

# 宇宙が生まれる前の話

R ver 1.0.0で関数を知る

TokyoR #75

?

- R歴7年目(無駄に年数だけ重ねてしまった)
  - クソ文系出身
  - 多変量解析から機械学習までテーブルデータ系はだいたいフルスクラッチ
- 最近パソコン買いました.
- Data Analyst(表の顔)
  - 調査屋の前処理・統計解析担当
  - コードを書く「なにやってるかわからない」人(社内評価)
- データ野郎(裏の顔)
  - 今年はいろいろ挑戦したい

# 【余談】 きっかけ

よーし新年一発目, **Lightning Talk** やっちゃおうか

## 4. 発表日とセッションを選んでください (Date & Session) \*

- ☐ 1月日付未定 (January), 応用セッション advanced/application session (30min)
- ☐ 1月日付未定 (January), LT Lightning Talk (30min)
- ☐ 3月日付未定 (March), 応用セッション advanced/application session (30min)
- ☐ 3月日付未定 (March), LT Lightning Talk (30min)

# 【余談】 きっかけ

なあ……

## 4. 発表日とセッションを選んでください (Date & Session) \*

- ☒ 1月日付未定 (January), 応用セッション advanced/application session (30min)
- ☐ 1月日付未定 (January), LT Lightning Talk (30min)
- ☐ 3月日付未定 (March), 応用セッション advanced/application session (30min)
- ☐ 3月日付未定 (March), LT Lightning Talk (30min)



# 宇宙が生まれる前の話

R ver 1.0.0で関数を知る

TokyoR #75

なぜ宇宙が生まれる前の話をするのか？

宇宙が生まれる前とか知らない



我々は何を知らないか  
=  
(我々は何を知っているか?)<sup>C</sup>



我々はどこから来たのか





我々は何者か





我々はどこへ行くのか

よくわからない

人が含まれる場: 地球



なぜ地球で生命が生まれたのか？





地球はどのようにしてできたのか？



よくわからない

地球が含まれる場: 宇宙



宇宙はなぜ生まれたのか？



宇宙が始まる前は何があったのか？

よくわからない

# 今日のお話

- よく考えたら自分は何もよく知らない(無知の知)
  - モダンなアイツら
    - 今日ないの
  - ベイズなアイツら
    - 今日ないの
  - RStudio
    - 今日ないの
- Rむかしばなしをします.
  - 応用セッションなのに応用しない
  - ハードルはくぐるもの

此 処 マ デ 前 座

ソ ロ ソ ロ ク ド イ ノ デ ハ ナ イ カ

# 宇宙が生まれる前の話

## R ver 1.0.0で関数を知る

TokyoR #75



# 初心者セッション(超基礎)

- Rを「古典」から学ぶ
- 早速Rをインストールするところから始めよう

# 初心者セッション(超基礎)



CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

About R

[R Homepage](#)

[The R Journal](#)

Software

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

Documentation

[Manuals](#)

[FAQs](#)

[Contributed](#)

[R 2.5.1](#) (July, 2007)

[R 2.5.0](#) (April, 2007)

[R 2.4.1](#) (December, 2006)

[R 2.4.0](#) (October, 2006)

[R 2.3.1](#) (June, 2006)

[R 2.3.0](#) (April, 2006)

[R 2.2.1](#) (December, 2005)

[R 2.2.0](#) (October, 2005)

[R 2.1.1](#) (June, 2005)

[R 2.1.0](#) (April, 2005)

[R 2.0.1](#) (November, 2004)

[R 2.0.0](#) (October, 2004)

[R 1.9.1](#) (June, 2004)

[R 1.8.1](#) (November, 2003)

[R 1.7.1](#) (June, 2003)

[R 1.6.2](#) (January, 2003)

[Installer for R 1.5.1](#) (June, 2002)

[Installer for R 1.4.1](#) (January, 2002)

[Installer for R 1.3.1](#) (September, 2001)

[Binary files for R 1.2.2](#) (March, 2001)

[Binary files for R 1.0.0](#) (February, 2000)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

---

Last change: 2017-10-08, by Uwe Ligges

# 初心者セッション(超基礎)

[Binary files for R 1.0.0](#) (February, 2000)

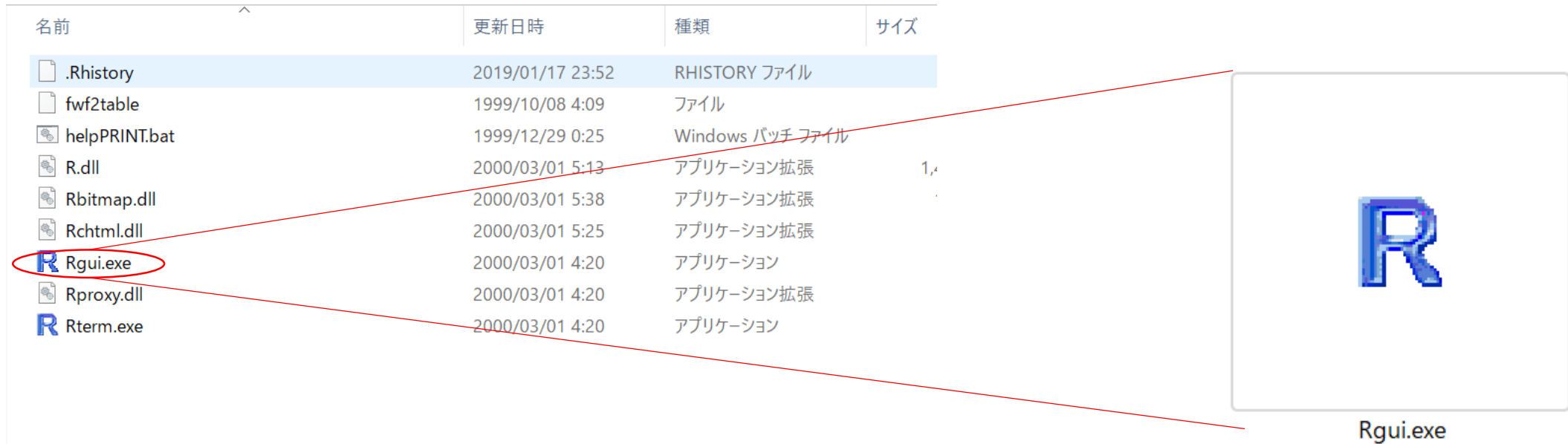
# 初心者セッション(超基礎)



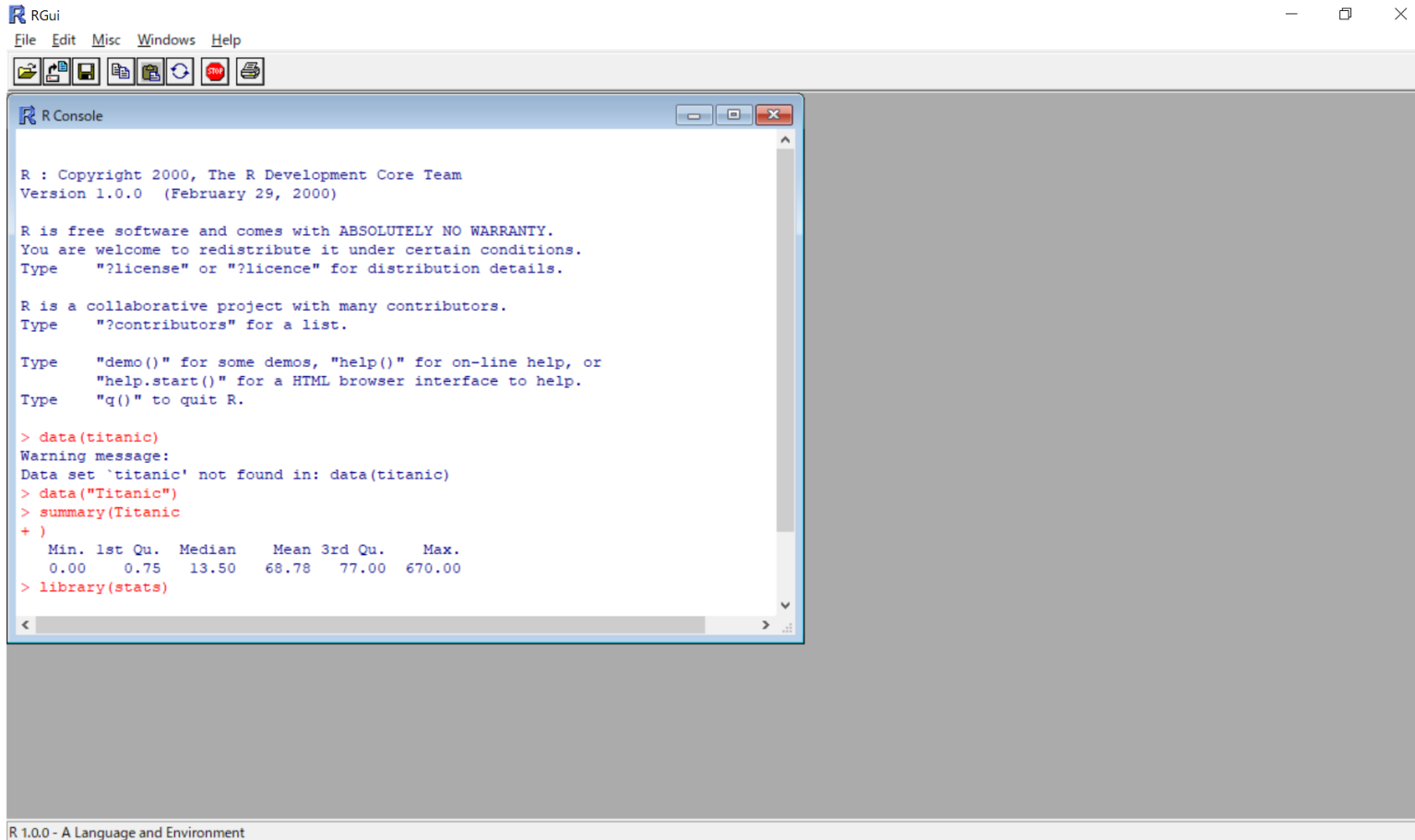
R1000.zip

# 初心者セッション(超基礎)

- ...¥R1000¥Documents and Settings¥murdoch¥Desktop¥BASE¥rw1000¥bin



# 初心者セッション(超基礎)



The screenshot shows the RGui application window. The title bar reads 'RGui'. The menu bar includes 'File', 'Edit', 'Misc', 'Windows', and 'Help'. Below the menu bar is a toolbar with icons for file operations and execution. The 'R Console' window is open, displaying the following text:

```
R : Copyright 2000, The R Development Core Team
Version 1.0.0 (February 29, 2000)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type "?license" or "?licence" for distribution details.

R is a collaborative project with many contributors.
Type "?contributors" for a list.

Type "demo()" for some demos, "help()" for on-line help, or
"help.start()" for a HTML browser interface to help.
Type "q()" to quit R.

> data(titanic)
Warning message:
Data set `titanic' not found in: data(titanic)
> data("Titanic")
> summary(Titanic)
+ )
  Min. 1st Qu.  Median    Mean 3rd Qu.   Max.
  0.00   0.75   13.50   68.78   77.00  670.00
> library(stats)
```

At the bottom of the RGui window, the text 'R 1.0.0 - A Language and Environment' is visible.

# 初心者セッション(超基礎)

```
R : Copyright 2000, The R Development Core Team  
Version 1.0.0 (February 29, 2000)
```

R : Copyright 2000, The R Development Core Team  
Version 1.0.0 (February 29, 2000)



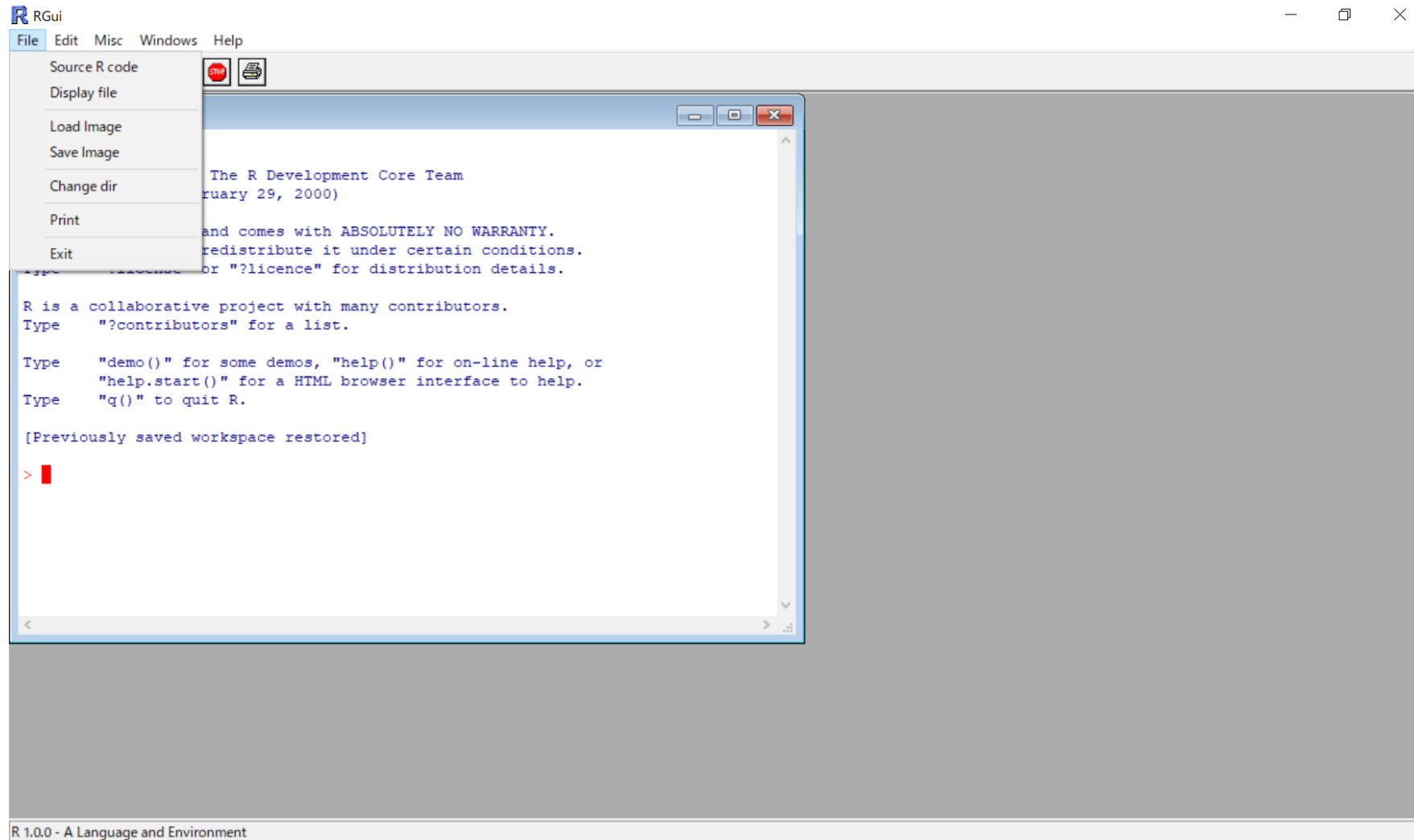


# R ver. 1.0.0

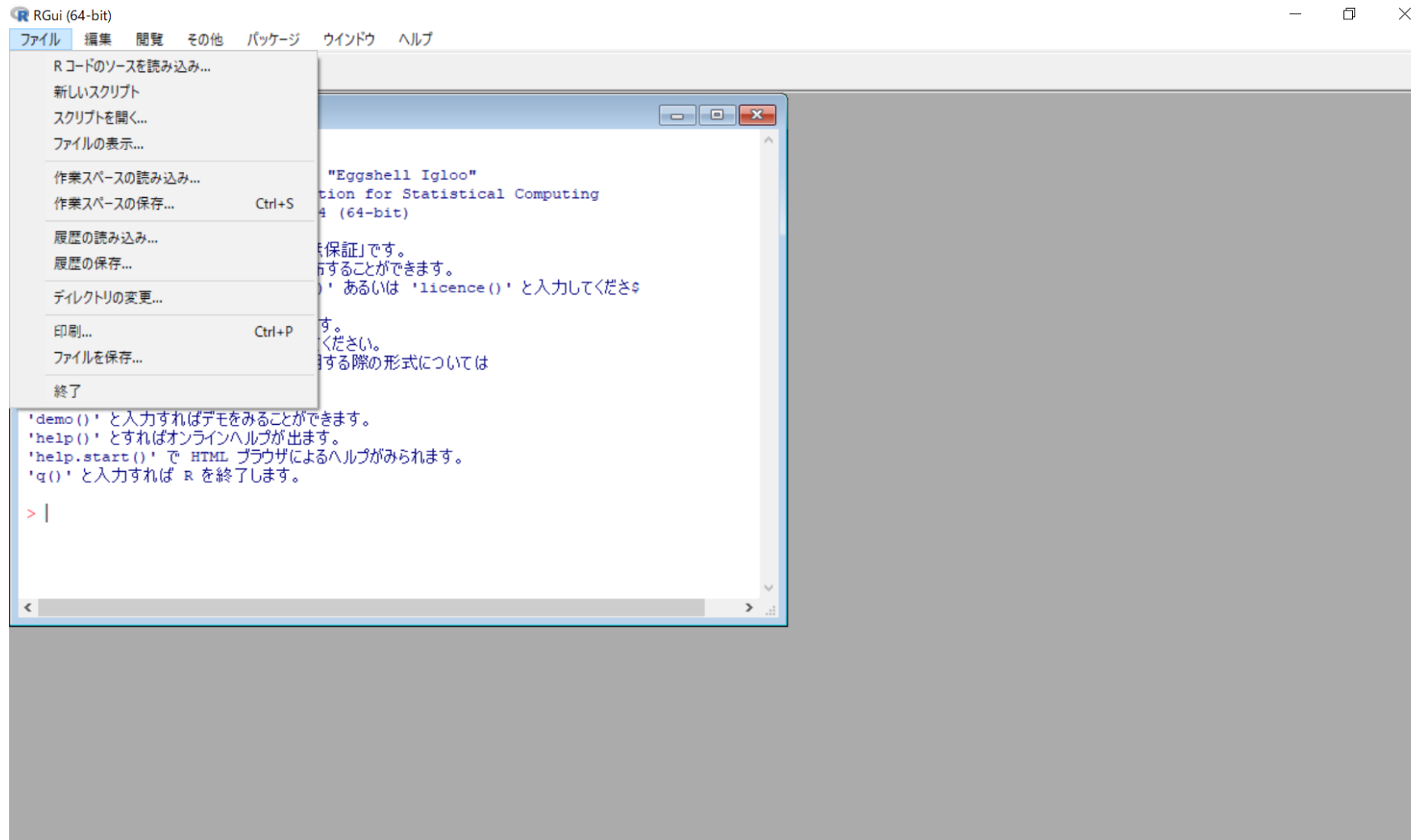
- 2000年2月リリース
  - Hadley(TidyR神)がRと出会う→2002年頃※
    - ※)ソースは以下(2019/01/11取得)  
統計言語「R」の神はなぜ無償で貢献したのか オープンコミュニティで活躍する“新人類”の誕生
      - <https://business.nikkeibp.co.jp/atcl/report/16/122700258/010900004/>
  - 2002年にはすでに1.4.1(1月), 1.5.1(6月)がリリース
- 実質旧約聖書(?)
- 実質欠史八代(??)
- 実質前1200年のカタストロフ(???)

翠 齋  
眺メテミル

# R ver 1.0.0

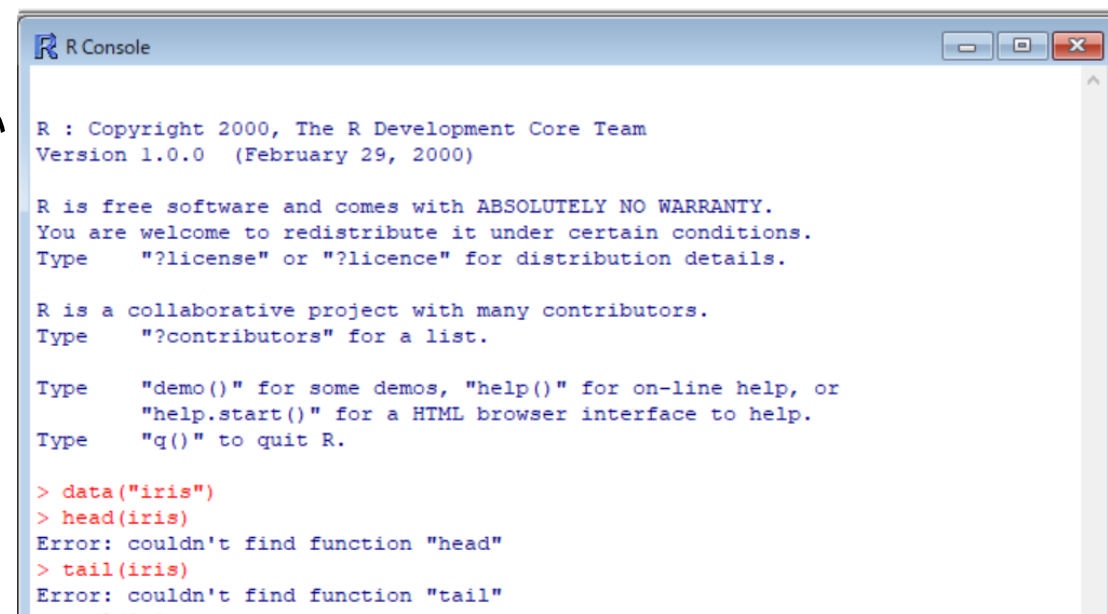


# 【参考】 R ver. 3.5.2



# R ver. 1.0.0

- 日本語版はない(あたりまえ)
  - 日本語(マルチバイト文字)を入力すると強制終了する
- GUI上でスクリプトを書けない(コンソール直打ち)
- 今ある関数が(一部)存在しない
  - `log1p()`はある
  - `expm1()`はない
  - `head/tail` もない
  - ほかにもない関数がありそう.



```
R Console

R : Copyright 2000, The R Development Core Team
Version 1.0.0 (February 29, 2000)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type    "?license" or "?licence" for distribution details.

R is a collaborative project with many contributors.
Type    "?contributors" for a list.

Type    "demo()" for some demos, "help()" for on-line help, or
        "help.start()" for a HTML browser interface to help.
Type    "q()" to quit R.

> data("iris")
> head(iris)
Error: couldn't find function "head"
> tail(iris)
Error: couldn't find function "tail"
```

紅 齋  
眺メテミル

# 宇宙が生まれる前の話

R ver 1.0.0で学ぶ関数

TokyoR #75

# というか

- なんで関数？
  - 関数とは動詞(@kilometer, TokyoR#75)
  - (普通は)動詞なしにコミュニケーションはできない
    - ~~まれに「ま新ヲ潤」など、動詞なしでコミュニケートできる人もいる~~
  - 現代は関数を %>% でつなぐ宇宙(コスモ)
  - 神代(1.0.0)は……？
- 「この処理実現するための関数わかんねえな」ってことがある
  - 「この気持ちを伝えたいけど適切な動詞がわからない」とき、自分で作る  
(「もによる」「サチる」「kaggr」など)
  - 別に特別なことではないのでは？



「関数」って  
何？

「動詞」って  
何？

Learning R Programming  
Become an efficient data scientist with R

# Rプログラミング 本格入門

達人データサイエンティストへの道

■ Kuhn Rem : 著    ■ 渡辺啓明・松村香子・市川大祐 : 訳    ■ 株式会社ホクソエム : 監訳



共立出版



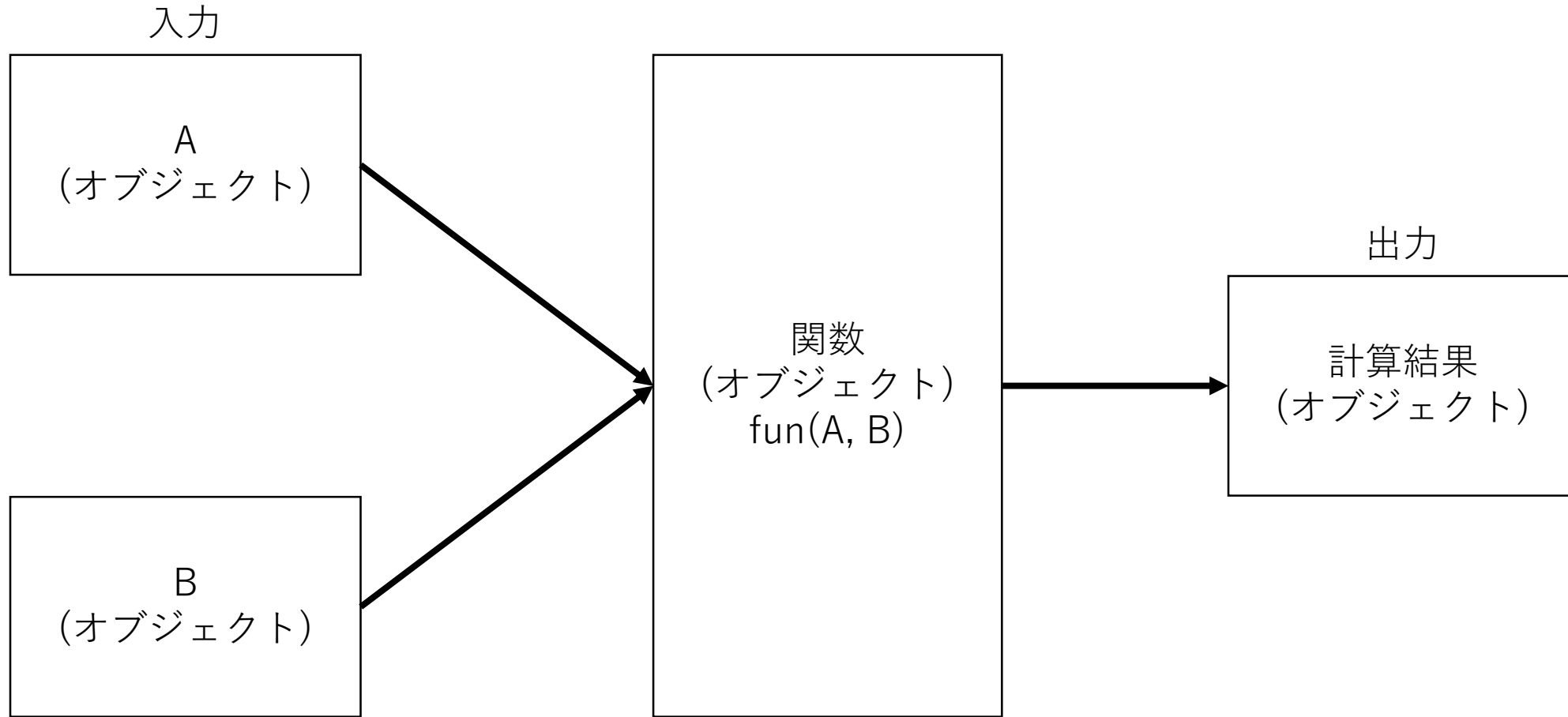
(Rの環境において)  
すべてはオブジェクトであり、  
すべては関数である。

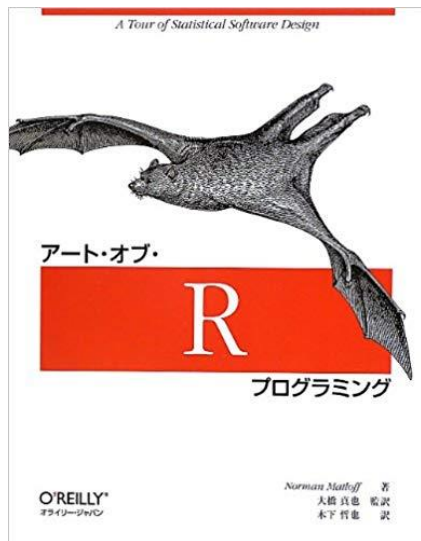
.....

すべての関数もまたオブジェクトである。  
(p. 53)

すべての関数は  
オブジェクトである  
って何？

# 結局関数ってどういうことなの……





入力を受け取り,  
それをつかって他の値を計算して  
結果を返す一連の命令です  
(p. 6)

# 関数とは in R

- 平均(代表値)を返す
  - `mean(x, na.rm = FALSE)` # 平均
  - `median(x, na.rm = FALSE)` # 中央値
  - `var(x, na.rm = FALSE)` # 分散
  - `sd(x, na.rm = FALSE)` # 標準偏差
  - `summary(x)` # なんかいろいろ
- 二項演算子
  - `1 + 2` など
  - ``+`(1,2)` → 3を返す(``-``, ``*``, ``/``もある)
  - 三項以上はErrorとなる
    - 具体的なエラーメッセージも存在しない時代

```
> `+`(1,2)
[1] 3
> `+`(1,2,3)
Error: syntax error
> ■
```



# 関数とは in R

- 配列を組む関数
  - `c(1,2,3)`
  - `list(1,2,3)`
  - `matrix(c(1,2,3,4,5,6), 2,3)`
  - `data.frame(data)`
- すべては関数である

# ない関数を作ってみる

- R ver. 1.0.0
  - head/tail がない
  - log1pはあるけどexpm1がない
  - 1.0.0触ってから3.5.2をいじると浦島太郎になれる
  - ver. 1.0.0の時代に今ある関数を作れば実質的に異世界転生
- Rの場合
  - おなまえ <- **function**(必要な引数){実行する処理}で関数ができる

実際にやってみた

# ない関数を作ってみる

- **Head**
  - 上から何個かほしい
- **Tail**
  - 下から何個かほしい
- **expm1**
  - $x$ を入力して  $\exp(x - 1)$ を出力したい

# ない関数を作ってみる

- `Head <- function(DataFrame, n = 5){  
 DataFrame[1:n,]  
}`
- `Tail <- function(DataFrame, n = 5){  
 END <- nrow(DataFrame)  
 START <- END-n  
 DataFrame[START:END,]  
}`
- `data.frame`, `matrix`型のみに有効
  - すべての型に適用できるようにするのは読者の練習問題とする(は?)
  - `github`にあげているコードにはベクトルにも適用できるように実装しました

# ない関数を作る

- `expm1`関数を再現する
- ```
EXPm1 <- function(x){  
    exp(x-1)  
}
```

# 関数の特徴 on R

- ver. 1.0.0でも動的型付け
- 割と自由に数値を入れてもエラーを吐かない
  - exp関数: 数値, ベクトル, 行列などもちゃんと返してくれる.
  - integer: `Expn1(2)` = 2.718282
  - vector : `Expn1(c(1,2,3,4,5))` = 1.000000 2.718282  
7.389056 20.085537 54.598150
  - matrix : `Expn1(matrix(c(1,2,3,4,5,6),2,3))`  
[,1] [,2] [,3]  
[1,] 1.000000 7.389056 54.59815  
[2,] 2.718282 20.085537 148.41316

# 関数の特徴 on R

- ver. 1.0.0でも動的型付け
- 割と自由に数値を入れてもエラーを吐かない
  - exp関数: 数値, ベクトル, 行列などもちゃんと返してくれたりする?
    - factor型, character型, logical型はエラー
    - 複数の型が入ってるdata.frame型もエラー
    - list方は要素の型があってもエラー

```
> Expml(list(1,2,3,4,5))
```

```
Error in x - 1 : non-numeric argument to binary operator
```



# 関数の特徴 in R

- 関数のデフォルト値
  - 引数として設定されているが、入力しなくてもどうにかしてくれる
  - ここは “<-” ではなく “=” で指定する.
- `Head <- function(DataFrame, n = 5){  
    DataFrame[1:n,]  
}`
  - デフォルト値があると、DataFrameさえ入れればどうにかしてくれる

演算子を作る

# 関数とは in R

- 平均を返す関数
  - `mean(x, na.rm = FALSE)`
    - `x`: 僕の作った最強のデータ(ベクトル)
    - `na.rm`: パラメータ(NAを含めて計算する? しない?)
- 二項演算子
  - Lispっぽい
  - ``+`(1,2)` → 3を返す(``-``, ``*``, ``/``もある)
  - 三項以上はErrorとなる
    - 具体的なエラーメッセージも存在しない時代

```
> `+`(1,2)
[1] 3
> `+`(1,2,3)
Error: syntax error
> ■
```

すべての演算子は  
関数じゃね？

# 演算子を作る

- Pythonなどにある演算子`+=`/`-=`(インクリメント/デクリメント)
  - Rにはデフォルトで定義されていなかったりする
- パイプ演算子 `%>%`
  - `magrittr`のない神代



(Rの環境において)  
すべてはオブジェクトであり、  
すべては関数である。

.....

すべての関数もまたオブジェクトである。  
(p. 53)

自己定義演算子は関数として  
定義できんじゃない？

# 演算子を作る

- 演算子は”%(好きな演算子の名前)%”としてつくる
- インクリメント/デクリメント
  - “%+=%” <- function(x,y){x + y}
  - “%-=%” <- function(x,y){x - y}
- パイプ演算子
  - “%>1%” <- function(x, Function){Function(x)}

# 動かしてみる

```
> c(1,2,3,4,5) %+=% 1
```

```
[1] 2 3 4 5 6
```

```
> c(1,2,3,4,5) %-= % 1
```

```
[1] 0 1 2 3 4
```

```
> c(1,2,3,4,5) %>% sum
```

```
[1] 15
```

```
> 「パイプ演算子定義できたやんけ！」 .....と思いきや
```



# 動かしてみる

```
>matrix(c(1,2,3,4,5,6),2,3) %>% apply(MARGIN = 1,  
FUN = sum)
```

Error in apply(MARGIN = 1, FUN = sum) : Argument "X"  
is missing, with no default

```
> (T_I)  
[1] TRUE ←これver 3.5.2で再現しなかった
```

```
> (^_^)  
Error: syntax error ← ! ? ! ?
```

```
> 
```

# 蛇足：演算子“\_”

```
> (T_T)
[1] TRUE
```

←これver 3.5.2で再現しなかった

- “\_”  $\Leftrightarrow$  “<-”
- > A\_1
- > A
- [1] 1
- R\_Linux氏からの情報提供

# 蛇足: 演算子”\_”



ホクそうむ

@R\_Linux

フォローする



返信先: @0\_u0さん

むかしのRだとunderscoreは代入記号として使われたので、これ、いまのバージョンの(T<-T)に相当するんじゃないでしょうか

13:13 - 2019年1月18日

1件のリツイート 5件のいいね



1



1



5



ホクそうむ

@R\_Linux

フォローする



返信先: @0\_u0さん

R-1.9.0 からunderscoreを代入には使えなくなっただけです

[cran.r-project.org/src/base/NEWS.1](https://cran.r-project.org/src/base/NEWS.1)

14:56 - 2019年1月18日

くやしいので

- `matrix`型にパイプ演算子で`apply()`を適用できるように改修したい
- `%>%`演算子どう動いてるんだ.....？

# 改修してみる

- %>%演算子どう動いているんだ.....?
  - igjit先生がすでにやっていた
  - [https://igjit.github.io/slides/2018/01/tiny\\_pipe/#/](https://igjit.github.io/slides/2018/01/tiny_pipe/#/)

パイプ演算子  
自作入門

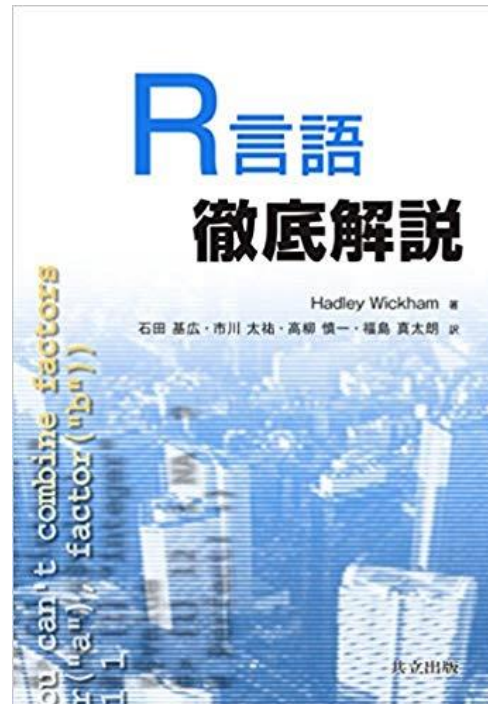
@igjit



# 改修してみる

- **igjit**先生に従ってみる

- Rの演算子とか遅延評価とか非標準評価などについては前スライドのURL
- または →



を参照してください.....

# 改修してみる

- **igjit**先生に従ってみる
  - 「R言語完全に理解した」体で↓を再現してみる

表現式がドットを含むかどうか確認する補助関数

```
has_dot <- function(expr) {  
  any(vapply(expr, identical, logical(1), quote(.)))  
}
```

```
has_dot(quote(1 + 2))  
# [1] FALSE
```

```
has_dot(quote(1 + .))  
# [1] TRUE
```


# 改修してみる

- **igjit**先生に従って`has_dot()`関数を作ってみる
  - 1.0.0で未実装な{base}関数
    - `identical()`
    - `vapply()`
  - オリジナルで作るしかなさそう.....



# 改修してみる

- `HasDot <- function(expr){  
 any(as.list(expr) == substitute(.))  
}`

```
> HasDot(quote(1+.))  
[1] TRUE  
> HasDot(quote(1+2))  
[1] FALSE  
> 
```

- 返り値の再現ができたしいけんじゃね？

# 改修してみる

- ドットがないときに挿入する関数
  - どうやら1.0.0でもそのまま使えそう(安心)
  - ```
InsertDot <- function(expr){  
  as.call(c(expr[[1]], quote(.), as.list(expr[-1])))  
}
```
- これで準備は整った

# 改修してみる

- ```
"%>2%" <- function(LeftHands, RightHands){  
  env <- parent.frame()  
  expr <- substitute(RightHands)  
  dotted <- if(HasDot(expr)) expr else  
InsertDot(expr)  
  eval(dotted, list(. = LeftHands), env)  
}
```
- 流れ変わったな

# 改修してみる

```
> matrix(c(1,2,3,4,5,6),2,3) %>% apply(., MARGIN = 1, sum)
[1]  9 12
```

～♪ 適宜完全勝利した例のBGMを脳内再生

# おまけ

NAを特定の数値で埋めたい

```
> "%fillna%" <- function(x,y){  
  x[is.na(x)] <- y  
  return(x)  
}  
> c(1,2,3,4,5,NA) %fillna% 1  
[1] 1 2 3 4 5 1  
> matrix(c(1,NA,3,4,NA,6),2,3) %fillna% 1  
      [,1] [,2] [,3]  
[1,]    1    3    1  
[2,]    1    4    6  
>
```

# まとめ

- 宇宙のできる前のRに異世界転生
  - 今当たり前に使っている関数が存在しない時代
  - 返り値から関数の中身を想像して試行錯誤するやつをする
    - これがたのしい
    - 既存の関数を拡張するきっかけにもなりそうだった
    - OSSへのコミットに活かせるかもしれない！
- 異世界で現代の技術を使って無双したかったが.....？
  - げんじつはきびしい

# 次回予告

- テーマの候補
  - 欠損値(**NA**)の処理の話(定数代入？多重代入？完全情報最尤法？)
  - データの匿名化の話(**k-anonymity**, **differential privacy**)
  - 制約付きの回帰分析とかの実装
  - 多言語とか**Web**アプリとの連携とかそういう実装してみた系とか

# 主な参考



## パイプ演算子 自作入門

@igjit





Enjoy!