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


```
rm(list = ls())
```

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
```

1.1 BayesFacotr 包版本

```
packageVersion("BayesFactor")
```

```
## [1] '0.9.12.4.3'
```

2 导入数据

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

2.1 数据长宽数据转换

2.1.1 因变量为 RT 的数据整理

```
# 分析因变量为 RT 的使用数据
df.RT <- df %>%
  # 选择被试信息以及 RT_ 开头的列
  dplyr::select(subj_idx, starts_with("RT_")) %>%
  #RT_Bad_Match 到 RT_Good_Nonmatch 列转换为长数据，列名为 condition，值名为 rt
  tidyr::pivot_longer(
    cols = RT_Bad_Match:RT_Neutral_Nonmatch,
    names_to = "condition",
    values_to = "rt"
  ) %>%
  # 将 condition 列拆分为三列，DV_Name 为因变量名称
  #Valence 是道德信息，Matchness 是匹配信息
  tidyr::separate(col = condition,
                  into = c("DV_Name", "Valence", "Matchness"),
```

```
                 sep = "_") %>%
  # 类型为 character 的转换为因子类型，便于后续分析
  dplyr::mutate_if(is.character, as.factor)

head(df.RT)
```

```
## # A tibble: 6 x 5
##   subj_idx DV_Name Valence Matchness    rt
##   <fct>    <fct>   <fct>   <fct>     <dbl>
## 1 v010001  RT      Bad     Match      775.
## 2 v010001  RT      Bad     Nonmatch   793.
## 3 v010001  RT      Good    Match      716.
## 4 v010001  RT      Good    Nonmatch   817.
## 5 v010001  RT      Neutral Match      747.
## 6 v010001  RT      Neutral Nonmatch   786.
```

2.1.2 因变量为 ACC 的数据整理

```
df.ACC <- df %>%
  dplyr::select(subj_idx, starts_with("ACC_")) %>%
  tidyr::pivot_longer(
    cols = ACC_Bad_Match:ACC_Neutral_Nonmatch,
    names_to = "condition",
    values_to = "ACC"
  ) %>%
  tidyr::separate(col = condition,
                  into = c("DV_Name", "Valence", "Matchness"),
                  sep = "_") %>%
  dplyr::mutate_if(is.character, as.factor)

head(df.ACC)
```

```
## # A tibble: 6 x 5
```

```
##   subj_idx DV_Name Valence Matchness   ACC
##   <fct>    <fct>   <fct>   <fct>     <dbl>
## 1 v010001  ACC     Bad     Match     0.723
## 2 v010001  ACC     Bad     Nonmatch  0.730
## 3 v010001  ACC     Good    Match     0.896
## 4 v010001  ACC     Good    Nonmatch  0.8
## 5 v010001  ACC     Neutral Match     0.838
## 6 v010001  ACC     Neutral Nonmatch  0.794
```

2.1.3 因变量为 dPrime 的数据整理

```
df.dPrime <- df %>%
  dplyr::select(subj_idx, starts_with("dPrime_")) %>%
  tidyr::pivot_longer(
    cols = dPrime_Bad:dPrime_Neutral,
    names_to = "condition",
    values_to = "dPrime"
  ) %>%
  tidyr::separate(col = condition,
                  into = c("DV_Name", "Valence"),
                  sep = "_") %>%
  dplyr::mutate_if(is.character, as.factor)

head(df.dPrime)
```

```
## # A tibble: 6 x 4
##   subj_idx DV_Name Valence dPrime
##   <fct>    <fct>   <fct>    <dbl>
## 1 v010001  dPrime  Bad       1.21
## 2 v010001  dPrime  Good      2.10
## 3 v010001  dPrime  Neutral   1.81
## 4 v010003  dPrime  Bad       1.02
## 5 v010003  dPrime  Good      2.05
```

```
## 6 v010003  dPrime  Neutral   1.04
```

3 正确的 BF 计算

```
bayesfactors <- BayesFactor::generalTestBF(
    rt ~ Valence*Matchness*subj_idx - subj_idx:Valence:Matchness,
    data = data.frame(df.RT),
    whichRandom = "subj_idx",
    neverExclude = "subj_idx",
    whichModels = "all")

bayesfactors
```

```
## Bayes factor analysis
## --------------
## [1] Valence + subj_idx + Valence:subj_idx + Matchness:subj_idx
## [2] Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx
## [3] Valence:Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx
## [4] Valence + Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx
## [5] Valence + Valence:Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx
## [6] Matchness + Valence:Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx
## [7] Valence + Matchness + Valence:Matchness + subj_idx + Valence:subj_idx + Matchnes
## [8] subj_idx + Valence:subj_idx + Matchness:subj_idx
##
## Against denominator:
##   Intercept only
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
# 感兴趣的效应都要先除零模型（仅包括随机效应的模型）
null <- bayesfactors[8]

# 全模型
```

```
full <- bayesfactors[7]
BF_full.n <- full/null# 全模型与 null 对比
BF_excinx.n <- bayesfactors[4]/null
BF_inx <- BF_full.n/BF_excinx.n
BF_inx
```

```
## Bayes factor analysis
## --------------
## [1] Valence + Matchness + Valence:Matchness + subj_idx + Valence:subj_idx + Matchnes
##
## Against denominator:
##   rt ~ Valence + Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
BF_m.n <- bayesfactors[2]/null
BF_excinx.n/BF_m.n
```

```
## Bayes factor analysis
## --------------
## [1] Valence + Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx : 2609.99
##
## Against denominator:
##   rt ~ Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
BF_v.n <- bayesfactors[1]/null
BF_excinx.n/BF_v.n
```

```
## Bayes factor analysis
## --------------
## [1] Valence + Matchness + subj_idx + Valence:subj_idx + Matchness:subj_idx : 501.779
##
## Against denominator:
```

```
##   rt ~ Valence + subj_idx + Valence:subj_idx + Matchness:subj_idx
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
ttestBF(df$RT_Good_Match, df$RT_Neutral_Match, paired = TRUE, nullInterval = c(Inf, 0))
```

```
## t is large; approximation invoked.
## t is large; approximation invoked.
```

```
## Bayes factor analysis
## --------------
## [1] Alt., r=0.707 !(0<d<Inf) : 14411.55 ±NA%
##
## Against denominator:
##   Null, mu = 0
## ---
## Bayes factor type: BFoneSample, JZS
```

```
a <- ttestBF(df$RT_Good_Match, df$RT_Neutral_Match, paired = TRUE, nullInterval = c(0,
```

```
## t is large; approximation invoked.
## t is large; approximation invoked.
```

```
a[2]
```

```
## Bayes factor analysis
## --------------
## [1] Alt., r=0.707 !(0<d<Inf) : 14411.55 ±NA%
##
## Against denominator:
##   Null, mu = 0
## ---
## Bayes factor type: BFoneSample, JZS
```

4 查看数据的被试信息

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

```
## [1] 20
```

#（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

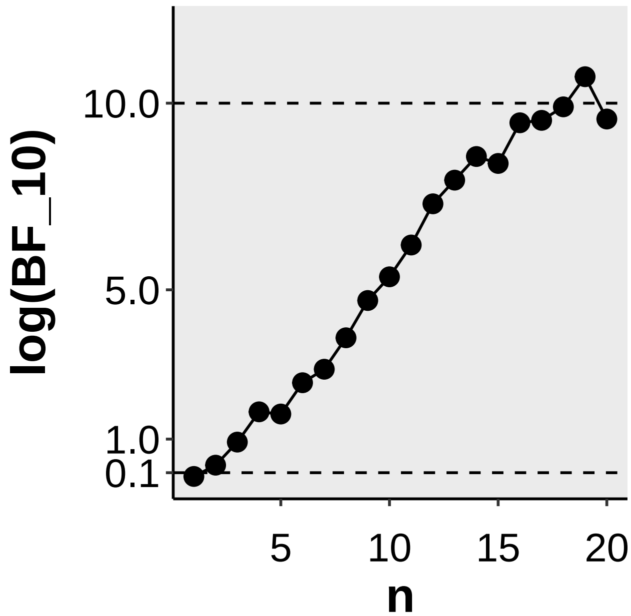
```
bf_output <- rep(1, length(subj_num)) ### 先建立一个列表
```

```
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$subj_idx)
  bayesfactors <- ttestBF(df.selected$RT_Good_Match,
                          df.selected$RT_Neutral_Match,
                          paired = TRUE,
                          nullInterval = c(0, Inf))# 计算贝叶斯因子
  bf_output[i] <- bayesfactors[2]
}
```

4.1 结果

```
tibble(bf_output) %>%
  dplyr::mutate(n = 1:nrow(.)) %>%
  dplyr::rename("Bayes Factor" = "bf_output") %>%
  dplyr::mutate(logBF = log(`Bayes Factor`)) %>%
   ggplot(aes(x = n, y = logBF)) +
  geom_point(size = 3) +
  geom_line() +
  geom_hline(aes(yintercept = 10), linetype = "dashed") +
  geom_hline(aes(yintercept = 1/10), linetype = "dashed") +
  scale_y_continuous(
    limits = c(0, 12),
    breaks = c(1/10, 1, 5, 10)) +
  ylab(label = "log(BF_10)") +
   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))
```



（重复测量）方差分析的 R 语言实现 ## 数据的基本信息

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

```
## [1] 20
```

生成三个向量用来储存两个主效应和交互项

```
BFs_match <- rep(1, length(subj_num))

BFs_valence <- rep(1, length(subj_num))

BFs_int <- rep(1, length(subj_num))
```

```
for (i in seq_along(subj_num)) {
  if (i == 1) {
    next
  }
```

```
  df.RT$subj_idx <- as.character(df.RT$subj_idx)
  id <- unique(df.RT$subj_idx)[1:i]
  df.selected <- df.RT %>% dplyr::filter(subj_idx %in% id)
  df.selected$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)

  null <- bayesfactors[8]
  full <- bayesfactors[7]# 全模型
  BF_full.n <- full/null# 全模型与 null 对比
  BF_excinx.n <- bayesfactors[4]/null
  BF_m.n <- bayesfactors[2]/null
  BF_v.n <- bayesfactors[1]/null
  BFs_match[i] <- BF_excinx.n/BF_v.n# 计算 Matchness 主效应的 BF
  BFs_valence[i] <- BF_excinx.n/BF_m.n# 计算 Valence 的 BF
  BFs_int[i] <- BF_full.n/BF_excinx.n# 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    1            1          1
```

```
## 2     3.42         0.58       0.76
## 3    22.8          0.66       0.71
## 4   144.          1.38       0.62
## 5    81.9         1.64       0.98
## 6 1728.           2.83       0.96
```

4.2 做折线图

4.2.1 长宽数据、log 变换

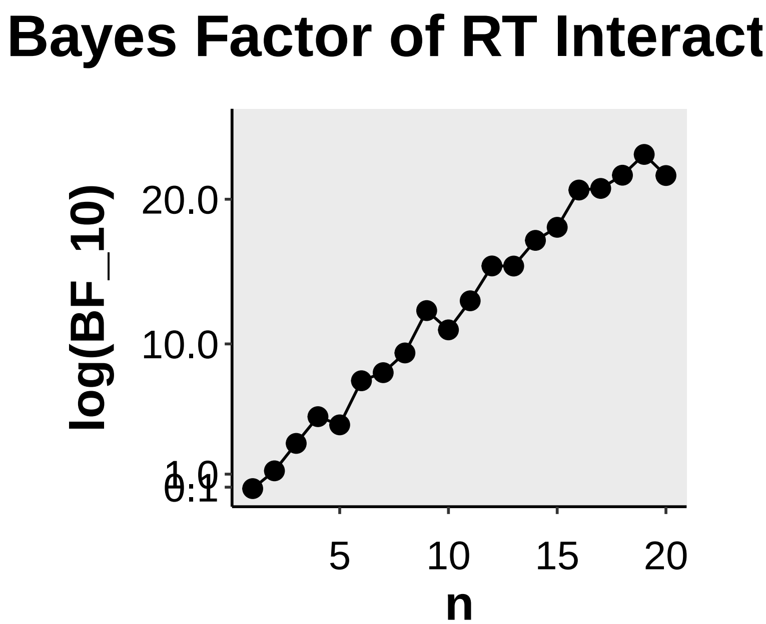
```r
dat_plot <- aov_output %>% dplyr::mutate(n = 1:nrow(.)) %>%
  tidyr::pivot_longer(BFs_int:BFs_match, names_to = "Effect",
               values_to = "Bayes Factor") %>%
  dplyr::mutate(`logBF` = log(`Bayes Factor`))
a <- dat_plot %>%
  filter(Effect == "BFs_int")
a
```

```
## # A tibble: 20 x 4
##          n Effect  `Bayes Factor` logBF
##      <int> <chr>            <dbl> <dbl>
##  1       1 BFs_int        1    e 0  0
##  2       2 BFs_int        3.42e 0  1.23
##  3       3 BFs_int        2.28e 1  3.13
##  4       4 BFs_int        1.44e 2  4.97
##  5       5 BFs_int        8.19e 1  4.41
##  6       6 BFs_int        1.73e 3  7.45
##  7       7 BFs_int        3.02e 3  8.01
##  8       8 BFs_int        1.18e 4  9.38
##  9       9 BFs_int        2.20e 5 12.3
## 10      10 BFs_int        5.81e 4 11.0
## 11      11 BFs_int        4.34e 5 13.0
## 12      12 BFs_int        4.82e 6 15.4
```

```
## 13    13 BFs_int        4.79e 6 15.4
## 14    14 BFs_int        2.83e 7 17.2
## 15    15 BFs_int        6.96e 7 18.1
## 16    16 BFs_int        9.16e 8 20.6
## 17    17 BFs_int        1.02e 9 20.7
## 18    18 BFs_int        2.55e 9 21.7
## 19    19 BFs_int        1.07e10 23.1
## 20    20 BFs_int        2.49e 9 21.6
```

```r
dat_plot %>% dplyr::filter(Effect == "BFs_int") %>%
  ggplot(aes(x = n, y = logBF)) +
  geom_point(size = 3) +
  geom_line() +
  ylab(label = "log(BF_10)") +
  ggtitle("The Bayes Factor of RT Interaction Effect") +
  scale_y_continuous(
    limits = c(0, 25),
    breaks = c(1/10, 1, 10, 20)) +
   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("RT_inx.png", width = 10, height = 7, dpi = 300)
```