R

重回帰分析

サンプルデータとスクリプト

・サンプルデータ：NICER 1.0 より学習者のエッセイデータ 287 個
・Criterion によるスコア、延べ語数、異なり語数、文数、TTR、ギローインデックス、平均単語長、平均文長、MATTR を各エッセイの特徴量として抽出
・ファイル名とともに結果をテキストファイルに保存するスクリプト

1.NICER1.0 を解凍したフォルダーの中の NICER_NNS に作業ディレクトリーを移動
2. スクリプトを実行
3. 結果を保存するファイルを作業ディレクトリーの外にファイル名をつけて「保存（作成）」
例：jpn4.txt
4.R の中に読み込む

```
> jpn4 <- read.table(choose.files())
> class(jpn4)
[1] "data.frame"
> head(jpn4)
          V1 V2  V3  V4 V5        V6       V7        V8       V9      V10
1 JPN501.txt  4 319 135 30 0.4231975 7.558549 0.5921317 4.304075 10.63333
2 JPN502.txt  4 356 161 29 0.4522472 8.532983 0.6649157 4.233146 12.27586
3 JPN503.txt  3 201 121 13 0.6019900 8.534682 0.7170149 4.746269 15.46154
4 JPN504.txt  4 260 140 27 0.5384615 8.682431 0.6877692 4.761538  9.62963
5 JPN505.txt  4 420 175 25 0.4166667 8.539126 0.6341905 3.995238 16.80000
6 JPN506.txt  3 261 124 20 0.4750958 7.675407 0.6390038 4.072797 13.05000
```

前処理

■見出しをつける

```
> names(jpn4) <- c("file", "Score", "Token", "Type", "NoS", "TTR", "GI", "MATTR", "AWL", "ASL")
> head(jpn4)
        file Score Token Type NoS       TTR       GI     MATTR      AWL      ASL
1 JPN501.txt     4   319  135  30 0.4231975 7.558549 0.5921317 4.304075 10.63333
2 JPN502.txt     4   356  161  29 0.4522472 8.532983 0.6649157 4.233146 12.27586
3 JPN503.txt     3   201  121  13 0.6019900 8.534682 0.7170149 4.746269 15.46154
4 JPN504.txt     4   260  140  27 0.5384615 8.682431 0.6877692 4.761538  9.62963
5 JPN505.txt     4   420  175  25 0.4166667 8.539126 0.6341905 3.995238 16.80000
6 JPN506.txt     3   261  124  20 0.4750958 7.675407 0.6390038 4.072797 13.05000
```

■データ数の確認

■欠損値の処理

・欠損値を調べる
・欠損値がいくつかるか調べる

```
> sum(is.na(jpn4))
[1] 2
```

・欠損値を除いたデータにする

```
> jpn4.b <- na.omit(jpn4)
[1] 285
```

■分析対象のオブジェクトをで「固定」しておくと便利

・jpn4.b について、各カラムを単位に操作する

```
> attach(jpn4.b)
```

・こうしておけば、いちいち jpn4.b$Score としなくても Score だけでよい。
・

重回帰分析 lm()

■ lm(**従属変数 ~ 独立変数１ + 独立変数２ + 独立変数３** + ...)

```
> lm(Score ~ Token + Type + NoS + TTR + GI + MATTR + AWL + ASL)
```

・オブジェクトを、attach() してなくても、data= でオブジェクトを指定てもよい

```
> lm(Score ~ Token + Type + NoS + TTR + GI + MATTR + AWL + ASL, data=jpn4.b)
```

・データの中の一つを従属変数にして、残りはすべて独立変数として全部入れるときは、省略して書ける

```
lm( 従属変数 ~ ., data= データ名 )
```

```
Call:
lm(formula = Score ~ Token + Type + NoS + TTR + GI + MATTR + AWL + ASL)

Coefficients:
 (Intercept)         Token          Type           NoS           TTR            GI        MATTR
AWL           ASL
    -0.145765      0.004008     -0.021620      0.003506     -9.147468      1.004200     -1.306580
0.556470     0.021069
```

・結果を詳しく見るには、一旦保存して、その summary() を見る。

```
> jukaiki <- lm(Score ~ Token + Type + NoS + TTR + GI + MATTR + AWL + ASL)
> summary(jukaiki)


Call:
lm(formula = Score ~ Token + Type + NoS + TTR + GI + MATTR +
    AWL + ASL)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9080 -0.2508  0.0035  0.2581  0.7614

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.145765   0.642548  -0.227 0.820705
Token        0.004008   0.002144   1.870 0.062603 .
Type        -0.021620   0.010421  -2.075 0.038953 *
NoS          0.003506   0.014089   0.249 0.803665
TTR         -9.147468   1.328107  -6.888 3.83e-11 ***
GI           1.004200   0.257697   3.897 0.000122 ***
MATTR       -1.306580   1.362782  -0.959 0.338519
AWL          0.556470   0.064559   8.620 5.32e-16 ***
```

```
ASL          0.021069   0.025071   0.840 0.401424
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.334 on 276 degrees of freedom
Multiple R-squared:  0.8145,    Adjusted R-squared:  0.8091
F-statistic: 151.5 on 8 and 276 DF,  p-value: < 2.2e-16
```

- Estimate が各変数の係数
- t 値とその右の p 値は、「それぞれの変数の係数が０（ゼロ）、つまり、その変数の影響はない」という帰無仮説が起きる確率
  - 「影響がない」確率が 5% とか 1% 未満で低ければ、そういうことは起きえない、つまり「影響がないとは言えない」という話になる。
  - というわけで、有意（星がついている）であれば、その変数は影響力がある、と考えられえる

■標準化

・変数の規模・単位が違うので、影響を等しく比べられるように「標準化（正規化）」する。

https://scrapbox.io/nishio/%E6%AD%A3%E8%A6%8F%E5%8C%96%E3%81%A8%E6%A8%99%E6%BA%96%E5%8C%96

関数は scale()

ただし、データは数値でないといけないので、まず、一番左にあるファイル名のカラムを除く。

```
> jpn4.c <- jpn4.b[,-1]
```

・様子を見てみる

```
> pairs(jpn4.c)
```

・標準化する

```
> jpn4.z <- scale(jpn4.c)
```

・型が matrix になっているので、データフレームに変換

```
> jpn4.z <- as.data.frame(jpn4.z)
```

・これをもとに、改めて、重回帰分析

```
> attach(jpn4.z)
> jukaiki.z <- lm(Score ~ Token + Type + NoS + TTR + GI + MATTR + AWL + ASL)
> summary(jukaiki.z)

Call:
lm(formula = Score ~ Token + Type + NoS + TTR + GI + MATTR +
    AWL + ASL)

Residuals:
     Min       1Q   Median       3Q      Max
-1.18748 -0.32797  0.00458  0.33760  0.99580

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.587e-17  2.588e-02   0.000 1.000000
Token        5.179e-01  2.770e-01   1.870 0.062603 .
Type        -9.364e-01  4.514e-01  -2.075 0.038953 *
NoS          3.299e-02  1.326e-01   0.249 0.803665
```

```
TTR         -7.898e-01  1.147e-01  -6.888 3.83e-11 ***
GI           1.282e+00  3.290e-01   3.897 0.000122 ***
MATTR       -7.775e-02  8.110e-02  -0.959 0.338519
AWL          2.423e-01  2.811e-02   8.620 5.32e-16 ***
ASL          7.988e-02  9.505e-02   0.840 0.401424
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4369 on 276 degrees of freedom
Multiple R-squared:  0.8145,    Adjusted R-squared:  0.8091
F-statistic: 151.5 on 8 and 276 DF,  p-value: < 2.2e-16
```

■ VIF

・ところが、独立変数間でお互いに相関が高いものがある。
  ・複数の相関が高いものを重ねて入れてしまうと、その影響が重複して評価されてしまう。
  ・VIF(Variance Inflation factor) をチェック（10 以上はダメ、ほとんど同じもの）
  ・パッケージ car をインストールする

```
> library(car)
```

・関数 vif() の実行

```
> vif(jukaiki.z)
    Token      Type       NoS       TTR        GI     MATTR       AWL       ASL
114.203516 303.150320  26.150429  19.565310 161.087442   9.787097   1.175870  13.443703
```

10 以上は外すべき。2 以下が望ましい。

・しかし、一つをはずすと、それに関連したものが変動して、値も変わるので、一概に 10 以上はすべてダメというわけではない。何を選ぶかは、判断による。で、やってみる。

・VIF の計算は自分でやってもよい（VIF のページ参照）

```
VIF = 1/(1 - 相関係数 ^2)
```

■変数選択 関数

```
> jukaiki.z.result <- step(jukaiki.z)
> summary(jukaiki.z.result)

Call:
lm(formula = Score ~ Token + Type + TTR + GI + AWL + ASL)

Residuals:
     Min       1Q   Median       3Q      Max
-1.17036 -0.34503 -0.00159  0.34247  0.98419

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.508e-17  2.583e-02   0.000   1.0000
Token        4.732e-01  2.183e-01   2.167   0.0311 *
Type        -7.950e-01  4.082e-01  -1.948   0.0525 .
TTR         -8.081e-01  1.057e-01  -7.648 3.36e-13 ***
GI           1.145e+00  2.709e-01   4.227 3.22e-05 ***
AWL          2.389e-01  2.768e-02   8.630 4.83e-16 ***
ASL          5.048e-02  2.918e-02   1.730   0.0848 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4361 on 278 degrees of freedom
Multiple R-squared:  0.8138,    Adjusted R-squared:  0.8098
F-statistic: 202.6 on 6 and 278 DF,  p-value: < 2.2e-16
```

・Token + Type + TTR + GI + AWL + ASL が変数の候補として残った。
　・ただし、Type と ASL は重要度が低い
・ここで、また、VIF を見てみる。

```
> vif(jukaiki.z.result)
    Token      Type       TTR        GI       AWL       ASL
71.198905 248.805646  16.675124 109.594573   1.144067   1.271753
```

・欠損値のためエラーが起きる場合は、欠損値を除いておく na.omit(データフレーム)

■試行錯誤

・Type と ASL を除いて、また、step() してみて、VIF を見てみる、、、という風に試行錯誤していく。

```
> jukaiki.z2 <- lm(Score ~ Token + TTR + GI + AWL)
> vif(jukaiki.z2)
    Token       TTR        GI       AWL
12.765221 10.876050  8.233309  1.081001
> jukaiki.z2.result <- step(jukaiki.z2)
> summary(jukaiki.z2.result)

Call:
lm(formula = Score ~ TTR + GI + AWL)

Residuals:
     Min       1Q   Median       3Q      Max
-1.24601 -0.34767  0.02472  0.32806  1.00389

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.446e-16  2.608e-02   0.000        1
TTR         -7.848e-01  2.817e-02 -27.861   <2e-16 ***
GI           7.205e-01  2.752e-02  26.178   <2e-16 ***
AWL          2.539e-01  2.716e-02   9.346   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4403 on 281 degrees of freedom
Multiple R-squared:  0.8082,    Adjusted R-squared:  0.8061
F-statistic: 394.6 on 3 and 281 DF,  p-value: < 2.2e-16
```

・TTR と GI と AWL が残る。

```
> jukaiki.z3 <- lm(Score ~ TTR + GI + AWL)
> vif(jukaiki.z3)
     TTR       GI      AWL
1.162202 1.109615 1.080821
```

・これで、VIF も問題ない。
・R2 も 0.8 とかなり高い。（8 割のデータがこのモデルで説明できる）

■結論

エッセイのスコアは TTR と GI と AWL で、8 割が説明できる。

■興味深い点

・TTR も GI も語彙の多様性を表す指標といわれているのに、、、
・相関を見てみる

```
> cor.test(TTR, GI)
```

```
        Pearson's product-moment correlation

data:  TTR and GI
t = 5.3729, df = 283, p-value = 1.62e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1949506 0.4060786
sample estimates:
      cor
0.3042462
```

・相関は弱い。

■もう一度やってみる。

・語彙の多様性指標は、MATTR にしてみる。
・Token はいろいろと相関が出るので外してみる。

データは正規分布していないといけないか

```
plot(jukaiki.ml2)
```

・Reference
  ・https://toukeier.hatenablog.com/entry/2019/09/08/224346

重回帰分析

https://cran.r-project.org/web/packages/pequod/index.html

・ランダム効果は入れられない

■交互作用の分析

・単純主効果

```
model.int <- simpleSlope(model.lmres, pred="説明２", mod1="説明３")
PlotSlope(model.int)
```

■データの中心化

・中心化をしたい項目について、オプションで指定する

```
, centered = c("項目A", "項目B")
```

■ Reference

・http://www.ner.takushoku-u.ac.jp/masano/class_material/waseda/keiryo/R18_reg1.html
・https://norimune.net/1856