

贝叶斯因子序列分析在 R 语言中的实现

郑元瑞

胡传鹏

目录

1 下载和安装需要的 R 语言程序包	1
2 导入数据	2
2.1 数据长宽数据转换	2
3 查看数据的被试信息	2
4 (重复测量) 方差分析的 R 语言实现	3
4.1 数据的基本信息	4

1 下载和安装需要的 R 语言程序包

```
# install.packages(c("tidyverse", "BayesFactor", "here"))
library(BayesFactor) # 计算  $T$  检验和方差分析的贝叶斯因子
library(tidyverse)
library(here)
here()

## [1] "/Users/zhengyuanrui/SBFA_Tutorial"
```

2 导入数据

```
df <- read_csv(here("2_Data", "df.js.sum_jasp.csv"))

## Rows: 20 Columns: 10
## -- Column specification -----
## Delimiter: ","
## chr (1): subj_idx
## dbl (9): Match_Bad, Match_Good, Match_Neutral, Mismatch_Bad, Mismatch_Good, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

#T 检验使用数据
```

2.1 数据长宽数据转换

```
df_anova <- df %>%
  select(subj_idx:Mismatch_Good) %>%
  pivot_longer(
    cols = Match_Bad:Mismatch_Good,
    names_to = c("Matchness", "Valence"),
    names_sep = "_", values_to = "rt"
  )#anova 使用数据
```

3 查看数据的被试信息

```
subj_num <- unique(df$subj_idx) # 每个被试的编号
n <- length(unique(df$subj_idx)) # 一共有 20 个被试
```

(配对样本) T 检验的 R 语言实现 ## good_match 条件与 bad_match

条件的对比先建立一个空的列表，用来储存后续的贝叶斯因子。列表长度为目前数据的样本量

```
bf_output <- rep(NA, length(subj_num)) ### 先建立一个列表
```

```
for (i in seq_along(subj_num)) {#i 遍历 subj_num
  if (i == 1) {
    next
    # 由于一个被试不能正确计算贝叶斯因子，所以当 i 等于 1 时，跳过
  }
  # 将 df 数据框中的 subj_idx 列转换为字符串型
  df$subj_idx <- as.character(df$subj_idx)
  # 提取出遍历到的被试编号
  id <- unique(df$subj_idx)[1:i]
  # 从原数据中筛选被试
  df.selected <- df %>% filter(subj_idx %in% id)
  # 转换为因子型
  df.selected$subj_idx <- as.factor(df.selected$subj_idx)
  bayesfactors <- ttestBF(
    x = df.selected$Match_Bad, y = df.selected$Match_Good,
    paired = TRUE
  )# 计算贝叶斯因子
  bf_output[i] <- bayesfactors[1]
}
```

```
bf_output
```

```
## [1] NA 1.079341 2.095059 3.339921 6.652698 19.097998
## [7] 31.855000 49.752936 130.961355 58.188628 93.428773 239.350270
## [13] 127.216734 250.586118 571.531998 1522.706898 2032.527877 2787.948423
## [19] 5664.722345 3113.214090
```

4 （重复测量）方差分析的 R 语言实现

4.1 数据的基本信息

```
subj_num <- unique(df_anova$subj_idx) # 每个被试的编号
n <- length(unique(df_anova$subj_idx)) # 一共有 20 个被试
```

生成三个列表用来储存两个主效应和交互项

```
BFs_match <- rep(NA, length(subj_num))
```

```
BFs_valence <- rep(NA, length(subj_num))
```

```
BFs_int <- rep(NA, length(subj_num))
```

```
for (i in seq_along(subj_num)) {
  if (i == 1) {
    next
  }
  df_anova$subj_idx <- as.character(df_anova$subj_idx)
  id <- unique(df_anova$subj_idx)[1:i]
  df.selected <- df_anova %>% filter(subj_idx %in% id)
  df.selected$subj_idx <- as.factor(df.selected$subj_idx)
  df.selected$Matchness <- as.factor(df.selected$Matchness)
  df.selected$Valence <- as.factor(df.selected$Valence)
  bayesfactors <- bf <- anovaBF(rt ~ Valence*Matchness + subj_idx,
    data = data.frame(df.selected),
    whichRandom = "subj_idx"
  )
  BFs_match[i] <- bayesfactors[1]
  BFs_valence[i] <- bayesfactors[2]
  BFs_int[i] <- bayesfactors[4] / bayesfactors[3]
}
```

```
BFs_match
```

```
## [1] NA 0.8953323 0.8650357 0.6663430 1.2198559 1.0933056
```

```
## [7] 1.9655117 2.6025054 4.1638685 11.1578821 4.0817921 9.8094105
## [13] 13.6773648 16.5886212 27.4503838 33.1264474 39.9170928 95.8418954
## [19] 131.4633835 394.5779112
```

```
BFs_valence
```

```
## [1] NA 0.5443669 0.7204119 2.3970343 2.0215187 5.1356384
## [7] 6.4251278 9.3556566 20.5462128 12.0952198 49.0483067 56.3505902
## [13] 26.8491331 29.9237771 38.6972787 83.8987965 115.9720459 120.5988359
## [19] 243.5398468 122.4772697
```

```
BFs_int
```

```
## [1] NA 1.462711e+00 3.289766e+00 4.816308e+00 7.550150e+00
## [6] 3.998117e+01 7.530245e+01 2.873114e+02 5.562483e+02 2.287760e+02
## [11] 5.631946e+02 1.390083e+03 1.649642e+03 5.738990e+03 1.777621e+04
## [16] 8.679439e+04 7.243691e+04 2.030460e+05 2.016384e+05 7.473701e+04
```