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　3

3.1　结果 4

4　（重复测量）方差分析的 R 语言实现　5

4.1　数据的基本信息 5

4.2　做折线图 7

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"
```

```
options(scipen = 9)# 将科学计数法改为在万后 9 位
```

2 导入数据

```
df <- readr::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 %>%
  dplyr::select(subj_idx:Mismatch_Neutral) %>%
  tidyr::pivot_longer(
    cols = Match_Bad:Mismatch_Neutral,
    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
```

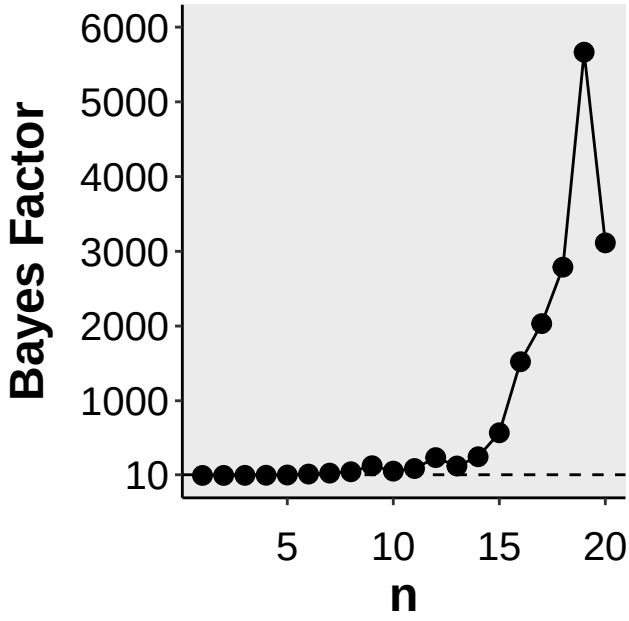
#（配对样本）T 检验的 R 语言实现 ## good_match 条件与 bad_match 条件的对比先建立一个空的列表，用来储存后续的贝叶斯因子。列表长度为目前数据的样本量

```
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$subj_idx)[1:i]
  # 从愿数据中筛选被试
  df.selected <- df %>% filter(subj_idx %in% id)
  # 转换为因子型
  df.selected$subj_idx <- as.factor(df.selected$subj_idx)
  bayesfactors <- BayesFactor::ttestBF(
    x = df.selected$Match_Bad, y = df.selected$Match_Good,
    paired = TRUE
  )# 计算贝叶斯因子
  bf_output[i] <- bayesfactors[1]
}
```

3.1 结果

```
tibble(bf_output) %>%
  dplyr::mutate(n = 1:nrow(.)) %>%
  dplyr::rename("Bayes Factor" = "bf_output") %>%
  dplyr::mutate(`Bayes Factor` = tidyr::replace_na(`Bayes Factor`, 1)) %>%
   ggplot(aes(x = n, y = `Bayes Factor`)) +
  geom_point(size = 3) +
  geom_line() +
  geom_hline(aes(yintercept = 10), linetype = "dashed") +
  scale_y_continuous(
    limits = c(1, 6000),
    breaks = c(10, 1000, 2000, 3000, 4000, 5000, 6000)) +
   theme(
    plot.margin = unit(c(1, 1, 1, 1), "cm"),
    plot.background = element_rect(fill = "white", color = NA),
    plot.title = element_text(size = 22, face = "bold",
                              hjust = 0.5,
                              margin = margin(b = 15)),
    axis.line = element_line(color = "black", size = .5),
    axis.title = element_text(size = 18, color = "black",
                              face = "bold"),
    axis.text = element_text(size = 15, color = "black"),
    axis.text.x = element_text(margin = margin(t = 10)),
    axis.title.y = element_text(margin = margin(r = 10)),
    axis.ticks = element_line(size = .5),
    panel.grid = element_blank(),
    legend.position = c(0.20, 0.8),
    legend.background = element_rect(color = "black"),
    legend.text = element_text(size = 15),
    legend.margin = margin(t = 5, l = 5, r = 5, b = 5),
    legend.key = element_rect(color = NA, fill = NA))
```



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
```

生成三个向量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$subj_idx)[1:i]
  df.selected <- df_anova %>% dplyr::filter(subj_idx %in% id)
  df.selected$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 <- BayesFactor::generalTestBF(
    rt ~ Valence*Matchness*subj_idx - subj_idx:Valence:Matchness,
    data = data.frame(df.selected),
    whichRandom = "subj_idx",
    neverExclude = "subj_idx",
    whichModels = "all", progress = FALSE)

  BFs_match[i] <- bayesfactors[4]/bayesfactors[1]# 计算 Matchness 主效应的 BF
  BFs_valence[i] <- bayesfactors[4]/bayesfactors[2]# 计算 Valence 的 BF
  BFs_int[i] <- bayesfactors[7] / bayesfactors[4]# 计算交互项的 BF
}
```

```
aov_output <- tibble::tibble(BFs_int, BFs_valence, BFs_match)# 整合为数据框
aov_output$BFs_int <- round(aov_output$BFs_int, digits = 2)# 保留两位小数
aov_output$BFs_valence <- round(aov_output$BFs_valence, digits = 2)
aov_output$BFs_match <- round(aov_output$BFs_match, digits = 2)
head(aov_output)# 查看数据
```

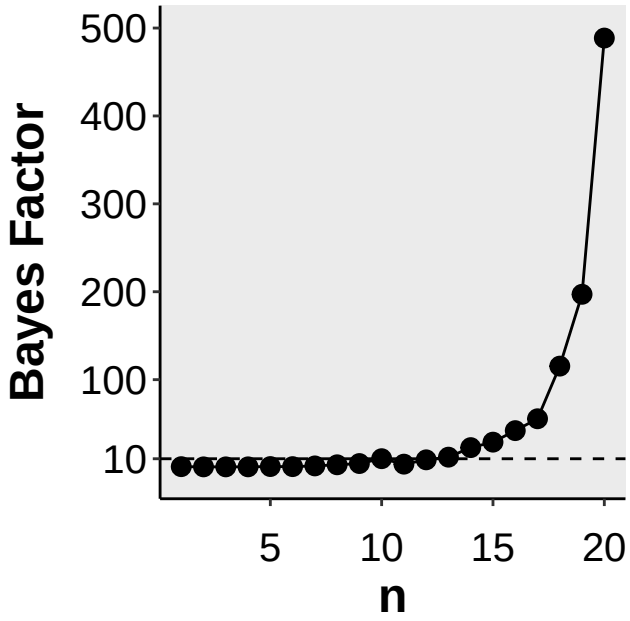
```
## # A tibble: 6 x 3
##   BFs_int BFs_valence BFs_match
##     <dbl>       <dbl>     <dbl>
## 1   NA          NA        NA
## 2    2.45        0.54      0.77
## 3   11.1         0.69      0.76
## 4   46.8         1.56      0.78
## 5   83.8         1.8       1.13
## 6 2591.          3.61      1.03
```

4.2 做折线图

```
dat_plot <- aov_output %>% dplyr::mutate(n = 1:nrow(.)) %>%
  tidyr::pivot_longer(BFs_int:BFs_match, names_to = "Effect",
                      values_to = "Bayes Factor") %>%
  dplyr::mutate(`Bayes Factor` = tidyr::replace_na(`Bayes Factor`, 1))
```

```
dat_plot %>% dplyr::filter(Effect == "BFs_match") %>%
  ggplot(aes(x = n, y = `Bayes Factor`)) +
  geom_point(size = 3) +
  geom_line() +
  geom_hline(aes(yintercept = 10), linetype = "dashed") +
  scale_y_continuous(
    limits = c(-10, 500),
    breaks = c(10, 100, 200, 300, 400, 500)) +
   theme(
    plot.margin = unit(c(1, 1, 1, 1), "cm"),
    plot.background = element_rect(fill = "white", color = NA),
    plot.title = element_text(size = 22, face = "bold",
                              hjust = 0.5,
                              margin = margin(b = 15)),
    axis.line = element_line(color = "black", size = .5),
    axis.title = element_text(size = 18, color = "black",
                              face = "bold"),
    axis.text = element_text(size = 15, color = "black"),
    axis.text.x = element_text(margin = margin(t = 10)),
    axis.title.y = element_text(margin = margin(r = 10)),
    axis.ticks = element_line(size = .5),
    panel.grid = element_blank(),
    legend.position = c(0.20, 0.8),
    legend.background = element_rect(color = "black"),
    legend.text = element_text(size = 15),
    legend.margin = margin(t = 5, l = 5, r = 5, b = 5),
```

```
    legend.key = element_rect(color = NA, fill = NA))
```




```
# ggsave("Match_maineffect.png", width = 10, height = 7, dpi = 300)
```