2017.11.25.SAT

R AnalyticFlowによる予測分析

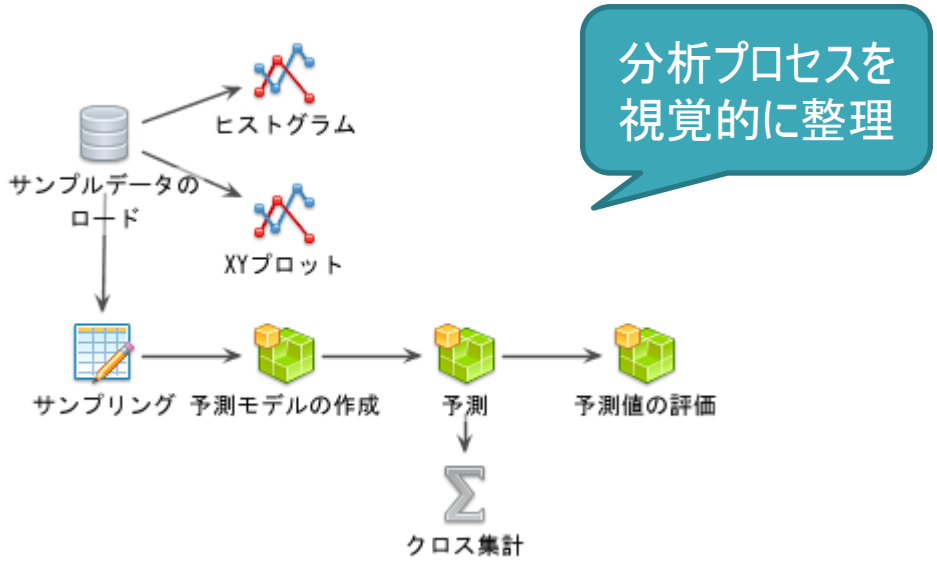
株式会社ef-prime

鈴木 了太

ef-prime, inc.

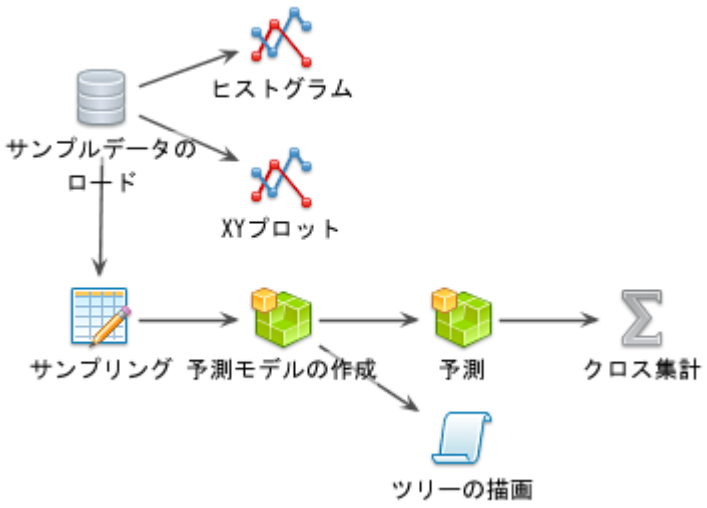
R AnalyticFlowとは

- 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
  - Javaで開発、マルチOS対応
    - Windows / Mac / Linux



R AnalyticFlowとは

分析フローを作成し、対応するRスクリプトを生成・実行

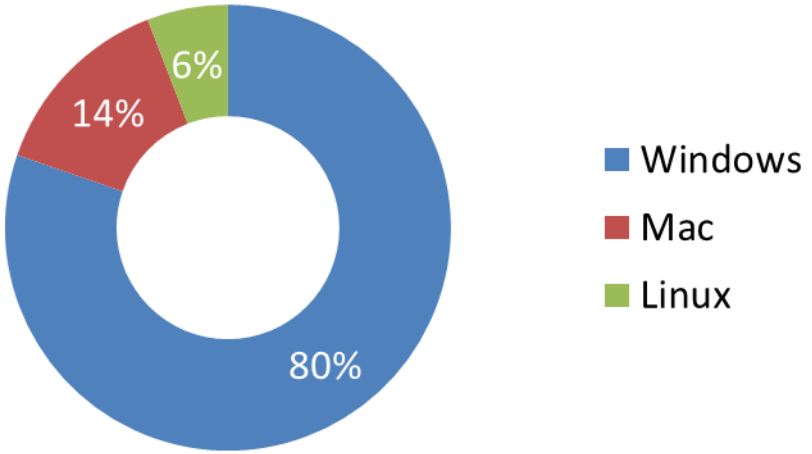



```r
# 1. データの読み込み
data(iris)

# 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)
```

多言語・マルチOS

表示言語として日本語と英語を選択可能
世界101ヶ国でダウンロード（2017年11月時点、直近1年間）
Windows / Mac / Linux をサポート

1. Japan
2. United States
3. India
4. United Kingdom
5. Peru
6. China
7. Taiwan
8. Germany
9. Italy
10. Brazil



想定するユーザー層

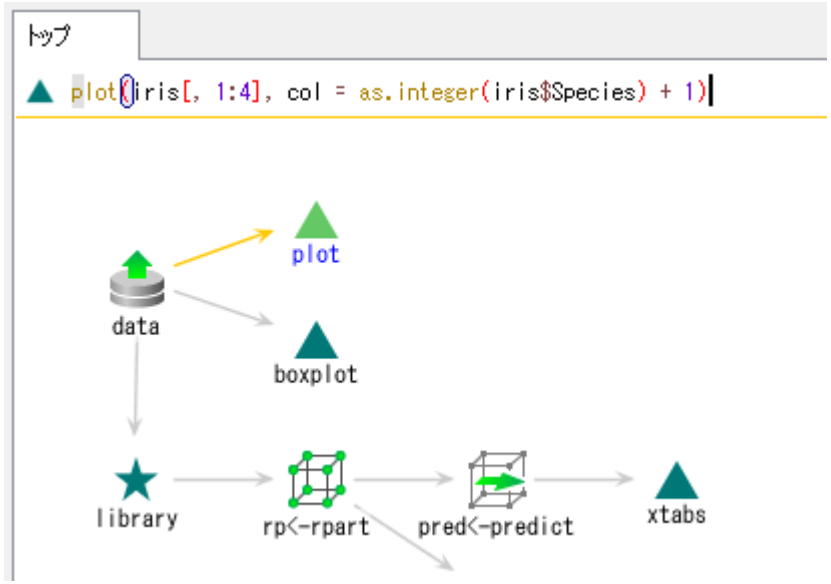
異なったスキルセットを持つユーザーを想定し、
ユーザー間の共同作業をサポートするように設計

ユーザー層	想定する特徴
エキスパート	Rの関数やライブラリを作成し、 新しい手法を開発することができる
パワーユーザー	Rスクリプトを記述し、 既存の分析手法を適用することができる
ライトユーザー	データ分析の概念は理解しているが、 Rに関する知識はない

ef-prime, inc.

当初の想定利用スタイル

補完機能などのサポートを利用しつつ、ユーザーがRスクリプトを直接記述することで分析フローを組み立てる形式



当初の想定利用スタイル

Rのスキルを持つユーザーが分析プロセスを組み立て、
ライトユーザーはそれを実行するのみ

ユーザー層	利用スタイル
エキスパート	視覚的に整理しながらRスクリプトを記述。 視覚化されたフローはユーザー間で共有しやすく、 ライトユーザーに処理を任せることもできる
パワーユーザー	
ライトユーザー	作成済みの分析フローをマウス操作で簡単に実行 （ファイル名など限られた部分のみ書き換え）

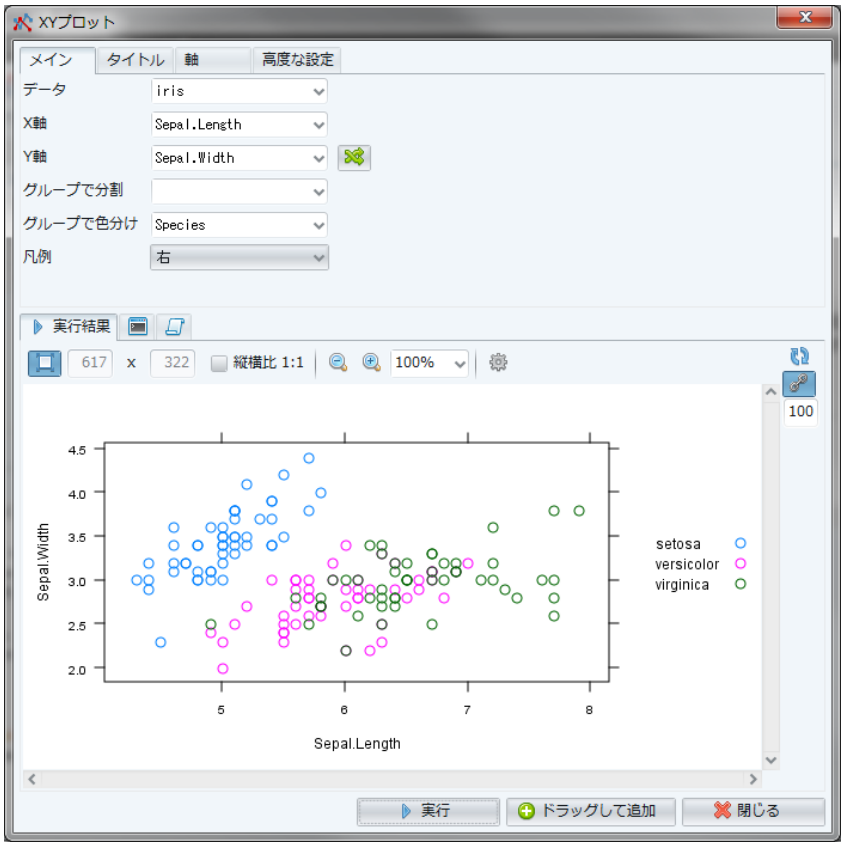
最近の改善点（バージョン3.x）

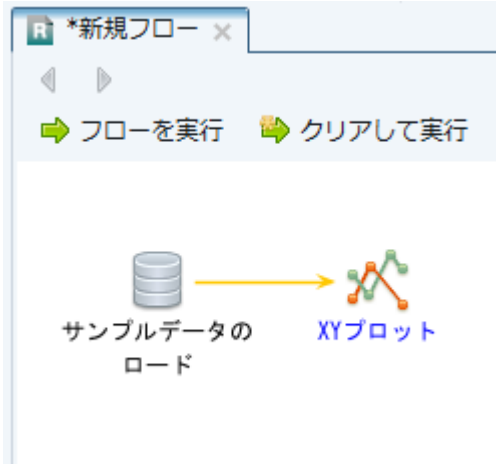
Rのスキルを持つユーザーについては利便性を高め、
ライトユーザーも自ら分析ができるように

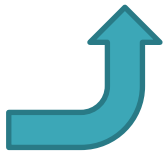
ユーザー層	改善点
エキスパート	関数やライブラリを開発するならIDEのほうが便利なので、併用できるようにしたい
パワーユーザー	よくある処理はGUIで完結させ、スクリプトの記述量を減らしたい
ライトユーザー	Rの知識がなくても自分で分析を設計し、実行できるようにしたい

充実した分析GUI

簡単なマウス操作でRスクリプトを記述せずに分析が可能。
Rの知識が必須ではなくなり、スクリプトの記述量も削減



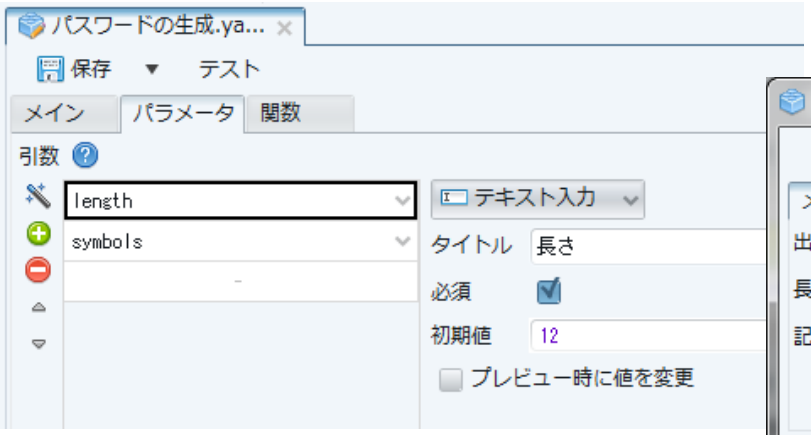


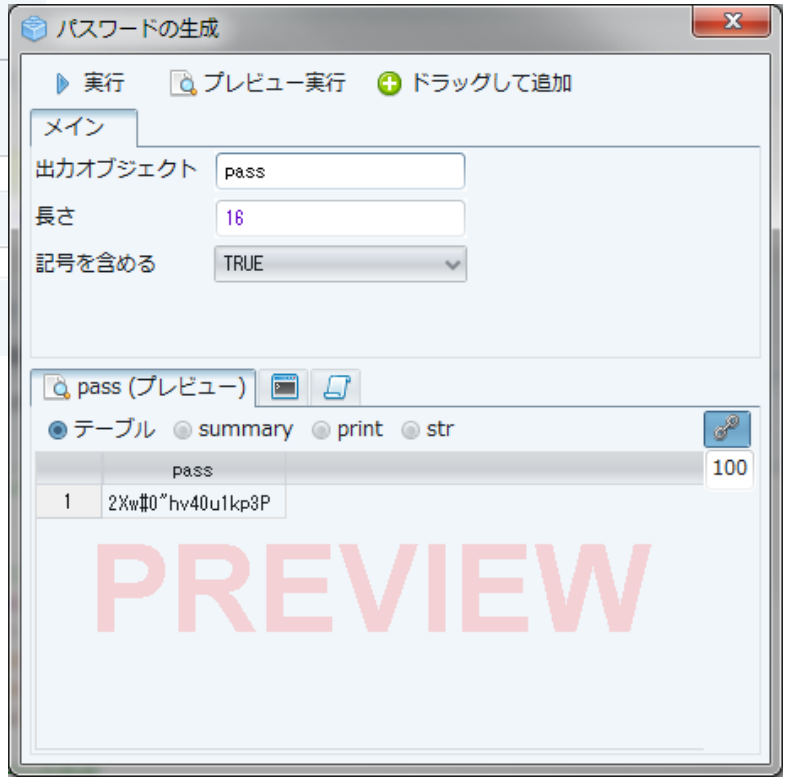


ドラッグ＆ドロップ

カスタムUIの作成

組み込みのGUIに存在しない機能を追加できる。
Javaプログラミングを行わず、ビルダーを使って簡単に作成可能

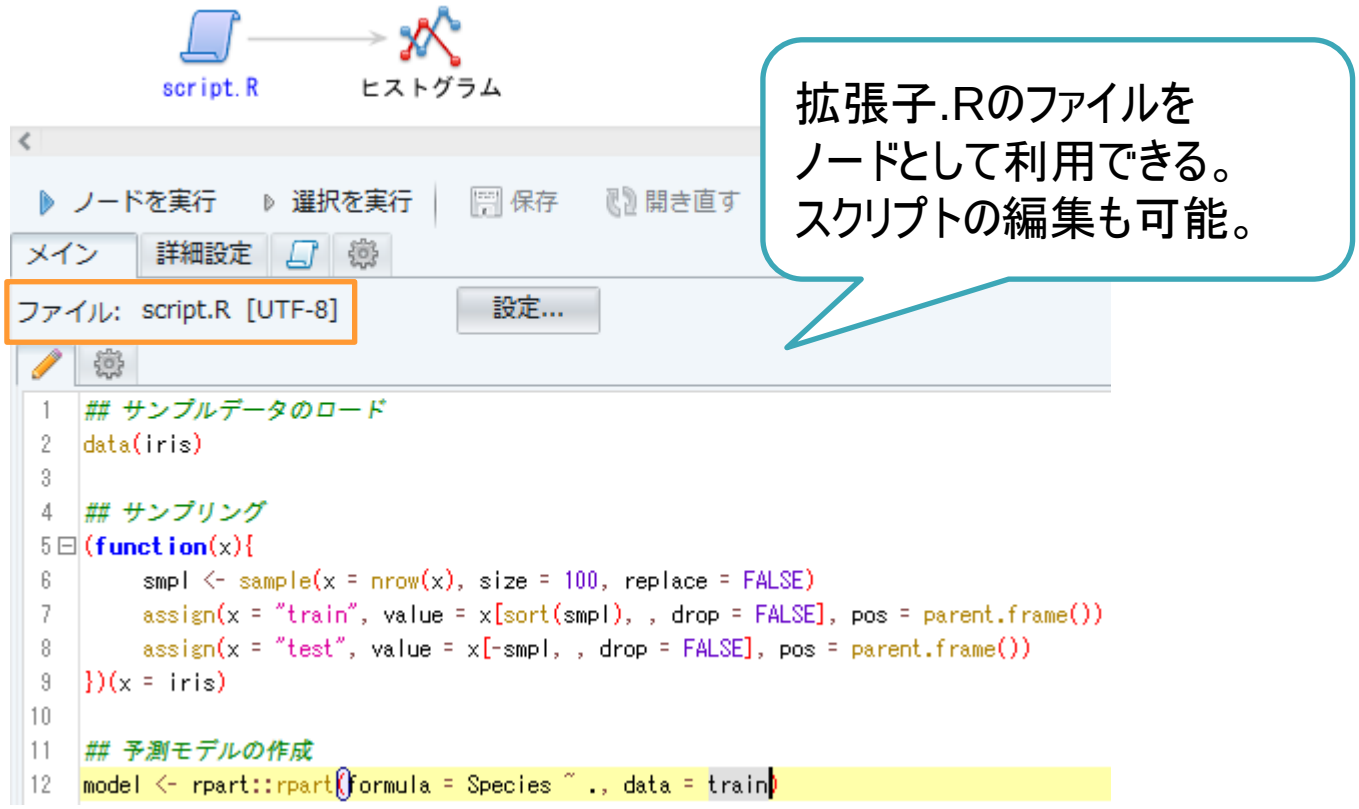




ビルダー上で関数（自作も可能）を指定し、
引数を設定用UIなどを指定して保存

外部ツールとの連携

IDEが使用するフォルダをプロジェクトフォルダとして設定し、
Rスクリプトファイルをノードとして利用可能

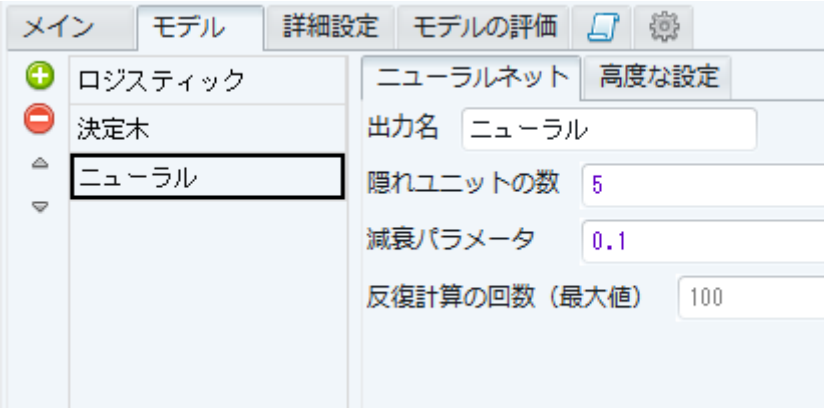


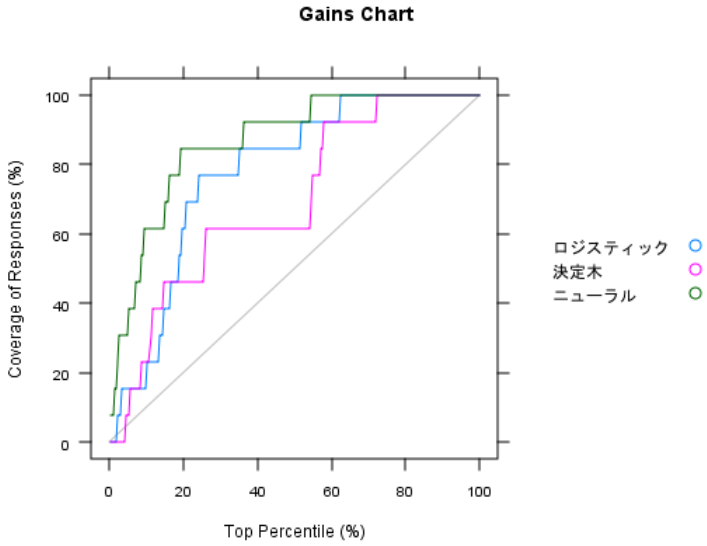
これまでの歩み

時期	バージョン	詳細
2007年夏ごろ	プロトタイプ	構想・開発着手
2007年12月	0.1.0	プレビュー版公開（Windowsのみ）
2008年3月	0.3.0	ユーザビリティの向上、ファイルエクスプローラ
2009年12月	1.0.0	オブジェクトブラウザ、ボックス、キャッシュ、コードエディタ
2012年12月	2.0.0	タブインターフェースへの統合、デバッグ補助、自動バックアップ、64bit対応
2015年12月	3.0.0	分析機能の完全GUI化、新インターフェース、プロジェクト管理、カスタムUI作成機能
2017年6月	3.1.0	分析機能の強化（予測分析、多変量解析、仮説検定など）、利便性の向上、安定性の向上（Rの別プロセス化）

予測分析のサポート

複数の予測手法やパラメータ設定のもとで予測モデルを作成し、
予測精度を評価して出力





	Accuracy
ニューラル	0.9409449
ロジスティック	0.9251969
決定木	0.9133858

Predict x Actual

	0	1
0	235	9
1	6	4

ここまではリリース済み（3.1.0～）。
次ページ以降が現在の取り組み

問題点

コードが複雑化し、もはや読むことが困難なケースも多数。
サポートしている予測手法や評価手法も限られる

```
## 予測モデルの作成
(function(){
    try_ <- function(expr, modelName) {
        eval(bquote(tryCatch(expr, error = function(e) {
            cl <- conditionCall(e)
            cl <- if (is.null(cl)) "" else deparse(cl, nlines = 1)
            stop(.(modelName), " ", cl, ": ", conditionMessage(e), call. = FALSE)
        })))
    }
    prune_ <- function(model) {
        if (all(c("xerror", "xstd") %in% colnames(model$cptable))) {
            rpart::prune(model, cp = model$cptable[which.min(model$cptable[, "xerror"] + model$cptable[, "xstd"]), "CP"])
        } else {
            warning("Could not prune the tree without cross validation result")
            model
        }
    }
    nnet_ <- function(formula, data, subset = NULL, na.action, ...) {
        mf <- match.call(expand.dots = FALSE)
        mf[[1]] <- quote(stats::model.frame)
        mf$formula <- update(terms(formula, data = data), . ~ 1)
        mf$... <- NULL
        y <- model.response(eval(mf))
        cl <- match.call()
        cl[[1]] <- quote(nnet::nnet)
```

予測分析のコード例。この例では3つのモデル作成と評価のため、1ノードで133行を生成

複雑化の理由（抜粋）

- 機能の共通化
  - 関数によって機能が微妙に異なり、カスタマイズが必要
    - 例：ある変数 z をモデルから除外するにはモデル式を `y ~ . - z` のように記述すればよいが、z が単一の値しか取らない factor のとき `nnet::nnet()` はエラーとなる。

- 安全性の担保
  - 既存オブジェクトの変更・破壊などを防ぐため、`local()` や無名関数を活用している。
    - これは安全である一方、コードの複雑化を招く

```
# iris を学習用と検証用に分割する
(function(x){
    smpl <- sample(x = nrow(x), size = nrow(x) / 100 * 50, replace = FALSE)
    assign(x = "train", value = x[sort(smpl), , drop = FALSE], pos = parent.frame())
    assign(x = "test", value = x[-smpl, , drop = FALSE], pos = parent.frame())
})(x = iris)
```

解決方法の案

- 処理の関数化
  - 共通化した機能を提供する関数を作成する
  - 既存オブジェクト等に関する問題も回避できる

```
# データ x をふたつのデータフレームに分割する関数
sample.data <- function(x, p = 0.5, train = "train", test = "test"){
     smpl <- sample(x = nrow(x), size = nrow(x) * p, replace = FALSE)
     assign(x = train, value = x[sort(smpl), , drop = FALSE], pos = parent.frame())
     assign(x = test, value = x[-smpl, , drop = FALSE], pos = parent.frame())
}

sample.data(iris, p = 0.5)
```

- パッケージの利用
  - 上記のように作成した関数をパッケージ化したり、類似の機能を提供するパッケージを利用する
    - データ加工は `dplyr`、予測モデルは `caret` など

参考：caret パッケージ

予測分析を効率化するパッケージ。さまざまなパッケージが
提供する200以上の手法を統一的なインターフェースで利用でき、
予測精度の評価やパラメータチューニングも可能

```
## caret による予測モデリングの例
result <- caret::train(medv ~ .,
        data = MASS::Boston,
        # glmnet パッケージの ElasticNet を使用
        method = "glmnet",
        # 5-fold クロスバリデーション
        trControl = caret::trainControl(
                method = "cv", number = 5),
        # α と λ の値を指定してグリッドサーチ
        tuneGrid = expand.grid(
                alpha = seq(0, 1, by = .1),
                lambda = .1^(1:5))
)
```

パッケージに関する従来の方針

- 生成されるコードが「どこでも確実に動く」
  - 標準でRに同梱されるパッケージのみを使用
    - base または recommended
    - それ以外のパッケージはインストールされていないと動かず、ポータビリティを損なうため。
  - 上記のパッケージであっても、名前空間を指定して `package::function()` のように呼び出す
    - `library()` などでロードしたとき特定の関数がマスクされ、コードの挙動が変わってしまう可能性をなるべく避けるため

実装する機能が複雑化し、限界を迎えつつある

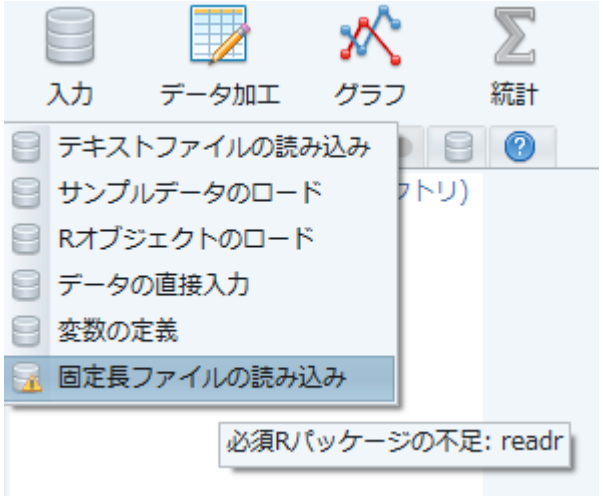
パッケージに関する新しい方針

基本方針は
変えない！

- 生成されるコードが「どこでも確実に動く」
  - 必要なパッケージを検出して通知
  - パッケージのインストールと管理をサポート
  - 必要であれば事前にパッケージをロード

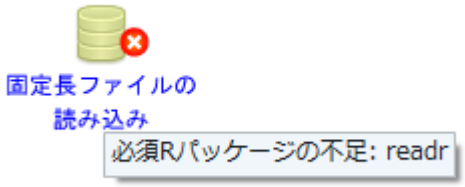
必要なパッケージを検出して通知

- メニューでの表示
  - 分析機能を呼び出す前にパッケージの有無をチェック



必要なパッケージを検出して通知

- 分析フロー上での表示
  - ノードごとに必要なパッケージを管理し、実行前に通知

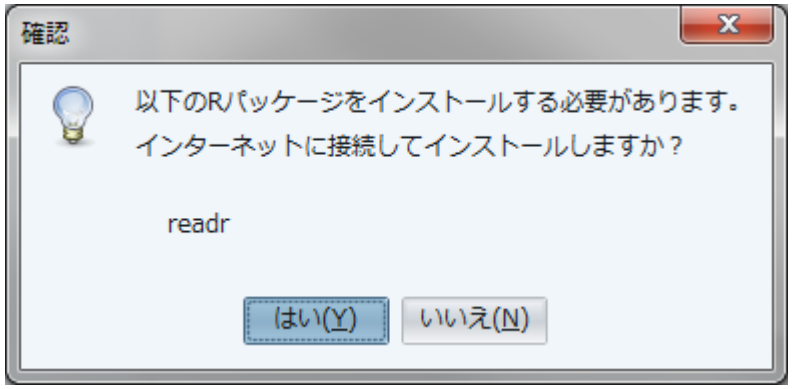


- スクリプトのエクスポート
  - 分析フローをRスクリプトに出力する際、パッケージをチェックするコードを含めるオプションを作成。
  - 必須パッケージがない場合、実行後すぐにエラーで停止

```
Error: Missing required package: readr. Try:
   install.packages("readr")
```

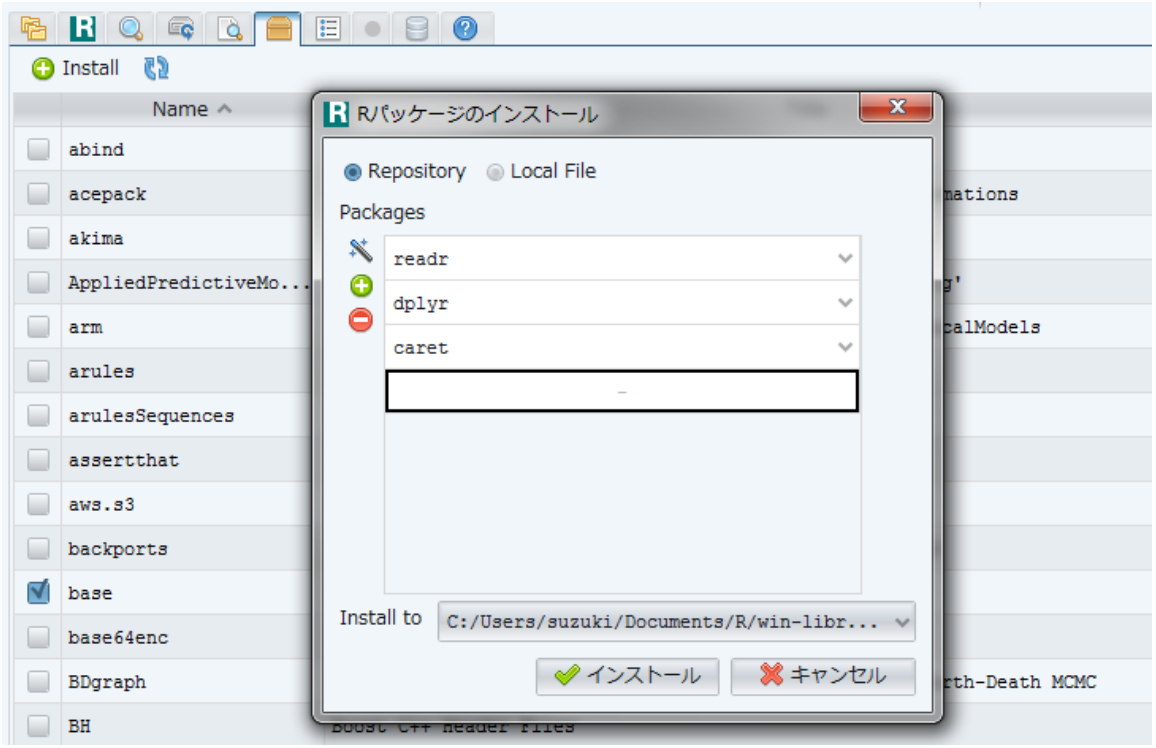
パッケージのインストールと管理をサポート

- インストールの提案
  - 不足するパッケージがあればダイアログを表示し、ワンクリックでインストール
    - 標準のCRANミラーを設定してあり、初心者にとっては選択に困るミラーサイトの選択も省略



パッケージのインストールと管理をサポート

- パッケージマネージャ
  - Rスクリプトを使わずにパッケージのインストールなどの管理を行うための仕組みを提供

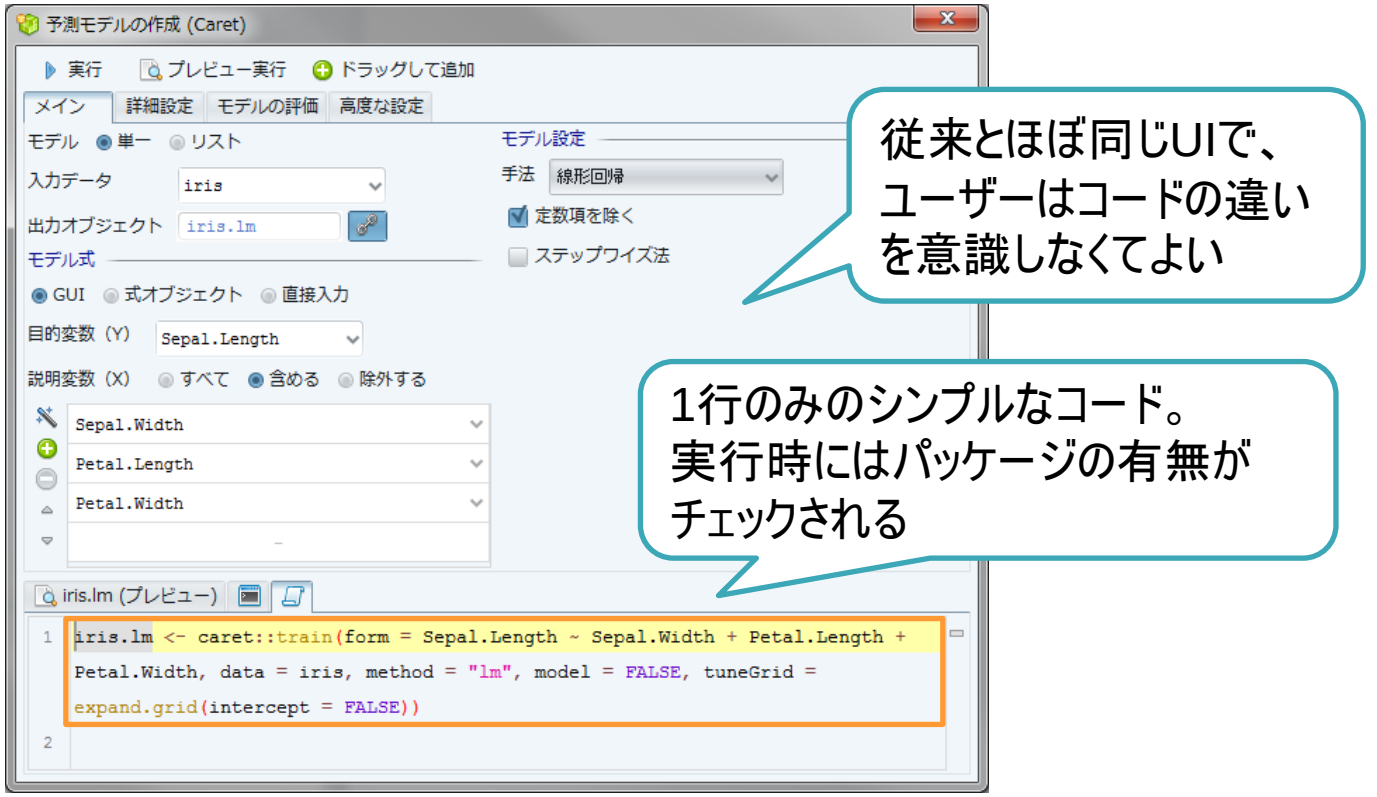


必要であれば事前にパッケージをロード

- パッケージの自動ロード
  - 必要がある分析機能に限り、実行時に自動的にパッケージをロードするコードを生成
    - `library()` または `require()`
  - 関数をマスクするなどの意図しない副作用を起こす可能性があるため、最低限の利用とする方向で検討
    - 基本的には従来通り `package::function()`

開発例：caret による予測分析

caret パッケージによる予測分析機能の開発例（プロトタイプ）。
lm() と同じインターフェースで caret 用のコードを生成しており、
ブートストラップやクロスバリデーションによる精度評価が行われる



今後について

- 開発状況
  - パッケージの通知、管理の仕組みを開発中
  - パッケージを利用した分析機能の開発は試行中
    - 例：`caret` でしあわせになれるか？自作するべきか？
      - 多数のパッケージを統一して使う仕組みのためか、
        個々の実装には不備もちらほら
      - `library()` でロードしないと動かない機能も
      - ただしGitHubでのプルリクエストへの対応は早く、
        カスタマイズの仕組みもあるので試す価値はありそう
  - Java + R な開発者の方、ご興味ありましたらご一報を！

ef-prime, inc.

ご清聴ありがとうございました

ef-prime, inc.

R AnalyticFlow

http://r.analyticflow.com

@ef-prime_jp

R AnalyticFlow