2019.12.21.SAT

データ解析の応用とツールの現在

株式会社ef-prime

鈴木 了太　@efprime_jp

ef-prime, inc.

自己紹介

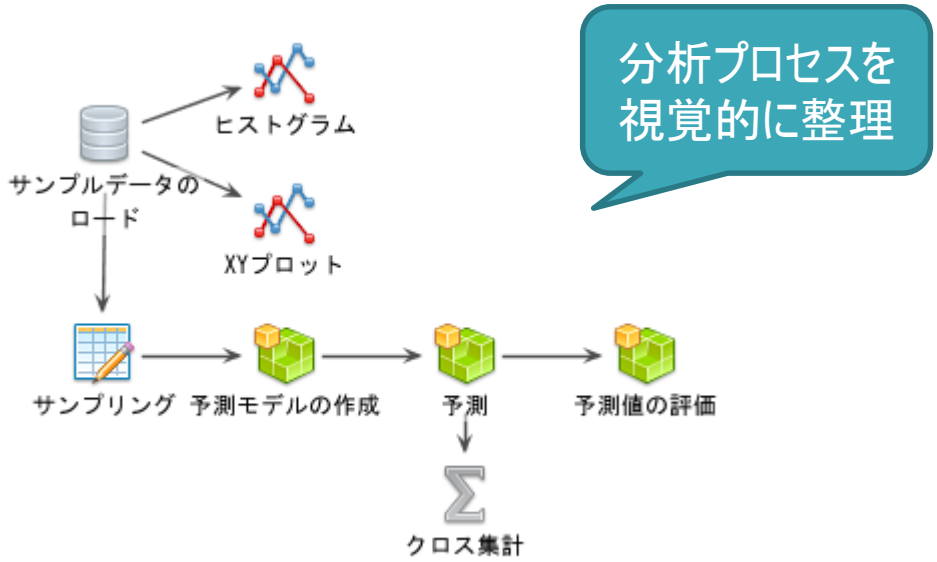
- 鈴木 了太
  - 株式会社ef-prime代表
  - Rユーザー歴約18年
    - 開発CRANパッケージ：pvclust
    - 書籍：Useful R 10 Rのパッケージおよびツールの作成と応用（共立出版、第IV部）

- 株式会社ef-prime（エフプライム）
  - データ分析コンサルティング（主に企業向け）
  - 2006年設立、多くのプロジェクトでRを活用
  - データ分析ツールを開発・公開

ef-prime, inc.

R AnalyticFlow

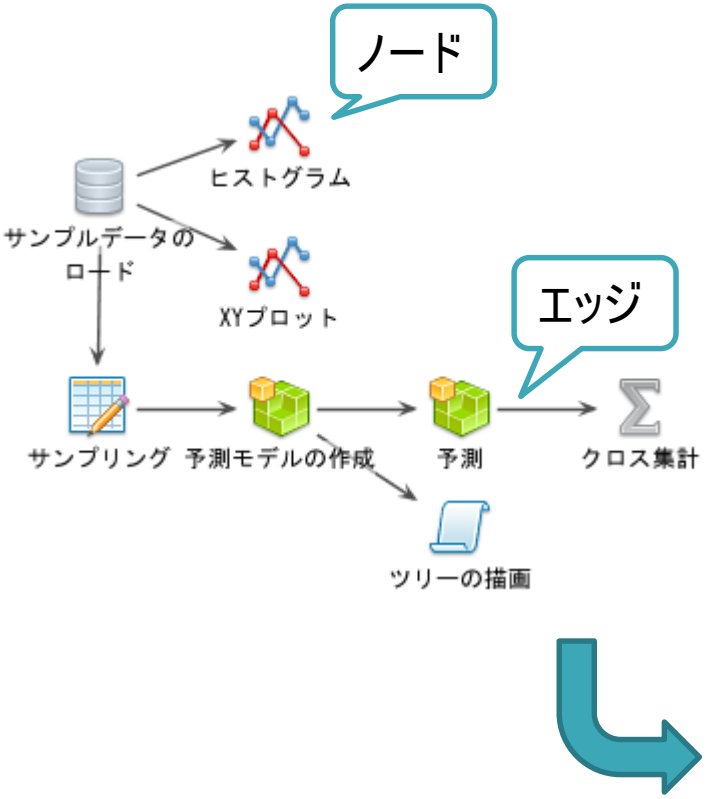
- データ解析のためのR GUI
  - 分析プロセスをワークフローで表現
  - オープンソース、無償公開
  - Javaで開発、マルチOS対応
    - Windows / Mac / Linux



https://r.analyticflow.com

R AnalyticFlow

ノードをエッジで繋いだ分析フローを作成、Rスクリプトを生成して実行




```r
# 1. データの読み込み
data(iris)

# 2. 探索的分析
plot(iris[, 1:4], col =
as.integer(iris$Species) + 1)
boxplot(Petal.Length ~ Species, data =
iris, col = 3, main = "Petal.Length")

# 3. モデリング
library(rpart)
rp <- rpart(Species ~ ., iris)

# 4. モデルの確認
plot(rp, margin = 0.1, branch = 0.3)
text(rp, fancy = T, all = T, use.n = T)

# 5. 予測および評価
pred <- predict(rp, type = "class")
xtabs(~pred + iris$Species)
```

データ解析の応用とツールの現在

- 事例紹介＆R Tips
  - 実際の業務（によく似たストーリー）と、関連するRの話題

- 背景
  - R AnalyticFlowは以下のユースケースに最適
    - データ分析を受託し、分析結果を納品
    - 作成した分析フローをRスクリプトとして納品
    - データはファイルやデータベースにあり、分析はローカル
  - 当てはまらない事例
    - データがクラウドにあり、分析もクラウド環境で行いたい
    - そもそもPython指定でRが使えない
    - Rの機能をインタラクティブに使いたいが、Rに触りたくない

ef-prime, inc.

ef-prime, inc.

事例紹介

実際の事例に基づいていますが、詳細は意図的に変更している箇所があります。

登場するソフトウェアおよびシステム等の名称は各社の登録商標を含みます。

事例：Rで分析GUI

- 要件
  - データに対して定型的な統計解析を行いたい
  - 統計を専門としないメンバーが探索的に解析できるようにしたい
- 解決策
  - Shinyを用いたウェブアプリとして実現
  - 多数のデータセットを柔軟に扱えるよう工夫

ef-prime, inc.

事例：Rで分析GUI

以下はirisデータを使ったイメージ。共通のID（row.names）を持つ
複数のテーブルがあり、これらの相互関係を分析したい

sepal

	length	width
1	5.1	3.5
2	4.9	3
3	4.7	3.2
4	4.6	3.1
5	5	3.6

petal

	length	width
1	1.4	0.2
2	1.4	0.2
3	1.3	0.2
4	1.5	0.2
5	1.4	0.2

info

	species
1	setosa
2	setosa
3	setosa
4	setosa
5	setosa

事例：Rで分析GUI

x軸はsepalテーブルの
length変数

Shiny Example

x

sepal

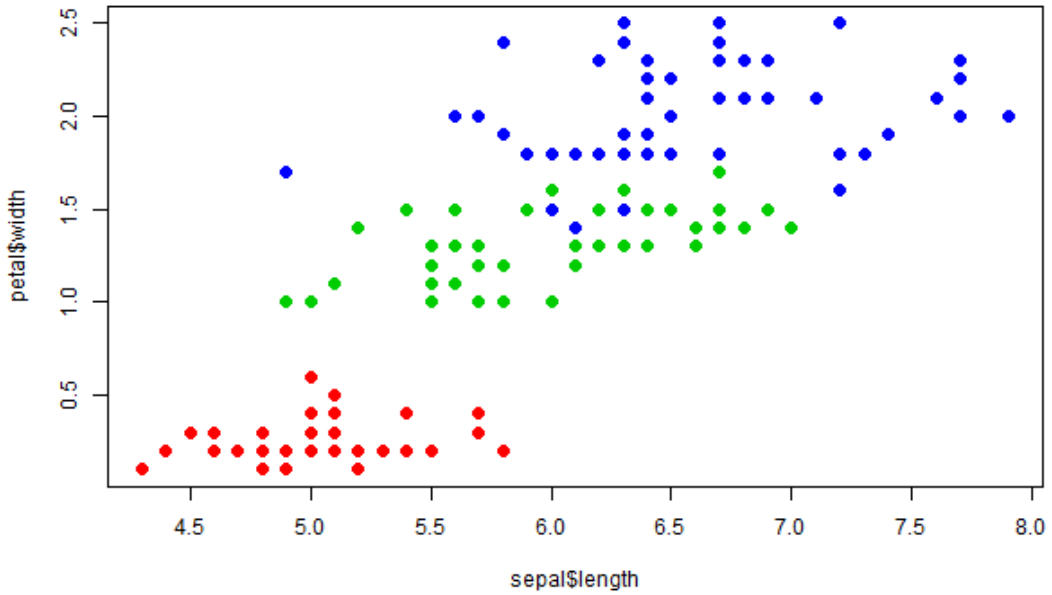
length

y

petal

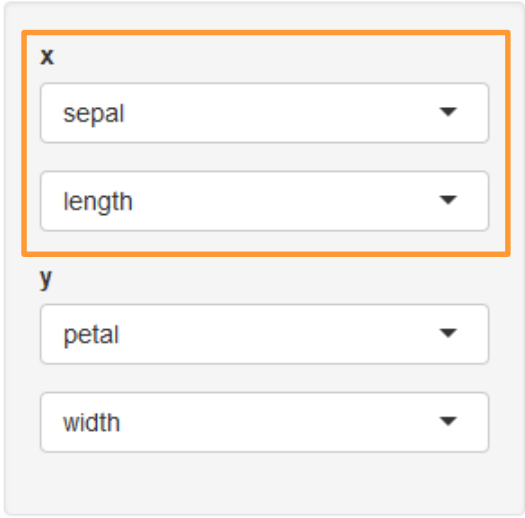
width

y軸はpetalテーブルの
width変数



解説用に単純化したイメージです。実際のプログラムとは大きく異なります。

事例：Rで分析GUI




```
ui <- fluidPage(

  titlePanel("Shiny Example"),

  sidebarLayout(
    sidebarPanel(
      selectInput("x_df", "x", choice = c("sepal", "petal")),
      selectInput("x_var", NA, choice = c("length", "width")),
      selectInput("y_df", "y", choice = c("sepal", "petal")),
      selectInput("y_var", NA, choice = c("length", "width"))
    ),

    mainPanel(
      plotOutput("Plot")
    )
  )
)
```

事例：Rで分析GUI

```
server <- function(input, output) {

  output$Plot <- renderPlot({
    x <- get(input$x_df, envir = globalenv())
    y <- get(input$y_df, envir = globalenv())

    plot(x[[input$x_var]], y[[input$y_var]],
         xlab = paste(input$x_df, input$x_var, sep = "$"),
         ylab = paste(input$y_df, input$y_var, sep = "$"),
         col = as.integer(info$species) + 1,
         pch = 20, cex = 2)
  })
}
```

指定されたテーブルを
グローバル環境から取得

テーブルごとに指定
された変数をプロット

事例：Rで分析GUI

あるテーブルの変数で条件を指定し、条件を満たすデータについて分析を行うことができるようにしたい

sepal

	length	width
1	5.1	3.5
2	4.9	3
3	4.7	3.2
4	4.6	3.1
5	5	3.6

petal

	length	width
1	1.4	0.2
2	1.4	0.2
3	1.3	0.2
4	1.5	0.2
5	1.4	0.2

info

	species
1	setosa
2	setosa
3	setosa
4	setosa
5	setosa

sepal$length >= 5

sepal

	length	width
1	5.1	3.5
5	5	3.6

petal

	length	width
1	1.4	0.2
5	1.4	0.2

info

	species
1	setosa
5	setosa

事例:Rで分析GUI

Shiny Example

x

sepal

length

y

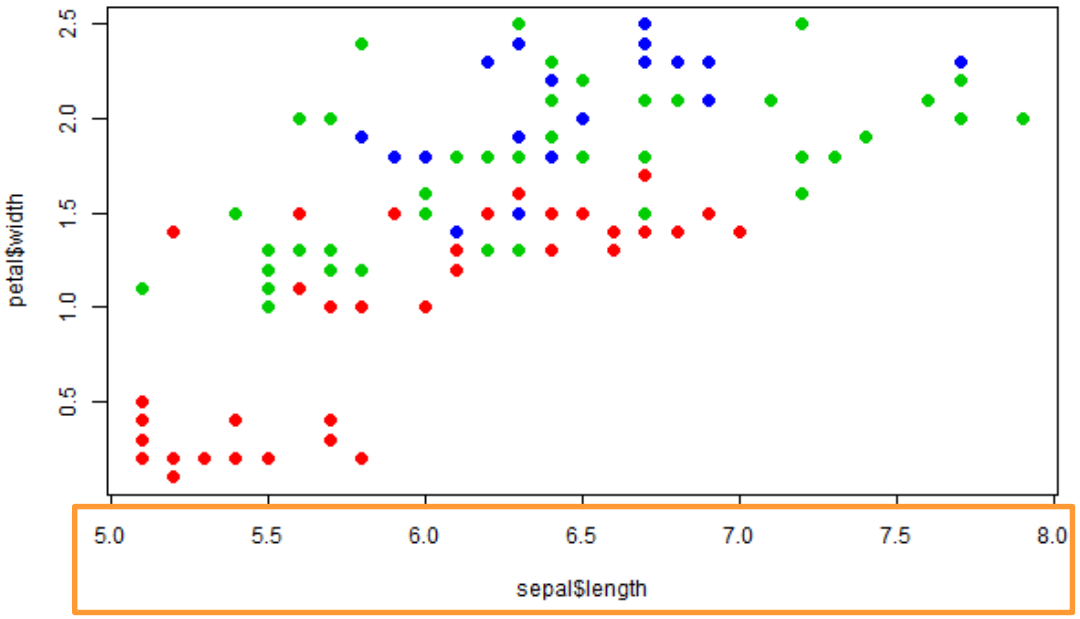
petal

width

condition

sepal$length > 5

apply



ef-prime, inc.

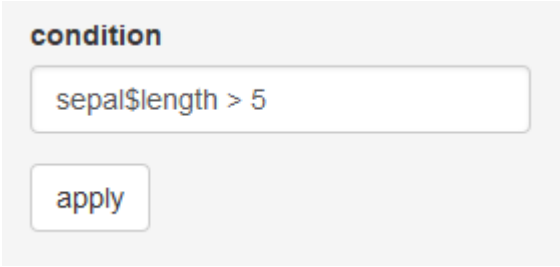
事例：Rで分析GUI

```
ui <- fluidPage(

  titlePanel("Shiny Example"),

  sidebarLayout(
    sidebarPanel(
      selectInput("x_df", "x", choice = c("sepal", "petal")),
      selectInput("x_var", NA, choice = c("length", "width")),
      selectInput("y_df", "y", choice = c("sepal", "petal")),
      selectInput("y_var", NA, choice = c("length", "width")),
      textInput("cond", "condition", value = ""),
      actionButton("apply", "apply")
    ),

    mainPanel(
      plotOutput("Plot")
    )
  )
)
```



事例：Rで分析GUI

データをグローバル環境ではなく
GetData() 関数から取得

```
output$Plot <- renderPlot({
   x <- get(input$x_df, envir = GetData())
   y <- get(input$y_df, envir = GetData())

   plot(x[[input$x_var]], y[[input$y_var]],
        xlab = paste(input$x_df, input$x_var, sep = "$"),
        ylab = paste(input$y_df, input$y_var, sep = "$"),
        col = as.integer(info$species) + 1,
        pch = 20, cex = 2)
 })
```

他のコードはそのまま
⇒　ロジックを分離している

事例：Rで分析GUI

```
server <- function(input, output) {

  GetData <- eventReactive(input$apply, try({
    e <- list2env(
      mget(c("sepal", "petal", "info"), envir = globalenv()),
      parent = globalenv()
    )

    if(input$cond != "") {
      use_flg <- eval(parse(text = input$cond), envir = e)
      for(i in names(e)) {
        e[[i]] <- subset(e[[i]], use_flg)
      }
    }

    return(e)
  }))

  output$Plot <- renderPlot({...
```

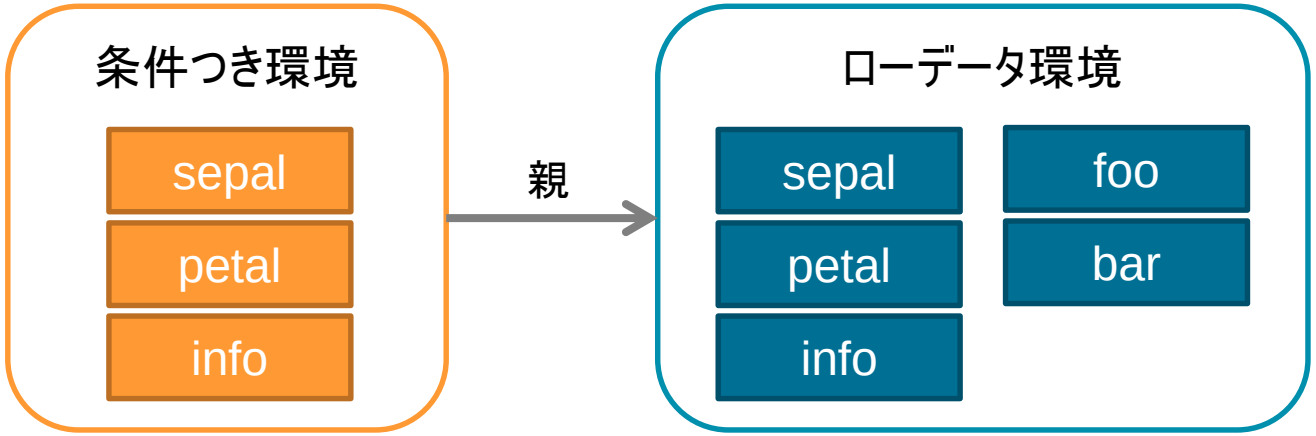
applyボタンを押すと
値が更新される関数を定義

グローバルから取得したオブジェクトを
環境オブジェクトに格納

環境内のオブジェクトに条件を
適用して返す

事例：Rで分析GUI

- ポイント
  - 複数のデータをまとめて返す「リアクティブな」関数を定義
    - UI操作に対応してデータがまるごと入れ替わる
    - プロットのコードはこの結果を受け取るだけでよい
  - このケースではリストで実装してもよい
    - 環境を使うことでオブジェクトのサーチパスを制御できる
    - 下の例では、条件つき環境でもfoo/barはローデータが見える



事例：Rで分析GUI

- まとめ
  - Shinyを用いたウェブアプリで、定型的な統計解析をUI操作で柔軟に実行
  - 多数のデータセットを柔軟に扱えるよう工夫しつつ、個々のコードをなるべく分離　⇒　スパゲティ化の回避

- こぼれ話
  - 元データの一部はExcelでしか得られず、かつRでの自動処理が困難　⇒　Excel VBAで変換処理
    - シート内にある特定のデータ領域を自動検出、うまくいかない場合は手動で領域選択して補助
    - Rで処理できる形式に変換して出力

分析プログラムの移植

- 概要
  - 既存の分析プログラムを別のツール・環境へ移植したい
- 移植のパターン
  - 商用パッケージからオープンソースへ
    - コスト削減、より詳細なコントロール、技術者層の厚さ
  - ローカルからクラウドへ
    - データがクラウドへ　⇒　分析環境も移行
    - クラウド環境で予測結果を使いたい、というパターンも

事例：SASからRへの移植

- 要件
  - SASで開発されたプログラムがあり、これをRに移植したい
  - 統計モデルの推定と予測、非線形最適化

- 結果
  - （疑似）マクロとRcppを駆使し、ほぼ同等な出力を実現
  - ただし最適化部分の実行速度に難あり
    - 本件ではSASの最適化機能が非常に優秀
      （主双対内点法による非線形最適化）

ef-prime, inc.

事例：SASからRへの移植

- 移植のポイント：マクロ
  - 元のSASプログラムがマクロを多用して書かれている
  - マクロ変数以外はコピペして実行するのと同じ
    ⇒ 「外側」の変数が書き換わる

```
fun <- function(x) {
  a <- x + a
  print(a)
}
```

```
a <- 100
fun(x = 10)
print(a)
```

aの値は100

もしfunがマクロなら…

```
a <- 100

### ここからマクロ ###
a <- 10 + a
print(a)
### ここまでマクロ ###

print(a)
```

マクロ変数xを10に書き換え

aの値は110

事例：SASからRへの移植

- Rで（疑似）マクロ
  - defmacro関数
    - 「マクロのように動作する関数」を返す
    - Lumley T. "Programmer's Niche: Macros in R", R News, 2001, Vol 1, No. 3, pp 11-13 https://www.r-project.org/doc/Rnews/Rnews_2001-3.pdf
    - gtoolsパッケージでも利用可能

マクロ変数

```
macro <- defmacro(x, expr = {
        a <- x + a
        print(a)
})
```

事例：SASからRへの移植

- 移植のポイント：（疑似）マクロ
  - defmacro関数を使うことで、R上でマクロを再現できる
    - ただしマクロが入れ子になった場合の変数のスコープなど、SASと仕様が一致するとは限らないため確認が必要

```
> fun <- function(x) {
+   a <- x + a
+   print(a)
+ }
>
> a <- 100
> fun(x = 10)
[1] 110
> print(a)
[1] 100
```

```
> macro <- defmacro(x, expr = {
+       a <- x + a
+       print(a)
+ })
>
> a <- 100
> macro(x = 10)
[1] 110
> print(a)
[1] 110
```

「外側」の値が変わった

事例：SASからRへの移植

- 移植のポイント：行ごとの処理
  - SASのDATAステップでは行単位で処理が記述される
  - 列をまとめて配列として扱うこともできるが、
    これをRでそのまま再現すると行単位のループで遅くなる

```
array a[100] a1-a100;

do i=1 to &n;
  do j=1 to &p;
    x=a[i];
    y=a[j];
    z=somefun(x, y)
  end;
end;
```

行ごとに、列a1～a100の値からなる配列aを定義

この二重ループも「行ごとに」処理される

事例：SASからRへの移植

- 移植のポイント：Rcpp
  - なるべく列単位の処理に置き換えるが、
    行のループが避けられない場合はC++に処理を任せる

```
for(int r_ = 0; r_ < n_; r_++) {

  double a[100];

  for(int i = 1; i <= 100; i++) {
      std::stringstream ss;
      ss << i;
      NumericVector tmp = df["a" + ss.str()];
      a[i] = tmp[r_];
  }
...
```

行のループ

データフレームからa1, a2… の
各列を取得し、行を指定して
配列に格納

事例：SASからRへの移植

- 移植のポイント：Rcpp
  - なるべく列単位の処理に置き換えるが、
    行のループが避けられない場合はC++に処理を任せる

```
...

  for(int i = 0; i < n[r_]; i++) {
    for(int j = 0; j < p[r_]; j++) {
      x[r_] = a[i];
      y[r_] = a[j];
      z[r_] = somefun(x[r_], y[r_])
    }
  }
}
```

行の中で二重ループ

事例：SPSS ModelerからPythonへの移植

- 概要
  - クラウドデータベースの採用に伴い、Pythonベースのクラウド分析環境に処理を移植
    - 残念ながらRは使用できない…
  - 複雑なデータ加工部分をいかに移植するかがポイント
    - IBM SPSS Modelerはデータ加工の機能が充実しており、自動的にSQLを生成して実行することができる
  - データベース上のテーブルを「あたかもデータフレームであるかのように」扱えるモジュールを開発し、これを用いて移植
    - R風に `summary()` や `getQuery()` なども作ってみた

事例：SPSS ModelerからPythonへの移植

- もしRなら…
  - dbplyrパッケージでdplyrコードをSQLに自動変換して実行
  - ローカルではdata.frameで開発し、動作を確認してから
    データベースに接続して実行、という手順も可能

ef-prime, inc.

```
# データフレーム
result <- read_csv("iris.csv") %>%
  select(sepal_length, petal_width, species) %>%
  head(100)

# データベース
result <- tbl(con, in_schema("somewhere", "iris")) %>%
  select(sepal_length, petal_width, species) %>%
  head(100)
```

違いはデータ読み込み部分のみ

おわりに（雑感）

- Rの柔軟性の高さ
  - 難しく見えても、工夫次第で色々できる懐の深さ
  - 「Rで何でもやる」のが正解とは限らないが、「わりと何でもできる」のは大きな利点

- R以外の選択肢
  - クラウドサービスでの予測分析ではPythonが優勢？
    - Google AI Platformはscikit-learnなどの機械学習モデルをアップロードして予測に利用できる
    - Microsoft AzureMLなどRと親和性が高いものもある
  - 「データがどこにあるか」によって決まる部分も大きい
    - 複数のツールを扱えるほうが望ましい

おわりに（雑感）

- データ分析とプログラミング
  - データ分析においてプログラミングは重要なツール
    - R：強力なプログラミング環境を持つ統計ソフトウェア
    - Python：強力な機械学習ライブラリを持つプログラム言語
    - でも「プログラミングより分析に集中したい」ときがある？
  - 「データから情報を引き出したい人」は誰？
    - 統計家、データサイエンティスト、ソフトウェアエンジニアの
      いずれでもない人達の存在
    - 「みんなが学ぶべき」なのか、「もっと簡単に」できるべきか。
      あるいは「チームでやる」

ご清聴ありがとうございました

ef-prime, inc.

R AnalyticFlow

https://r.analyticflow.com

@ef-prime_jp　R AnalyticFlow