書き出すか, html に CSS を書き出すか). このオプションは .Rnw と .Rhtml でのみ機能します[*15].
- **unnamed.chunk.label** (unnamed-chunk; character): ラベルを設定していないチャンクのラベルの接頭語を指定します.
- **upload.fun** (identity; function): ファイルパスを引数にとり, ファイルに対して処理を行い出力フォーマットが HTML または Markdown の場合に文字列を返す関数を指定します. 典型的な使い方として, 画像をアップロードしそのリンクを返す関数を指定します. 例えば knitr::opts_knit$set(upload.fun = knitr::imgur_upload) でファイルを http://imgur.com にアップロードできます (?knitr::imgur_upload を参照してください).
- **verbose** (FALSE; logical): 情報を冗長に詳細するか (例えば各チャンクで実行された R コードやメッセージログなど), チャンクラベルとオプションのみ表示するかを指定します.

## 1.3 グローバル R オプション

グローバル R オプションとは, base R で options() で設定されるもののことです. 以下は, **knitr** の挙動に影響するオプションの一覧です.

- **knitr.bib.prefix** (R-; character): knitr::write_bib() で生成される書誌情報のキーの値の接頭辞です.
- **knitr.child.warning** (TRUE; logical): child を使用して子文書を読み込んでいるコードチャンクで, コードチャンクが空になっていない場合に警告を発します. このようなコードチャンクに書かれたコードは

[*13] 訳注: R Markdown ではヘッダの設定は YAML フロントマターで行います
[*14] 訳注: R Markdown ではこの機能もやはり YAML フロントマターが担当しています
[*15] 訳注: R Markdown では出力フォーマット関数に同様のオプションが用意されていることが多いです

無視されるためです．FALSE を指定して警告を抑制できます．

- **knitr.digits.signif** (FALSE; logical) インライン R 式内の数値をフォーマットする方法を指定します．TRUE は format() を，FALSE は round() を意味します．を指定できます．前者はグローバルオプション digits で有効桁数を指定することを意味します．後者は digits で小数点以下の桁数を指定することを意味します．options(digits =) でグローバルオプションを設定できます．
- **knitr.duplicate.label** (NULL): "allow" を指定すると，同一文書内でのチャンクラベルの重複を許容します．
- **knitr.include.graphics.ext** (FALSE; logical): LaTeX 出力時に，\includegraphics{} へ出力されるファイルパスにファイル拡張子を含めるかどうかを指定します．
- **knitr.progress.simple** (NULL; logical): 出力時に進捗バーを表示するかどうかを指定します．
- **knitr.progress.fun** (knitr:::txt_pb; function): function(total, labels) {} という形式の関数を指定します．この関数は，チャンクの数が与えられる total 引数と，チャンクラベルのベクトルが与えられる labels を受け取り，かつ，list(update = function(i) {}, done = function() {}) のように 2 つのメソッドのリストを返す必要があります．update() メソッドは現在のチャンクのインデックス i を引数に取り，進捗バーの更新を返します．done() メソッドは進捗バーを閉じます．以下は **cli** パッケージの進捗バーを使用する場合の例です．

```
function(total, labels) {
  id <- cli::cli_progress_bar(
    total = total, .auto_close = FALSE
  )
  list(
    update = function(i) {
      cli::cli_progress_update(id = id)
    },
    done = function() {
      cli::cli_process_done(id)
    }
  )
}
```

そして以下は，Windows のプログレスバーを使用する例です．つまり，Windows でのみ動作します．

```
function(total, labels) {
  pb <- winProgressBar("Knitting...", max = total)
  list(
    update = function(i) {
```

```
      setWinProgressBar(pb, i, label = labels[i])
    },
    done = function() {
      close(pb)
    }
  )
}
```

- **knitr.progress.linenums** (FALSE; logical): 進捗バーに行数を表示するかどうかを指定します. デフォルトでは, チャンクラベルのみが表示されます.
- **knitr.progress.output** (""; character または connection): **knitr** のデフォルトのテキスト形式の進捗バーに対して, 進捗バーの出力先を指定できます. デフォルトでは, 進捗バーは stdout() に出力されます. もし代わりに stderr を使うなら, stderr() を使うことになります.
- **knitr.purl.inline** (FALSE; logical): knitr::purl() 出力にインライン R コードを含めるかどうかを指定します.
- **knitr.svg.object** (FALSE; logical): TRUE ならば, svg 形式のプロットは, self-contained で HTML を出力する場合は <svg> タグで HTML に直接埋め込まれ, self-contained でない場合は<object> タグで参照されます. FALSE の場合は,<img> タグが使用されます.

# 第 2 章

# フック

コードチャンク実行前後のカスタマイズ関数, 出力の調整, チャンクオプションの操作について

オリジナルのページ: https://yihui.org/knitr/hooks/

オリジナルの更新日: 2017-02-03

---

The object `knit_hooks` in the **knitr** package is used to set hooks; the basic usage is `knitr::knit_hooks$set(name = FUN)` (see objects for details) where `name` is the name of a chunk option (can be arbitrary), and `FUN` is a function. There are two types of hooks: chunk hooks and output hooks. Hook functions may have different forms, depending what they are designed to do. **knitr** パッケージの `knit_hooks` オブジェクトはフックの設定に使用します. 基本的には, `knitr::knit_hooks$set(name = FUN)` というふうに使います. `name` の部分は任意の引数名にすることが可能であり, チャンクオプションの名前を指定します. 詳細はオブジェクト (A 章) を見てください. `FUN` は関数です. フックには 2 種類あります. チャンクフックと出力フックです. フック関数は, その設計次第で様々な形態をとります.

## 2.1 チャンクフック

チャンクフックは対応するチャンクオプションの値が `NULL` ではないとき, コードチャンクの前後に実行されます. そしてフック関数には以下のような引数を使用できます.

```
hook_foo <- function(before, options, envir, name, ...) {
  if (before) {
    ## チャンク実行前の処理
  } else {
    ## チャンク実行後の処理
```

```
  }
}
```

**knitr** が文書を処理する時, 各チャンクの直前に `hook_foo(before = TRUE)` が呼び出され, チャンク直後に `hook_foo(before = FALSE)` が呼び出されます. チャンクがキャッシュされていたり評価しないように設定されていない限り, そのように動作します. そのため, `hook_foo(before = FALSE)` はチャンクの後に呼び出されます.

引数の `options` は現在のチャンクに設定された**オプション** (1 章) のリストです. 例えば `options$label` は現在のチャンクのラベルです.

引数の `envir` はコードチャンクが評価させる環境です.

引数の `name` は `knit_hooks` のフックが関連付けられた名前です.

全ての引数はオプションです. 例えば, `function(before)` も `function(options, ...)` フック関数として有効なシグネチャです. フック関数に対応する引数が存在するならば, リスト `list(before, options, envir, name)` の値はそれぞれの引数に与えられます.

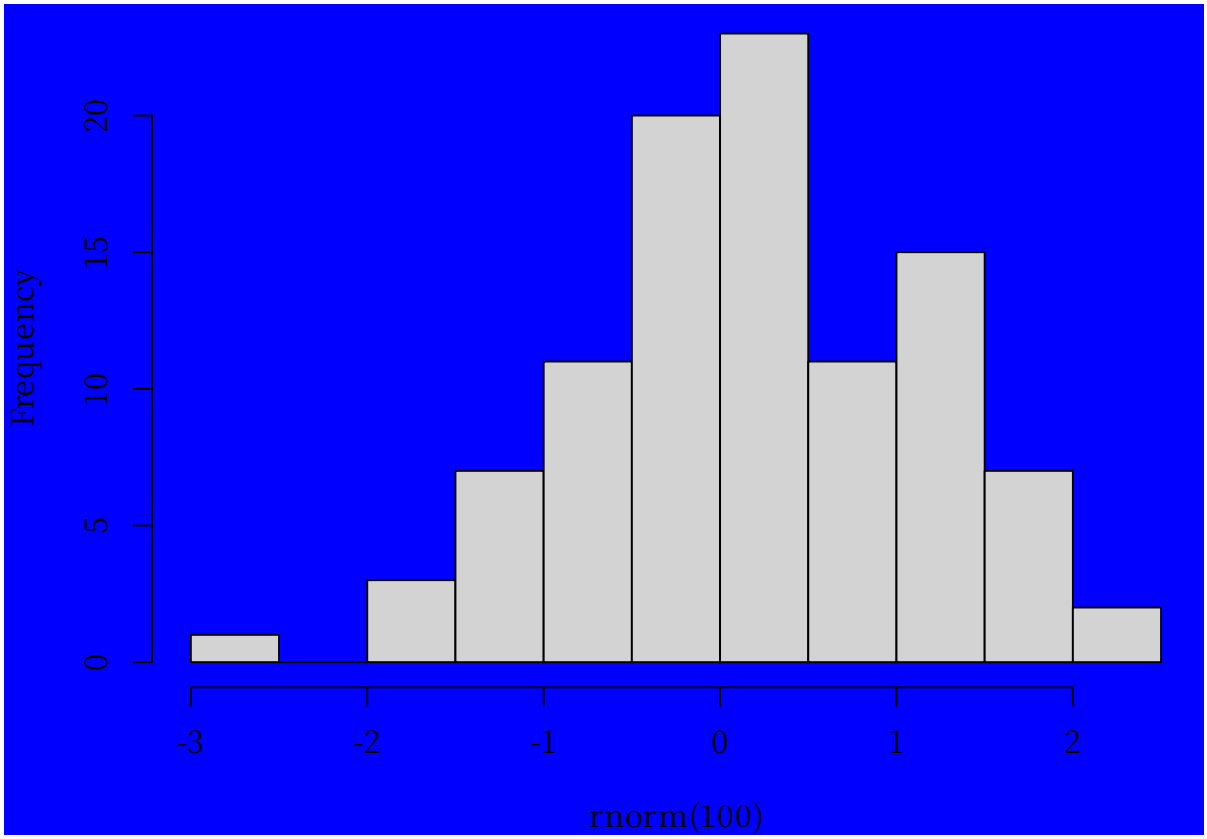
たとえば, `small_mar` というオプションに対してフックを設定したいなら, 以下のようにします. 後の 2 つの引数はチャンクフックのオプション引数です. 例えば, 次のように `small_mar` オプションにフックを設定します.

**訳注**: マージン調整だけでは違いがわかりづらい, というかこのドキュメントを生成している R Markdown はデフォルトでマージン調整するので, 違いをわかりやすくするため, 背景色を変更する処理も追加しています.

```
knitr::knit_hooks$set(small_mar = function(before, ...) {
  if (before) {
    par(mar = c(4, 4, .1, .1))
  } # 上と右のマージンを狭める
  par(bg = "blue") # 背景色を青にする
})
```

そしてフックに設定した関数はこのように呼び出されます. `small_mar` オプションの値に必ず `TRUE` を設定する必要はありません. `NULL` 以外の任意の値さえ与えられていれば動作します.

````
```{r, myplot, small_mar=TRUE}`r ''`
hist(rnorm(100), main = '')
```
````




**knitr** のフックは出力にテキストを挿入することにも使えます．そのため，このタイプのフック関数は文字列を返す必要があります．この機能はフックの能力を大いに広げます．**rgl** パッケージを例に取りましょう．**rgl** によって生成された 3D グラフを Markdown または HTML 文書に挿入したい時，このタイプのフック関数を考える事になるでしょう．なお，この例よりも洗練された `hook_rgl()` 関数が **rgl** パッケージにあるので参考にしてください．

```
knitr::knit_hooks$set(rgl = function(before, options, envir) {
  if (!before) {
    # チャンクコードが評価された後の処理
    if (rgl.cur() == 0) {
      return()
    } # アクティブなデバイスがないかどうか
    name <- paste0(options$fig.path, options$label, ".png")
    rgl.snapshot(name, fmt = "png")
    return(paste0("![rgl plot](", name, ")\n"))
  }
})
```

そしてコードチャンクはこのようになります.

````
```{r, fancy-rgl, rgl=TRUE}`r ''`
library(rgl)  # 用例は ?plot3d から
open3d()
x = sort(rnorm(1000)); y = rnorm(1000); z = rnorm(1000) + atan2(x,y)
plot3d(x, y, z, col = rainbow(1000))
```
````

Markdown の場合 `![rgl plot](fancy-rgl.png)` と出力されているでしょう.

ここまでの話を要約します.

1. フックは `knit_hooks` で, `knit_hooks$set(foo = FUN)` という構文で設定されます.
2. あるチャンクで `foo` というチャンクオプションが `NULL` 以外の値をとる場合, このフック関数 `FUN` が実行されます.
3. フックはチャンクの直前と直後に実行できます.
4. フックによって返される文字列は修正が加えられることなく出力ブロックに書き出されます.

さらなる用例は 045-chunk-hook.md (source) を参照してください.

## 2.2 出力フック

出力フックはチャンクからの**生の**出力をカスタマイズし洗練するために使います. 様々な種類の出力に対処するための 8 つの出力フック関数が存在します.

- **source**: ソースコード
- **output**: 通常の R の出力で, 警告文, メッセージ文, エラー文を除くもの (つまり, 通常の R ターミナルで出力されていたものです)
- **warning**: `warning()` による警告文
- **message**: `message()` によるメッセージ文
- **error**: `stop()` によるエラー文 (コードチャンクとインライン R コードの両方に適用されます)
- **plot**: 出力されるグラフ
- **inline**: インライン R コードの出力
- **chunk**: チャンクの全ての出力 (つまりその前のフックにも生み出されたもの)
- **document**: 文書全体の出力 (デフォルトでは `base::identity` が適用されます)

これらのフックは全て `function(x, options)` という形式をとります (例外として, `inline` と `document` のフックのみ引数は `x` の 1 つです), `x` が出力の文字列で, `options` が現在のチャンクのオプションのリストです. 出

力フックに関する情報と用例をさらに詳しく知りたい場合は R Markdown クックブックの出力フックの章を参考にしてください.

以下は error フックの用例になります. これは R Markdown 上で, エラー文に対して追加の整形処理を実行するフックです.

```
knitr::knit_hooks$set(error = function(x, options) {
  paste(c(
    '\n\n:::{style="color:Crimson; background-color: SeaShell;"}',
    gsub("^## Error", "**Error**", x),
    ":::"
  ), collapse = "\n")
})
```

このようなチャンクでフックの動作をテストします

```
```{r, error=TRUE}`r ''`
1 + "a"
```
```

**Error** in 1 + “a”: 二項演算子の引数が数値ではありません

デフォルトではチャンクフックは空ですが, 出力フックはデフォルト設定があり, 以下のようにしてリセットできます.

```
knitr::knit_hooks$restore()
```

本パッケージは出力を異なる部品にわけてそれぞれにデフォルトのフックを設定し, さらに LaTeX, HTML, Jekyll といった異なる出力フォーマットごとに用意しています. render_*() という一連の関数群は, 例えば render_latex(), redner_html(), など出力フォーマットごとにそれぞれ異なる, 組み込みの出力フックを提供するためにあります. 出力フックはドキュメント内で設定すべきですが, knitr::knit() が文書を処理する前にフックを設定したなら render_*(), たとえば render_markdown(), render_html() を最初に呼び出さなければなりません. hooks_markdown() などの hooks_*() 関数で, 設定を変えることなくこれらの出力フックにアクセスすることができます.

**訳注**

R Markdown の場合, 基本的な出力フォーマットにもデフォルトでフックが定義されており, 処理内容によっては予期せぬ結果になることがあるため, 単純な上書きや`$restore()` は意図しない動作につながることがあります. 詳細は “R Markdown Cookboox” Ch. 12[α] を確認ください.

[α] 翻訳版: https://gedevan-aleksizde.github.io/rmarkdown-cookbook/output-hooks.html

本パッケージは出力を異なる部品にわけてそれぞれにデフォルトのフックを設定し, さらに LaTeX, HTML, Jekyll といった異なる出力フォーマットごとに用意しています. `render_*()` という一連の関数群は, 例えば `render_latex()`, `redner_html()`, など出力フォーマットごとにそれぞれ異なる, 組み込みの出力フックを提供するためにあります. 出力フックはドキュメント内で設定すべきですが, `knitr::knit()` が文書を処理する前にフックを設定したなら `render_*()` を先に呼び出す必要があります. この `*` には出力フォーマットの名前あてはまります. 例えば `render_markdown()`, `render_html()` があります. `hooks_markdown()` などの `hooks_*()` 関数で, 設定を変えることなくこれらの出力フックにアクセスすることができます.

以降は各フォーマットでの詳細を記します.

### 2.2.1 LaTeX: `render_latex()`

出力ファイルタイプが LaTeX の場合, デフォルトのフックはほとんどのチャンク出力を `verbatim` 環境で囲んで出力し, `inline` 出力における数値を指数表記で出力します. 詳細はチャンク出力の制御のデモを参照してください). `plot`, `chunk` フックはより複雑な処理をしています.

- デフォルトでは `plot` フックは出力の信頼性を維持するため多くの要因に影響されます.
  - たとえばグラフィックデバイスが `tikz` ならば, `\input{}` コマンドが使われますし[*1], それ以外では通常は `\includegraphics{}` コマンドが使われます.
  - `out.width`, `out.height` オプションに依存して, フックはグラフのサイズを設定し直します. 例えば `\includegraphics[width=.8\textwidth]{file}` のように変更されます. 1 つのチャンクに複数のグラフがある場合, `fig.show='hold'` を設定するとともに, 複数の画像を適切なサイズで横に並べて表示できるように設定できます (たとえば `.45\textwidth`[*2] とすれば横に 2 つのグラフを並べられます).
  - tikz のグラフは `\input{}` で挿入するため, このやり方は正しくありませんが, チャンクオプション `resize.width` と `resize.height` は複数の tikz グラフを横に並べられます (`\resizebox{resize.width}{resize.height}{file.tikz}` という書き方によって. もしいずれかのオプションが `NULL` なら`!` で置き換えられます. 詳細は LaTeX パッケージの `graphicx` のドキュメントを参照してください). このフック関数によってユーザーは自動レポート生成の全能力を使いこ

[*1] 訳注: tikz の画像は LaTeX ソースコードで記述されるため, `\input{}` でテキストファイルとして読み込む必要があります.
[*2] 訳注: 本文幅の 45%

なせます. 単一チャンクの複数グラフとグラフのサイズの設定が可能になるだけでなく, base R のグラフィックスや grid 系のグラフィックス (例: **ggplot2**), あるいはグリッド系のグラフを並べて表示することもできるためです. この機能がなかったら, R で 1 つのウィンドウにこういった複数のグラフを 1 つにまとめるのがどんなに難しいことか考えても見てください[*3].

  - グラフのアラインメントを決めるために `fig.align` には (`default`, `left`, `right`, `center`) の 4 つの値が用意され, `fig.align='center'` を指定すれば簡単に画像を中央揃えにできます.

- デフォルトの `chunk` フックは主にチャンクの装飾に使われています.
  - LaTeX の `framed` パッケージがユーザーの TeX ソフトウェアパッケージ (TeXLive か MikTeX か他の何か[*4]) にインストールされているなら, `chunk` フックはカスタマイズした背景色 (デフォルトでは薄灰色) にした `kframe` 環境に全ての出力を挿入することで, チャンクの視認性を向上させます (他の地の文よりも強調されますが, とても目立つというほどでもないはずです).
  - 最後に, 全ての出力が `knitrout` 環境で囲まれます. この環境はユーザーが LaTeX で再定義できます.

### 2.2.2 Sweave: `render_sweave()`

ソースコードを `Sinput` 環境に挿入し, その出力を `Sioutput` 環境に挿入し, そしてチャンク全体を `Schunk` 環境に挿入します. このテーマの使用にはスタイルファイル `Sweave.sty` か, 少なくともこれら 3 つの環境を定義することが必要です.

### 2.2.3 Listings: `render_listings()`

Sweave 同様に, `Sweavel.sty` が使用されます.

### 2.2.4 HTML: `render_html()`

HTML ファイルに書き出すにあたって, フックは出力を自動で調整します. 基本的にチャンクによる出力はクラス付きの `div` レイヤーに挿入されます. たとえば, ソースコードは`<div class="knitr source"></div>`, チャンク全体は`<pre></pre>` に, インラインの出力は`<code class="knitr inline"></code>` に書き出されます[*5].

### 2.2.5 Markdown: `render_markdown()`

ソースコードとその出力はスペース 4 つでインデントされます. GitHub Flavored Markdown のため, ソースコードは ```` ```r ```` と ```` ``` ```` の間に挿入され, 出力部分は ```` ``` ```` と ```` ``` ```` の間に挿入されます.

---

[*3] 訳注: 現在は **patchwork** や **cowplot** パッケージなどの登場により, そこまで難しいことではなくなりつつあります.
[*4] 訳注: あるいは Yihui 氏による TinyTeX が使いやすいでしょう.
[*5] 訳注: R Markdown では最終的に出力される HTML はさらに pandoc などの処理を経由しているため, これとは異なります

### 2.2.6 Jekyll: render_jekyll()

このサイト[*6]を構築するために, Jykell 用に特別にいくつかのフックを用意する必要がありました. これらは実際かなり単純なものです. R ソースコードはハイライト環境に挿入し言語を r に設定する, 残りの出力部分は text 言語に設定したハイライト環境 (ほとんど何もハイライトしない) に挿入するだけです. 現在, グラフは Markdown の構文に従って書き出されます.

### 2.2.7 reStructuredText: render_rst()

コードは :: の後に挿入され, スペース 4 個でインデントされるか, sourcecode ディレクティブに挿入されます.

## 2.3 オプションフック

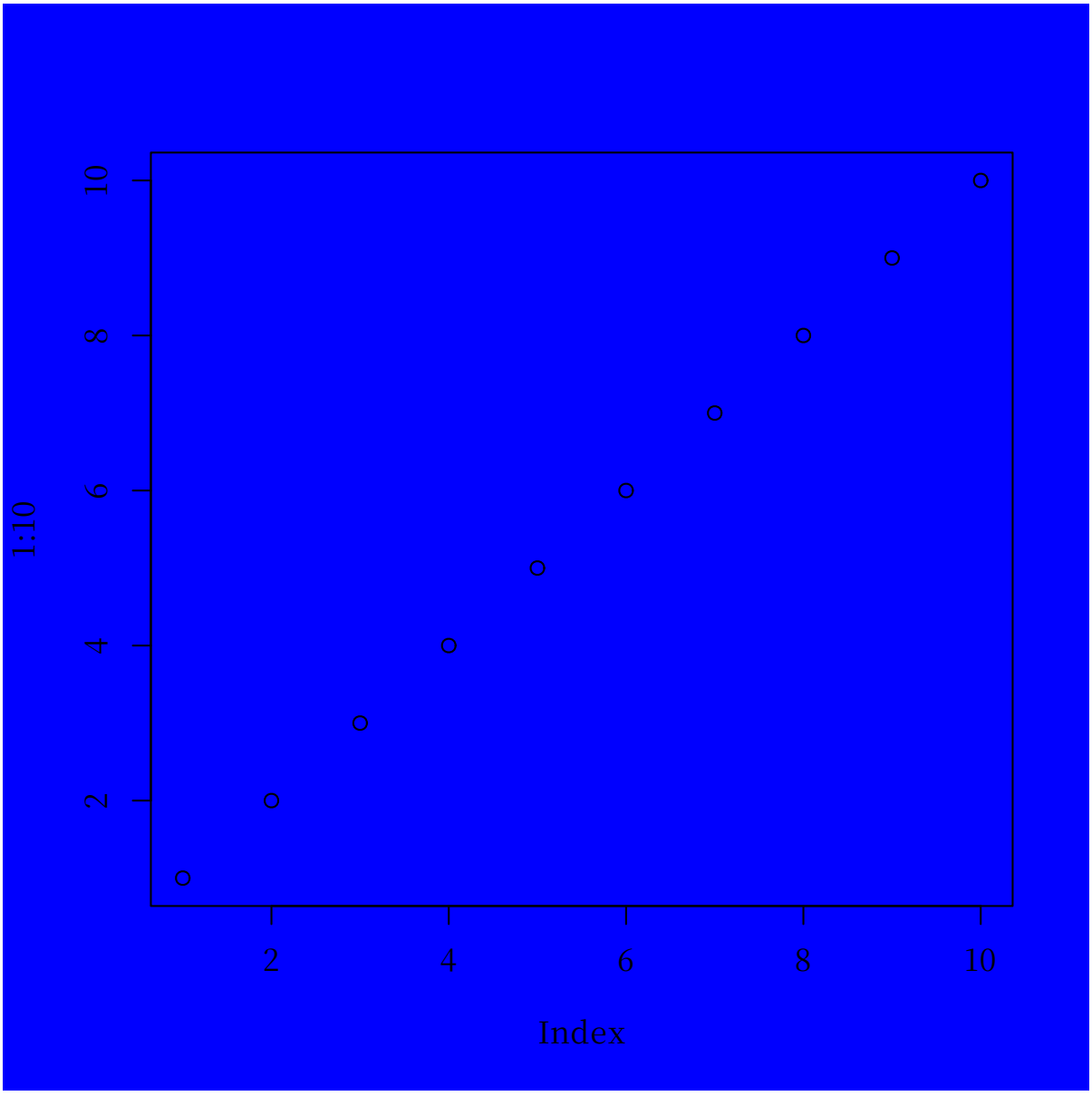
他のチャンクオプションの値に応じて別のチャンクオプションの値を変えたいとき, opts_hooks をつかってそれを実行することがあるかもしれません. オプションフックは対応するチャンクオプションの値が NULL 以外であるときに実行されます. たとえば fig.width を常に fig.height 以上の値に調整できます.

```
knitr::opts_hooks$set(fig.width = function(options) {
  if (options$fig.width < options$fig.height) {
    options$fig.width <- options$fig.height
  }
  options
})
```

fig.width は NULL になることがないため, このフック関数は常にチャンクの直前の, チャンクオプションが確認される前に実行されます. 以下のコードチャンクは上記のフックを設定することで, fig.width の実際の値は初期値の 5 の代わりに 6 が適用されます.

```
```{r fig.width = 5, fig.height = 6}`r '''
plot(1:10)
```
```

[*6] 訳注: オリジナルが掲載されている Yihui 氏のサイトのこと




訳注: knit_hooks 同様に, opts_hooks にも restore() メソッドが用意されています.

# 第 3 章

# 使用例

ソースと出力のデモ

オリジナルのページ: https://yihui.org/knitr/demos/

オリジナルの更新日: 2017-02-03

---

**訳注**: 現在は knitr の主な利用場面は R Markdown との併用だと思うので, それらと関係の薄いページは翻訳していません. また, 編集上の問題から, ここで挙げられているページのうち翻訳済みのものは全てナビゲーションバーの「用例」パートでもリンクされています.

Github の knitr-examples はより豊富なコレクションになっています. このページはむしろドキュメント用途として作られています. 他のユーザーによる **knitr** のショーケースを見ることもできます.

- 2011-12-03 (未翻訳) Minimal examples - Examples for Rnw, Markdown, HTML and LaTeX
- 2011-12-04 NA - キャッシュ機能の使用例について
- 2011-12-05 NA - パッケージマニュアルについて
- 2011-12-06 (未翻訳) LyX - Using knitr with LyX
- 2011-12-07 (未翻訳) Code Externalization - Use an external R script with your document
- 2011-12-08 (未翻訳) Beamer - Using knitr in beamer slides
- 2011-12-09 NA - **knitr** におけるグラフィックスの力について
- 2011-12-10 NA - `listings` と `knitr` の併用
- 2012-01-14 NA - チャンクの再利用方法
- 2012-01-18 (未翻訳) Child documents - Input child files into the main document
- 2012-01-22 (未翻訳) Package vignettes - How to build package vignettes with knitr
- 2012-01-25 NA - チャンクの 6 種類の出力とインライン出力を操作する
- 2012-01-26 (未翻訳) Quick reporting - Build a report based on an R script
- 2012-02-01 (未翻訳) Org-mode - Use knitr in Org-mode

- 2012-02-02 (未翻訳) RStudio - knitr support in RStudio
- 2012-02-11 (未翻訳) Pretty printing - Print highlighted source code of a function
- 2012-02-12 (未翻訳) Upload images - Publish images from chunks in the web
- 2012-02-24 (未翻訳) Sweave - Transition from Sweave to knitr
- 2012-02-27 (未翻訳) Eclipse - Configure Eclipse to work with knitr
- 2012-02-29 NA - **knitr** における LaTeX のデフォルトスタイル
- 2012-03-16 NA - Emacs, TeX Maker, TeXShop, WinEdt, そして TextMate などについて
- 2012-05-01 (未翻訳) HTML5 slides - making HTML5 slides with pandoc and knitr
- 2012-05-04 (未翻訳) Language engines - Use other languages in knitr
- 2012-11-09 (未翻訳) JavaScript - Combine R and JS applications like D3
- 2013-02-10 (未翻訳) WordPress - Publish blog posts from R + knitr to WordPress
- 2013-03-06 (未翻訳) Pandoc - Convert Markdown to other formats via Pandoc
- 2013-03-11 NA - ユーザーたちによる使用例

# 第 4 章

# よくある質問 (FAQ)

オリジナルのページ: https://yihui.org/knitr/faq/

オリジナルの更新日: 2017-02-17

---

この FAQ は issues や私 (Yihui) のブログや E メールに届いた質問などを蓄積したものです. 個人的な考えとして, 私は FAQ という概念の大ファンでもありませんし, FAQ とはほとんどバグのような存在であると思っているため, このページのリストはなるべく少なくしたいです.

1. 「**knitr** が動かないんだけど…」
   - まず最初に, あなたの R パッケージ (`update.packages()` を使います) とたぶん R 本体も (ところで, 現在のあなたの R のバージョンは何ですか?) 更新してください. それから動くかどうかを確認してください. もしそれでも動かなかったら,「必要最低限の再現例 (minimal reprex)」のファイルと `xfun::session_info('knitr')` の実行結果を issue に投稿してください.
2. 「パッケージのサイトの説明が役に立たないときはどこで質問するのがいいですか?」
   - 何を質問したいかにもよりますが, 以下のような選択肢があります[*1](特に最初の 2 つは私もよく巡回しています).
     - (推奨) **Stack Overflow**: 一般的な質問 (より専門的で早い回答がつきます). `r` と `knitr` タグを忘れずにつけてください.
     - **Github issues**: バグ報告と機能追加の要望のみにしてください.
     - **knitr mailing list** または **R-help** のメーリングリスト: 一般的な質問とその回答が, 一般公開される E メールによってやりとりされます.
     - **私の個人的な E メール**: 本当にプライベートな問題でない限り非推奨です[*2].
     - **Twitter** (@xieyihui): 本当に簡単な問題だと確信があるなら.
3. 「**knitr** のソースドキュメントを書くのに最適のエディタソフトは何ですか?」

[*1] 訳注: 英語が苦手な場合は, Stack Oveflow 日本語版, または R-wakalang などのコミュニティがあります

[*2] 訳注: リンク先の投稿を要約すると, Yihui 氏個人で次から次へと来る質問をさばくのは限界があるし, オープンコミュニティは多くの回答者がいて既出の質問に対する答えも共有でき効率的だ, ということです

- 初心者にとってはたぶん RStudio が良いです．**knitr** は LyX, Emacs/ESS, WinEdt, Tinn-R and やその他多くのエディタでサポートされています．

4. 「> とか + とかのプロンプト記号はどこへいったんですか? 出力にこれがないと落ち着きません」
   - 私はこれらが意味をなさないと考えているので, デフォルトではこれらは除去されています．R の本に> や + を載せるのを嫌う理由はこのようなものです: 本に書かれた R コードを読む際に, **こいつらは私の精神をめちゃくちゃにひねりちぎろうとし, 私の両眼に流血を欲してきます**．マジで `1 + 1` ではなく`> 1+1` という表記を読むのがお好みな奇特な方は, チャンクオプション の章にあるように `prompt` オプションで設定できます[*3]．
5. 「ワーキングディレクトリとはなんですか? コードチャンク内でワーキングディレクトリを変更できますか?」
   - あなたはそういうことをしないほうがよいです．ワーキングディレクトリは常に `getwd()` で分かります (出力ファイルは全てここに書き出されます) が, コードチャンクは入力ファイルがあった場所で評価しています．R コード実行中のワーキングディレクトリ変更は一般的に, よくない使い方 (バッドプラクティス) です．詳しくは issue #38 での議論をご覧ください．また, できることなら常に, 絶対パスによるディレクトリ指定も回避するべきです．代わりに相対パス指定を使ってください．そのようなコードは再現性を損なうからです．ここまで言ってもまだコードチャンク内で `setwd` を使おうというのなら, 新しく設定したワーキングディレクトリは指定したチャンクにのみ適用され, 以降のコードチャンクでは本来のワーキングディレクトリに差し戻されることを留意してください．
6. 「出力された灰色の背景色ボックスが狭すぎます.」
   - それはボックスが狭すぎるからではありません．ボックス幅は現在行の幅が適用されます．つまりあなたの出力のほうが広すぎるのです．ページのマージンを超えるような出力を避けるため, もっと小さな `width` オプションを設定してください (例: `options(width = 60)`, 詳細は example 038 を参照してください.)
7. 「コードチャンクにリテラルや生のコードを書く方法は? たとえばパースせずにコードチャンクを書きたいです．チュートリアルに便利だと思います.」
   - コードチャンクで `verbatim` エンジンを使えば可能です．これがその例です．

````
01 ````{verbatim}
02 ```{r, eval=TRUE}
03 1 + 1
04 ```
05 ````
````

   - インラインの R コードでは `knitr::inline_expr()` を使うことになるでしょう (**knitr** ver. 1.8 以降で使用できます)．R Markdown を使っているなら, あるトリックが可能です．`` `r `` の**直後に，ス**

[*3] 訳注: Yihui 氏がこれを嫌う理由はここでも簡潔に書かれています．R Markdown においては, それ以外の方法でもコードの装飾をカスタマイズできます．たとえばこの翻訳版でなされているようにコピーボタンや行番号をつけたりできます．

**ペースなどを挟まずすぐ**改行を入れ, 二重のバッククオート (バックティック) のペアでインラインの評価式全体を囲みます. 例えば以下のようになります. この挙動の説明に興味があるならこのページを見てください (訳注: 未翻訳).

**訳注**: ソースコードでの記述

````
````
```{r, eval=TRUE}`r ''`
1 + 1
```
````
````

実際に表示されるもの

````
```{r, eval=TRUE}`r ''`
1 + 1
```
````

インライン式では以下のようになります

**ここに生のインライン `R` 評価式を出力**: `` `r 1+1` ``.

ここに生のインライン R 評価式を出力: `r 1+1`.

8. 「何かお役に立てることはありますか?」
   - paypal でドネートできますし, たのしい GIF アニメを私にツイートしたり, Github で **knitr** のリポジトリをフォークしたりコードの改善に貢献したりできます.
   - パッケージや knitr book を引用してください. R で `citation('knitr')` をしてみてください.
9. 「このドキュメントの修正や小さな変更を投稿するにはどうすればいいですか?」
   - R パッケージを修正したい場合はリポジトリ へ移動し, ツールバー右上の Edit ボタンを押します. それから必要な修正をします. 投稿の概要を書いて, **Propose file change** をクリックして, プルリクエストを投稿してください.
   - このウエブサイト (https://yihui.org/knitr) 上の修正や変更提案は, ページ左側の `Edit this page` を押して, あとは Github の手順に従ってください[*4].

[*4] 訳注: 原典ではなくこの翻訳版に対する修正提案は https://github.com/Gedevan-Aleksizde/knitr-doc-ja で受け付けています

# 用例

## knitr のエディタ

Emacs, TeX Maker, TeXShop, WinEdt, そして TextMate などについて

オリジナルの記事: https://yihui.org/knitr/demo/editors/

オリジナルの更新日: 2012-03-16

---

**訳注**: このページは R Markdown ではなく, Rnw を想定した説明です. R Markdown は基本的に RStudio での編集が最も使いやすいと思われます.

私は以前 Lyx (未翻訳), RStudio (未翻訳) Emacs Org-mode (未翻訳), Ecripse (未翻訳) について書きました. その他にも, Texmaker や Windt といったエディタで **knitr** を使うことができます. ポイントは R を呼び出して **knitr** パッケージを読み込み, それから `knit()` または `knit2pdf()` を呼び出すことです.

## Texmaker

`User --> User Commands --> Edit User Commands` から, Rnw 文書を処理するためのカスタムコマンドを定義できます.

R の実行ファイルパスが `PATH` 環境変数にない場合, 上記のコマンドに `Rscript.exe` のフルパスを書く必要があります. こんなかんじに:

```
"C:/Program Files/R/R-2.14.2/bin/Rscript.exe" -e "knitr::knit2pdf('%.Rnw')"
```

`Rscript.exe` の場所さえ知っていれば, R を開いて `R.home('bin')` を実行すれば見つけられます. そうすればどんな Rnw 文書ファイルに対しても, この `knitr` コマンドでコンパイルすることができます.

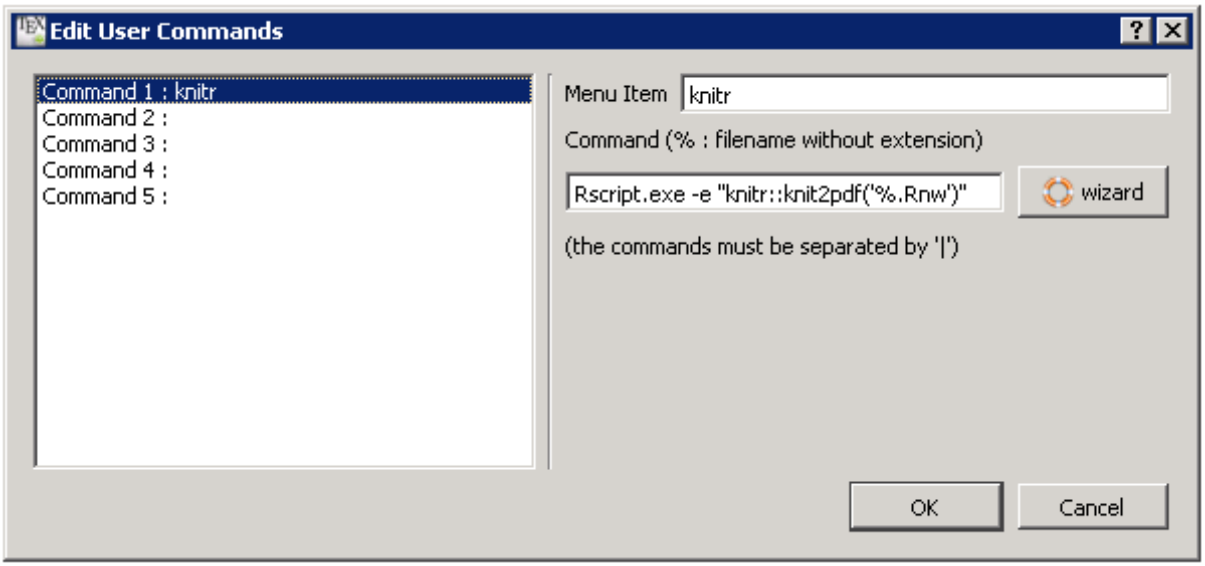



図 4.1　Texmaker でユーザーコマンドを定義する

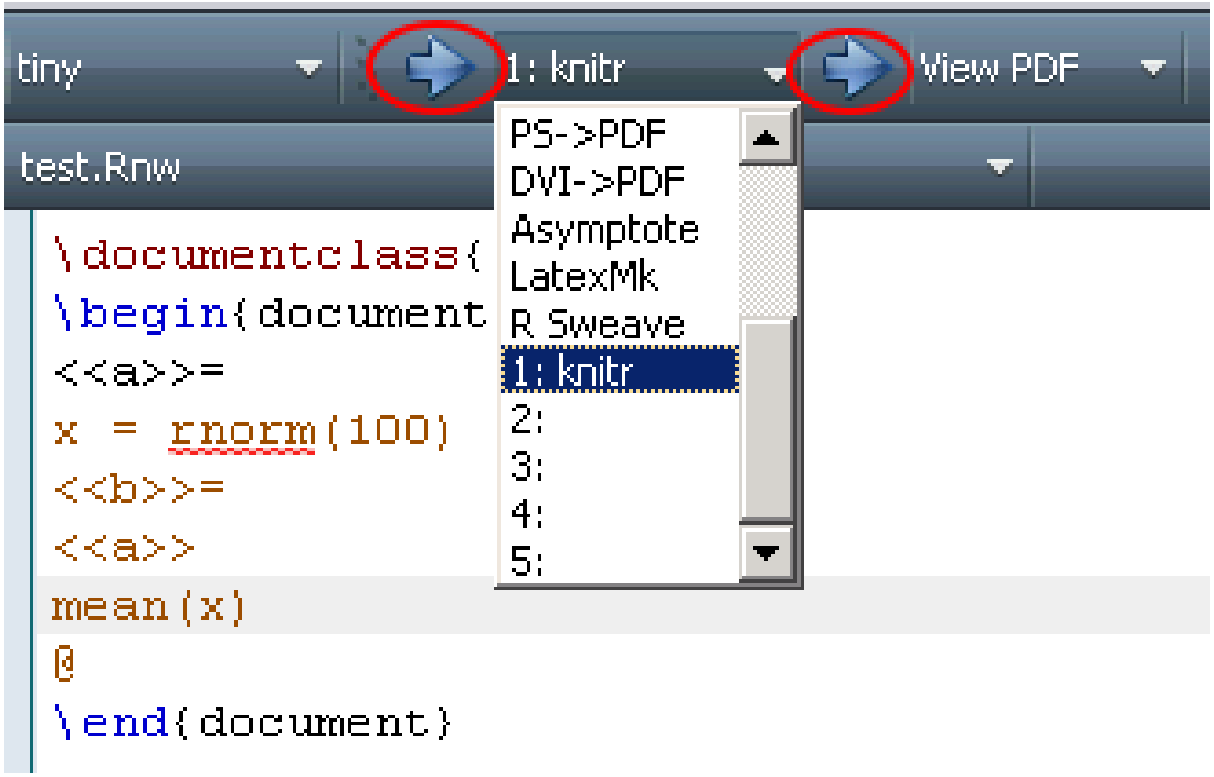



図 4.2　Texmaker で knitr コマンドでコンパイルする

文書をコンパイルするには左向きの矢印をクリックし, PDF を表示するのに右矢印をクリックします. もちろん上記の設定は Windows のものですが, 他のシステムでも同じ要領です. `Rscript.exe` を `Rscript` に置き換えてください (実際は Windows 環境でも `Rscript` が使えます).

## TeXStudio

一例として, 基本的には Texmaker と同様に設定することができます (Henrik Nyhus と Paul J. Hurtado に感謝)

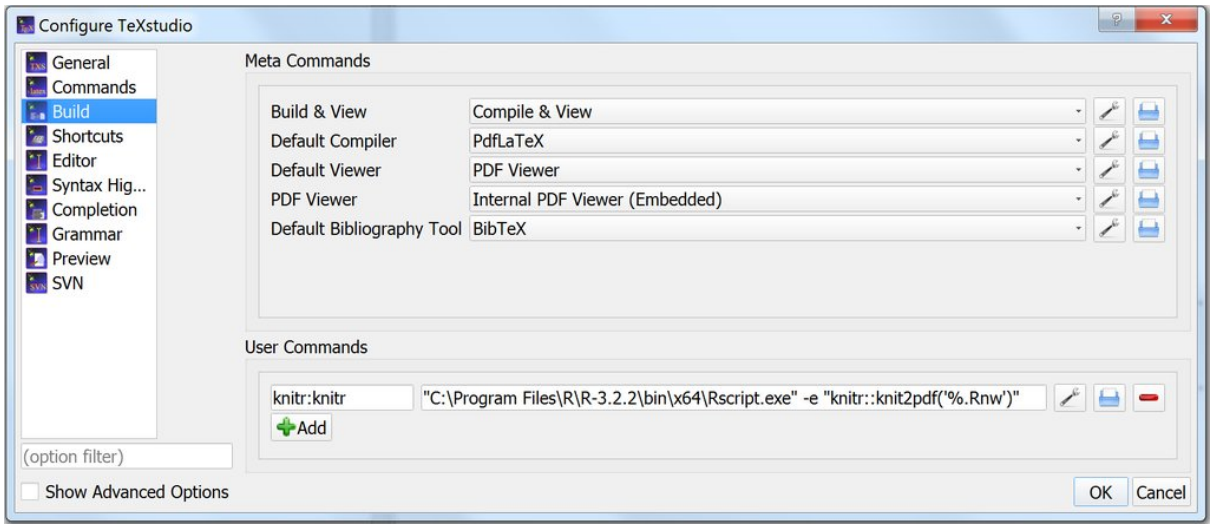



図 4.3　TeXStudio での knitr

上級者向けオプション (左下) を解放すれば, 下部に `Commands ($PATH)` が現れるでしょう. ここに `R` フォルダのパスを入力できます (例: 引用符なしで `C:\Program Files\R\R-3.3.2\bin\x64`). それからユーザーコマンド (そして他のコマンドからも呼び出せる) に `Rscript.exe -e "knitr::knit2pdf('%.Rnw')"` とするだけです.

いつでもこのコマンドを実行できます (ホットキーは Alt-Shift-F1) し, サイレントモードでも knitr で PDF を生成できます. F7 キーでいつでも表示されている PDF を更新できます. 代わりの方法として, コマンドの末尾に `| txs:///view-pdf` を追加する方法があります. これはあるコマンドの実行後に実行する別のコマンドを分けるパイプ記号です. よって基本的に F7 を押すだけで事足りるようになります.

しかし, もし BibLaTeX のような文献引用パッケージを使うならば, まだかなり非効率です. コンパイルの前に bib ファイルの処理を手動で行う必要があり, よって最低でも 2 回手動でコンパイルする必要があり (そしてあなたは何回必要か正確に分からないでしょう). その間に不要なビューアの呼び出しが発生します. TeXstudio は文献処理ツール起動したりコンパイルしたり, それらを繰り返すのに優れていますが, ビルドと閲覧 (F5) を何度も押す度にこれが必要になり, そこでデフォルトのコンパイラを `Rscript.exe -e "knitr::knit2pdf('%.Rnw')"` で置き換えることで **knitr** を使用し続けられるわけです (`R` フォルダを `$PATH` に設定している前提です). その隣の “Repeat contained compilation commands” が押されたままであることを確認してください.

もちろん, これならば少なくとも F5 (または F6) を押すだけで TeXstudio デフォルトの標準の LaTeX コンパイラの代わりに `knitr` が実行されます. この方法のおそらく唯一の欠点は, ほとんどの人にとって LaTeX コードのデバッグがいっそう困難になることです. ほとんどのエラーが「`texify.exe` "had status 1"」という曖昧な警告に置き換わってしまうことです (ユーザーコマンド経由で **knitr** を実行するのではこれは改善できません). そのような状況でもログファイルも時には有用ですが, ログファイルは`.Rnw` ファイルではなく **knitr** の生成した `.tex` を参照するので, このファイルを開いてどこがおかしいのかを見つけようとする必要があります.

しかし物事の全体で見れば, TeXstudio ならば 他のあらゆる LaTeX 書き込み機能の恩恵を得つつ `knitr` を使うことができます. その上, 不便な回避策が必要な RStudio と比較すれば, `BibLaTeX-Chicago` のような `Biber` ベースの文献引用パッケージの使用もはるかに簡単になります.

## WinEdt

WinEdt の R-Sweave モードは現在 **knitr** をサポートしています. 自分自身で WinEdt の設定をしたいなら, よくお読みください.

以下の手引書は Phil Chalmers によるものです. 私は全てを確認していませんが, おおよそ良さそうだと思います.

1. `Options -> Execution Modes -> PDFTeXify` へ移動する. それから実行可能な `Rscript.exe` (例: `C:\Program Files\R\R-2.14.2\bin\Rscript.exe`) を探し, それを選択する
2. `Switches` で `-e` を入力し, `Parameters` で`"knitr::knit2pdf('%n%t')"` を入力する.

そして `F9` をタイプすれば, PDF を開くのを含めて全てが一度に実行されます.

Phil に感謝.

## Emacs/ESS

12/9 以降, **knitr** は公式に ESS にサポートされています. Debian/Ubuntu をお使いの場合, 以下のようにしてインストールできます.

```
sudo apt-get install ess
```

ESS で **knitr** を使う方法に関する短い動画があります.

(歴史的背景に興味がある人向け) Simon Potter と lucialam の両氏が Emacs/**knitr** についてブログに書いています.

## Gedit

gedit で外部ツールとして定義することができます. 以下は David Allen による方法です. 感謝.

```
Rscript -e "library(knitr); knit('$GEDIT_CURRENT_DOCUMENT_NAME')"
```

## Sublime

Andrew Heiss による KnitrSublime パッケージは Sublime Text 2 で LaTeX で **knitr** を使用する基本的な機能をサポートしています.

## Vim

Jakson Aquino のおかげで Vim-R-Plugin **knitr** の包括的なサポートをしています.

## TextMate

Applescript for TextMate 2 で使用するアプローチとして #252 や Chris Fonnesbeck のリポジトリ knitr.tmbundle を参照してください.

## TeXShop

TeXShop で **knitr** を動作させる設定は簡単です. TeXShop の `Engines` ディレクトリ (大抵の場合は `~/Library/TeXShop/Engines/`) の `Knitr.engine` ファイルに以下を書き込むだけです.

```
#!/bin/bash
export PATH=$PATH:/usr/texbin:/usr/local/bin
Rscript -e "library(knitr); knit('$1')"
latexmk -pdf "${1%.*}"
```

Cameron Bracken と Fabian Greimel の厚情に感謝.

## TeXworks

追加ツールの設定に関して TeXworks も Texmaker と似ています．以下は Ubuntu でのスクリーショットです．StackExchange 回答してくれた Speravir に感謝．(Windows/Mac OS も `Rscript` が `PATH` にある限り同様にできるはずです)．

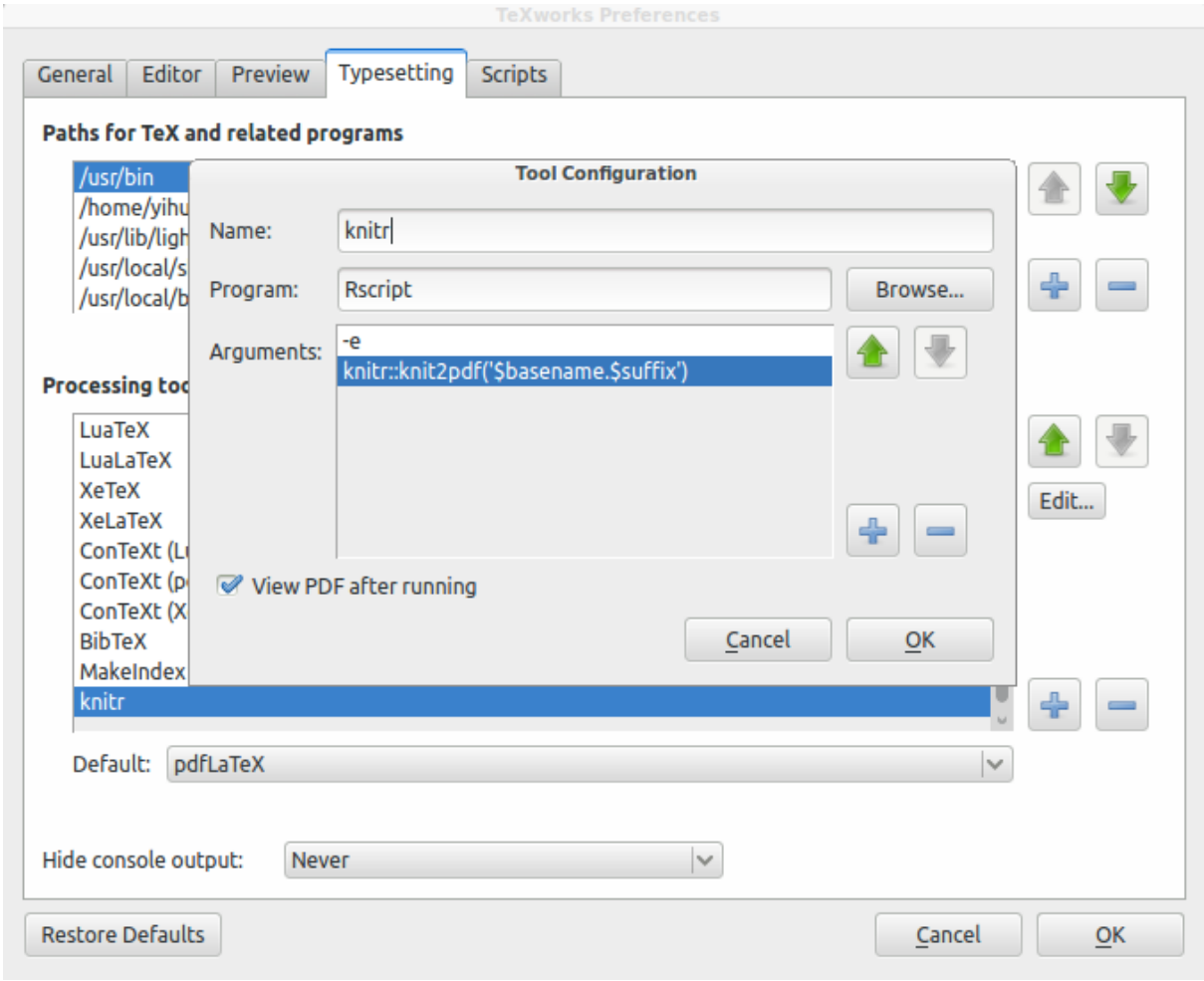


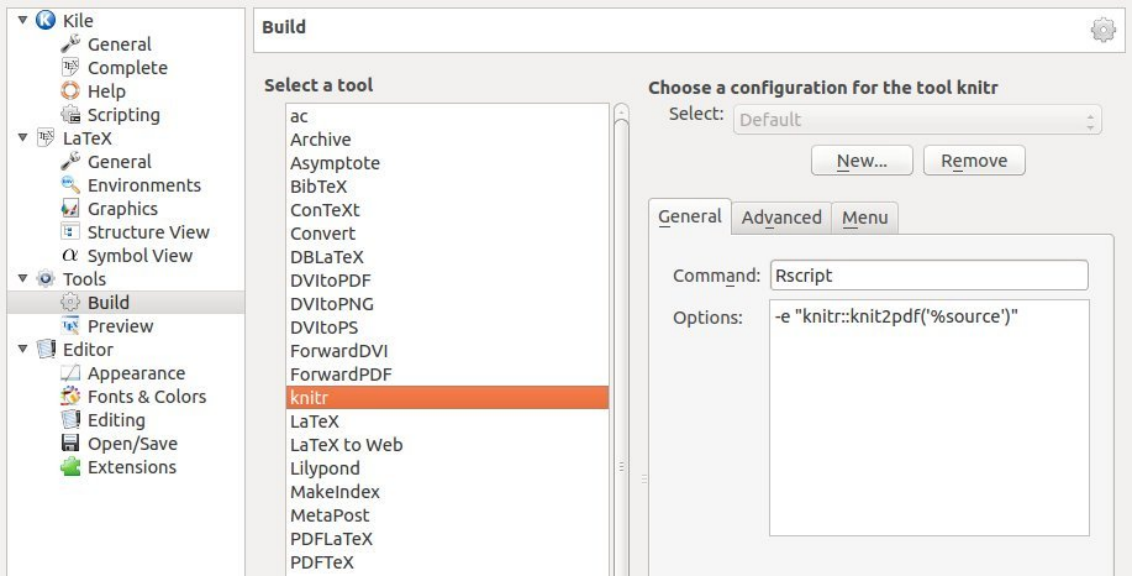

図 4.4　TeXworks で **knitr** を使う

## Kile

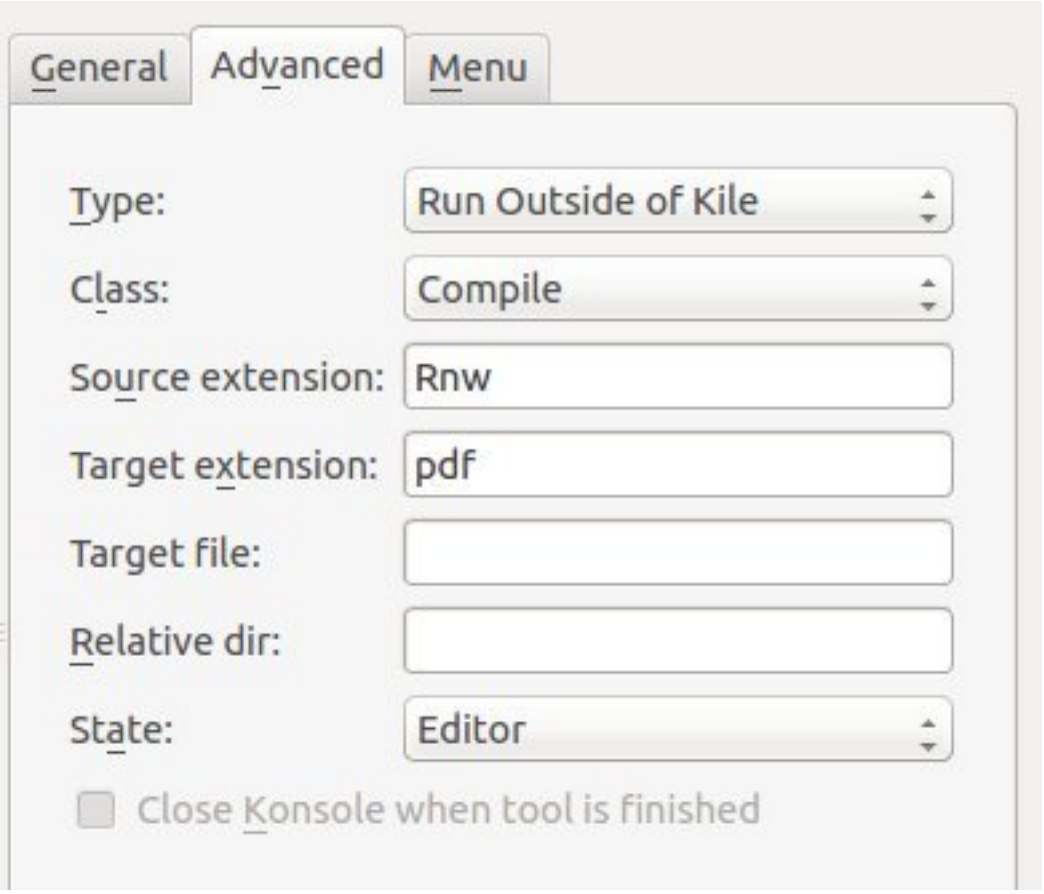
tirip01 が以下のような方法を指摘しています．

1. `Build` タブを開き，`New..` を選んで，`knitr` とタイプし `Finish` を押します．`General` タブから `Comannd` タブで `Rscript` と入力し，その下の `Options` フィールドで`-e "knitr::knit2pdf('%source')"` とタイプし

ます.




2. Advanced タブへ移動し, Rnw を Source extension に設定し, pdf を Target extension に設定します.




3. メニューから Compile を選択します.

Dr Marek Gągolewski も Configure Kile for knitr under GNU/Linux というブログ投稿でもっと複雑なア

プローチを解説しています.

## Tinn-R

Tinn-R は **knitr** v2.3.7.3 以降からサポートを開始しました.

## `framed` パッケージ

**knitr** における LaTeX のデフォルトスタイル

オリジナルのページ: https://yihui.org/knitr/demo/framed/

オリジナルの更新日: 2012/2/29

---

このページは `knitr` 単体の場合を解説しています. R Markdown の場合, `framed` が関わるのは基本的にコードチャンクの表示スタイルのみです.

デフォルトでは **knitr** はタイプセットに framed という LaTeX パッケージを使用しています. 代表的な特徴として, 薄灰色の影がつけられます. このページではいくつかのトリックと既知の問題を紹介します.

よくある質問で挙げたように, 影付きボックスからはみ出ることがあるかもしれません. その際は `options('width')` でより小さい値を設定してください.

## 要素の概要

テキストのはみ出しはさておき, 図もまた影付きの余白を超えるかもしれません. 図の幅が広すぎる場合, LaTeX は `kframe` 環境で問題が合った旨を警告します. `kframe` は **knitr** チャンク出力を包むために使用します. 既知の問題では, issue #154 で PNG を使った場合があります. 確実にページ余白を超えないようにするため, **knitr** は以下のコマンドを LaTeX プリアンブルで使用します.

```
%% maxwidth is the original width if it's less than linewidth
%% otherwise use linewidth (to make sure the graphics do not exceed the margin)
\makeatletter
\def\maxwidth{ %
  \ifdim\Gin@nat@width>\linewidth
    \linewidth
```

```
  \else
    \Gin@nat@width
  \fi
}
\makeatother
```

出力が LaTeX の場合, チャンクオプション `out.width` はデフォルトで`'\\maxwidth'` 設定されます.

## 影付きボックスのパディング

もしデフォルトのレイアウト (パディングがまったくない) が倹約しすぎると感じた場合, この LaTeX コマンドでパディングを 5mm に設定します.

```
\setlength\fboxsep{5mm}
```

**framed** パッケージのデフォルトのスタイルが気に入らない場合, listings や自分自身で定義した出力フックに切り替えられます.

## framed と互換性のない環境

二段組の文書内での `figure*` とは相性がよくありません. この状況に対処するアプローチに 1 つとして, knitr-twocolumn.pdf を参照してください.

Tufte handout/book クラスを使う場合, `fullwidth` 環境と **framed** パッケージは併用できません. 可能性のある解決策として, issue #222 の議論を参照してください.

**lineno** パッケージとも併用できません. Michael's の投稿を参照してください.

### Listings

`listings` と `knitr` の併用

オリジナルのページ: https://yihui.org/knitr/demo/listings/

オリジナルの更新日: 2011/12/10

---

このページは主に R Markdown ではなく Rnw を想定していることに注意してください.

**knitr** では, LaTeX の `listings` パッケージで結果を装飾するためにの出力フックを簡単に定義することができます. 以下のようなスニペットを使うことになるでしょう.

```
## a common hook for messages, warnings and errors
hook_lst_bf <- function(x, options) {
  paste("\\begin{lstlisting}[basicstyle={\\bfseries}]\n", x,
    "\\end{lstlisting}\n",
    sep = ""
  )
}
knit_hooks$set(source = function(x, options) {
  paste("\\begin{lstlisting}[language=R,numbers=left,stepnumber=2]\n", x,
    "\\end{lstlisting}\n",
    sep = ""
  )
}, output = function(x, options) {
  paste("\\begin{lstlisting}[basicstyle={\\ttfamily}]\n", x,
    "\\end{lstlisting}\n",
    sep = ""
  )
}, warning = hook_lst_bf, message = hook_lst_bf, error = hook_lst_bf)

## empty highlight header since it is not useful any more
set_header(highlight = "")
```

見て分かるように, **knitr** は全てをユーザーに公開してます. 必要なのはこれらの R コードの部品と出力を適切な環境で包むことです.

**ちょっと待ってください**, 上記のコードをコピペしないでください. これはすでに少々の機能を追加した上で `render_listings()` 関数として **knitr** に組み込まれています. これが使用例になります.

- **knitr** で `listings` をつかう
  - Rnw ソース: knitr-listings.Rnw
  - LyX ソース: knitr-listings.lyx
  - PDF 出力: knitr-listings.pdf

PDF の出力のスクリーショットを 1 つお見せします:

```
set.seed(1121)  # for reproducibility
options(width = 85)
x ← rnorm(20)
x
```

```
 [1]  0.14496  0.43832  0.15319  1.08494  1.99954 -0.81188  0.16027  0.58589  0.36009
[10] -0.02531  0.15088  0.11008  1.35968 -0.32699 -0.71638  1.80977  0.50840 -0.52746
[19]  0.13272 -0.15594
```

```
mean(x)
```

```
[1] 0.3217
```

```
sqrt(-1)  # this will give you a warning message
```

```
Warning message: NaNs produced
```

```
[1] NaN
```

Another chunk:

```
hook_rgl ← function(before, options, envir) {
    ## after a chunk has been evaluated
    if (before || !require("rgl") || rgl.cur() == 0)
        return()  # no active device
    name ← paste(valid_prefix(options$fig.path), options$label,
        sep = "")
    par3d(windowRect = 100 + options$dpi * c(0, 0, options$width,
        options$height))
    Sys.sleep(0.05)  # need time to respond to window size change
```

図 4.5　knitr での listings の使用

LaTeX のスタイルファイル `Sweavel.sty` を提供してくれた Frank Harrell に感謝します.

## 4.1　さらなる listings オプション.

**listings** パッケージには膨大な数の使用可能なオプションがあるので, その全性能を引き出すにはマニュアルを読み込んでください. 以下はエラーメッセージで開業する方法の例を提示しています. この Rnw ソースをダウンロードすることができます. ポイントは `breaklines=true` オプションです.

## チャンク出力の制御

チャンクの 6 種類の出力とインライン出力を操作する

オリジナルのページ:

オリジナルの更新日: 2012-01-25

---

```
# the error messages are written in the Serror
#   environment defined in preamble
knit_hooks$set(error = function(x, options) {
    sprintf("\\begin{Serror}\n%s\\end{Serror}\n", x)
})
```

```
stop("I am a long long long long long long long long long long long
```

```
Error: I am a long long long long long long long long long long
    long long long long long long string
```

```
sessionInfo()
```

```
R version 2.14.2 (2012-02-29)
Platform: x86_64-pc-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  base

other attached packages:
[1] knitr_0.4.1

loaded via a namespace (and not attached):
 [1] codetools_0.2-8 digest_0.5.2    evaluate_0.4.1  formatR_0.3-4
 [5] highlight_0.3.1 methods_2.14.2  parser_0.0-14   plyr_1.7.1
 [9] Rcpp_0.9.10     stringr_0.6     tools_2.14.2
```

図 4.6　listings の出力の改行

main manual の導入の通り, **knitr** は R コードチャンクの評価のために **evaluate** パッケージを使用し, その出力は 6 種類あります. **ソースコード**, **通常テキスト出力**, **メッセージ文**, **警告文**, **エラー文**, そして**グラフ**です. 以降ではこれらをうまく制御する方法を要約します.

1. **ソースコード**: チャンクオプション `echo` を使います, 例: `echo=FALSE` は R コードを隠します
2. **通常テキスト出力**: `results` オプションを使います (`markup` はテキストをマークアップし, `asis` はテキストをそのまま出力し, `hide` は結果を隠します)
3. **メッセージ文**: `message` オプションを使います (`FALSE` は出力されるメッセージ文を隠します)
4. **警告文**: `warning` オプションを使います (`FALSE` は出力される警告文を隠します)

5. **エラー文**: error オプションを使います (FALSE はエラー発生時点で R の処理を停止し, TRUE は出力にエラー文を表示します)
6. **グラフ**: fig.keep オプションを使います. (none は全てのグラフを破棄し, all は全ての低水準プロットに分けて保存し, high は高水準作図として保存します)

これらのオプションは互いに独立しており, 他のタイプの出力への影響を気にすることなく自由に切り替えることができます.

**knitr** の全てのオプションは R の評価式から値を取ることができます. これは main manual で条件評価 (conditional evaluation) の機能として紹介したものです. 端的に言うなら eval=dothis は実際の値 value はグローバル環境の dothis とい変数から取られた値になるということです. この変数を操作することで, 1 つ 1 つのチャンクの評価をオン・オフ切り替えることができます.

## echo オプションの発展的な使い方

チャンクオプション echo は TRUE/FALSE だけでなく, 数値のベクトルを取ることで出力文を選択することができます. このベクトルのインデックスはコードチャンクを完全な R 評価式単位でインデックスします. たとえば echo=1 は出力内の最初のソースコード出力のみを含めることを意味します. 以下はこの例の完全版です.

```
<<hide-par, echo=3:4>>=
## 表示させたくない「醜い」コード
par(mar = c(4, 4, 0.1, 0.1), cex.lab = 0.95, cex.axis = 0.9,
    mgp = c(2, 0.7, 0), tcl = -0.3)
plot(mtcars[, 1:2])
plot(mtcars[, 4:5])
@
```

評価式 par() はこのコードチャンクに必須ではなく, 読み手の集中力を逸らすことすらあります. そこで出力からこれを隠したいとなるでしょう. このケースでは, 最初の評価式 (コメントのことです) も見せたくありません. echo=3:4 ならば 3, 4 番めの評価式が出力に含まれることを意味します. 評価式のインデックスが行番号と同じである必要はありません. 代わりに 1,2 番めの評価式を削除するという意味で echo=-(1:2) とすることもできます.

ソースコードを部分部分に分けて選択すると, 読み手は混乱するかもしれません. (相対的に) 完全な部分集合を選ぶために, ほとんどの場合は a:b または -(a:b) を使うべきでしょう. しかし, 誰もあなたに禁じることはできません.

```
% 3, 5 番めの評価式を選択
<<hide-par, echo=c(3, 5)>>=
## 表示させたくない「醜い」コード
par(mar = c(4, 4, 0.1, 0.1), cex.lab = 0.95, cex.axis = 0.9,
    mgp = c(2, 0.7, 0), tcl = -0.3)
plot(mtcars[, 1:2])
par(mar = c(4, 4, 1, 0.5)) # reset margins
plot(mtcars[, 4:5])
@
```

## インライン出力

チャンクの出力とは別の出力タイプがあります．インライン R コードの出力です (例: `\Sexpr{t.test(x)$p.value}`)．数値の出力は特別な扱われ方をします．非常に大きいか小さい数値は指数表記で書き出されます．指数表記になるしきい値は R オプションの `scipen` (詳細は`?options` を参照してください) によります．基本的に $10^4$ より大きいか，$10^{-4}$ より小さい場合に指数表記になります (負の数でも絶対値が同様に評価されます)．出力フォーマット (LaTeX とか HTML とか) に応じて，**knitr** は`$3.14 \times 10^5$` や `3.14 &times; 10<sup>5</sup>` というように適切なコードで出力します．

もう 1 つの R オプション `digits` は，どの桁で丸めるべきかを制御します．デフォルトの `options(scipen = 0, digits = 4)` が気に入らないならば，こんなふうに最初のチャンクで変更できます．

```
## 10^5 以上ならば指数表記され，2 桁で丸める
options(scipen = 1, digits = 2)
```

R Markdown での例:

```
インラインコードでは「`r 1+1`」と表示される
```

インラインコードでは「2」と表示される

上記の例では他の地の文とともに 2 と表示されます．

Rnw の例 (LaTeX):

```
Inline code looks like this \Sexpr{1+1}
```

R HTML の例:

```
<p>Inline code looks like this <!--rinline 1+1 --></p>
```

R HTML 文書では, デフォルトでは結果の文字列は<code></code> で囲まれます. 出力から<code></code> タグをなくしたいなら, R コードを `I()` で囲むだけです. 例:

```
<p>Inline code looks like this <!--rinline I(1+1) --></p>
```

さらに別の用例をスタックオーバーフローの投稿で見ることができます.

### Long lines of text output

通常, R はテキスト出力時に `width` オプション (`options(width = ??)` で設定されたもの) を尊重します. たとえば `rnorm(100)` として見てください. `width` のデフォルト値は `75` に設定されていますが, あなたが LaTeX を使っているならこれより小さい値を望むかもしれません. いくつかのケースでは, 小さな値を設定していたとしても実際の出力の幅が広すぎることがあるかもしれません. それは大抵の場合, R がこのオプションを尊重してないためです. 私はこの問題にあまりできることはありませんが, **knitr** 側でハックする方法がいくつかあります. その例の 1 つは issue #421 で見られます[*5].

## 1 メッセージに 1 コメントを

R でメッセージ文を書き出すのに `cat()` を使うのが好きな人がたくさんいますが, これは大変よくない使い方です. このような形にメッセージ文は非表示にするのが大変だからです. 本当に**メッセージ** を提示したい場合は, `message()` 関数を使うべきです. 正規のメッセージは `suppressMessages()` で表示を抑制したり, **knitr** で補足したりすることができ便利です. パッケージ開発者の中にはこの問題に注意を払ってない人がいます. そのようなパッケージを読み込んだりパッケージの関数を使ったりすると, 表示を抑制できない偽物のメッセージ文を見かけます. パッケージのスタートアップメッセージはじっさい必要ですが, それは `packageStartupMessage()` で表示すべきです. **knitr** の `message=FALSE` でメッセージを消せないなら, それはパッケージ開発者に変更を要求するときです.

同様に, 警告を発したいならまさに `warning()` を使うべきです.

[*5] 訳注: `citation()` の出力が `width` に対応していない場合の問題です

というわけで, あなたの `cat` の振る舞いに気をつけてください!

## その他

チャンクで `echo=FALSE, results='hide'` を使用した時, 出力に余分な空白行があるかもしれません. 空白行がいらないのなら, issue #231 で解決できるかもしれません.

## チャンク参照 / マクロ

チャンクの再利用方法

オリジナルのページ: https://yihui.org/knitr/demo/reference/

オリジナルの更新日 2012/01/14

---

**訳注**: このページは全編を通して Sweave の構文で書かれていますが, R Markdown のチャンクでも同様のことが可能です.

Sweave には`<<chunk-label>>` (`<<>>=` と違い`=` がない点に注意) という構文でチャンクを再利用するためにチャンクを参照する機能があります. たとえば

```
<<chunk1>>=
1 + 1
@

<<chunk2>>=
<<chunk1>>
@
```

`chunk2` では `chunk1` のコードが挿入されます. この機能は **knitr** でも有効ですが **knitr** はさらに任意の (制限なし) 階層での再帰的なチャンク参照をサポートしています (Sweave は 1 段階までしかサポートしていませんでした). つまりあるチャンクはさらに別のチャンクを参照しているチャンクを参照できるということです.

この`<<chunk-label>>` 構文と同様のものは markdown の構文でも機能します. それ以前の, 他のチャンクを含んでいる, 名前を付けた markdown チャンクを再利用することができます. あなたの使っている R Markdown エディタが「予期しないトークンです」と構文に警告していたとしてもです.

**訳注**: R Markdown でもチャンク中に`<<`, `>>` で囲んでラベルを書くことでチャンク参照ができます.

**knitr** でチャンクを再利用するアプローチは他にもあります.

1. 再利用したいチャンクと同じ名前のラベルを使う
2. そのチャンクを参照する `ref.label` オプションを使う

## 同じラベルを使用する

1 番目のアプローチの例です

```
<<chunk1, echo=TRUE, results='hide'>>=
1 + 1
@

<<chunk1, echo=FALSE, results='markup'>>=
@
```

2 番目のチャンクは空なので, **knitr** は同じ名前を持ち, 空でないチャンクを探し, そのチャンクのコードを使用します. ポイントは他のチャンクを使用するためにはチャンクの中身を空にしておくことです. 問題は, この 2 つのチャンクの MD5 ハッシュ値が異なるため, 両方のキャッシュを残すことができないというものです. **knitr** はラベル 1 つにつき 1 まとまりのキャッシュしか取ることができません.

## チャンクオプション `ref.label`

2 番目のアプローチの例です.

```
<<chunk1, echo=TRUE, results='hide'>>=
1 + 1
@

<<chunk2, ref.label='chunk1', echo=FALSE, results='markup'>>=
@
```

2 番目のチャンクはラベルが異なるので, キャッシュが取れない問題はなくなりました. 明らかに, 第 2 のアプローチはより汎用的な解決法です.

この機能は出力文書において R のコードと出力を分離することを可能とします. 簡単な例として, 論文を書く際に本文中に R コードを表示しないために `echo=FALSE` を設定することができます. そして補遺 (Appendix)

のセクションでこの R コードを掲載するため, チャンク参照を使います (`eval=FALSE`, `ref.label=...` オプションを使います).

## グラフィックス

**knitr** におけるグラフィックスの力について

オリジナルの記事: https://yihui.org/knitr/demos/graphics

オリジナルの更新日: 2011/12/9

---

このページは主に R Markdown ではなく Rnw を想定していることに注意してください.

## グラフィックスマニュアル

グラフィックスマニュアルでは **knitr** のグラフィックスに彩りを加えるものを紹介します.

- グラフィックスマニュアルのソースと出力は以下です
  - Rnw ソース: knitr-graphics.Rnw
  - LyX ソース: knitr-graphics.lyx
  - PDF 出力: knitr-graphics.pdf

刊行にあたって R グラフィックスの改善の余地が大いにあることに気づくかもしれません. R があなたにもたらすものを鵜呑みにしないでください. あなたのグラフを美しくプロフェッショナルにすることを考える時間です.

マニュアルからいくつかスクリーショットを提示します.

`tufte-handout` クラスの作者に感謝します. 上記の例はこれを利用しています. そして **tikzDevice** パッケージは文書クラスと一貫したフォントスタイルのグラフをもたらしてくれます (セリフフォントを使用しています).

## カスタムグラフィックデバイスについての補足

チャンクオプション の `dev` は 3 つの引数をとる R 関数として定義されたカスタムグラフィックデバイスに対応します. これは `pointsize` 10 を使用した PDF デバイスの例です[*6].

[*6] 訳注: R Markdown の場合, 日本語表示は後述するように `cairo_pdf` のほうが簡単です.

*The tikz Device*

The main advantage of using tikz graphics is the consistency of styles between texts in plots and those in the main document. Since we can use native LaTeX commands in plots, the styles of texts in plots can be very sophisticated.

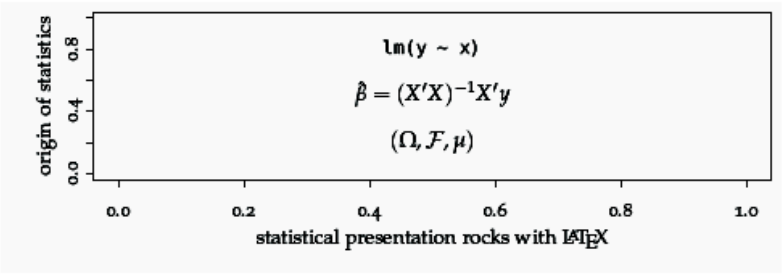



Figure 8: A plot created by **tikzDevice** with math expressions and typewriter fonts. Note the font style consistency – we write the same expressions in LaTeX here: $\hat{\beta} = (X'X)^{-1}X'y$ and $(\Omega, \mathcal{F}, \mu)$; also `lm(y ~ x)`.

図 4.7　**knitr** 上の tikz グラフィックス

```
suppressMessages(library(ggplot2))
pie <- ggplot(diamonds, aes(x = factor(1), fill = cut)) +
    xlab("cut") + geom_bar(width = 1)
pie + coord_polar(theta = "y")  # a pie chart
pie + coord_polar()  # the bullseye chart
```

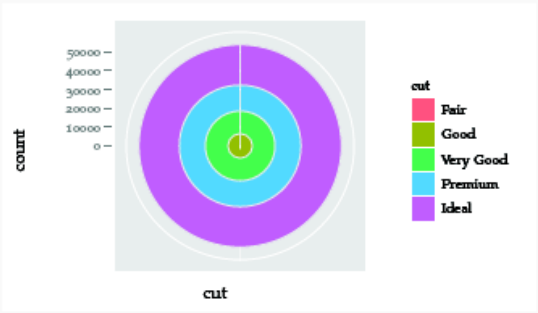



Figure 6: Two plots were produced in this chunk, but only the last one is kept. This can be useful when we experiment with many plots, but only want the last result. (Adapted from the **ggplot2** website)

図 4.8　(ref:demo-graphics-ggplot2

```
01 my_pdf <- function(file, width, height) {
02   pdf(file, width = width, height = height, pointsize = 10)
03 }
```

これでチャンクオプションでこのデバイスを使えるようになりますが, 1 つ重要なことを覚えておいてください. **knitr** はグラフファイルに対して適切なファイル拡張子を推測できないため, `fig.ext` オプションも同時に指定する必要があります. 最終的に, このカスタムデバイスはこのように使われます.

```
```{r dev='my_pdf', fig.ext='pdf'}
plot(rnorm(100))
```
```

文書全体でこのデバイスを使用したい場合は, もちろん `\SweaveOpts{}` を使ってグローバルに設定することもできます.

## デバイスに追加の引数を与える

`dev.args` オプションを通してグラフィカルデバイスをよく制御できます. `pointsize = 10` とハードコーディングする代わりに, チャンクに `dev.args = list(pointsize = 10)` を与えることができます. これが例です.

```
```{r dev='pdf', dev.args=list(pointsize=10)}
plot(rnorm(100))
```
```

`dev.args` はリストなので, デバイスの引数として可能なものを取るべきです. たとえば `pdf()` には `dev.args=list(pointsize=11, family='serif')`. `dev.args` の全ての要素はチャンクのグラフィカすデバイスに与えられます.

## R グラフィックスにハイパーリンクを付ける

**tikzDevice** パッケージのおかげで, R グラフィックスではほとんどの LaTeX コマンドを使うことができます. ハイパーリンクを付ける例を示します: links.Rnw (Jonathan Kennel に感謝).

特記事項として, あなたは `\usepackage{hyperref}` を **tikzDevice** パッケージのメトリックのリストに与える必要があります. そうでなければ `\hyperlink` や `\hypertarget` コマンドは認識されません.

## マルチバイト文字のエンコーディング

あなたのグラフにマルチバイト文字が含まれている場合, `pdf()` デバイスの `encoding` オプションを指定する必要があります. issue #172 を参照してください. 可能なエンコーディングのリストは以下で確認できます.

```
01 list.files(system.file("enc", package = "grDevices"))
```

```
##  [1] "AdobeStd.enc"  "AdobeSym.enc"  "CP1250.enc"    "CP1251.enc"
##  [5] "CP1253.enc"    "CP1257.enc"    "Cyrillic.enc"  "Greek.enc"
##  [9] "ISOLatin1.enc" "ISOLatin2.enc" "ISOLatin7.enc" "ISOLatin9.enc"
## [13] "KOI8-R.enc"    "KOI8-U.enc"    "MacRoman.enc"  "PDFDoc.enc"
## [17] "TeXtext.enc"   "WinAnsi.enc"
```

```
## 例: pdf.options(encoding = 'CP1250')
```

このような警告メッセージを目にした場合, エンコーディングの設定が必要かもしれません.

```
Warning: conversion failure on '<var>' in 'mbcsToSbcs': dot substituted for <var>`.
```

別の手段として, pdf の代わりに cairo_pdf を使うというものがあります. (issue #436 を参照してください)[*7][*8]:

```
options(device = function(file, width = 7, height = 7, ...) {
  cairo_pdf(tempfile(), width = width, height = height, ...)
})
```

もし Windows 環境でこれが失敗するのなら, issue #527 を確認してください.

## 装飾フォント

pdf() のドキュメントによれば, useDingbats 引数は小さな円を含む PDF のファイルサイズを減らしてくれる可能性があります. RStudio 上で **knitr** を使っている場合, このオプションはデフォルトで無効になっています. 巨大な散布図を含む場合, ソース文書に pdf.options(useDingbats = TRUE) と書くことで有効になるでしょう. 詳細な議論は issue #311 を参照してください.

## アニメーション

チャンクオプション fig.show='animate' が設定されコードチャンクで複数のグラフが生成されている場合, 全てのグラフが統合されアニメーションになります. LaTeX の出力では, LaTeX パッケージの **animate** が使われ, HTML/Markdown の出力に対しては, デフォルトでは FFmpeg が使われ WebM 動画が作られま

[*7] 訳注: 投稿したい論文雑誌などから特別な要件がない限り, 現在は cairo_pdf のほうが簡単だと思います.
[*8] 訳注: R Markdown であれば, dev = 'cairo_pdf' で十分です.

す．FFmpeg をインストールする際に **libvpx** のサポートを有効にする必要があることに注意してください．Linux および Windows ユーザーは FFmpeg ウェブサイトのダウンロードリンクを確認してください (バイナリ版では **libvpx** は有効になっています)．OSX ユーザーは，Homebrew 経由で FFmpeg をインストールできます．

```
brew install ffmpeg --with-libvpx
```

## マニュアル

パッケージマニュアルについて

オリジナルのページ https://yihui.org/knitr/demos/manual

オリジナルの更新日: 2011/12/5

訳注: このページは **knitr** ドキュメントのトップページではありません．

パッケージのマニュアルそれ自体は **knitr** のほとんどの機能の良い使用例となりえます．このマニュアルは LyX で書かれ ((knitr-manual.lyx))，Rnw ソース (knitr-manual.Rnw) にエクスポートして PDF ファイル knitr-manual.pdf を生成できます．

マニュアルのコンパイルには 3 つのパッケージが追加で必要です．**rgl**, **animation**, **tikzDevice** です．

LyX で **knitr** を使う手順は LyX page を参照してください．

## キャッシュ

キャッシュ機能の使用例について

オリジナルのページ: https://yihui.org/knitr/demo/cache/

オリジナルの更新日: 2011-12-04

---

チャンクオプションの `cache=TRUE` でキャッシュを有効にでき，`cache.path` でキャッシュ用ディレクトリを指定できます．1 章『オプション』を参照してください．

## キャッシュの例

キャッシュ機能は私の文書の多くで広く使われています. たとえば **knitr** の メインマニュアル やグラフィックス です. ここでは, もう少しいくつかの例を挙げます.

- 基本的な例
  - 巨大なデータをキャッシュする 056-huge-plot.Rmd (出力
  - Rtex の構文を使った例: knitr-latex.Rtex
- キャッシュの依存関係の自動的な構成
  - Rnw ソース: knitr-dependson.Rnw
  - チャンクオプション `autodep=TRUE` と関数 `dep_auto()` によって, **knitr** チャンク間の依存関係を解決することが可能となり, `dependson` オプションを指定するための手作業をいくらか省くことができます.

## 重要な補足事項

キャッシュがいつ再構成されるかと, キャッシュされないチャンクについてよく理解するため, メインマニュアル (英語版 PDF) のキャッシュに関するセクションを, とても注意深く読まなければなりません.

キャッシュに影響する要素を今一度繰り返します (いかなる変更も古いキャッシュを無効にします):

1. `include` を除く全てのチャンクオプション, たとえば `tidy=TRUE` を `FALSE` に変えるだけでも古いキャッシュは破棄されますが, `include` は例外です.
2. チャンク内の R コードのわずかな変更, スペースや空白行の追加・削除であっても, 古いキャッシュを削除することにつながります

極めて重要な注意事項として, 副次的な作用をもつチャンクはキャッシュ**すべきでない**ということを挙げます. **knitr** は `print()` による副次作用を維持しようとしますが, さらなる別の副次作用は保存されません. ここでチャンクをキャッシュすべきでないケースをいくつか挙げます.

1. `options('width')`, `pdf.options()`, または `opts_chunk$set()`, `opts_knit$set()`, `knit_hooks$set()` のような他のなんらかの **knitr** のオプションを設定する時
2. キャッシュされたチャンクでの `library()` によるパッケージの読み込みと読み込まれたパッケージはキャッシュされないチャンクによっても扱われます. (キャッシュされたチャンクでパッケージを読み込み, 使うのは全く OK です. **knitr** はキャッシュされたチャンクでパッケージのリストを保存するためです. しかし, キャッシュされないチャンクが, それまでのキャッシュされたチャンクでどのパッケージをロードしたかを知ることはできません.)

さもなければ次回からはチャンクはスキップされ, そこでの設定はすべて無視されます. このようなチャンクでは明示的に cache=FALSE を使わなければなりません.

source() と setGeneric() は, コードがローカル環境で評価された場合でもグローバル環境にオブジェクトを作成するという副次作用を持ちます. **knitr** 0.4 より前ではこれらのグローバルオブジェクトをキャッシュすることはできませんでした (たとえば issue #138 を見てください). しかし ver. 0.4 以降は **knitr** が globalenv() で作成されたオブジェクトを確認し, 他のオブジェクト同様に保存するようになったため, キャッシュできるようになりました.

キャッシュされたチャンクで使われるパッケージのリストは保存されますが, パッケージ名をキャッシュする方法としては完璧ではありません. かりにパッケージを読み込み, あとで除外したとしても, **knitr** はそれを知ることはできません (新たに読み込まれたパッケージしか捕捉できません). Issue #382 で解説されているように, キャッシュディレクトリにある__packages ファイルを手動で編集しなければなりません.

## キャッシュにはまだ何かありますか?

以上のオブジェクトがキャッシュに影響するのはいかにも納得できることですが, 再現可能性のある研究ではキャッシュは他の変更によって無効化されるという点でよりいっそう厳格になることがあります. 典型例の 1 つはソフトウェアのバージョンです. 2 つのバージョンが異なる R に, 異なる結果を出させるのは不可能ではありません. この場合, 我々はこう設定するかもしれません.

```
knitr::opts_chunk$set(cache.extra = R.version.string)
knitr::opts_chunk$set(cache.extra = R.version) # あるいはプラットフォームの違いも考慮する
```

これによって, キャッシュされた結果は特定のバージョンの R でのみ適用されます. R をアップグレードし文書を再コンパイルしたとき, 全ての結果は再計算されます.

同様に, キャッシュが特定の環境でのみ保存されるように, このオプションに他の変数も設定したいかもしれません. これは野心的な例です.

```
## キャッシュは特定の R のバージョンとセッションでのみ有効
## キャッシュは最大で 1 ヶ月間保存される (来月は再計算される)
knitr::opts_chunk$set(cache.extra = list(
  R.version, sessionInfo(), format(Sys.Date(), "%Y-%m")
))
```

この cache.extra オプションの別の良い使用例が issue #238 で示されています. この例ではキャッシュはファイルの更新日時と関連付けられています. つまり, データファイルが変更されれば, キャッシュは自動的に

再構成されることになります[*9].

注: キャッシュ条件にさらにオブジェクトを導入する際に, cache.extra 以外の任意のオプション名を使用することができます. たとえば cache.validation も呼び出せます. 全てのチャンクオプションはキャッシュの確認時に考慮されるためです.

## キャッシュディレクトリを入力ファイル名と連動させる

ときとしてデフォルトとは異なる入力ファイルに対して異なるキャッシュディレクトリを使いたいことがあるかもしれません. 解決策の 1 つが issue #234 で提示されています. しかし, 自己完結性を高めるため, この設定はソースドキュメントの内部で行うことを推奨します (opts_chunk$set(cache.path = ...) を使ってください.).

## より細かいキャッシュのとり方

上級者はチャンクオプション cache に対して TRUE か FALSE かだけでなく, 数字で設定するなどしてもっと粒度の細かいキャッシュが欲しいと考えるかもしれません. cache = 0, 1, 2, 3 として, 0 は FALSE, 3 は TRUE と同じで, cache = 1 は計算の結果のみキャッシュから読み込む (evaluate::evaluate() から), よってコードは再評価されませんが, 他の箇所, 出力フックや作成されたグラフの保存といった箇所は評価されます. cache = 2 ならば, 1 とほとんど同じですが, 唯一の違いはグラフファイルがすでに存在する場合, 再保存されない点です. これはグラフが大きい場合, いくらか時間の削減になります. 以前の R セッションで記録されたグラフが別の R セッション, あるいは別のバージョンで安全に再保存されるという保証はないため, cache = 1 よりは cache = 2 を推奨します.

cache = 1, 2 の場合は少数のチャンクオプションだけがキャッシュに影響します. オプション名は knitr:::cache1.opts, knitr:::cache2.opts で確認してください. 基本的に, コード評価に影響しないチャンクオプションが変更されてもキャッシュは無効になりません. たとえば echo を TRUE から FALSE に変えるとか, fig.cap = '**新しいキャプション**' と設定するなどの変更です. しかし, eval を TRUE から FALSE に変えた場合, あるいは cache.path='foo/' を'bar/' に変えた場合, キャッシュは再構成されます.

いくつかの例が, example #101 (出力) で見られます.

この方法では, 計算と文書の出力レンダリングを分離することが可能となり, キャッシュを破棄することなく出力を調整するのに役立ちます. issue #396, #536 を確認してください.

## 乱数生成器 (RNG) の再現可能性

乱数生成器 (RNG) が関係するチャンクの再現性を維持するため, **knitr** は .Random.seed をキャッシュし, 各チャンクの評価直前に復元します. しかし 1 つ問題があります. チャンク A, B がキャッシュされていたとし

[*9] 訳注: ファイルの更新日時を取得するには base R の file.mtime 関数が便利とのことです

ます. この時 A, B の間に新たにチャンク C を挿入したとします (3 つのチャンクは全て内部で RNG を使用しています). RNG が変更されたため B が更新されるはずですが, .Random.seed は副次作用であるため, これに反して実際には B は更新されません. はっきり言うと, B の再現性は偽りのものとなります.

RNG の関係する再現可能性を保証するために, .Random.seed とキャッシュを関連付ける必要があります. これが変更される度にキャッシュも更新されなければなりません. これは cache.extra オプションで **評価されない** R 評価式を参照することで簡単にできます. たとえば以下のように.

```
01 knitr::opts_chunk$set(cache.extra = rand_seed)
```

?rand_seed を確認してください (これは評価されない R 評価式です). この場合は各チャンクは最初に .Random.seed が最後の実行から変更されたかどうかを確認します. .Random.seed が異なるならキャッシュは再構成されます.

## knitr のショーケース

ユーザーたちによる使用例

オリジナルのページ: https://yihui.org/knitr/demo/showcase/

オリジナルの更新日: 2013-03-11

---

訳注: 更新日時からも分かるように, このリストは古く, また膨大であるため個別の翻訳はいたしません. R に限らず, 活発に利用されているオープンソースソフトウェアは頻繁に更新されるため, 非公式の解説はしばしば out-of-date になりうる, ということに注意してください.


以下のリンク集は外部サイトによる knitr の使用事例集です (もしリンクしてほしくない, あるいはしてほしいということであれば私 (Yihui) に気軽に連絡してください)

## ウエブサイト

- RPubs: Easy web publishing from R
- knitr in a knutshell, a short tutorial by Karl Broman
- R learning resources at UCLA by Joshua Wiley et al (dynamically built with **knitr**)
- knitr on ShareLaTeX (an online LaTeX editor)
- Rcpp Gallery: Articles and code examples for the **Rcpp** package

- Slidify: reproducible HTML5 slides made easy
- One Page R Literate Data Science by Graham Williams
- Reproducible graphics with R and ggplot2 by Baptiste Auguié
- ML/stats notes by John Myles White
- A French introduction to R by Julien Barnier (also see CRAN)
- Applications of R in Business Contest (knitr's entry; announcement)

## 書評

Dynamic Documents with R and knitr に対する書評のリストです.

- A book review in the *Journal of Statistical Software* by Amelia McNamara
- A book review in *MAA Reviews* by Peter Rabinovitch
- A book review on TUGboat by Boris Veytsman
- A book review on RPubs by RK
- A book review in *The American Statistician* by Quan Zhang

## `knitr` によるソリューション

- a knitr Howto page in Vanderbilt Biostatistics Wiki
- Plain Text, Papers, Pandoc by Kieran Healy
- R Markdown output formats for Tufte-style handouts by Michael Sachs
- Blogging with Rmarkdown, knitr, and Jekyll by Brendan Rocks
- Blog with Knitr and Jekyll by Jason C Fisher
- Creating HTML5 slides with RStudio, knitr and pandoc by Gaston Sanchez
- A framework to create bootstrap styled HTML reports from knitr Rmarkdown by Jim Hester (a preview)
- Creating a Business Dashboard in R by Bart Smeets
- Carl Boettiger has cool blog posts demonstrating how to publish a post to Wordpress.com with knitr and RWordPress purely in R, with images uploaded to Imgur and Flickr respectively
- knitr + cactus + TwitterBootstrap + Jquery by Barry Rowlingson (includes a smart use of jQuery to add links to R functions)
- Interactive reports in R with knitr and RStudio and Interactive HTML presentation with R, googleVis, knitr, pandoc and slidy by Markus Gesmann
- Stacked bar plots with several descriptive nodes by ADP
- Blogging from R to Wordpress by William K. Morris
- A demo on using `tidy=TRUE` or the listings environment so code chunks can stay inside page margins (link to StackExchange)

- How to Use Knitr with a Rakefile by Lincoln A. Mullen
- Reproducible Research with Word? by Eric P. Green
- knitr, slidify, and Popcorn.js by Ramnath Vaidyanathan (#466)
- Transparent, reproducible blogging with nanoc and knitr by Charles Hogg
- Using knitr and R to make instructor/student handout versions by Luke Miller
- Thesis template to generate LaTeX files using R with knitr by Alexis Sarda

## R パッケージ

- The Github Wiki of the cda package by Baptiste Auguie
- website of ggbio package by Tengfei Yin
- The sampSurf Package by Jeffrey H. Gove
- the RHadoop Wiki by Revolution Analytics
- The ggmcmc package examples by Xavier Fernández-i-Marín
- tabplot: Tableplot, a visualization of large datasets by Martijn Tennekes and Edwin de Jonge (see its PDF vignette)
- the ggplot2 transition guide to version 0.9.0 by Dennis Murphy et al
- a few packages on Bioconductor: ReportingTools and RGalaxy
- the dendextend package by Tal Galili
- An Introduction to CHNOSZ vignette by Jeffrey M. Dick using Tufte style

## 教材

以下は Roger Peng の厚情によってシェアされた Cousera の講座 *Computing for Data Analysis* 向けの **knitr** の講義です.

さらに以下も **knitr** と関連する講座の資料です.

- BMI 826-003 (Tools for Reproducible Research) by Karl Broman, University of Wisconsin-Madison
- Reproducible Research by Roger Peng, Coursera
- Introduction to data science by Mahbubul Majumder, University of Nebraska at Omaha
- BIOS 301 (Introduction to Statistical Computing) by Chris Fonnesbeck, Vanderbilt University
- Math 344 (Probability and Statistics) by Randall Pruim, Calvin College
- Stat 506 (Advanced Regression) by Jim Robison-Cox, Montana State University
- STT 3820 by Alan T. Arnholt, Appalachian State University
- Math 747 (Topics in math biology) by Ben Bolker, McMaster University
- STAT 319 (Applied Statistics in Science) by Yuan Huang, Penn State University (also see the tutorial Create Dynamic R Statistical Reports Using R Markdown)

- some notes on reproducible research by Aedin Culhane, Harvard University
- STAT 622 (Bayesian Data Analysis) by Marina Vannucci, Rice University
- STA613/CBB540 (Statistical methods in computational biology) by Barbara Engelhardt, Duke University
- Stat 590 (Statistical Computing) by Erik Erhardt, University of New Mexico
- STAT 497C (Topics in R Statistical Language) by Eric Nord, Penn State University
- CSSS-Stat 567 (Statistical Analysis of Social Networks) by Peter Hoff, University of Washington
- Math 15 (Statistics) by David Arnold, College of the Redwoods
- STAT 545A Exploratory Data Analysis by Jennifer Bryan, University of British Columbia
- STAT545 Introduction to Computational Statistics, by Vinayak Rao, Purdue University
- Sta 101 Data Analysis and Statistical Inference, by Mine Çetinkaya-Rundel, Duke University
- GEOL 6370 Data Analysis in the Geosciences, by Steven M. Holland, University of Georgia

## ワークショップ・プレゼンテーション

以下は Joshua Wiley の厚情によって作成・シェアされたチュートリアル資料です.

文字に起こされたプレゼンテーション:

- New tools and workflows for data analysis by Jennifer Bryan (video)
- Geospatial Data in R and Beyond by Barry Rowlingson
- Broom Spatial R Class by Frank Davenport (PDF)
- Visualizing Categorical Data by Michael Friendly
- ggplot2 workshop notes by Josef Fruehwald for AVML 2012
- R Introduction for UCL PhDs by Florian Oswald at University College London
- Introduction to R lectures for ECPR Winter School 2013 by Zoltán Fazekas, University of Southern Denmark
- R for the brave by Will Pearse
- Introduction to knitr: The R Markdown (Rmd) format by L. Collado Torres for JHSPH Biostat computing club
- Stop Clicking, Start typing by Matt Frost
- そろそろ RStudio の話でもしてみようと思う by 和田 計也
- Introduction to Data Analysis and Visualization using R by Vinayak Hedge
- Creating publication quality graphics using R by Tim Salabim
- Reproducible Research Using Knitr/R by Keith Hughitt

## 書籍

- The Analysis of Data by Guy Lebanon (written with R Markdown)

- Dynamic Report Generation with R and knitr by Yihui Xie (written with LyX + the `knitr` module)
  - Dynamic Report Generation with R and knitr, Second Edition
- Text Analysis with R for Students of Literature by Matthew L. Jockers
- Data Analysis for the Life Sciences by Rafael Irizarry and Michael Love
- Using R for Introductory Statistics, Second Edition by John Verzani
- Learning R: A Step-by-Step Function Guide to Data Analysis by Richard Cotton (written with AsciiDoc + `knitr`)
- Introductory Fisheries Analysis with R by Derek H. Ogle
- The Statistical Sleuth In R by Nicholas Horton, Kate Aloisio, and Ruobing Zhang (`knitr` + LaTeX)
- Regression Modeling Strategies (`knitr` + LaTeX)
- Latent Variable Modeling using R: A Step-By-Step Guide
- Biolostatistical Design and Analysis using R
- Statistics for Experimental Economists: Elegant Analysis with R by Mark A. Olson
- R 과 Knitr 를 활용한 데이터 연동형 문서만들기
- El arte de programar en R: un lenguaje para la estadística by Julio Sergio Santana and Efraín Mateos Farfán
- PH525x series - Biomedical Data Science by Rafael Irizarry and Michael Love
- Data Science for Fundraising: Build Data-Driven Solutions Using R by Ashutosh Nandeshwar and Rodger Devine

## 論文・レポート

- Our path to better science in less time using open data science tools by Julia S. Stewart Lowndes *et al*, Nature Ecology & Evolution **1**, Article number: 0160 (2017)
- Eglen, SJ; Weeks, M; Jessop, M; Simonotto, J; Jackson, T; Sernagor, E. A data repository and analysis framework for spontaneous neural activity recordings in developing retina. GigaScience 2014, 3:3 http://dx.doi.org/10.1186/2047-217X-3-3
  - plus an interview to the first author
  - Q&A on dynamic documents
- Programming tools: Adventures with R by Sylvia Tippmann, *Nature* **517**, 109–110 (01 January 2015) doi:10.1038/517109a
- Rebooting review, *Nature Biotechnology* **33**, 319 (2015)
- Rule rewrite aims to clean up scientific software, *Nature* **520**, 276–277 (2015)
- A Guide to Reproducible Code (Guides to Better Science), by the British Ecological Society
- 2017 Employer Health Benefits Survey by Kaiser Family Foundation (2017)
  - Referenced by the New York Times article "While Premiums Soar Under Obamacare, Costs of Employer-Based Plans Are Stable"
- Irregularities in LaCour (2014) by David Broockman, Joshua Kalla, and Peter Aronow, a rebuttal paper with retraction letter from Donald P. Green

  - LaCour, Michael J. & Donald P. Green. 2014. “When contact changes minds: An experiment on transmission of support for gay equality[2].” Science 346(6215): 1366.
- Granger-causality analysis of integrated-model outputs, a tool to assess external drivers in fishery by Margarita Rincón, Rachele Corti, Bjarki Elvarsson, Fernando Ramos, and Javier Ruiz.
- Harry Alastair V., Butcher Paul A., Macbeth William G., Morgan Jess A. T., Taylor Stephen M., Geraghty Pascal T. (2019) Life history of the common blacktip shark, Carcharhinus limbatus, from central eastern Australia and comparative demography of a cryptic shark complex. Marine and Freshwater Research. https://doi.org/10.1071/MF18141
- Piwowar HA, Vision TJ. (2013) Data reuse and the open data citation advantage. PeerJ 1:e175 http://dx.doi.org/10.7717/peerj.175
- Some great short courses on R, generalized additive models, and machine learning, etc, by Michael Clark, Center for Social Research, Notre Dame
- An Introduction to Mediation Analysis by Joshua F. Wiley
- ORANGE REPORT: Annual Report of the Swedish Pension System by the Swedish Pensions Agency
- 2011 Census Open Atlas Project by Alex Singleton
- openWAR: An Open Source System for Evaluating Overall Player Performance in Major League Baseball
- Design and Analysis of Bar-seq Experiments by Robinson et al., 2014
- Data and program code for meta-analyses of population health and health services research questions by Tim Churches
- Genomic analysis using R and knitr by Konrad Rudolph
- Assessing the 2016 Budget reforms by John Daley and Brendan Coates
- CFPB Data Point: Becoming Credit Visible by the CFPB Office of Research
- A parametric texture model based on deep convolutional features closely matches texture appearance for humans by Wallis et al.
- Revisiting the effect of red on competition in humans (supplementary information) by Laura Fortunato and Aaron Clauset
- Epiviz Web Components: reusable and extensible component library to visualize functional genomic datasets by Jayaram Kancherla, Alexander Zhang, Brian Gottfried, and Hector Corrada Bravo

## 多言語でのラッパー

- knitr-ruby: a Ruby wrapper
- Flask-FlatPages-Knitr: Knitr preprocessing for Flask-FlatPages

## ブログの投稿

- Using knitr and pandoc to create reproducible scientific reports by Peter Humburg
- Reproducible research, training wheels, and knitr by Jerzy Wieczorek
- Don't R alone! A guide to tools for collaboration with R by Noam Ross
- Getting Started with R Markdown, knitr, and Rstudio 0.96, How to Convert Sweave LaTeX to knitr R Markdown and Converting Sweave LaTeX to knitr LaTeX: A case study by Jeromy Anglim
- Tools for making a paper by Will Lowe
- Integrate data and reporting on the Web with knitr by me as a guest blog post on Revolution Analytics
- knitr: A flexible R authoring tool (HTML5 slides) by Josef Fruehwald
- Planting seeds of reproducibility with knitr and markdown by Mine Çetinkaya-Rundel (the Citizen-Statistician blog)
- A closer look at "How economists get tripped up by statistics" by Laurie Samuels
- Latex Allergy Cured by knitr
- knitr Performance Report-Attempt 1
- Easier literate programming with R by Christophe Lalanne
- knitR - eine Alternative zu Sweave? by Christian B.
- Better R support in pygments by monkey patching SLexer by f3lix
- 被 knitr 包给震撼到了 by @xccds
- Reproducible Research by Tom Torsney-Weir (on Vim and Marked)
- 为什么 Markdown+R 有较大概率成为科技写作主流？ by 阳志平
- Governance Indicators by Russell Shepherd
- Petrol prices adjusted for inflation by Matt Cooper
- Creating beautiful reports from R with knitr by David Smith
- An R-based Research Notebook by Tom Torsney-Weir
- knitR, Markdown, and Your Homework by Jarrett Byrnes
- Color Palettes in HCL Space by Trestle Technology, LLC
- Introduction to R and Biostatistics by Leonardo Collado Torres
- Reproducible Research using R and Bioconductor by Paolo Sonego
- Bioinformaticians Need Lab Notebooks Too by Nacho Caballero
- From OpenOffice noob to control freak: A love story with R, LaTeX and knitr by Christoph Molnar
- Including an interactive 3D rgl graphic in a html report with knitr by Stéphane Laurent
- Create HTML or PDF Files with R, Knitr, MiKTeX, and Pandoc by Justin Meyer
- Reproducible research with R, Knitr, Pandoc and Word by Rolf Fredheim
- Visualizing Farmers' Markets Geo Data using googleVis, plyr, knitr and Markdown using R by Peter Chen

- 2013 NSF Graduate Research Fellowship statistics by Elson Liu
- Ben Bolker's notes on workflows, pipelines, reproducible research, etc. by Steve C Walker
- Playing with R, ggplot2 and knitr by Mladen Jovanović
- A simple bootstrap-based knitr template by Sean Davis
- Automated Blogging by Romain François
- How to avoid scandals using knitr by Mango Solutions
- Fast-track publishing using knitr: Part I, Part II, Part III by Max Gordon
- Basic data-frame manipulations in R by THE ROSTRUM
- Reproducibility is not just for researchers by Kevin Markham
- Tools for statistical writing and reproducible research by Bill Gardner
- knitr ではじめるデータ分析レポート作成~基礎編~ by Yu ISHIKAWA
- Starting data analysis/wrangling with R: Things I wish I'd been told by Stian Håklev
- Knitr's best hidden gem: spin by Dean Attali
- Why use KnitR for scientific publishing? by Rob Les Davidson
- From Code to Reports with knitr & Markdown by Andrew Brooks
- Top 10 data mining algorithms in plain R by Ray Li
- Knotes on Knitr by Jon Zelner
- A reproducibility horror story
- Reproducible Analytical Pipeline by Matt Upson
- Composing reproducible manuscripts using R Markdown by Chris Hartgerink, Tilburg University

# 付録 A

# オブジェクト

オプションを操作するオブジェクト, パターン, そしてフックについて

オリジナルのページ: https://yihui.org/knitr/objects/

オリジナルの更新日: 2017-02-03

---

**knitr** パッケージはオプションと設定を制御する特別なオブジェクト (以下では `obj` と表記) を使用します. このオブジェクトは次のようなメソッドを持ちます.

- **`obj$get(name)`**: `name` という名前のオプションを返します. または, `name` が 2 以上の長さの名前つきベクトルである場合はリストを返します. そして `name` が与えられなかった場合は全てのオプションのリストを返します
- **`obj$set(...)`**: オプションを永続的に変更します. 引数`...` には `tag = value` の形式, または `list(opt1 = value1, opt2 = value2)` のようなオプションのリストとして与えられます
- **`obj$merge(values)`**: 新しいオプションリストを現在のリストに一時的にマージして, そのリストを返します (もとのリストは変更されません)
- **`obj$restore()`**: オブジェクトを元に戻します

**knitr** のこれらのオブジェクトはユーザーから見ることができます.

- **`opts_chunk`**, **`opts_current`**: チャンクオプションを管理します
- **`opts_knit`**: **knitr** パッケージ全体のオプションを管理します
- **`knit_hooks`**: フック関数を管理します
- **`knit_patterns`**: 入力ドキュメントから R コードを取りだすための正規表現を管理します
- **`knit_engines`**: R 以外の他の言語の処理に対する関数を管理します

`knit_patterns` 以外の他の全てのオブジェクトはデフォルトの初期値が設定されており, `knit_paterns` は特に指定がない場合は入力ファイルのタイプに基づいて自動的に決定されます. `knit_hooks` オブジェクトは 1 番よ

く使うことになるでしょう．そして残りの 3 つは直接使用する機会は少ないと思います．たとえば opts_chunk はたいていはコマンドラインで直接実行するよりも入力ファイル内で設定します．

各チャンクが依存している **knitr** の設定は，そのチャンクが実行されるまえに設定**されなければなりません**．スクリプトの一番最初の，他のどのチャンクよりも先に実行される位置に「knit の設定用チャンク」を書き，その中で cache = FALSE, include = FALSE オプションを設定することをおすすめします．設定用チャンクには，このチャンク自身の実行時に有効化される設定を想定する命令文は含めては**いけません**[*1]．設定用チャンクはこのようになるでしょう[*2]．

```
```<<setup, cache=FALSE, include=FALSE>>=
library(knitr)
opts_knit$set(upload.fun = imgur_upload, self.contained = FALSE,
              root.dir = '~/R/project')
@
```
```

技術的な注意点として，これらのオブジェクトはクロージャに近い性質であることを挙げます．これらは関数によって返される，関数リストです．詳細は，エクスポートされてない knitr:::new_defaults 関数を確認してください．チャンクオプションはクロージャによっても管理されています．

[*1] 訳注: チャンクオプションの評価はそのチャンクの実行直前になるため，ここでの変更は次のチャンクまで有効にならない，という意味です

[*2] 訳注: この例は.Rnw のものであり，<< ... >>= というヘッダの記法も.Rnw で使われることに注意してください．R Markdown では {r, ...} を使います (参考: 1.1 章)

# 付録 B

# パターン

入力文書から R コードとチャンクオプションを取得するための正規表現のリストについて

オリジナルのページ: https://yihui.org/knitr/patterns/

オリジナルの更新日: 2017-02-03

---

オブジェクト の `knit_patterns` は **knitr** のパターンを管理します. たとえば現在の正規表現パターンのリストを取得するには `knit_patterns$get()` が使えます. パターンリストには以下のコンポネントが含まれています.

- **`chunk.begin`**: コードチャンクの開始時点のパターンです. チャンクオプションを取りだすための, `()` で定義されたグループが含まれていなければなりません.
- **`chunk.end`**: チャンクの終了時点のパターンです (文芸的プログラミングにおける本来の意味とは異なります. 本来は通常のテキストの開始時点を意味していました. 詳細は 1 章の `filter.chunk.end` オプションを参照してください).
- **`chunk.code`**: このパターンにマッチする文字を除去することで, チャンクから R コードを取り出します.
- **`inline.code`**: 地の文のなかにまぎれたインライン R コードを取りだすためのパターンです (つまり, コードチャンクの分離には使いません). `chunk.begin` 同様に, グループが含まれていなければなりません.
- **`inline.comment`**: インラインコメントのパターンです (このパターンにマッチするインライン R コードのトークンは, 各行から除去されます)
- **`header.begin`**: 文書のヘッダの開始点を見つけるためのパターンです. 出力文書にヘッダ情報を挿入する際に使われます (例: LaTeX のプリアンブルコマンド, HTML の CSS スタイル)
- **`document.begin`**: 文書の本文開始点を見つけるためのパターンです (たとえば LaTeX プリアンブルを取りだすのに使うとして, tikz のグラフィックスを外部化したり, シンタックスハイライトのためにコードを挿入したりすることができます)

パターンが `NULL` である場合, 何もマッチしません.

Sweave のように, **knitr** には 2 種類の R コードがあります. パラグラフのようなまとまりである「コードチャンクと」, 地の文の中にあって実行される「インライン R コード」です. 文書内のチャンクのオプションは `label, opt1=TRUE, opt2=FALSE, opt3='character.string'` という形式になります (オプションは , と = で繋げられ, ラベルのみが `label` が暗黙に `value` とみなされるため, ラベルのみが `value` を必要としません).

## B.1 ビルトインパターン

**knitr** には `all_patterns` を保存するいくつかのビルトインパターンがあります.

```
01 library(knitr)
02 str(all_patterns)
```

**Knitr** は最初に適切なパターンのセットを決定するために入力文書の中身を探索します. もしこの自動検出が失敗したら, 入力文書のファイル拡張子名から判定し, **knitr** は自動で上記のリストから適切な形式を選び出します. たとえば入力ファイルが `file.Rnw` ならは `all_patterns$rnw` を, `file.html` ならば `all_patterns$html` を, というふうに使用します.

`pat_rnw()`, `pat_html()`, `pat_md()`, `pat_tex()`, `pat_brew()` という一連の便利な関数はビルトインパターンを設定するために使われます.