R でプログラミング

-R 入門書に書いていないプログラミングの常識-

名古屋大学大学院 生命農学研究科
森林生態生理学研究分野

安部 晃生

プログラミング

- コンピュータに対して処理を委譲
  - 反復処理
  - 演算速度
- プログラミング言語を用いてプログラム (指示書) を作成
  - 誰がプログラムを読み書きするのか？
    - 書く: **人間**
    - 読む: コンピューター, **人間**

人間の人間による人間のためのコーディング

- なぜプログラムは難しく見えるのか？
  - 難しく見えるように書いている
- 難しくないように見えるプログラムとは？
  - 読みやすい (可読性)
  - 見やすい (視認性)
- 人間が読むことを意識したプログラムを書くことが大事
  - プロトコールは自分だけがわかれば良いのか？

可読性 (intelligibility)

- ソースコードを読むことにより，プログラムの目的･処理内容が容易に理解できること
  - 自然な処理の流れ
  - 適切な命名 (変数名，関数名など)
  - コメント
- 可読性の低いソースコードの問題
  - 理解するための時間コスト
  - バグ発見の遅れ

可読性
処理の流れの例

```
three()
five()
two()
one()
four()
```

```
one()
two()
three()
four()
five()
```

わかりやすいのはどっち？

ただし，処理の順番に依存性がある場合は注意

可読性
命名の例

```
function (fp, rp, c)
{
   tm_f <- tm(fp, c)
   tm_r <- tm(rp, c)
   return abs(tm_f - tm_r)
}
```

わかりやすいのは
どっち？

```
function (primer.forward, primer.reverse, condition)
{
   tm.forward <- calculate.tm(primer.forward, condition)
   tm.reverse <- calculate.tm(primer.reverse, condition)
   return abs(tm.forward - tm.reverse)
}
```

可読性
コメントの例

```
cleanup <- function(file) {
    ...
}
```

わかりやすいのは
どっち？

```
# 一時データを保存したファイルをクリーンアップします。
# クリーンアップされたファイルはサイズが 0 になりますが
# 削除されずに残ります。
# ファイルを削除するには base::unlink 関数を使用します。
cleanup <- function(file) {
    ...
}
```

視認性 (visibility)

- ソースコードが文字・単語・ブロック単位ではっきりと読めること
  - インデント
  - 空行 (段落化)
  - 等幅フォント
- 視認性の低いソースコードの問題
  - バグ発見の遅れ
  - 眼精疲労

視認性
例

```
quicksort <- function(vector) {
    if(length(vector) < 2) {
        return(vector)
    }
    if(length(vector) == 2) {
        return(
            ifelse(vector[1] <= vector[2], vector, vector[2:1])
        )
    }

    pivot <- sample(vector, 1)
    parity <- vector[vector == pivot]
    less <- quicksort(vector[vector < pivot])
    greater <- quicksort(vector[vector > pivot])
    return(c(less, parity, greater))
}
```

視認性
例

```
quicksort<-function(vector){
if(length(vector)<2)return(vector)
if(length(vector)==2)return(ifelse(vector[1]<=vector[2],vector,vector[2:1]))
pivot<-sample(vector,1);parity<-vector[vector==pivot];
less<-quicksort(vector[vector<pivot]);greater<-quicksort(vector[vector>pivot]);
return(c(less,parity,greater))
}
```

わかりやすいのはどっち？

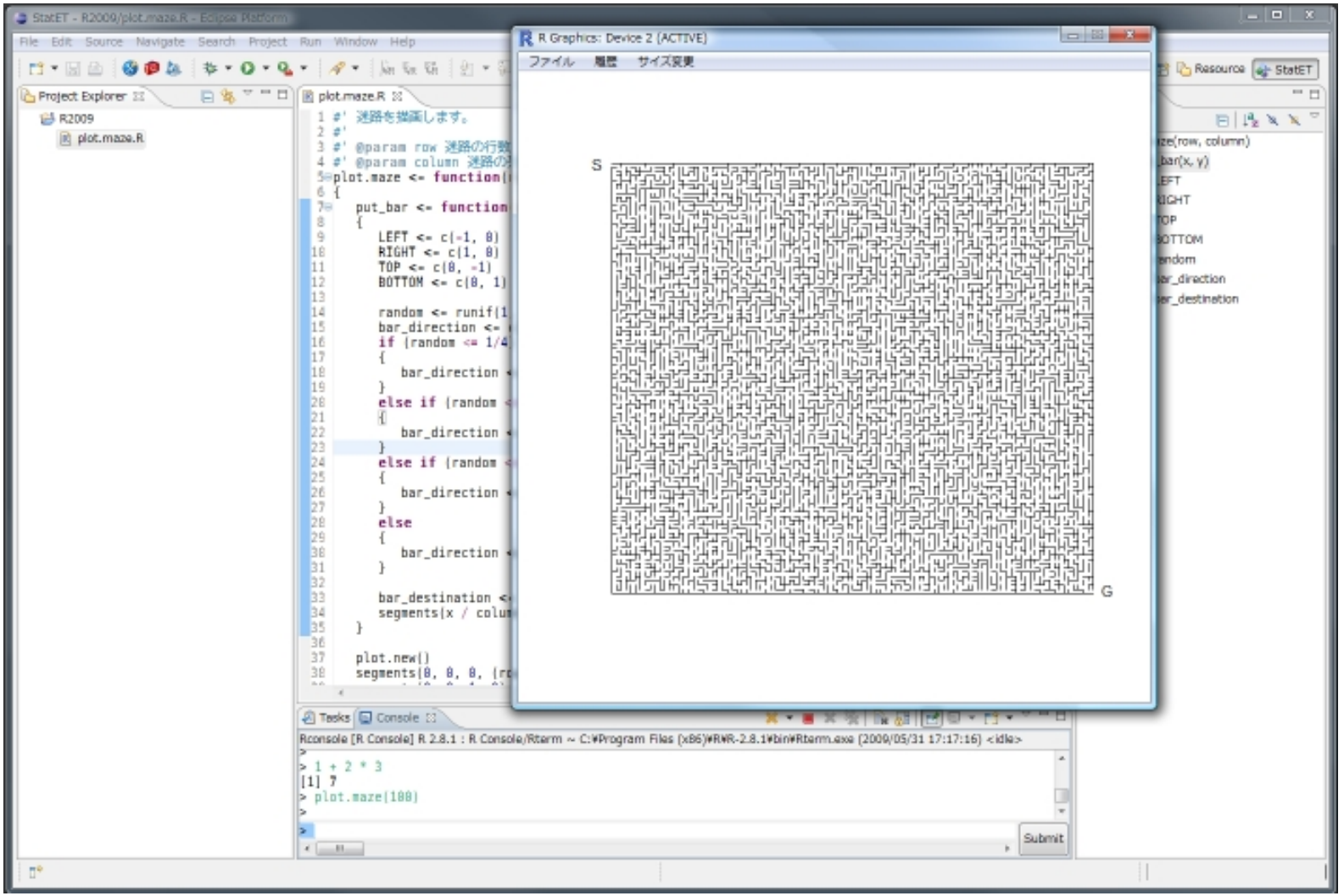
エディタ

- エディタの機能
  - タブ切り替え (もしくはマルチバッファ)
  - (複数ファイルの) 正規表現検索･置換
  - オートインデント
  - シンタックスハイライト
  - コードフォールディング
  - アウトライン表示
  - 単語補完
  - 実行環境 (外部アプリケーションとの連携)

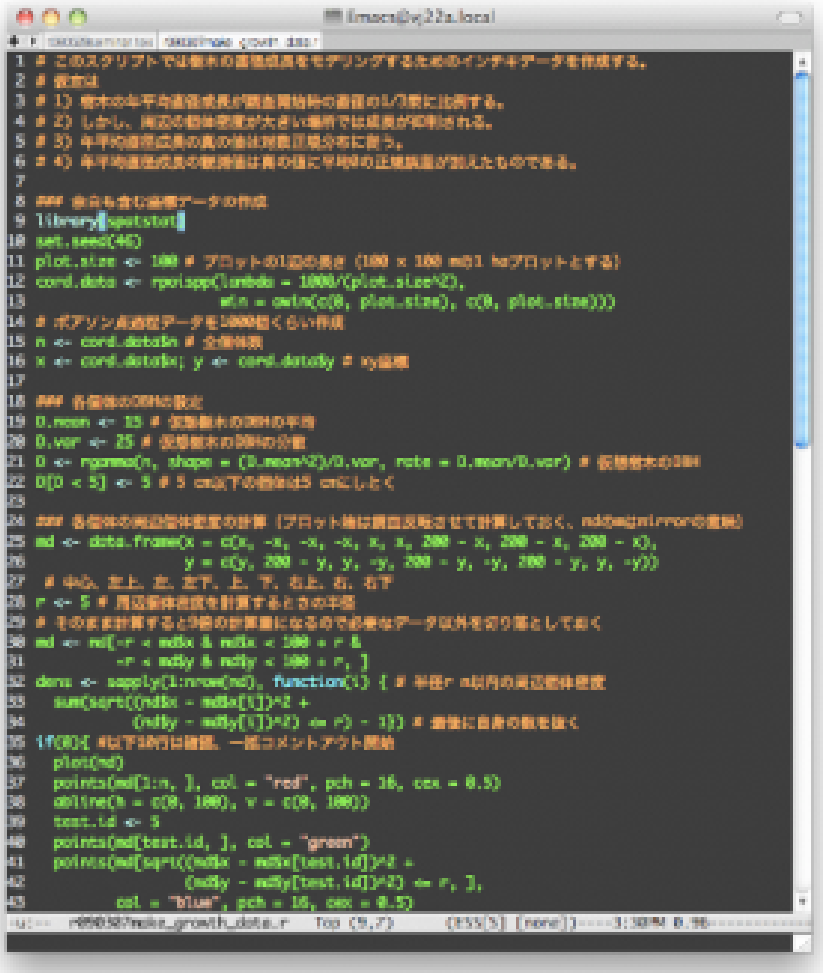
R で使われるエディタ

- Windows のメモ帳はメモのためのアプリケーションであって文書を書くためのアプリケーションではない！
- 目的にあったエディタを用いる
  - RjpWiki の「エディタでR」を参照
- 自分の好みのエディタを見つける
  - 投票所によると Emacs や秀丸あたりが人気らしい
  - ちなみに発表者は Eclipse が好きです

Eclipse + StatET



Emacs



(玉木さん提供)

参考すると良いウェブ文献

- コーディング技法 (http://msdn.microsoft.com/ja-jp/library/aa291593.aspx)
- 頑健なJavaプログラムの書き方 (http://www.alles.or.jp/~torutk/oojava/codingStandard/writingrobustjavacode.html)
- 命名規則 (http://ja.wikipedia.org/wiki/命名規則_(プログラミング))
- 字下げスタイル (http://ja.wikipedia.org/wiki/字下げスタイル)
- プログラミング作法 (http://ja.wikipedia.org/wiki/プログラミング作法)

今後 R を習得する上で役に立つ項目

- 一貫性
- リファクタリング
- 単体テスト
  - ➔ R では RUnit というパッケージが存在
- バージョン管理システム
- バグ管理システム
  - ➔ バグ, デバッグ
- その他