Kandai.R #1
18/06/2016

非変態ユーザーのための R Tips

中西大輔 (広島修道大学)
@daihiko

まだRで消耗してるの?

- 心理統計に用事があるのなら、こんな会合に出て来ないで黙ってHADを使っていればいいと思う。
- Rを使うのは時間の無駄
- SPSSやSASを使うのは金の無駄

人はなぜRを使わないのか

- (たくさんの) 変態が変態のために作っているから。
- ユーザーがどういう出力がほしいのか、よく分からない変態が作っている。
  - でも、いろんなユーザーがいるから仕方がない。汎用的なシステムというのはそういうものだ。

人はなぜHADを使うのか

- (1人の) 変態がパンピーのために作っているから。
- どういう出力がほしいのか、よくわかっている変態が作っている。
- 非汎用的なシステムの利点。

ほしい出力がほしい

- 論文を書くときに必要な情報は分野によりだいたい決まっている。心理学だと、
- 実験参加者、調査対象者の性別ごとの人数
- 実験条件ごとの人数
- 条件ごとの平均値と標準偏差 (標準誤差)

やりたい加工がやりたい

- 「この値が99のやつは欠損値にしたい」

- 「この値が50以上のやつには1、未満のやつには0とコーディングしたい」

- でも情報量減らすからそういう分析をやろうとしたら立ち止まったほうがいいね。

TIPSの重要性

- 本来分析とは論文を書くために行うことなのに、論文を書くという視点から書かれた解説書やサイトが少なすぎる (その点『外国語教育研究ハンドブック』は良書)。

- 実験にしろ調査にしろ、報告することは決まっているので、方法や結果のセクションに書くための出力をどう得るかどういう視点から解説した本やサイトが必要。

- なので、入門者講習をクリアしたみなさまが最初につまづくであろうポイントをTIPSにしました。

なぜHADでないのか

- HADは人類からコードを書く力 (コードう力) を奪おうとする陰謀だから。
  - コードが書けないアホになる危険性。
- HADには依存性があるが開発者は1人だけ。
  - 明日から1ライセンス20万円になったらどうする。
  - クスリだって最初はタダですよ?

TIPS1: カテゴリカル変数にはちゃんと名前をつけておく

d$sex <- factor(d$sex, labels=list(“male", "female"))

- たとえば、男は0、女は1としていたり、男は1、女は2としている場合。逆だったらmaleとfemaleの位置を反対にする。

- 1が男だっけ? それともポリコレ意識して2を男にしてたんだっけ? 1週間後の自分を信用するな。

TIPS2: table関数

table (d$sex)

- 男女の人数をカウントする

table (dsex, dcond)

- 男女別・条件別の人数をカウントする

TIPS3: グループごとの処理

tapply (d$item, d$sex, mean, na.rm=TRUE)

- sexごとのitemのmeanを求める。na.rm=TRUEは「欠損値は無視する」。

性別、条件ごとには?

- dplyr入れたほうが幸せ。

```
install.packages("dplyr")

library("dplyr")
```

やり方 (1)

```
d01<-dplyr::group_by(d, sex, cond)

dplyr::summarise(d01, mean(item))
```

- どうしてもアメリカ英語が好きな人はsummarize でもOK。

やり方 (2)

```
dplyr::summarise(dplyr::group_by(d, sex, cond),
mean(item))
```

- どうしてもアメリカ英語が好きな人はsummarize でもOK。

TIPS4: 一部のひとたちだけを抜き出したデータフレームを作りたい

```
d02<-d01[d01$hoge!=10,]
```

- d01データフレームの変数hogeの値が10でないオブザベーション (行) をd02データフレームとして取り出す。

TIPS5: クロンバックのアルファ係数を求める

install.packages(“psych") <-1回やればよい

library(“psych")

item_all<-data.frame(item1, item2, item3, item4, item5, item6, item7, item8, item9, item10)

psych::alpha(item_all)

なぜこうできない!

```
proc corr alpha;

  var item1-item10;
```

dplyrを使ったやり方

```
library (dplyr)

item_all<-dplyr::select(df, num_range("item", 1:10, 1))
```

- itemの後ろ1桁を連番の数値とみなして、item1からitem10までを格納したデータフレームitem_allを作る。

```
psych::alpha(item_all)
```

慣れたら1行で

```
psych::alpha(dplyr::select(df, num_range("item", 1:10, 1)))
```

TIPS6: 合成変数を作る

```
d$item_all<-apply(dplyr::select(df, num_range("item", 1:10, 1)), 1, sum)
```

- 「1」は行ごと、「2」は列ごと。

- dplyrとかapply関数を知る前のあなた

```
d$item_all<-
item1+item2+item3+item4+item5+item6+item7+item8+
item9+item10
```

(ただし、可読性は高い)

TIPS7: 値を変換したい

(SASでは)

if item_all<30 then item_all=0;

else item_all=1;

- item_allの値が30未満のものを0に、30以上のものを1にする (逆にやると全部0になっちゃうよ!)。

Rの場合

d$item_all[d$item_all<30]<-0

d$item_all[d$item_all>=30]<-1

- item_allの値が30未満のものを0に、30以上のものを1にする (逆にやると全部0になっちゃうよ!)。

- SASと比べてコードが自然言語に遠いので分かりにくいけど、そこはぐっと我慢。

データハンドリングは

- この前のHijiyama.R #4の@kazutanのスライドをご覧ください。
  - https://github.com/kazutan/HijiyamaR4
- ポイントは、とにかく初心者もdplyrを使うことをためらわないことです。

結論

HADがおすすめ!