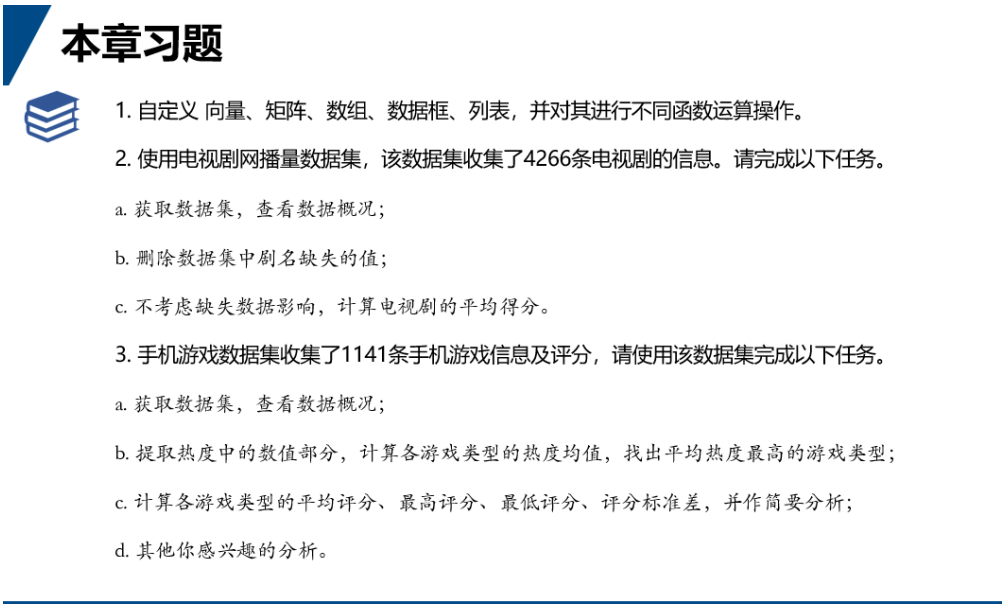
main_homework

AUTHOR
何啻

零、作业详情预览



本章习题

1. 自定义 向量、矩阵、数组、数据框、列表，并对其进行不同函数运算操作。

2. 使用电视剧网播量数据集，该数据集收集了4266条电视剧的信息。请完成以下任务。

a. 获取数据集，查看数据概况；

b. 删除数据集中剧名缺失的值；

c. 不考虑缺失数据影响，计算电视剧的平均得分。

3. 手机游戏数据集收集了1141条手机游戏信息及评分，请使用该数据集完成以下任务。

a. 获取数据集，查看数据概况；

b. 提取热度中的数值部分，计算各游戏类型的热度均值，找出平均热度最高的游戏类型；

c. 计算各游戏类型的平均评分、最高评分、最低评分、评分标准差，并作简要分析；

d. 其他你感兴趣的分析。

一、自定义不同数据结构并进行函数运算

这一部分较杂乱，详情见附录下***附录：学习笔记.pdf***文件的R语言学习笔记，其中涵盖了大部分的数据结构。

二、电视剧网播量数据集

作业题目解答指路：

a. 见***2.2探索性数据分析***

b. 见***2.2.2数据预处理-缺失值检查***部分

c. 见***2.2.3描述性统计*** 表格部分

数据清洗最后得到的数据集效果预览：

	剧名	播放量	点赞	差评	得分	采集日期	言情剧	穿越剧	网络剧	古装剧	喜剧	武侠剧	悬疑剧	青春剧	偶像剧	宫廷剧
1	花千骨2015	3.07亿	992342	357808	7.3	2015-09-23 23:48:48	1	1	1	0	0	0	0	0	0	0
2	还珠格格2015	73.3万	2352	7240	2.5	2015-09-23 23:48:48	0	0	1	1	1	0	0	0	0	0
3	天局	3454万	38746	3593	9.2	2015-09-23 23:48:52	0	0	1	1	0	1	1	0	0	0
4	明若晓溪	1.57亿	518660	72508	8.8	2015-09-23 23:48:51	1	0	0	0	0	0	0	1	1	0
5	多情江山	1126万	22553	6955	7.6	2015-09-23 23:48:52	1	0	0	1	0	0	0	0	0	1
6	我是机器人	1660万	37905	3605	9.1	2015-09-23 23:48:51	0	0	0	0	1	0	0	0	1	0
7	琅琊榜	1.29亿	401973	15829	9.6	2015-09-23 23:48:53	0	0	0	1	0	0	0	1	1	
8	伪装者	7.36亿	996680	102042	9.1	2015-09-23 23:49:03	0	0	0	0	0	0	1	0	0	
9	花千骨未删减版	3.91亿	256631	84710	7.5	2015-09-23 23:49:02	1	0	0	1	0	0	0	0	1	0
10	刺蝶	8298万	43746	17323	7.2	2015-09-23 23:49:02	0	0	0	0	0	0	1	0	0	
11	生死血符	644万	5141	3111	6.2	2015-09-23 23:49:03	0	0	0	0	0	0	0	0	0	
12	历史永远铭记	117万	1383	535	7.2	2015-09-23 23:49:03	0	0	0	0	0	0	0	0	0	0
13	最佳前男友	5.78亿	599349	96435	8.6	2015-09-23 23:49:05	1	0	0	0	0	0	0	0	1	0
14	鸳鸯佩	2.62亿	257062	96732	7.3	2015-09-23 23:49:04	1	0	0	0	0	0	0	0	0	0
15	我的媳妇是女王	2.27亿	275287	52397	8.4	2015-09-23 23:49:05	1	0	0	0	0	0	0	0	0	0
16	二胎时代未删减版	2163万	23706	3010	8.9	2015-09-23 23:49:04	1	0	0	0	0	0	0	0	0	0
17	二胎时代	2.44亿	329236	54983	8.6	2015-09-23 23:49:06	1	0	0	0	0	0	0	0	0	0
18	绝命追踪	6466万	73433	2684	9.6	2015-09-23 23:49:15	0	0	0	0	0	0	1	0	0	0
19	少年神探狄仁杰	3.85亿	653103	75272	9.0	2015-09-23 23:49:15	0	0	0	1	0	0	1	0	0	0
20	神犬小七未删减版	2402万	63344	7292	9.0	2015-09-23 23:49:17	0	0	0	0	0	0	0	0	1	0

2.1 库准备与数据导入

```
#加载包
library(readr)  #读取数据
library(dplyr)  #数据处理
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)  #数据可视化
library(stringr)
library(VIM)  #缺失值处理
```

```
Loading required package: colorspace

Loading required package: grid

VIM is ready to use.

Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues


Attaching package: 'VIM'

The following object is masked from 'package:datasets':

    sleep
```

```
library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
library(kableExtra)
```

```
Attaching package: 'kableExtra'

The following object is masked from 'package:dplyr':

    group_rows
```

```
library(moments)    # 计算偏度和峰度
library(knitr)      # 生成表格
library(ggrain)
```

```
Registered S3 methods overwritten by 'ggpp':
  method                  from
  heightDetails.titleGrob ggplot2
  widthDetails.titleGrob  ggplot2
```

数据导入

```
#导入电视剧数据
data_tv <- read_csv("9月4日作业/CH2_TV.csv", locale = locale(encoding = "GBK"))
```

```
Rows: 4266 Columns: 7
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (7): 剧名, 类型, 播放量, 点赞, 差评, 得分, 采集日期

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.
```

2.2 探索性数据分析（EDA）

2.2.1 数据预览

```
print(data_tv)
```

```
# A tibble: 4,266 × 7
   剧名           类型                          播放量 点赞   差评   得分   采集日期
   <chr>          <chr>                         <chr>  <chr>  <chr>  <chr>  <chr>
 1 花千骨2015     "言情剧\n/\n穿越剧\n/\n网络… 3.07亿 992,… 357,… 7.3    2015-9-…
```

```
 2 还珠格格2015    "古装剧\n\n/\n喜剧\n\n/\n网… 73.3万 2,352 7,240 2.5   2015-9-…
 3 天局            "武侠剧\n/\n古装剧\n/\n悬疑… 3454万 38,7… 3,593 9.2   2015-9-…
 4 明若晓溪        "青春剧\n/\n言情剧\n/\n偶像… 1.57亿 518,… 72,5… 8.8   2015-9-…
 5 多情江山        "言情剧\n/\n古装剧\n/\n宫廷… 1126万 22,5… 6,955 7.6   2015-9-…
 6 我是机器人      "偶像剧\n\n/\n喜剧"           1660万 37,9… 3,605 9.1   2015-9-…
 7 琅琊榜          "历史剧\n/\n古装剧\n/\n偶像… 1.29亿 401,… 15,8… 9.6   2015/9/…
 8 伪装者          "谍战剧\n/\n悬疑剧"           7.36亿 996,… 102,… 9.1   2015-9-…
 9 花千骨未删减版  "言情剧\n/\n古装剧\n/\n偶像… 3.91亿 256,… 84,7… 7.5   2015-9-…
10 刺蝶            "军旅剧\n\n/\n谍战剧\n\n/\n… 8298万 43,7… 17,3… 7.2   2015-9-…
# ℹ 4,256 more rows
```

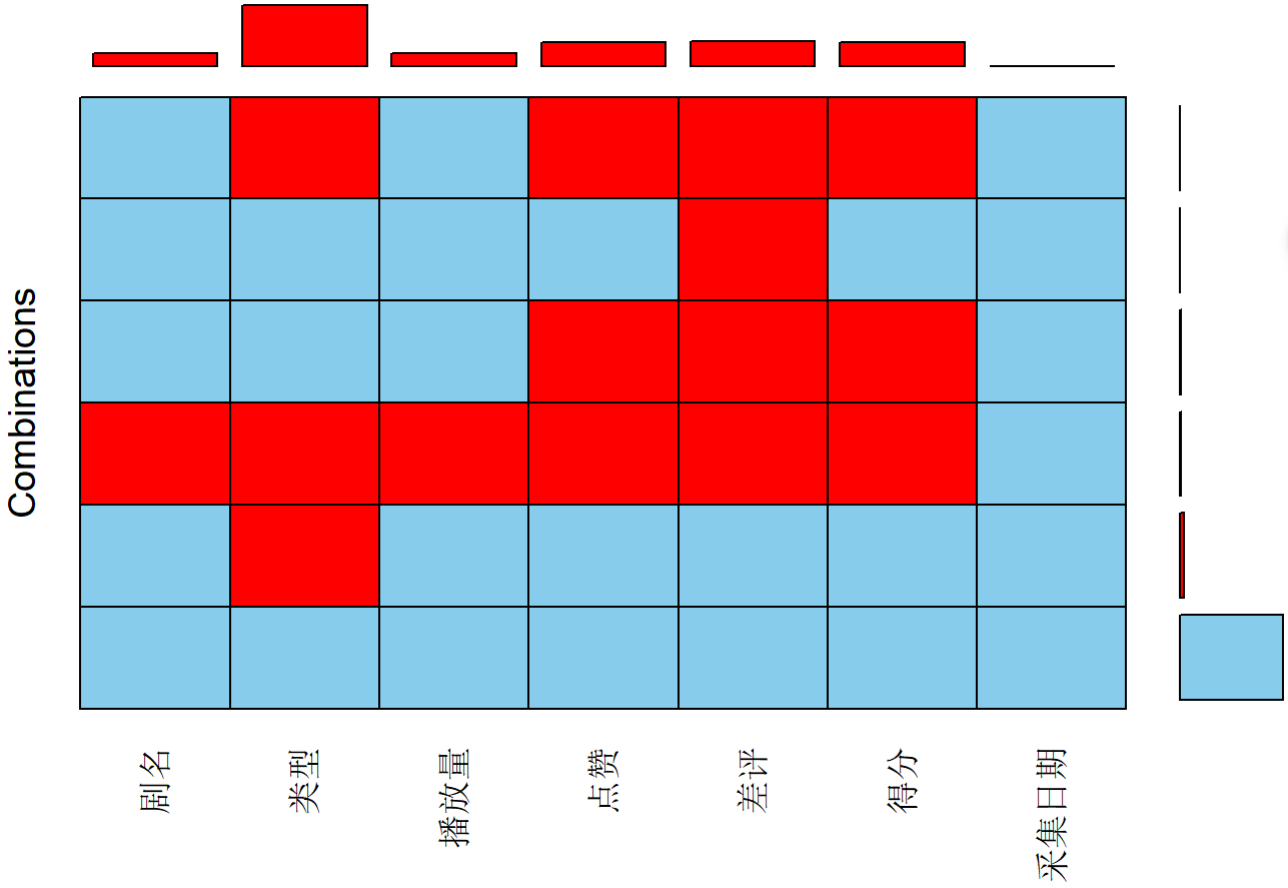
2.2.2 数据预处理

- 缺失值检查

```
#先将所有空值、null、0转变为缺失值，便于查看
data_tv[data_tv[1:nrow(data_tv), ] == ""] <- NA
data_tv[is.null(data_tv)] <- NA
data_tv[data_tv[1:nrow(data_tv), ] == 0] <- NA
data_tv[data_tv[1:nrow(data_tv), ] == "null"] <- NA
data_tv[data_tv[1:nrow(data_tv), ] == '.'] <- NA

#查看哪些变量存在缺失值
aggr(data_tv,plot=FALSE)
```

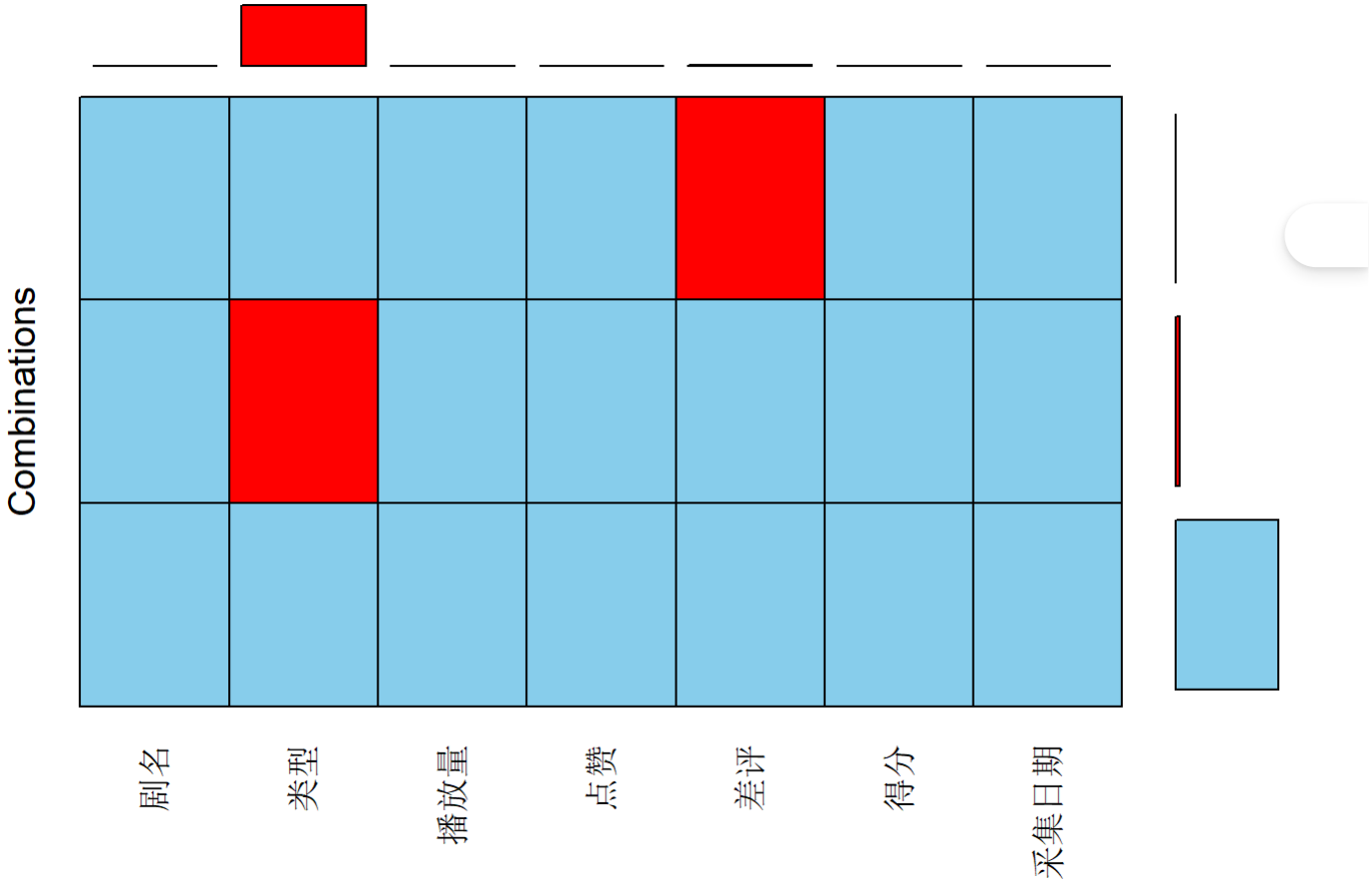
```
Missings in variables:
Variable Count
    剧名    47
    类型   220
  播放量    47
    点赞    88
    差评    90
    得分    88
```

```
aggr(data_tv,combined=T)
```



将'**剧名**'**缺失**或者'**点赞**'、'**差评**'、'**得分**'**缺失**的部分删去

```
data_tv_clean <- data_tv %>%
  filter(!is.na(剧名) & (!(is.na(点赞) & is.na(差评) & is.na(得分))))
aggr(data_tv_clean,combined=T)
```



- 将'类型'转化为**独热编码（One-Hot）格式**

提取所有类型

```
# 创建新数据，保留列名
data_tv1 <- data_tv_clean %>%
  select(剧名,播放量)
# 转换点赞、差评、得分为数值
data_tv1$点赞 <- as.numeric(gsub(",", "", data_tv_clean$点赞))
data_tv1$差评 <- as.numeric(gsub(",", "", data_tv_clean$差评))
data_tv1$得分 <- as.numeric(data_tv_clean$得分)
# 转换采集日期为日期格式
data_tv1$采集日期 <- ymd_hms(data_tv_clean$采集日期)

# 提取所有类型并去重
(all_types <- str_extract_all(data_tv_clean$类型, "[^\\n/]+") %>%
  unlist() %>%
  trimws() %>%
  unique())
```

```
 [1] "言情剧"     "穿越剧"     "网络剧"     "古装剧"     "喜剧"
 [6] "武侠剧"     "悬疑剧"     "青春剧"     "偶像剧"     "宫廷剧"
[11] "历史剧"     "谍战剧"     "军旅剧"     "年代剧"     "家庭剧"
[16] "粤语电视剧" "科幻剧"     "罪案剧"     "神话剧"     NA
[21] "农村剧"
```

编码

```
# 根据提取的类型创建新列
for (type in all_types) {
  if (!is.na(type)) {
  data_tv1[[type]] <- ifelse(
                        !is.na(data_tv_clean$类型) & grepl(type, data_tv_clean$类型),
                        1,
                        ifelse(is.na(data_tv_clean$类型), NA, 0)
                        )
}}
```

2.2.3 描述性统计

```
summary(data_tv1[,3:5])
```

```
     点赞              差评              得分        
Min.   :     13   Min.   :      2   Min.   : 2.400  
1st Qu.:  10500   1st Qu.:   3640   1st Qu.: 6.900  
Median :  31501   Median :   8167   Median : 7.700  
Mean   : 228066   Mean   :  43377   Mean   : 7.593  
3rd Qu.: 122064   3rd Qu.:  28562   3rd Qu.: 8.400  
Max.   :8613347   Max.   :1773345   Max.   :10.000  
                  NA's   :2
```

```
tab_01 = data.frame(
  Mean  = c(colMeans(data_tv1[,3:5])),
  SD = c(sapply(data_tv1[,3:5], sd)),
  Median = c(sapply(data_tv1[,3:5], median)),
  Min = c(sapply(data_tv1[,3:5], min)),
  # 计算数据的偏度
  skew = c(sapply(data_tv1[,3:5], skewness)),
  # 计算数据的峰度
  kurt = c(sapply(data_tv1[,3:5], kurtosis)),
  Max = c(sapply(data_tv1[,3:5], max))
)
## table for descriptive statistics
kable(
  tab_01,
  col.names = c("均值", "标准差", "中位数","偏度","峰度", "最小", "最大"),
  digits = 2,
  caption = "\\label{tab2}Summary Statistics",
  booktabs = T
)
```

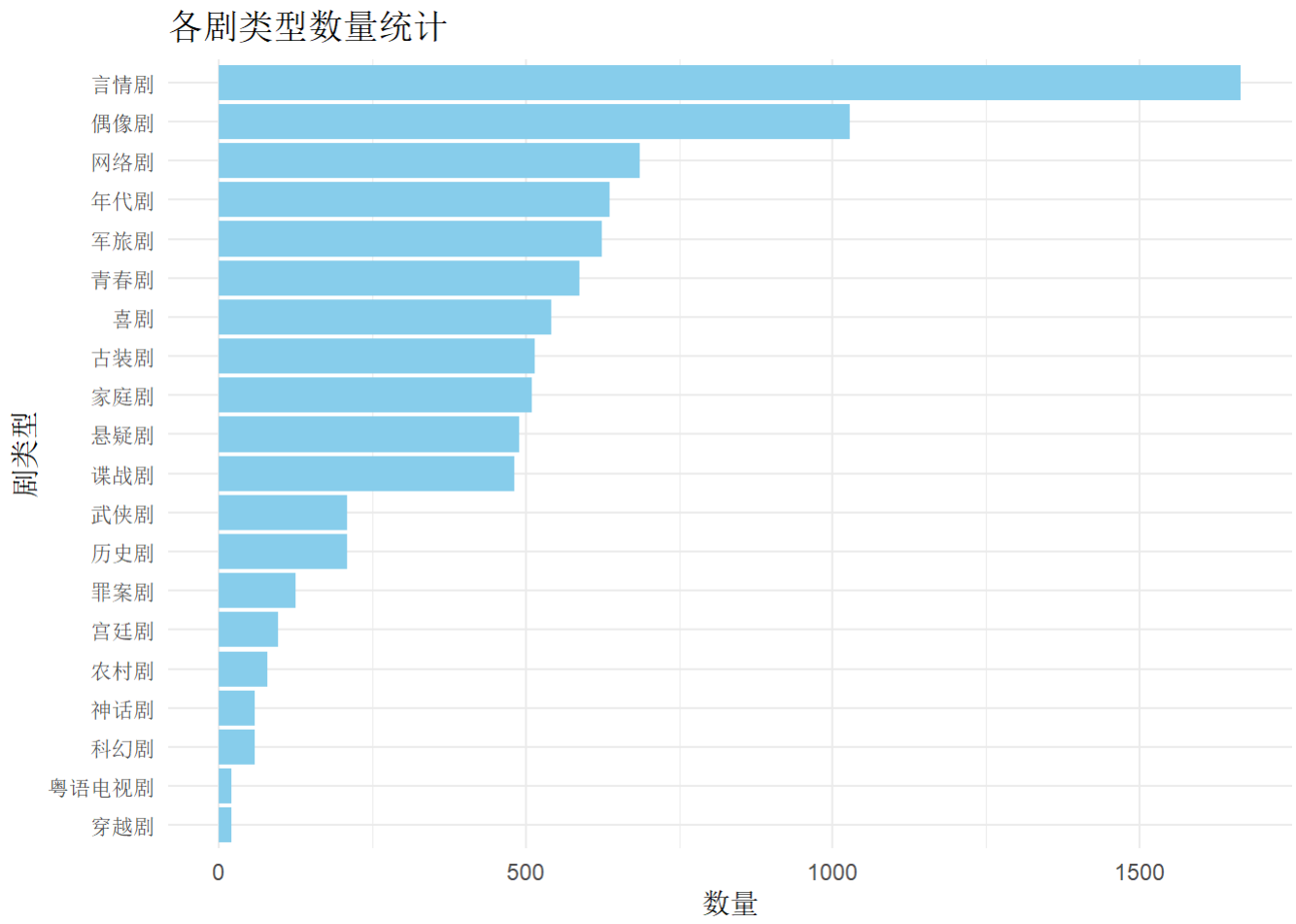
Summary Statistics

	均值	标准差	中位数	偏度	峰度	最小	最大
点赞	228065.88	743059.82	31501.0	13.0	7.65	74.80	8613347
差评	NA	NA	NA	NA	NA	NA	NA
得分	7.59	1.12	7.7	2.4	-0.98	5.08	10

2.2.4 剧类型分析

```
# 选择all_types对应的列名
selected_columns <- all_types[!is.na(all_types)]

# 删除所有这些列均为NA的行
data_tv2 <- data_tv1 %>%
  filter(rowSums(is.na(select(., all_of(selected_columns)))) != length(selected_columns))
```

```
# 统计各剧类型的数量
type_counts <- colSums(data_tv2[selected_columns], na.rm = TRUE) # 计算各列的和
type_counts <- data.frame(类型 = names(type_counts), 数量 = type_counts)

# 绘制柱状图
ggplot(type_counts, aes(x = reorder(类型, 数量), y = 数量)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "各剧类型数量统计", x = "剧类型", y = "数量") +
  theme_minimal()
```



三、手机游戏数据集

作业题目解答指路：

a. 见**3.2 探索性数据分析**与**3.4 数据挖掘**

b. 计算游戏平均热度，见 **3.4.1对'热度'和'类别'的分组探索**

c. 计算游戏评分相关统计量与分析，详情见**3.4.2对'评分'和'类别'的分组探索**和**3.4.3 数据分析**：

d. 见**3.2 探索性数据分析**其余部分与 **3.5机器学习预测分析方法**

数据清洗和特征工程最后得到的效果预览：

	游戏版本1	最后更新年	最后更新月	最后更新日	游戏名称	评分	类别	热度	资费	开发商	支持系统	评论数	喜欢数	中文	英文
1	1	2015	7	29	捕鱼达人	7.6	益智休闲	95	道具收费	触控科技	2.3以上	6654	114	1	0
2	1	2014	5	27	NBA2K14	7.5	体育运动	94	完全免费	2K Games	4.0以上	16190	3712	1	0
3	6	2016	9	29	愤怒的小鸟六周年版	5.9	益智休闲	93	道具收费	拓维信息系统股份有限公司	2.3以上	7187	280	1	0
4	1	2012	7	31	三国志无双战	5.1	动作游戏	90	完全免费	PlayBean Co., Ltd.	2.2以上	9372	1608	1	0
5	2	2016	9	23	地铁跑酷-冰岛	8.4	益智休闲	89	道具收费	乐逗	4.0以上	8804	888	1	0
6	1	2016	8	30	火箭飞人	8.2	动作游戏	89	完全免费	Halfbrick Studios	4.0.3以上	7371	544	0	1
7	1	2015	2	17	暴力摩托完整版	5.1	动作游戏	89	完全免费	Adrenaline Crew	2.0.1以上	4461	118	0	1
8	2	2015	7	6	愤怒的小鸟太空版	9.5	益智休闲	87	完全免费	Rovio	2.3以上	5238	127	0	1
9	0	2016	9	30	NBA 2K17免验证版(含数据包)	8.0	体育运动	85	完全免费	2K Games	4.3以上	2411	907	0	1
10	1	2016	7	22	天天酷跑	6.9	益智休闲	84	完全免费	腾讯	2.2以上	8245	1085	1	0
11	4	2016	7	21	杀手2-影子阴谋	6.5	射击游戏	84	道具收费	北京通联天地科技有限公司	2.1以上	2457	1912	1	0
12	0	2016	9	21	我的世界移动版	8.3	养成游戏	80	完全免费	Mojang	4.2以上	66893	7989	1	1
13	1	2013	12	5	NBA2K13(含数据包)	8.1	体育运动	80	完全免费	2K Games	4.0以上	12011	725	0	1
14	1	2016	9	28	极品飞车:无限(含数据包)	6.5	竞速游戏	79	完全免费	EA	4.0.3以上	5680	3534	1	1
15	3	2016	9	26	神庙逃亡2	8.1	益智休闲	79	道具收费	乐逗	2.3以上	8642	582	1	0
16	1	2014	1	15	空中打击高清完整版	5.0	飞行游戏	79	完全免费	Art In Games	2.0.1以上	4525	285	0	1
17	1	2016	3	16	鳄鱼小顽皮爱洗澡	9.0	益智休闲	78	免费试玩	触控科技	2.2以上	8789	335	1	0
18	1	2016	4	1	GTA侠盗猎车手:圣安地列斯(含数据包)	8.1	动作游戏	77	完全免费	Rockstar Games	3.0以上	96354	11410	0	1
19	2	2015	9	15	狂野飙车8:极速凌云	8.2	竞速游戏	76	道具收费	北京万盛和泰科技有限公司	2.3以上	14640	5039	1	0
20	1	2016	9	27	极品飞车17:最高通缉	8.5	竞速游戏	75	完全免费	EA	2.3以上	27449	7235	1	0
21	2	2016	7	17	跳跃忍者	5.3	动作游戏	75	完全免费	Backflip Studios	4.0.3以上	4093	105	0	1
22	1	2015	4	2	GTA侠盗猎车手:罪恶都市(含数据包)	7.8	动作游戏	75	完全免费	Rockstar Games	2.3以上	39297	4549	0	1
23	1	2016	1	13	捕鱼达人2	8.1	益智休闲	74	道具收费	触控科技	2.2以上	11035	114	1	0
24	1	2013	12	7	背刺商店版(含数据包)	7.9	动作游戏	74	完全免费	Gameloft	2.1以上	10150	2293	1	0
25	1	2014	8	26	愤怒的小鸟星球大战	8.9	益智休闲	73	完全免费	Rovio	2.3以上	5745	234	0	1
26	6	2016	9	28	愤怒的小鸟季节版	9.0	益智休闲	72	完全免费	Rovio	4.1以上	7241	369	0	1
27	0	2016	9	24	Pokemon GO	7.3	益智休闲	72	完全免费	Niantic, Inc.	4.4以上	656	27	0	1

3.1 库准备与数据导入

```
#加载包
library(readr)  #读取数据
library(dplyr)  #数据处理
library(tidyr)
library(ggplot2)  #数据可视化
library(stringr)
library(VIM)  #缺失值处理
library(lubridate)
library(kableExtra)
library(moments)     # 计算偏度和峰度
library(knitr)       # 生成表格
library(ggrain)
```

导入数据

```
#导入手机游戏数据
data_game <- read_csv("9月4日作业/CH2_game.csv", locale = locale(encoding = "GBK"))
```

```
Rows: 1141 Columns: 12
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (8): 游戏名称, 类别, 语言, 热度, 游戏版本, 资费, 开发商, 支持系统
```

```
dbl  (3): 评分, 评论数, 喜欢数
date (1): 最后更新时间

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.
```

3.2 探索性数据分析（EDA）

3.2.1 数据预览

```
print(data_game)
```

```
# A tibble: 1,141 × 12
   游戏名称    评分 类别   语言   热度   最后更新时间 游戏版本 资费   开发商 支持系统
   <chr>      <dbl> <chr>  <chr>  <chr>  <date>       <chr>    <chr>  <chr>  <chr>
 1 捕鱼达人     7.6 益智… 中文   95℃    2015-07-29   1.9.0    道具… 触控…  2.3以上
 2 NBA2K14      7.5 体育… 中文   94℃    2014-05-27   1.3      完全… 2K Ga… 4.0以上
 3 愤怒的小…    5.9 益智… 中文   93℃    2016-09-29   6.0.3    道具… 拓维…  2.3以上
 4 三国志无…    5.1 动作… 中文   90℃    2012-07-31   1.3      完全… PlayB… 2.2以上
 5 地铁跑酷…    8.4 益智… 中文   89℃    2016-09-23   2.54.0   道具… 乐逗   4.0以上
 6 火箭飞人     8.2 动作… 英文   89℃    2016-08-30   1.9.10   完全… Halfb… 4.0.3以…
 7 暴力摩托…    5.1 动作… 英文   89℃    2015-02-17   1.11     完全… Adren… 2.0.1以…
 8 愤怒的小…    9.5 益智… 英文   87℃    2015-07-06   2.2.1    完全… Rovio  2.3以上
 9 NBA 2K17…    8   体育… 英文   85℃    2016-09-30   0.0.21   完全… 2K Ga… 4.3以上
10 天天酷跑     6.9 益智… 中文   84℃    2016-07-22   1.0.35.0 完全… 腾讯   2.2以上
# ℹ 1,131 more rows
# ℹ 2 more variables: 评论数 <dbl>, 喜欢数 <dbl>
```

3.2.2 数据预处理

- 缺失值检查

为什么这里'评论数'和'喜欢数'列中将0视为缺失?

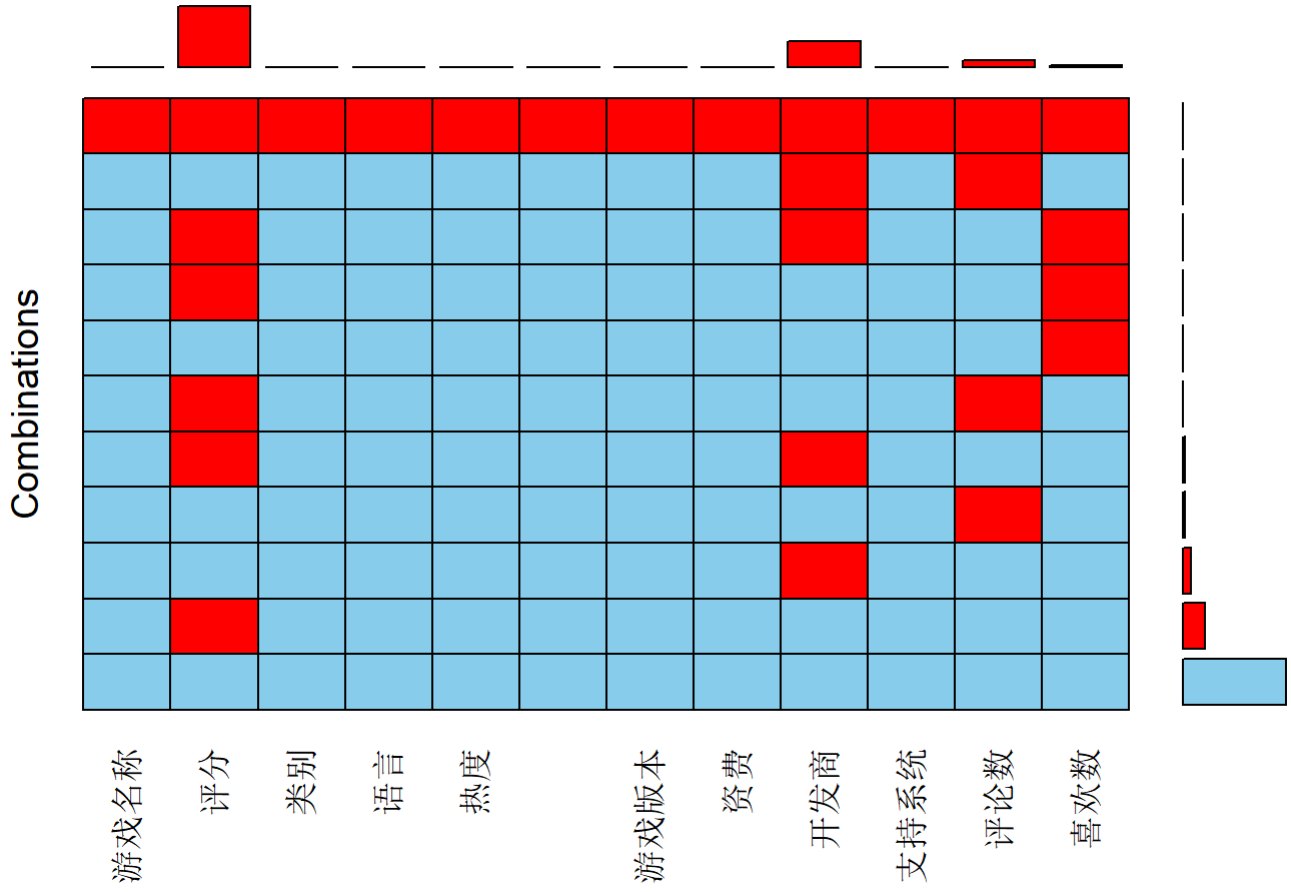
当某游戏该列（如评论数）的值为0时，比较表中另一列具有大致相同数值的（喜欢数）其他游戏，可以发现其他游戏的该列（评论数）一般都显著多于0，故而具有几乎相同评论数的该游戏不太可能该列值为0（喜欢数），猜测应该是搜集数据时遇到的搜集失败默认的赋值，为了数据的准确性和真实性，这里需要转化为NA来处理。

```
#先将所有空值、null、0转变为缺失值，便于查看
data_game[data_game[1:nrow(data_game), ] == ""] <- NA
data_game[is.null(data_game)] <- NA
data_game[data_game[1:nrow(data_game), ] == 0] <- NA

#查看哪些变量存在缺失值
aggr(data_game,plot=FALSE)
```

```
Missings in variables:
    Variable Count
    游戏名称     1
```

评分	203
类别	1
语言	1
热度	1
最后更新时间	1
游戏版本	1
资费	1
开发商	86
支持系统	1
评论数	22
喜欢数	7

```
aggr(data_game,combined=T)
```

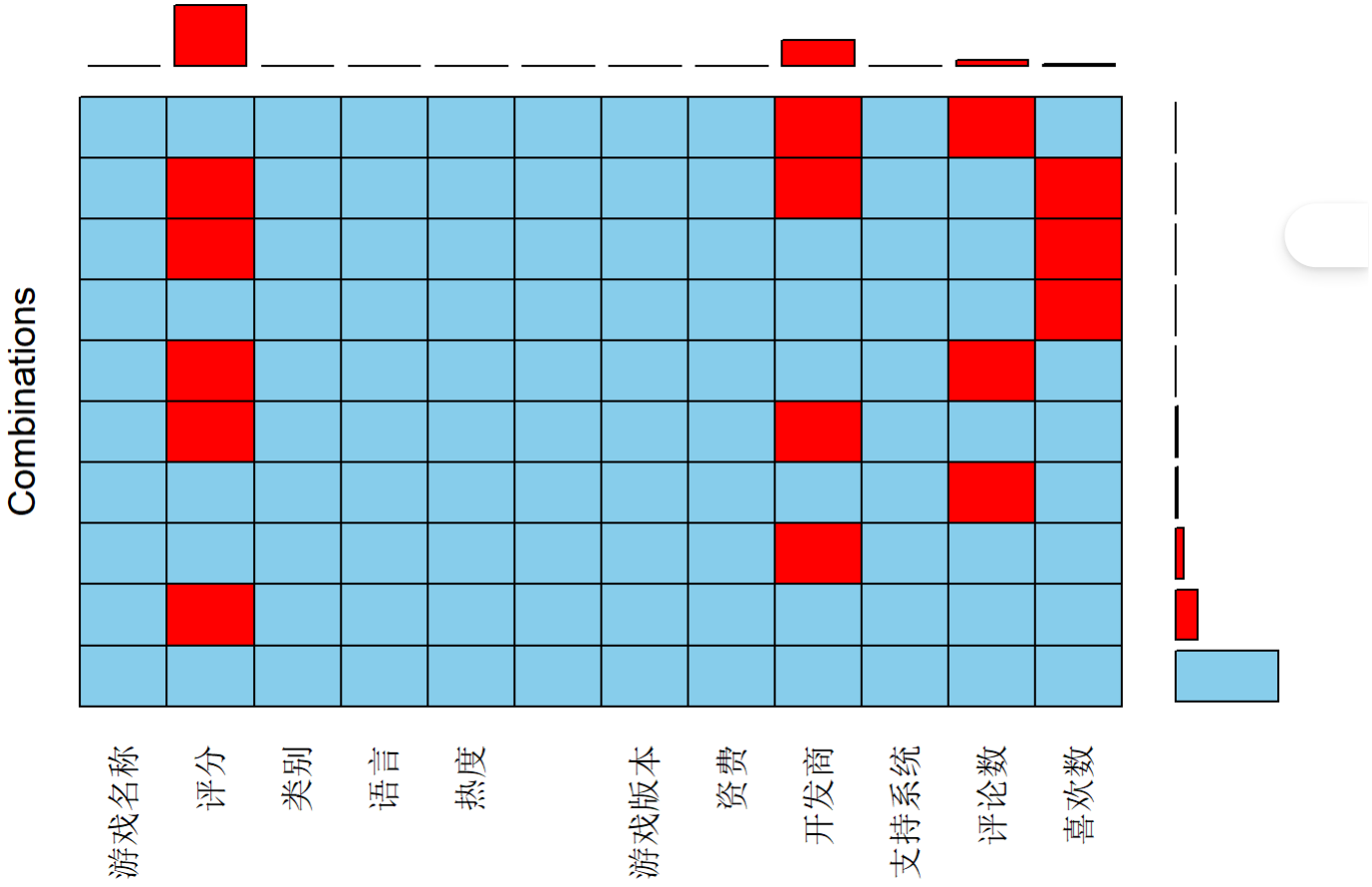


将**全部缺失**的部分删去

```
data_game_clean <- data_game %>%
  filter(rowSums(is.na(.)) != ncol(.))
print(paste('删去的行数为：',nrow(data_game)-nrow(data_game_clean)))
```

```
[1] "删去的行数为： 1"
```

```
aggr(data_game_clean,combined=T)
```



- 将'语言'转化为**独热编码（One-Hot）格式**

提取所有语言

```
# 创建新数据，排除“语言”列
data_game1 <- data_game_clean %>%
                              select(-语言)

# 提取所有类型并去重
(all_types <- str_extract_all(data_game_clean$语言, "[^, ]+") %>%
  unlist() %>%
  trimws() %>%
  unique())
```

```
[1] "中文" "英文" "韩文" "日文" "其他"
```

编码

```
# 根据提取的类型创建新列
for (type in all_types) {
  if (!is.na(type)) {
  data_game1[[type]] <- ifelse(
                  !is.na(data_game_clean$语言) & grepl(type, data_game_clean$语言),
                  1,
                  ifelse(is.na(data_game_clean$语言), NA, 0)
                  )
}}
```

3.3 特征工程

查看当前数据格式

```
print(data_game1)
```

```
# A tibble: 1,140 × 16
   游戏名称  评分 类别  热度  最后更新时间 游戏版本 资费  开发商 支持系统 评论数
   <chr>    <dbl> <chr> <chr> <date>       <chr>    <chr> <chr>  <chr>      <dbl>
 1 捕鱼达人   7.6 益智… 95℃   2015-07-29   1.9.0    道具… 触控…  2.3以上     6654
 2 NBA2K14    7.5 体育… 94℃   2014-05-27   1.3      完全… 2K Ga… 4.0以上    16190
 3 愤怒的…    5.9 益智… 93℃   2016-09-29   6.0.3    道具… 拓维…  2.3以上     7187
 4 三国志…    5.1 动作… 90℃   2012-07-31   1.3      完全… PlayB… 2.2以上     9372
 5 地铁跑…    8.4 益智… 89℃   2016-09-23   2.54.0   道具… 乐逗   4.0以上     8804
 6 火箭飞人   8.2 动作… 89℃   2016-08-30   1.9.10   完全… Halfb… 4.0.3以…   7371
 7 暴力摩…    5.1 动作… 89℃   2015-02-17   1.11     完全… Adren… 2.0.1以…   4461
 8 愤怒的…    9.5 益智… 87℃   2015-07-06   2.2.1    完全… Rovio  2.3以上     5238
 9 NBA 2K1…   8   体育… 85℃   2016-09-30   0.0.21   完全… 2K Ga… 4.3以上     2411
10 天天酷跑   6.9 益智… 84℃   2016-07-22   1.0.35.0 完全… 腾讯   2.2以上     8245
# ℹ 1,130 more rows
# ℹ 6 more variables: 喜欢数 <dbl>, 中文 <dbl>, 英文 <dbl>, 韩文 <dbl>,
#   日文 <dbl>, 其他 <dbl>
```

3.3.1 将热度提取为数值

```
convert_t_count <- function(t) {
    as.numeric(gsub("℃", "", t))
}
data_game1$热度 <- sapply(data_game1$热度, convert_t_count)
```

3.3.2 将‘最后更新时间’，‘游戏版本’分别分割提取成新列

```
# 1. 将所有数据（除了‘最后更新时间’，‘游戏版本’）保存至data_game2
data_game2 <- data_game1 %>%
  select(-`最后更新时间`, -`游戏版本`)

# 2. 将‘最后更新时间’提取为新列‘最后更新年’，‘最后更新月’，‘最后更新日’并加入到data_game2
data_game2 <- data_game1 %>%
  mutate(
    最后更新年 = as.integer(format(`最后更新时间`, "%Y")),
    最后更新月 = as.integer(format(`最后更新时间`, "%m")),
    最后更新日 = as.integer(format(`最后更新时间`, "%d"))
  ) %>%
  select(-colnames(data_game1)) %>%
  bind_cols(data_game2)

# 3. 提取游戏版本中的第一个“.”前的数值
data_game2 <- data_game1 %>%
  mutate(
    游戏版本1 = as.integer(ifelse(grepl("^[0-9]+", `游戏版本`),
                                  str_extract(`游戏版本`, "^[0-9]+"),
```

```
                                    NA))
  )%>%
  select(-colnames(data_game1)) %>%
  bind_cols(data_game2)

# 4. 将特定列转化为因子类型
factor_columns <- c('中文', '英文', '韩文', '日文', '其他', '支持系统', '开发商', '资费', '类别
data_game2 <- data_game2 %>%
  mutate(across(all_of(factor_columns), as.factor))

# 查看结果
print(data_game2)
```

```
# A tibble: 1,140 × 18
   游戏版本1 最后更新年 最后更新月 最后更新日 游戏名称     评分 类别    热度 资费
       <int>      <int>      <int>      <int> <chr>      <dbl> <fct> <dbl> <fct>
 1         1       2015          7         29 捕鱼达人     7.6 益智…    95 道具…
 2         1       2014          5         27 NBA2K14      7.5 体育…    94 完全…
 3         6       2016          9         29 愤怒的小…    5.9 益智…    93 道具…
 4         1       2012          7         31 三国志无…    5.1 动作…    90 完全…
 5         2       2016          9         23 地铁跑酷-…   8.4 益智…    89 道具…
 6         1       2016          8         30 火箭飞人     8.2 动作…    89 完全…
 7         1       2015          2         17 暴力摩托…    5.1 动作…    89 完全…
 8         2       2015          7          6 愤怒的小…    9.5 益智…    87 完全…
 9         0       2016          9         30 NBA 2K17…    8   体育…    85 完全…
10         1       2016          7         22 天天酷跑     6.9 益智…    84 完全…
# ℹ 1,130 more rows
# ℹ 9 more variables: 开发商 <fct>, 支持系统 <fct>, 评论数 <dbl>, 喜欢数 <dbl>,
#   中文 <fct>, 英文 <fct>, 韩文 <fct>, 日文 <fct>, 其他 <fct>
```

3.3.3 描述性统计

```
select_colnames <- c('游戏版本1','评分','热度','评论数','喜欢数')
numerical_data_game2 <- data_game2[select_colnames]
summary(numerical_data_game2)
```

```
  游戏版本1             评分            热度          评论数
 Min.   :       0   Min.   :1.000   Min.   :28.0   Min.   :   22
 1st Qu.:       1   1st Qu.:5.200   1st Qu.:33.0   1st Qu.:  494
 Median :       1   Median :6.600   Median :39.0   Median :  880
 Mean   :   18354   Mean   :6.611   Mean   :41.7   Mean   : 2365
 3rd Qu.:       2   3rd Qu.:7.900   3rd Qu.:48.0   3rd Qu.: 2558
 Max.   :20150513   Max.   :9.600   Max.   :95.0   Max.   :96354
 NA's   :3          NA's   :202                    NA's   :21
    喜欢数
 Min.   :    1.00
 1st Qu.:   68.25
 Median :  235.00
 Mean   :  790.45
 3rd Qu.:  860.75
```

```
Max.   :13323.00
NA's   :6
```

```
tab_01 = data.frame(
  Mean  = c(colMeans(numerical_data_game2, na.rm = TRUE)),
  SD = c(sapply(numerical_data_game2, sd, na.rm = TRUE)),
  Median = c(sapply(numerical_data_game2, median, na.rm = TRUE)),
  Min = c(sapply(numerical_data_game2, min, na.rm = TRUE)),
  # 计算数据的偏度
  skew = c(sapply(numerical_data_game2, skewness, na.rm = TRUE)),
  # 计算数据的峰度
  kurt = c(sapply(numerical_data_game2, kurtosis, na.rm = TRUE)),
  Max = c(sapply(numerical_data_game2, max, na.rm = TRUE))
)

# 生成描述性统计表格
kable(
  tab_01,
  col.names = c("均值", "标准差", "中位数","偏度","峰度", "最小", "最大"),
  digits = 2,
  caption = "\\label{tab2}Summary Statistics",
  booktabs = TRUE
)
```

Summary Statistics

	均值	标准差	中位数	偏度	峰度	最小	最大
游戏版本1	18353.73	597810.75	1.0	0	33.64	1133.21	20150513.0
评分	6.61	1.35	6.6	1	-0.04	2.04	9.6
热度	41.70	11.61	39.0	28	1.32	5.27	95.0
评论数	2365.30	4902.85	880.0	22	9.82	155.01	96354.0
喜欢数	790.45	1395.16	235.0	1	3.65	20.90	13323.0

绘制云雨图

```
# 自定义云雨图函数
"%||%" <- function(a, b) {
  if (!is.null(a)) a else b
}


geom_flat_violin <- function(mapping = NULL, data = NULL, stat = "ydensity",
                             position = "dodge", trim = TRUE, scale = "area",
                             show.legend = NA, inherit.aes = TRUE, ...) {
  layer(
    data = data,
    mapping = mapping,
    stat = stat,
    geom = GeomFlatViolin,
    position = position,
    show.legend = show.legend,
```

```
    inherit.aes = inherit.aes,
    params = list(
      trim = trim,
      scale = scale,
      ...
    )
  )
}
GeomFlatViolin <-
  ggproto("GeomFlatViolin", Geom,
          setup_data = function(data, params) {
            data$width <- data$width %||%
              params$width %||% (resolution(data$x, FALSE) * 0.9)


            # ymin, ymax, xmin, and xmax define the bounding rectangle for each group
            data %>%
              group_by(group) %>%
              mutate(ymin = min(y),
                     ymax = max(y),
                     xmin = x,
                     xmax = x + width / 2)


          },


          draw_group = function(data, panel_scales, coord) {
            # Find the points for the line to go all the way around
            data <- transform(data, xminv = x,
                              xmaxv = x + violinwidth * (xmax - x)) #利用transform函数为数据框r


            newdata <- rbind(plyr::arrange(transform(data, x = xmaxv), -y),plyr::arrange(trans
            newdata_Polygon <- rbind(newdata, newdata[1,])
            newdata_Polygon$colour<-NA


            newdata_Path <- plyr::arrange(transform(data, x = xmaxv), -y)


            ggplot2:::ggname("geom_flat_violin", grobTree(
              GeomPolygon$draw_panel(newdata_Polygon, panel_scales, coord),
              GeomPath$draw_panel(newdata_Path, panel_scales, coord))
            )
          },


          draw_key = draw_key_polygon,


          default_aes = aes(weight = 1, colour = "grey20", fill = "white", size = 0.5,
                            alpha = NA, linetype = "solid"),
```

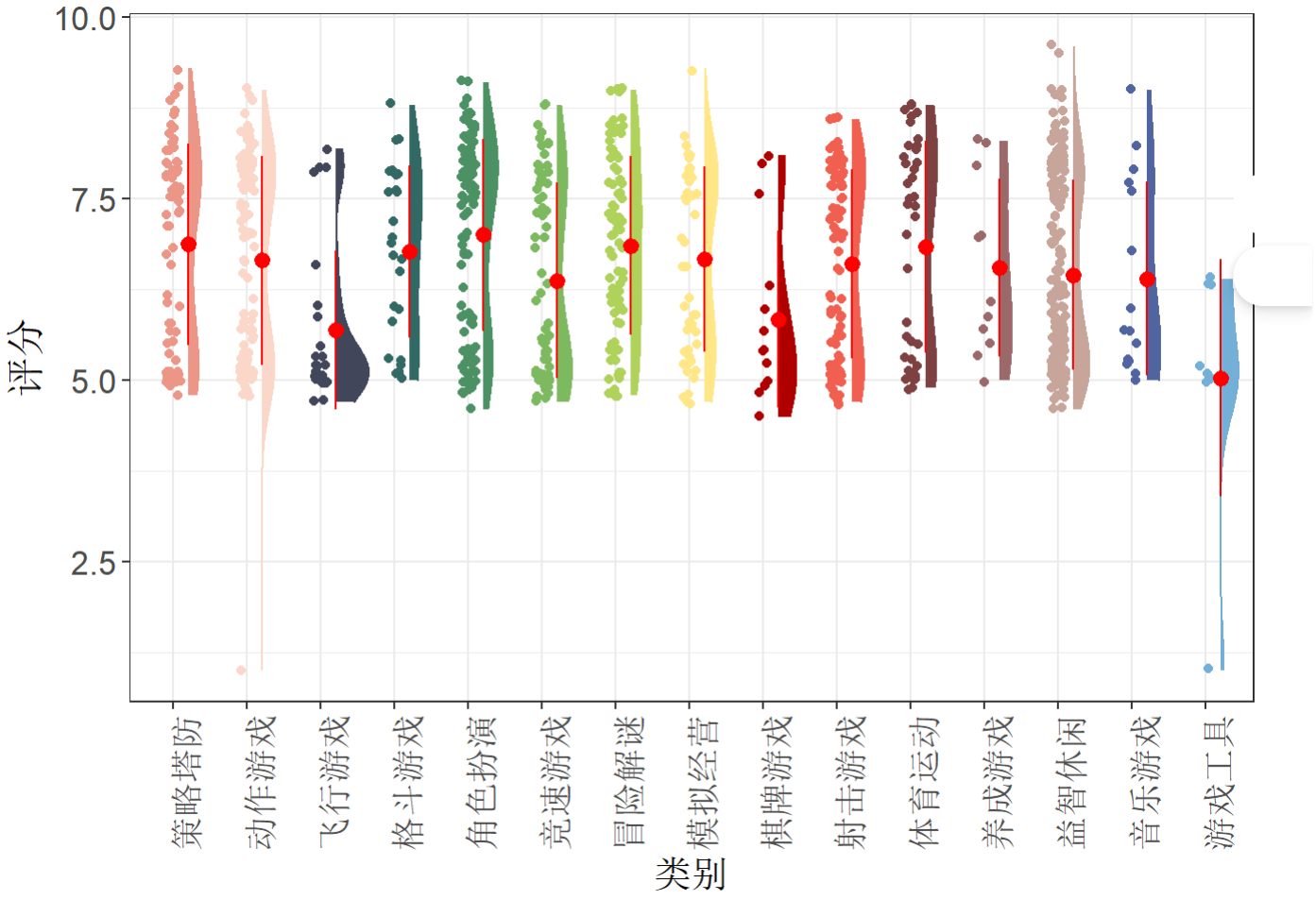
```
        required_aes = c("x", "y")
  )
```

```
library(grid)
library(RColorBrewer)
library(SuppDists) #提供rJohnson()函数

#为了画图方便，舍弃评分列缺少的行
data_game3 <- filter(data_game2, !is.na(`评分`))

f2.data <- data_game3[c('类别','评分')]
colnames(f2.data) <- c("类别","评分")
table(f2.data$类别)
```

```
策略塔防 动作游戏 飞行游戏 格斗游戏 角色扮演 竞速游戏 冒险解谜 模拟经营
      71      118       25       25      103       69       84       49
棋牌游戏 射击游戏 体育运动 养成游戏 益智休闲 音乐游戏 游戏工具
      14       96       39       11      210       15        9
```

```
# 定义颜色
mycol22 <- c("#eb998b", "#fddbc8", "#42465c", "#356d67", "#4c9568",
             "#7fb961", "#b0d45d", "#ffe788", "#b20000", "#f06152",
             "#7d4444", "#9e6c69", "#cca69c", "#5066a1", "#76afda",
             "#abddff", "#dcf2ff", "#e8743c", "#ffc556")

# 计算每个类别的均值和标准差
d <- group_by(f2.data, 类别) %>%
  summarize(mean = mean(评分),
            sd = sd(评分))

# 创建图形
ggplot(f2.data, aes(类别, 评分, fill = 类别, color = 类别)) +
  geom_flat_violin(position = position_nudge(x = .2)) +
  geom_jitter(width = .1) +
  geom_pointrange(aes(y = mean, ymin = mean - sd, ymax = mean + sd),
                  data = d, size = 0.5, position = position_nudge(x = .2),color = 'red') +
  scale_fill_manual(values = mycol22) +  # 设置填充颜色
  scale_color_manual(values = mycol22) +  # 设置点的颜色
  theme_bw() +
  theme(axis.text.x = element_text(size = 13, angle = 90, hjust = 1),  # 设置x轴标签垂直
        axis.text.y = element_text(size = 13),
        axis.title = element_text(size = 15),
        legend.position = "none")
```

```
Warning: Using the `size` aesthetic with geom_polygon was deprecated in ggplot2 3.4.0.
ℹ Please use the `linewidth` aesthetic instead.

Warning: Using the `size` aesthetic with geom_path was deprecated in ggplot2 3.4.0.
ℹ Please use the `linewidth` aesthetic instead.
```



3.4 数据挖掘

3.4.1 对'热度'和'类别'的分组探索

```
# 对'热度'进行分组统计
tab_01 <- data_game2 %>%
  group_by(类别) %>%
  summarise(
    Mean = mean(热度, na.rm = TRUE),
    SD = sd(热度, na.rm = TRUE),
    Median = median(热度, na.rm = TRUE),
    Min = min(热度, na.rm = TRUE),
    skew = skewness(热度, na.rm = TRUE),
    kurt = kurtosis(热度, na.rm = TRUE),
    Max = max(热度, na.rm = TRUE)
  )

# 生成描述性统计表格
kable(
  tab_01,
  col.names = c("类别", "均值", "标准差", "中位数", "最小", "偏度", "峰度", "最大"),
  digits = 2,
  caption = "\\label{tab2}Summary Statistics by Category",
  booktabs = TRUE
)
```

Summary Statistics by Category

类别	均值	标准差	中位数	最小	偏度	峰度	最大
策略塔防	40.89	10.28	39.0	28	0.83	2.97	70
动作游戏	42.60	12.85	40.0	28	1.54	5.79	
飞行游戏	41.65	10.06	41.0	29	1.63	6.69	7
格斗游戏	40.57	9.85	40.0	28	0.60	2.61	
角色扮演	42.19	10.21	39.0	28	0.56	2.19	68
竞速游戏	45.61	12.56	44.0	28	0.78	3.03	79
冒险解谜	39.52	10.09	37.0	28	0.77	2.58	64
模拟经营	38.32	7.83	37.0	28	0.52	2.30	57
棋牌游戏	37.06	8.83	33.0	28	0.83	2.37	56
射击游戏	39.96	9.73	39.0	28	1.47	6.57	84
体育运动	45.98	16.18	41.0	28	1.09	3.57	94
养成游戏	42.33	14.62	37.5	29	1.53	4.64	80
益智休闲	41.80	12.57	39.0	28	1.49	5.72	95
音乐游戏	40.60	8.86	40.0	28	0.32	2.42	58
游戏工具	43.62	10.60	48.0	28	-0.14	1.93	62

```
# 对 tab_01 按 Mean 进行降序排序，并提取平均热度最高的游戏类型
highest_mean_category <- tab_01 %>%
  arrange(desc(Mean)) %>%
  slice(1) %>%
  pull(类别)

# 打印结果
print(paste('平均热度最高的游戏类型是：', highest_mean_category))
```

```
[1] "平均热度最高的游戏类型是： 体育运动"
```

3.4.2 对'评分'和'类别'的分组探索

```
# 对'评分'进行分组统计
tab_01 <- data_game2 %>%
  group_by(类别) %>%
  summarise(
    Mean = mean(评分, na.rm = TRUE),
    SD = sd(评分, na.rm = TRUE),
    Median = median(评分, na.rm = TRUE),
    Min = min(评分, na.rm = TRUE),
    skew = skewness(评分, na.rm = TRUE),
    kurt = kurtosis(评分, na.rm = TRUE),
    Max = max(评分, na.rm = TRUE)
  )
```

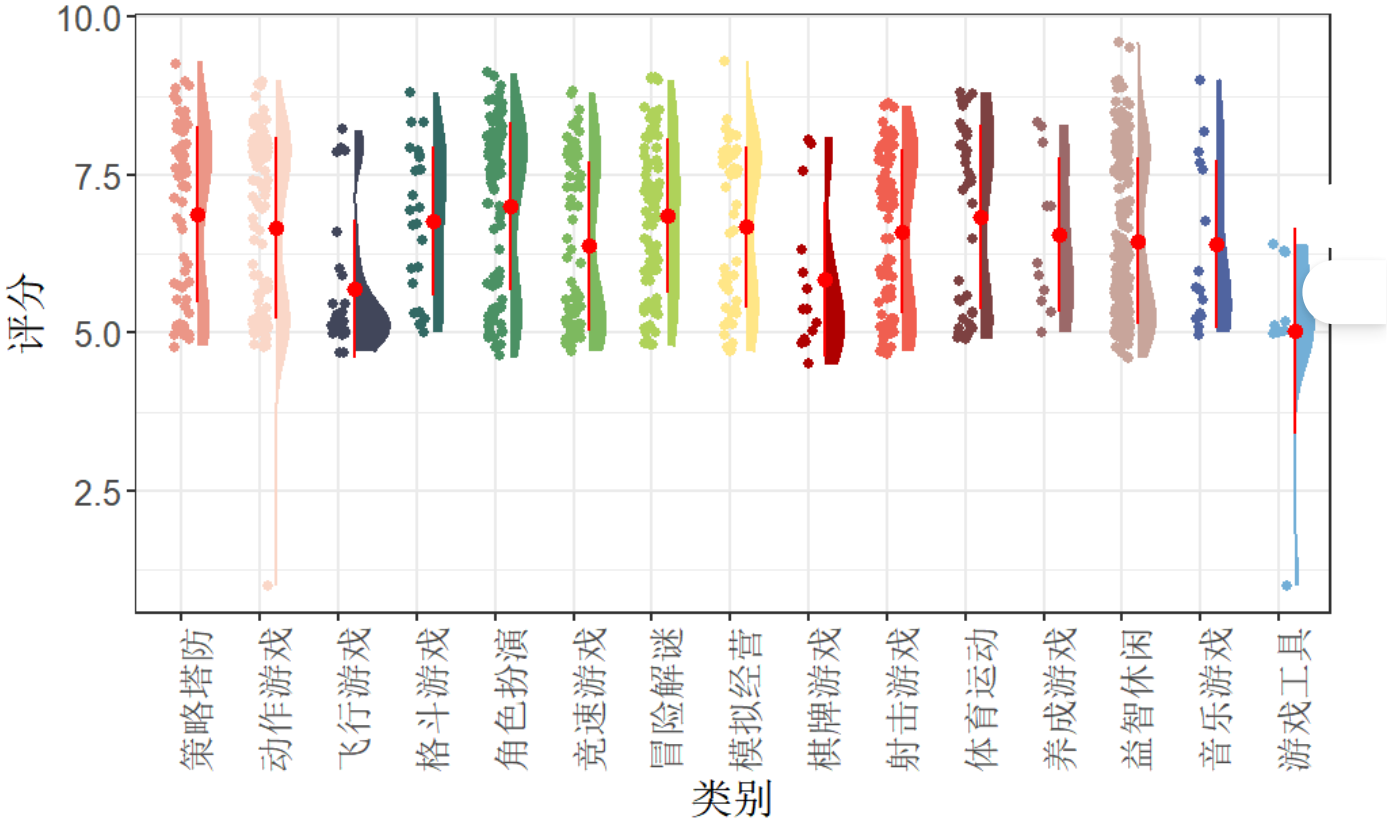
```r
# 生成描述性统计表格
kable(
  tab_01,
  col.names = c("类别", "均值", "标准差", "中位数", "最小", "偏度", "峰度", "最大"),
  digits = 2,
  caption = "\\label{tab2}Summary Statistics by Category",
  booktabs = TRUE
)
```

Summary Statistics by Category

类别	均值	标准差	中位数	最小	偏度	峰度	最大
策略塔防	6.88	1.39	7.30	4.8	-0.17	1.54	9.3
动作游戏	6.65	1.44	7.00	1.0	-0.50	3.09	9.0
飞行游戏	5.70	1.09	5.20	4.7	1.41	3.47	8.2
格斗游戏	6.78	1.18	6.90	5.0	-0.13	1.72	8.8
角色扮演	7.00	1.33	7.50	4.6	-0.35	1.66	9.1
竞速游戏	6.37	1.35	5.80	4.7	0.30	1.48	8.8
冒险解谜	6.86	1.23	6.95	4.8	0.03	1.89	9.0
模拟经营	6.67	1.28	6.90	4.7	-0.06	1.60	9.3
棋牌游戏	5.84	1.22	5.40	4.5	0.92	2.41	8.1
射击游戏	6.61	1.30	7.00	4.7	-0.09	1.40	8.6
体育运动	6.84	1.46	7.40	4.9	-0.13	1.34	8.8
养成游戏	6.55	1.23	6.10	5.0	0.34	1.62	8.3
益智休闲	6.46	1.31	6.10	4.6	0.36	1.70	9.6
音乐游戏	6.40	1.34	5.70	5.0	0.62	1.90	9.0
游戏工具	5.03	1.63	5.10	1.0	-1.79	5.38	6.4

3.4.3 数据分析

回顾前面的云雨图：



1. **游戏特性分析**

- **角色扮演游戏（RPG）**：

  高平均评分（均值 7.00）和**中位数**（7.50）表明，RPG 游戏普遍受欢迎。RPG 通常具有深入的剧情、复杂的角色成长系统，这些特点吸引了偏好长时间投入游戏的用户。

  较低的偏度和峰度：评分分布较为均匀，说明大部分 RPG 游戏都有稳定的玩家群体，不会因为少数特别热门或冷门的游戏拉大差距。

  用户画像：这些游戏通常吸引偏好深度体验、喜欢投入时间发展的用户。用户可能偏向中青年群体，男性用户比例较高，且有一定的收入和消费能力，愿意在游戏中进行内购以提升游戏体验。

- **策略塔防**：

  高评分且分布较为均匀：策略塔防游戏的平均评分较高，说明玩家对这类需要深思熟虑和战略规划的游戏有较高的需求。

  用户画像：这类游戏通常吸引喜欢挑战思维和策略的玩家，可能偏向高智商和耐心的用户群体。

- **飞行游戏**：

  低平均评分和**较高偏度**：飞行游戏整体评分较低，但存在少数非常受欢迎的游戏（如模拟飞行或空战游戏），但大部分游戏评分均比较低。

  用户画像：这类游戏可能吸引的是寻求刺激感和成就感的用户，例如飞行爱好者或军事迷，故而对游戏的特定部分具有较高的仿真要求，只有少数制作精美的游戏才能够获得玩家认可，侧面反应了该领域游戏制作的门槛较高。

- **游戏工具**：

低评分和高偏度：游戏工具的评分分布显示出较大的波动，主要是由于该类型的软件数量不足、质量参差不齐的原因。说明这个类别的产品质量或用户需求差异较大，开发市场前景较高。部分工具类应用可能非常有用，而其他则可能因为功能或体验不足而被冷落，用户可能更多是硬核玩家，他们需要辅助工具来提升游戏体验。不过相较于传统的游戏，辅助工具获得的评分一般来说并不和游戏具有统一的参考标准。

2. **市场趋势与用户行为**

- **多样化需求与长尾效应**：

  从数据中的高峰度和高偏度现象可以看出，许多游戏类型可能存在"长尾效应"，即大部分游戏的评分较为平均，但少数特别受欢迎的游戏则占据了大量市场份额。这表明，虽然玩家对主流游戏类型有共识，但对于特定类型的利基市场需求依然存在。

- **游戏工具的应用**：

  尽管平均评分较低，游戏工具类应用显示出较大的潜力，特别是对于专业玩家或某些硬核游戏的玩家群体。随着游戏复杂度和对专业化需求的增加，这一类应用可能会进一步发展。

- **变化的用户兴趣**：

  例如，休闲益智游戏虽然整体评分适中，但最大评分达到9.6，表明这类游戏依然能产生爆款。结合移动互联网的发展，轻度和快餐式的游戏体验仍然是一个重要的市场。

3. **进一步的商业策略建议**

- **针对硬核玩家的深度产品**：

  RPG、策略塔防和射击游戏等类型可以通过更高质量的内容、更复杂的游戏机制吸引核心玩家，并通过内购和订阅模式增加收入。

- **扩展休闲游戏的市场**：

  休闲游戏的开发者可以考虑通过简单的游戏机制和社交互动功能吸引更广泛的用户群体，特别是通过社交媒体平台推广，以便迅速获取用户。

- **工具类应用的商业潜力**：

  为游戏工具应用开发更专业和更精确的功能，特别是在电竞领域，可以吸引那些对游戏提升有强烈需求的用户群体，从而增加用户粘性和变现能力。

3.4.4 其他有趣的性质1：变量相关热图

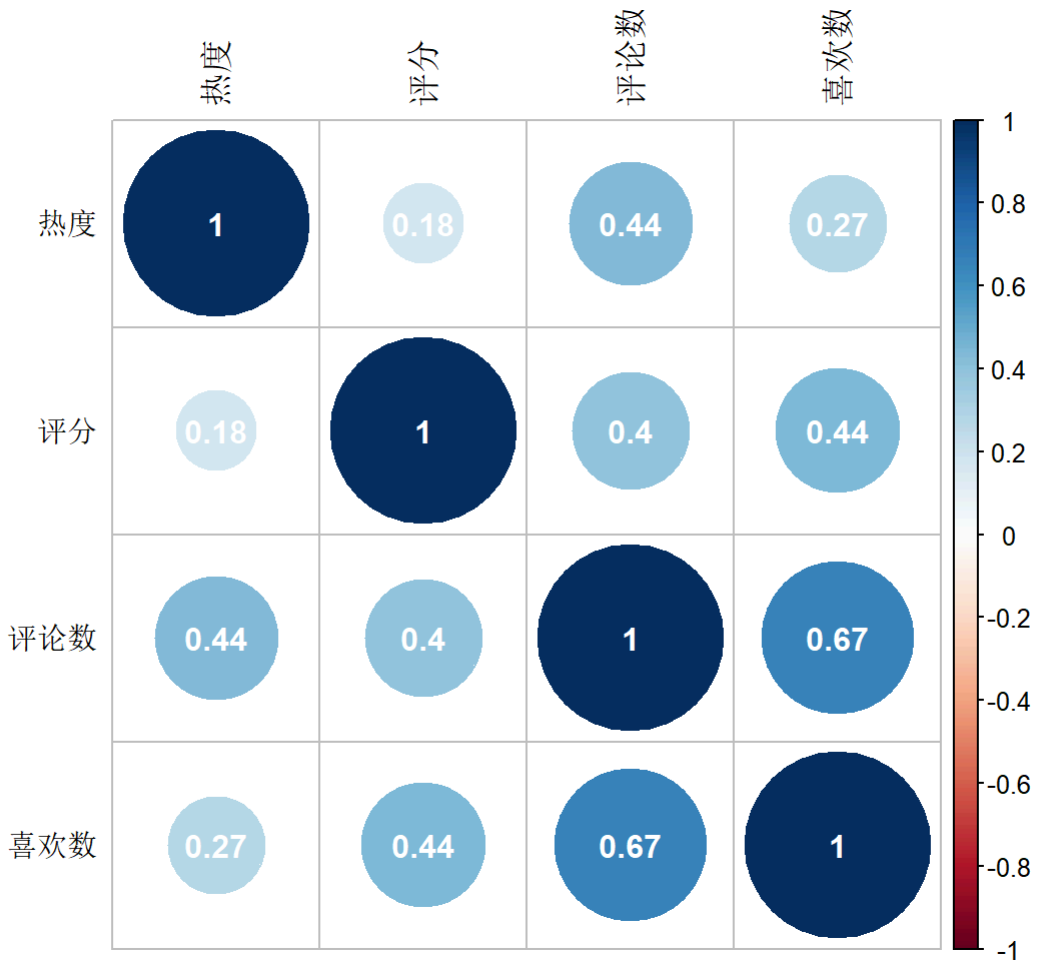
```
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
data_game4 <- filter(data_game3,!is.na(`评分`)&!is.na(`热度`)&!is.na(`评论数`)&!is.na(`喜欢数`
# 相关系数矩阵
correlations <- cor(data_game4[c('评分','热度','评论数','喜欢数')])
dim(correlations)
```

```
[1] 4 4
```

```
# 可视化相关系数矩阵
corrplot(correlations, order = "hclust",tl.col = "black", addCoef.col = "white")
```



通过该图可以发现某些性质：

- 喜欢数和评论数相关性较高，可能因为喜欢数和评论数都是在游戏介绍界面的交互，会在游戏商品栏界面点击“喜欢”的人往往也会顺便进行评论留言，反之亦然。
- 评分和热度之间相关性并不高，可能热度往往表现的是一个游戏短期内的“流量”而并非长期的“质量”。
- 评分与“评论数”、“喜欢数”的相关系数相近，也就意味着后两者所反应的“游戏质量”的程度相近，但是评论数和热度的相关系数明显高于喜欢数和热度的相关系数，也说明了其实一个游戏的热度本质上被如何衡量，高热度的游戏往往引发更多的讨论，但这些褒贬不一的评论并不会带来等效程度的喜欢数，即高热度并不意味着高评分。

3.4.5 其他变量的云雨图

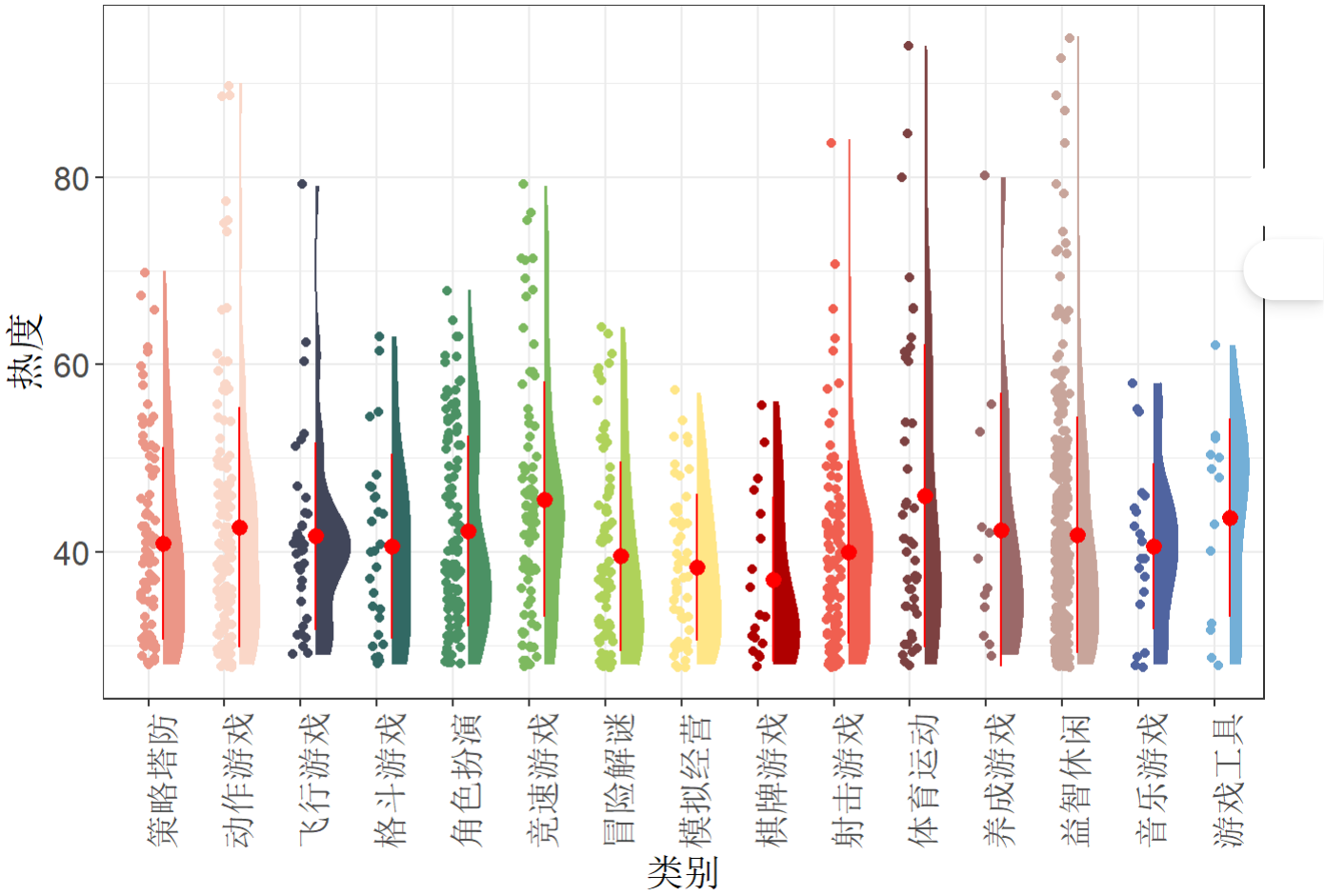
1. **热度**

```
library(grid)
library(RColorBrewer)
library(SuppDists) #提供rJohnson()函数

#为了画图方便，舍弃评分列缺少的行
```

```r
data_game3 <- filter(data_game2, !is.na(`热度`))

f2.data <- data_game3[c('类别','热度')]
colnames(f2.data) <- c("类别","热度")
table(f2.data$类别)
```

```
策略塔防 动作游戏 飞行游戏 格斗游戏 角色扮演 竞速游戏 冒险解谜 模拟经营
      90      136       37       28      114       85       86       53
棋牌游戏 射击游戏 体育运动 养成游戏 益智休闲 音乐游戏 游戏工具
      18      110       46       12      292       20       13
```

```r
# 定义颜色
mycol22 <- c("#eb998b", "#fddbc8", "#42465c", "#356d67", "#4c9568",
             "#7fb961", "#b0d45d", "#ffe788", "#b20000", "#f06152",
             "#7d4444", "#9e6c69", "#cca69c", "#5066a1", "#76afda",
             "#abddff", "#dcf2ff", "#e8743c", "#ffc556")

# 计算每个类别的均值和标准差
d <- group_by(f2.data, 类别) %>%
  summarize(mean = mean(热度),
            sd = sd(热度))

# 创建图形
ggplot(f2.data, aes(类别, 热度, fill = 类别, color = 类别)) +
  geom_flat_violin(position = position_nudge(x = .2)) +
  geom_jitter(width = .1) +
  geom_pointrange(aes(y = mean, ymin = mean - sd, ymax = mean + sd),
                  data = d, size = 0.5, position = position_nudge(x = .2),color = 'red') +
  scale_fill_manual(values = mycol22) +  # 设置填充颜色
  scale_color_manual(values = mycol22) +  # 设置点的颜色
  theme_bw() +
  theme(axis.text.x = element_text(size = 13, angle = 90, hjust = 1),  # 设置x轴标签垂直
        axis.text.y = element_text(size = 13),
        axis.title = element_text(size = 15),
        legend.position = "none")
```



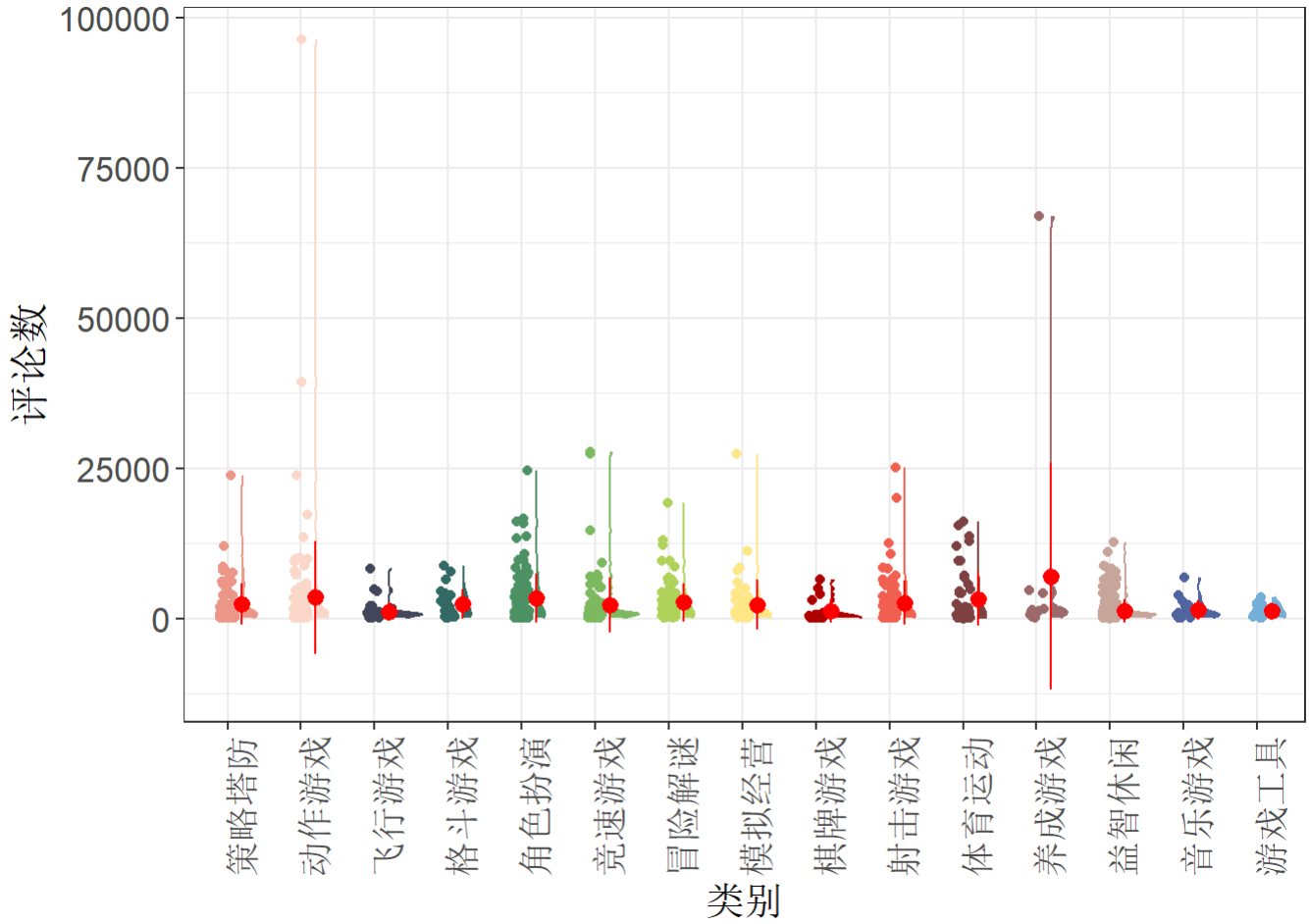
首先，热度往往呈现的是非对称、有偏度的分布，主要的极端值都聚集在高热度上，也可能因为过低的热度往往被截断为0而不会出现较低值。

进一步，可以看到近期游戏热度大部分集中在**动作游戏、体育运动、益智休闲**等类型游戏，可能意味着此类型游戏是近期游戏市场热度风口，有一定挖掘潜力的发展空间。反之**模拟经营、棋牌游戏、音乐游戏**等类别热度较集中，没有出现极端热度的情况，说明该类型游戏的领域整体在此时并不怎么流行，再对比评分的分布情况，可以发现这些类型游戏普遍评分也不高，猜测这些类型的游戏公司还没有做出足够优秀的作品，整体上较难获得玩家的认可。

2. **评论数**

```
library(grid)
library(RColorBrewer)
library(SuppDists) #提供rJohnson()函数

#为了画图方便，舍弃评论数列缺少的行
data_game3 <- filter(data_game2, !is.na(`评论数`))

f2.data <- data_game3[c('类别','评论数')]
colnames(f2.data) <- c("类别","评论数")
# 定义颜色
mycol22 <- c("#eb998b", "#fddbc8", "#42465c", "#356d67", "#4c9568",
             "#7fb961", "#b0d45d", "#ffe788", "#b20000", "#f06152",
             "#7d4444", "#9e6c69", "#cca69c", "#5066a1", "#76afda",
             "#abddff", "#dcf2ff", "#e8743c", "#ffc556")

# 计算每个类别的均值和标准差
```

```r
d <- group_by(f2.data, 类别) %>%
  summarize(mean = mean(评论数),
            sd = sd(评论数))

# 创建图形
ggplot(f2.data, aes(类别, 评论数, fill = 类别, color = 类别)) +
  geom_flat_violin(position = position_nudge(x = .2)) +
  geom_jitter(width = .1) +
  geom_pointrange(aes(y = mean, ymin = mean - sd, ymax = mean + sd),
                  data = d, size = 0.5, position = position_nudge(x = .2),color = 'red') +
  scale_fill_manual(values = mycol22) +  # 设置填充颜色
  scale_color_manual(values = mycol22) +  # 设置点的颜色
  theme_bw() +
  theme(axis.text.x = element_text(size = 13, angle = 90, hjust = 1),  # 设置x轴标签垂直
        axis.text.y = element_text(size = 13),
        axis.title = element_text(size = 15),
        legend.position = "none")
```



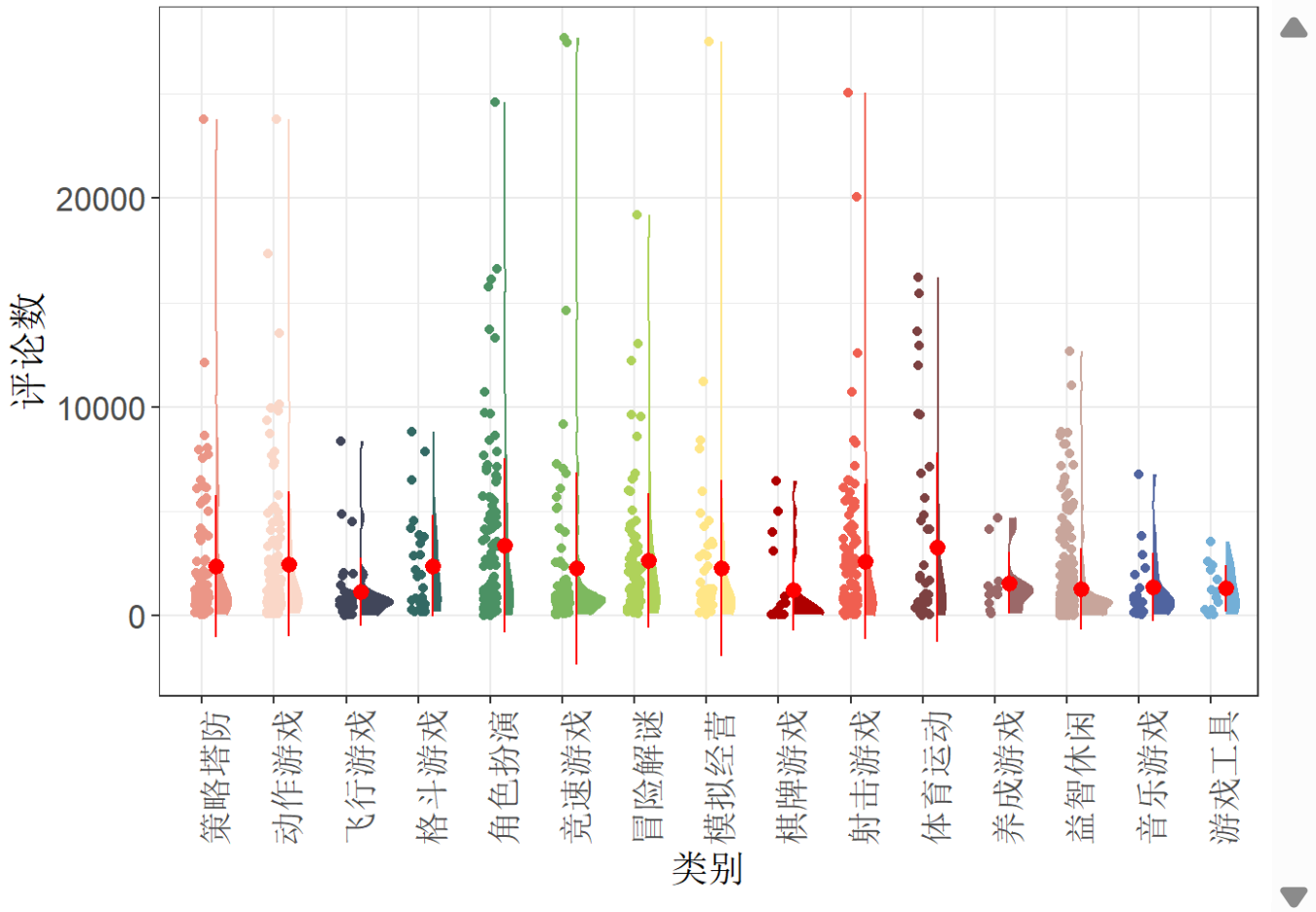
这里图画出来显然不太好看，因为有极端的评论数破坏了整体的范围，我们先找出这些极端的评论数并尝试删去最多的3个，再进行画图：

```r
# 找出评论数最高的三个数据点
(top_3_outliers <- data_game3  %>%
  top_n(3, wt = 评论数))
```

```
# A tibble: 3 × 18
  游戏版本1 最后更新年 最后更新月 最后更新日 游戏名称      评分 类别     热度 资费
```

```
      <int>      <int>      <int>      <int> <chr>       <dbl> <fct> <dbl> <fct>
1         0       2016          9         21 我的世界移…   8.3 养成…    80 完全…
2         1       2016          4          1 GTA侠盗猎…   8.1 动作…    77 完全…
3         1       2015          4          2 GTA侠盗猎…   7.8 动作…    75 完全…
# ℹ 9 more variables: 开发商 <fct>, 支持系统 <fct>, 评论数 <dbl>, 喜欢数 <dbl>,
#   中文 <fct>, 英文 <fct>, 韩文 <fct>, 日文 <fct>, 其他 <fct>
```

```r
# 从数据中删除这三个数据点
f2.data <- anti_join(data_game3, top_3_outliers, by = c("类别", "评论数"))[c('类别','评论数')]
colnames(f2.data) <- c("类别","评论数")
# 计算每个类别的均值和标准差
d <- group_by(f2.data, 类别) %>%
  summarize(mean = mean(评论数),
            sd = sd(评论数))

# 创建图形
ggplot(f2.data, aes(类别, 评论数, fill = 类别, color = 类别)) +
  geom_flat_violin(position = position_nudge(x = .2)) +
  geom_jitter(width = .1) +
  geom_pointrange(aes(y = mean, ymin = mean - sd, ymax = mean + sd),
                  data = d, size = 0.5, position = position_nudge(x = .2),color = 'red') +
  scale_fill_manual(values = mycol22) +  # 设置填充颜色
  scale_color_manual(values = mycol22) +  # 设置点的颜色
  theme_bw() +
  theme(axis.text.x = element_text(size = 13, angle = 90, hjust = 1),  # 设置x轴标签垂直
        axis.text.y = element_text(size = 13),
        axis.title = element_text(size = 15),
        legend.position = "none")
```

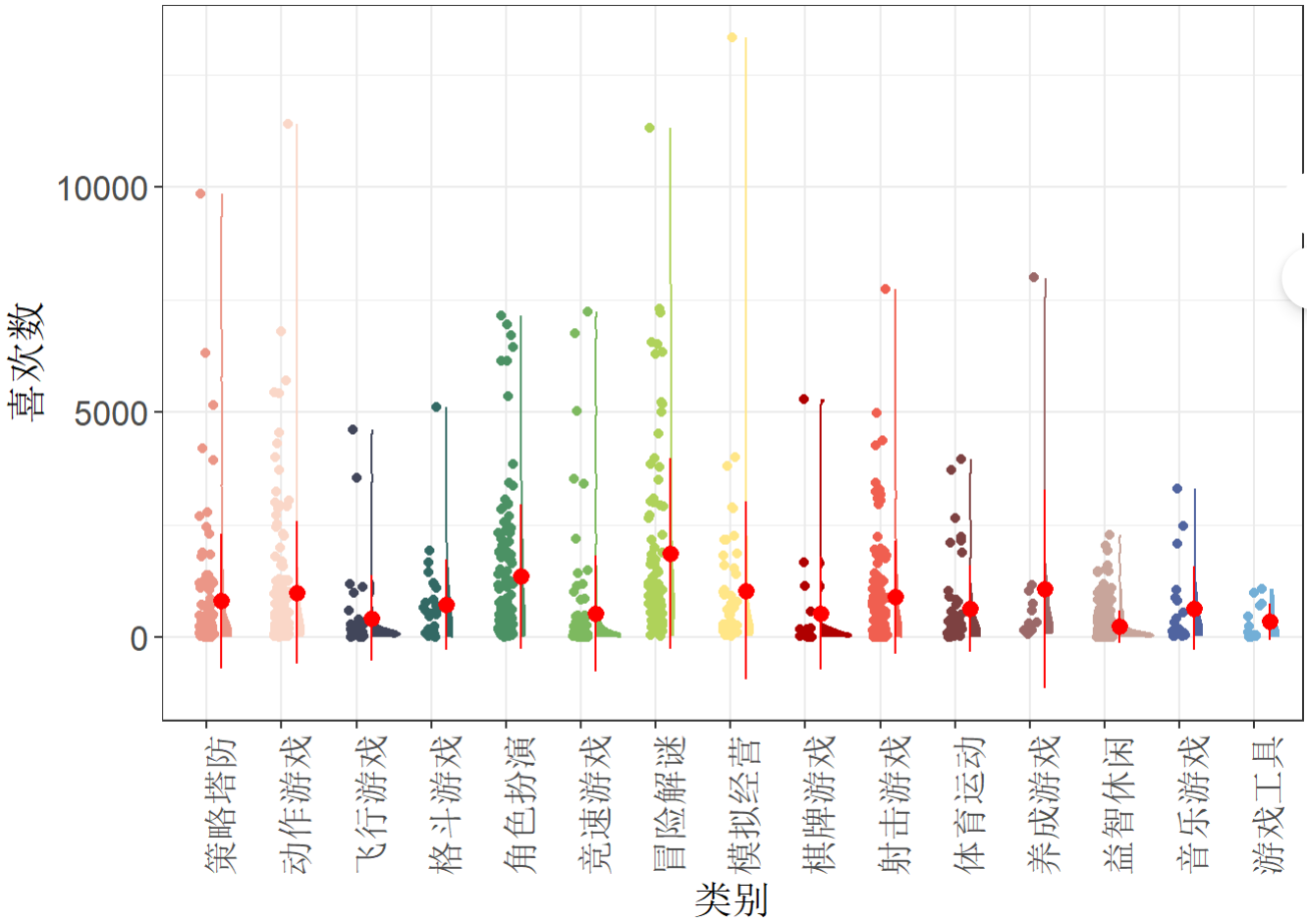


评论数断崖式领先的三个游戏分别为：**GTA侠盗猎车手:圣安地列斯、我的世界移动版、GTA侠盗猎车手:罪恶都市**，这些作为当年的知名游戏拥有大量的评论并不奇怪，甚至当之无愧。

对于其他类型的游戏，大部分游戏都较为集中，同样只有少部分游戏拥有几乎"异常"高的评论数，这正说明了游戏市场的厚尾性，即少部分高质量游戏断崖式遥遥领先大部分平庸的游戏。

3. **喜欢数**

```
#为了画图方便，舍弃评论数列缺少的行
data_game3 <- filter(data_game2, !is.na(`喜欢数`))

f2.data <- data_game3[c('类别','喜欢数')]
colnames(f2.data) <- c("类别","喜欢数")
# 计算每个类别的均值和标准差
d <- group_by(f2.data, 类别) %>%
  summarize(mean = mean(喜欢数),
            sd = sd(喜欢数))

# 创建图形
ggplot(f2.data, aes(类别, 喜欢数, fill = 类别, color = 类别)) +
  geom_flat_violin(position = position_nudge(x = .2)) +
  geom_jitter(width = .1) +
  geom_pointrange(aes(y = mean, ymin = mean - sd, ymax = mean + sd),
                  data = d, size = 0.5, position = position_nudge(x = .2),color = 'red') +
  scale_fill_manual(values = mycol22) +  # 设置填充颜色
  scale_color_manual(values = mycol22) +  # 设置点的颜色
  theme_bw() +
  theme(axis.text.x = element_text(size = 13, angle = 90, hjust = 1),  # 设置x轴标签垂直
        axis.text.y = element_text(size = 13),
        axis.title = element_text(size = 15),
        legend.position = "none")
```



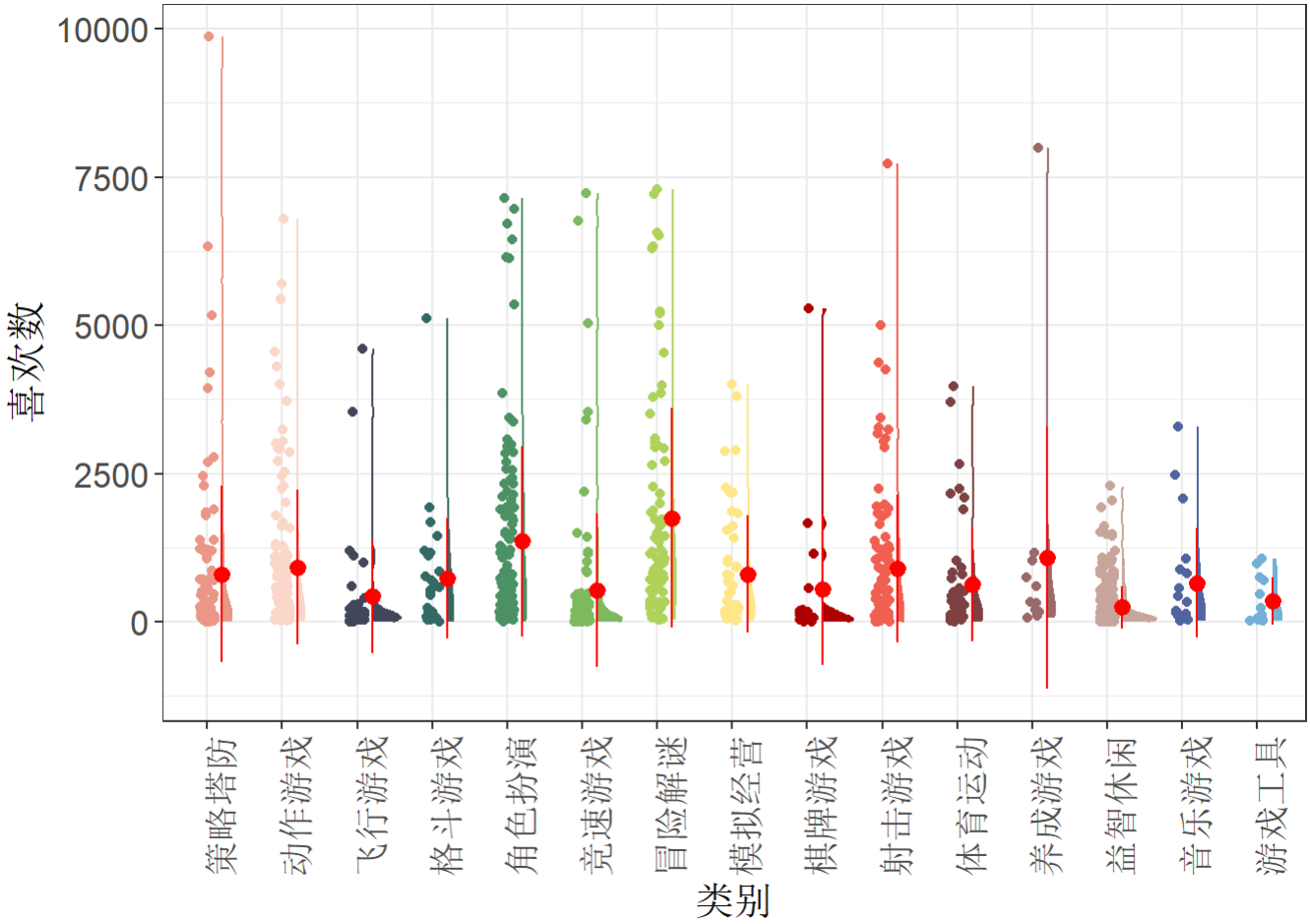
还是通过肉眼筛查极端异常值的数量，去掉最高的3个

```
# 找出评论数最高的三个数据点
(top_3_outliers <- data_game3  %>%
  top_n(3, wt = 喜欢数))
```

```
# A tibble: 3 × 18
  游戏版本1 最后更新年 最后更新月 最后更新日 游戏名称      评分 类别    热度 资费
      <int>      <int>      <int>      <int> <chr>       <dbl> <fct> <dbl> <fct>
1         1       2016          4          1 GTA侠盗猎…     8.1 动作…    77 完全…
2         1       2014         12         11 勇敢的心:…     9   冒险…    59 完全…
3         1       2016          3         18 这是我的战…    9.3 模拟…    54 完全…
# ℹ 9 more variables: 开发商 <fct>, 支持系统 <fct>, 评论数 <dbl>, 喜欢数 <dbl>,
#   中文 <fct>, 英文 <fct>, 韩文 <fct>, 日文 <fct>, 其他 <fct>
```

```
# 从数据中删除这三个数据点
f2.data <- anti_join(data_game3, top_3_outliers, by = c("类别", "喜欢数"))[c('类别','喜欢数')]
colnames(f2.data) <- c("类别","喜欢数")
# 计算每个类别的均值和标准差
d <- group_by(f2.data, 类别) %>%
  summarize(mean = mean(喜欢数),
            sd = sd(喜欢数))

# 创建图形
ggplot(f2.data, aes(类别, 喜欢数, fill = 类别, color = 类别)) +
  geom_flat_violin(position = position_nudge(x = .2)) +
  geom_jitter(width = .1) +
```

```
  geom_pointrange(aes(y = mean, ymin = mean - sd, ymax = mean + sd),
                  data = d, size = 0.5, position = position_nudge(x = .2),color = 'red') +
  scale_fill_manual(values = mycol22) +  # 设置填充颜色
  scale_color_manual(values = mycol22) +  # 设置点的颜色
  theme_bw() +
  theme(axis.text.x = element_text(size = 13, angle = 90, hjust = 1),  # 设置x轴标签垂直
        axis.text.y = element_text(size = 13),
        axis.title = element_text(size = 15),
        legend.position = "none")
```



数据分布的结论大致与评论数类似，但是除了游戏"GTA侠盗猎车手:圣安地列斯"仍然位居最高喜欢数的前几位，其余游戏并不和原来评论数的最高前3位一致，这也进一步说明了喜爱数和评论数并非高度相关。

3.5 机器学习预测分析方法

3.5.1 库导入与数据集分割

```
# 加载必要的包
library(dplyr)
library(caret)  # 用于数据集的分割
```

```
Loading required package: lattice
```

```
library(lightgbm)  # 用于LightGBM模型
library(Metrics)
```

```
Attaching package: 'Metrics'

The following objects are masked from 'package:caret':

    precision, recall
```

```r
# 将指定列转换为因子类型
factor_columns <- c('类别', '资费', '开发商', '支持系统', '中文', '韩文', '英文', '日文', '其他
data_game3 <- filter(data_game2, !is.na(`评分`))
data_game3 <- data_game3 %>%
  mutate_at(factor_columns, as.factor)

# 分割数据集为训练集和测试集7:3
set.seed(123)
train_index <- createDataPartition(data_game3$评分, p = 0.7, list = FALSE)
train_data <- data_game3[train_index, ]
test_data <- data_game3[-train_index, ]

# 准备LightGBM所需的数据格式
# 1. 将训练集和测试集中的目标列（评分）和特征分开
y_train <- train_data$评分
y_test <- test_data$评分

# 再将训练集分为训练子集和验证集，按照80:20的比例
train_sub_index <- createDataPartition(train_data$评分, p = 0.8, list = FALSE)
train_data <- train_data[train_sub_index, ]
valid_data <- train_data[-train_sub_index, ]
y_train <- y_train[train_sub_index]
y_valid <- y_train[-train_sub_index]


# 删除目标列和 '游戏名称' 列，保留自变量
train_data <- train_data %>% select(-评分, -游戏名称)
valid_data <- valid_data %>% select(-评分, -游戏名称)
test_data <- test_data %>% select(-评分, -游戏名称)

train_matrix <- as.matrix(train_data)
valid_matrix <- as.matrix(valid_data)
test_matrix <- as.matrix(test_data)
train_lgb <- lgb.Dataset(data = train_matrix, label = y_train)
```

```
Warning in storage.mode(data) <- "double": NAs introduced by coercion
```

```r
valid_lgb <- lgb.Dataset(data = valid_matrix, label = y_valid)
```

```
Warning in storage.mode(data) <- "double": NAs introduced by coercion
```

3.5.2 构建预测模型

```
# 设置LightGBM的参数
params <- list(
  boosting_type = "gbdt",
  objective = "regression",
  metric = "rmse",
  learning_rate = 0.01,
  feature_fraction = 0.9
)
# 设置验证集
valids <- list(train = train_lgb, valid = valid_lgb)

# 训练LightGBM模型
lgb <- lgb.train(
  params = params,
  data = train_lgb,
  nrounds = 500,  # 训练轮数
  valids = valids,
  early_stopping_rounds = 10,  # 如果验证集10轮没有提升，提前停止
  verbose = -1
)
```

3.5.3 模型效果评估

我们这里只构建一个简单的预测模型baseline来观察一下数据之间在“预测”上的交互联系，通过简单的调参，大致可以得知拟合效果一般，原因有可能是

- 数据集过小大致只有900多条，传统的机器学习模型一般在万为单位的数据集上表现更好，在小数据集上容易出现参数过多导致的过拟合。
- 数据集本身内含的信息过少，不足以提供足够的细节来预测评分。

```
# 对训练集进行预测并计算各项指标
predicted_train <- predict(lgb, train_matrix)
```

```
Warning in storage.mode(data) <- "double": NAs introduced by coercion
```

```
train_rmse <- rmse(y_train, predicted_train)
train_mse <- mean((predicted_train - y_train)^2)
sst_train <- sum((y_train - mean(y_train))^2)
sse_train <- sum((y_train - predicted_train)^2)
r_squared_train <- 1 - sse_train / sst_train

# # 打印训练集结果
# print(paste("训练集 RMSE:", train_rmse))
# print(paste("训练集 MSE:", train_mse))
# print(paste("训练集 R²:", r_squared_train))

# 对测试集进行预测并计算各项指标
predicted_test <- predict(lgb, test_matrix)
```

```
Warning in storage.mode(data) <- "double": NAs introduced by coercion
```

```
test_rmse <- rmse(y_test, predicted_test)
test_mse <- mean((predicted_test - y_test)^2)
sst_test <- sum((y_test - mean(y_test))^2)
sse_test <- sum((y_test - predicted_test)^2)
r_squared_test <- 1 - sse_test / sst_test
#
# # 打印测试集结果
# print(paste("测试集 RMSE:", test_rmse))
# print(paste("测试集 MSE:", test_mse))
# print(paste("测试集 R²:", r_squared_test))

# 创建数据框
lgb_model_eval <- data.frame(
  model = rep("LGB", 2),
  mse = c(train_mse, test_mse),
  rmse = c(train_rmse, test_rmse),
  R2 = c(r_squared_train, r_squared_test),
  dataset = factor(c("train", "test"))
)

# 打印结果
print(lgb_model_eval)
```

```
  model       mse      rmse        R2 dataset
1   LGB 0.3759316 0.6131326 0.8016026   train
2   LGB 0.8707640 0.9331473 0.5116258    test
```

3.5.4 模型解释性分析

由于模型拟合效果不强，我们只能简单地利用shap值做一个草率的分析，所得到的信息无法完全来解释每一个变量对预测的贡献：

首先我们计算shap值：

```
library(shapviz)
shp_lgb <- shapviz(lgb,X_pred =train_matrix)
```

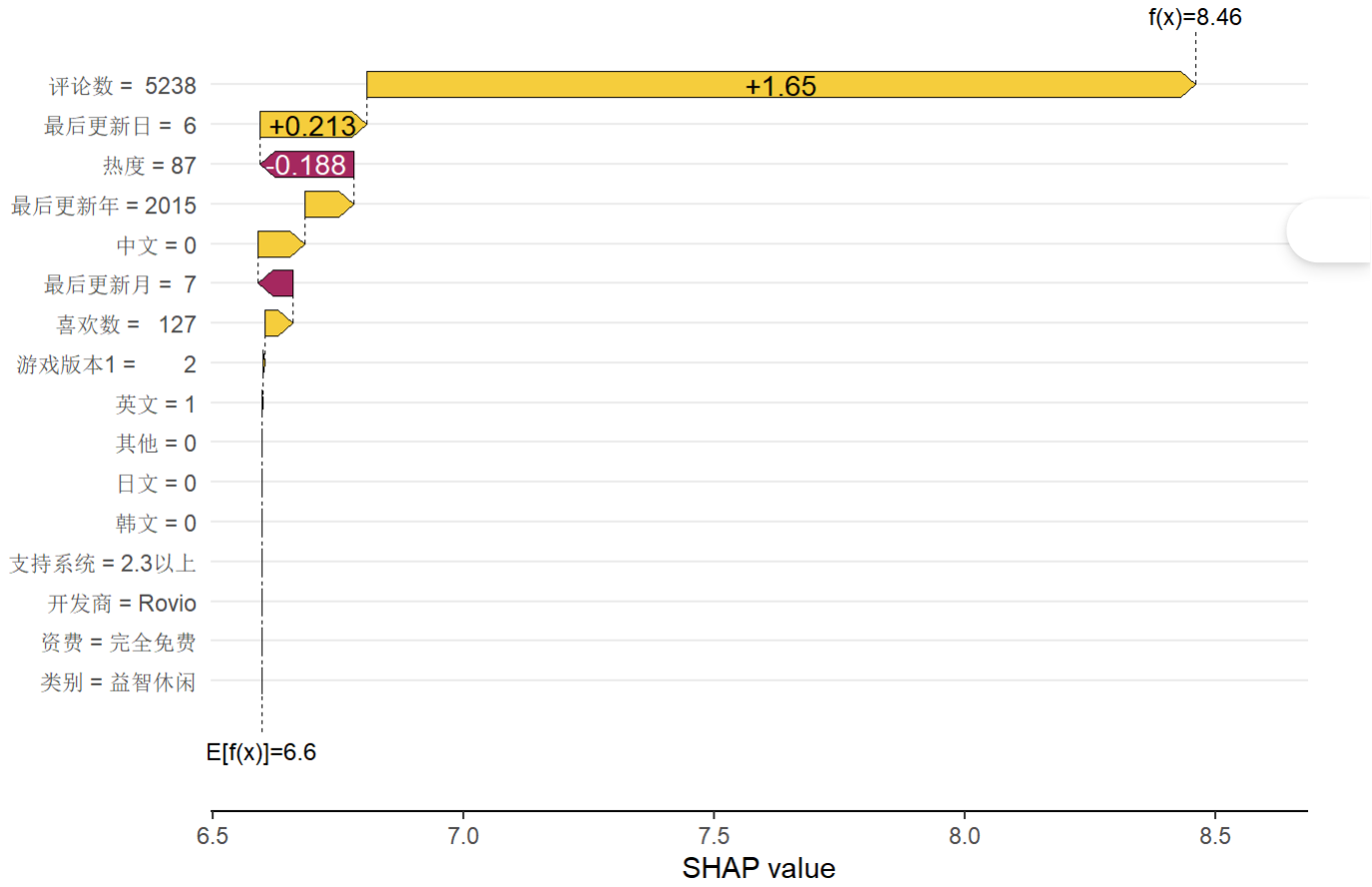
```
Warning in storage.mode(data) <- "double": NAs introduced by coercion
```

我们选择训练集的第二条数据来展示，该游戏的各项特征如何影响最终评分的输出，可以看到：

- 主要的贡献由评论数提供。对此的解释为评论本身需要玩家玩完游戏进一步思考和打字才能实现的交互，玩家往往更乐意对更优秀/评分高的游戏多费功夫来发表自己的感受
- 最后更新年数也提供了对评分的一定共享，通常来说最后更新年数越晚说明游戏仍然在频繁运营和更新，侧面反应了该公司看好该游戏的发展前景，其原因正来自于玩家对该游戏的喜爱与认可

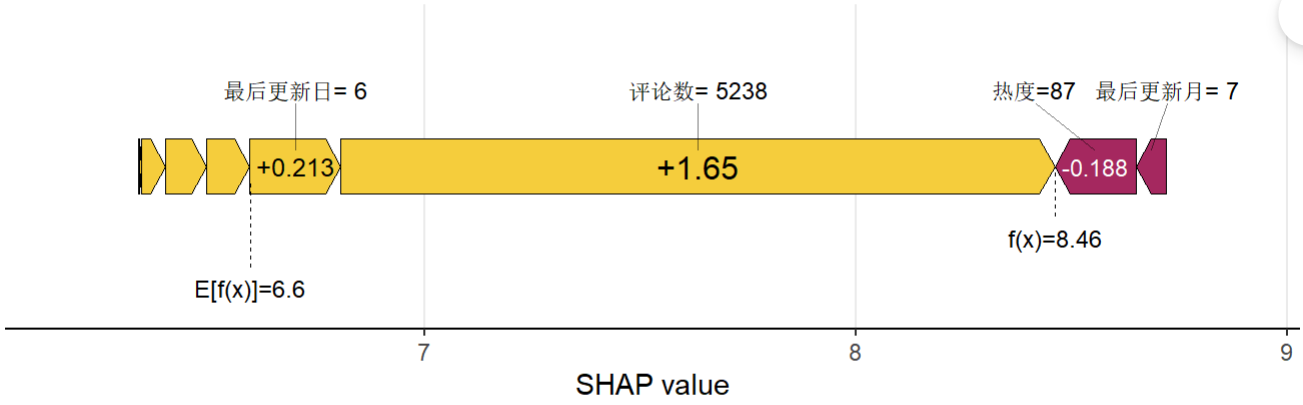
但也正因为模型总体拟合效果较差，所以该瀑布图并无法很好的解释所有变量的作用。

```
#展示shap值
choose_row <- 2
sv_waterfall(shp_lgb,row_id = choose_row,max_display = 20)#瀑布图
```



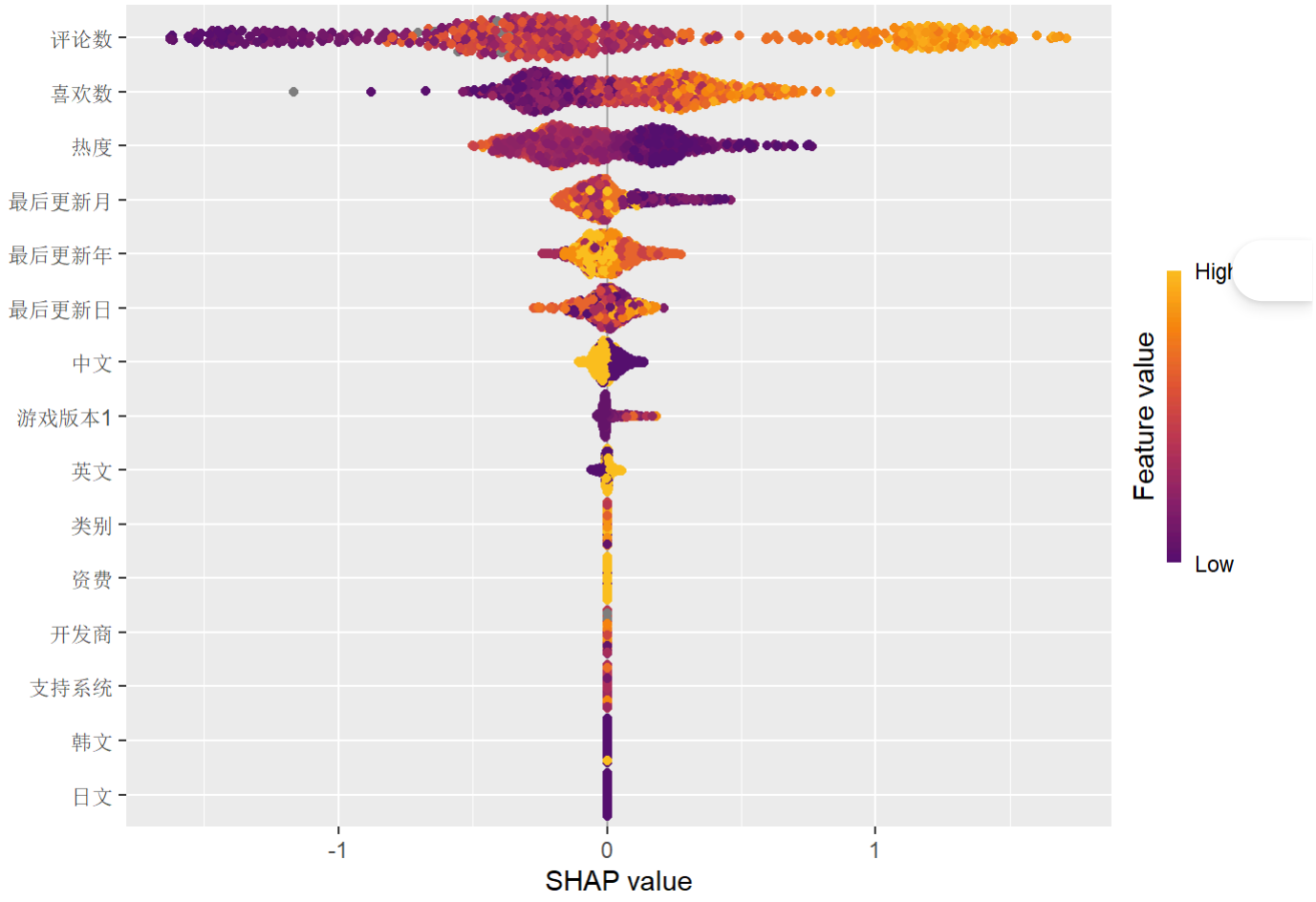

```
sv_force(shp_lgb,row_id = choose_row,max_display = 20)#缩合图
```

```
Warning: ggrepel: 12 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



那么对于每一个特征来说，模型在判断评分时依赖的每个特征的重要程度应该怎么衡量呢？通过该方法我们也可以简单列出一个散点图和柱状图来展示这一情况，以下两张图是采用不同表现形式对重要性的展示，总的来说，**评论数、喜欢数、热度**贡献最大、依次递减，这一结果是符合直觉的，毕竟他们都是数据中为数不多的数值型特征，至于最后更新的时间信息为什么占据很高的贡献度呢？一种解释正如上面所说，这间接反应了游戏开发者收到游戏市场反馈的决策，进而可以度量出游戏本身受到的认可。

```
#特征重要性评估
sv_importance(shp_lgb, kind = "beeswarm")#散点总结图
```




```
sv_importance(shp_lgb,fill="#fee08b")#条形图
```