第 11 章作业 MovieLense 数据集分析

目录

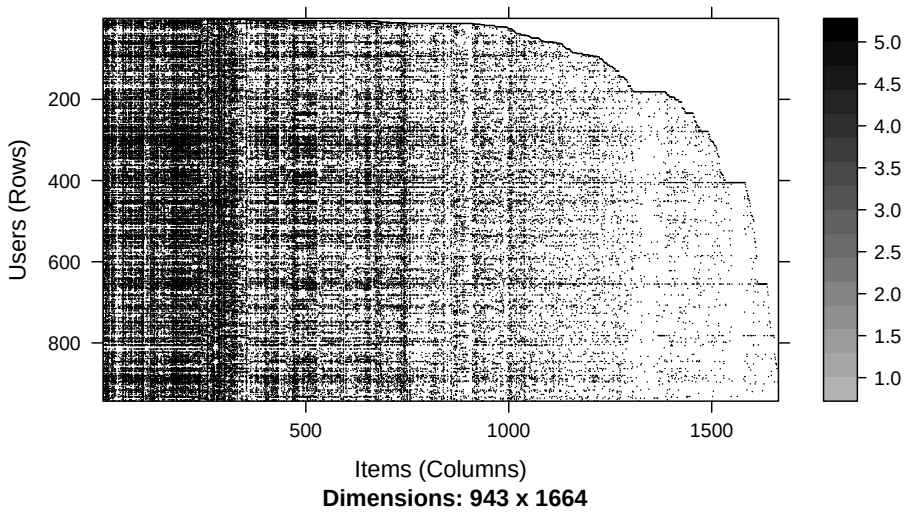
1 前言

R 的 recommenderlab 包可以实现协同过滤算法。这个包中有许多关于推荐算法建立、处理及可视化的函数。选用 recommenderlab 包中内置的 MovieLense 数据集进行分析，该数据集收集了网站 MovieLens（movielens.umn.edu）从 1997 年 9 月 19 日到 1998 年 4 月 22 日的数据，包括 943 名用户对 1664 部电影的评分。

```
library(recommenderlab)
library(ggplot2)
```

2 数据处理与数据探索性分析

```
data(MovieLense)
image(MovieLense)
```

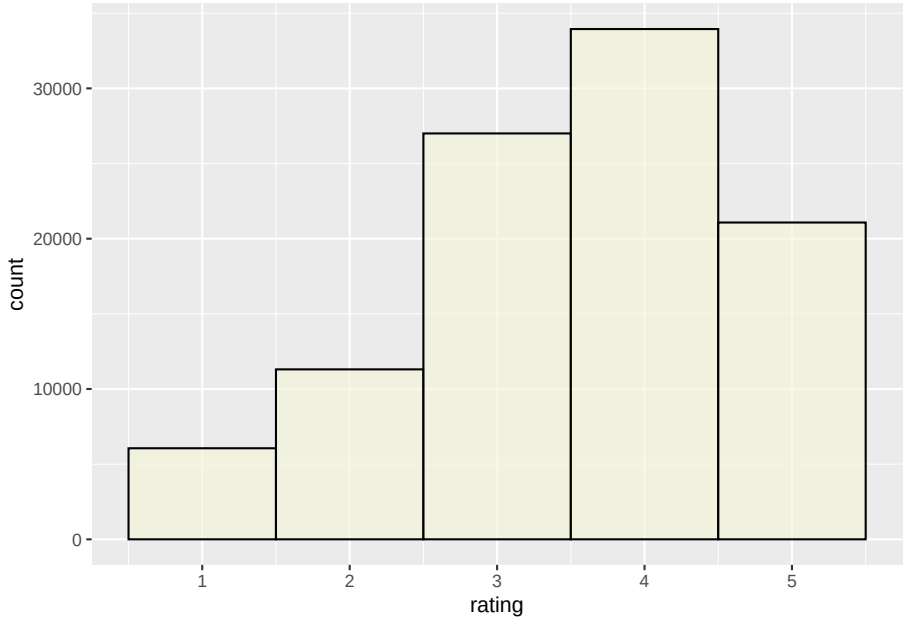


Dimensions: 943 x 1664

```
# 获取评分
ratings.movie <- data.frame(ratings = getRatings(MovieLense))
summary(ratings.movie$ratings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.00    4.00    3.53    4.00    5.00
```

```
ggplot(ratings.movie, aes(x = ratings)) +
  geom_histogram(fill = "beige", color = "black",
    binwidth = 1, alpha = 0.7) + xlab("rating") + ylab("count")
```



利用 summary() 获取评分数据，可知最大值为 5，最小值为 1，平均值为 3.53。并将其柱状图进行绘制，如下所示。

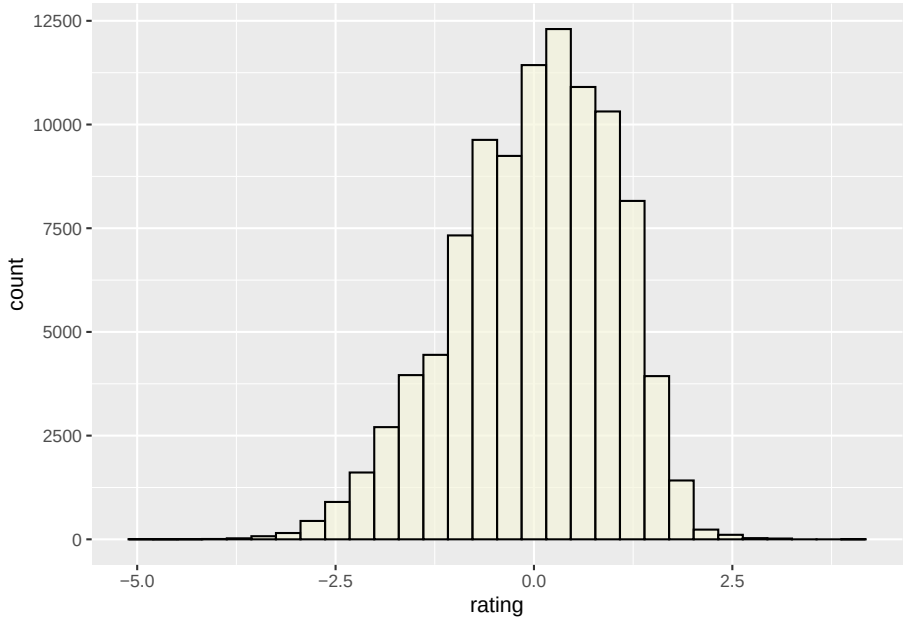
2.1 数据标准化

在进行数据分析前，利用 normalize() 我们将数据进行标准化，并进行绘制。

```
ratings.movie1 <- data.frame(ratings =
    getRatings(normalize(MovieLense, method = "Z-score")))
summary(ratings.movie1$ratings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -4.852  -0.647   0.108   0.000   0.751   4.128
```

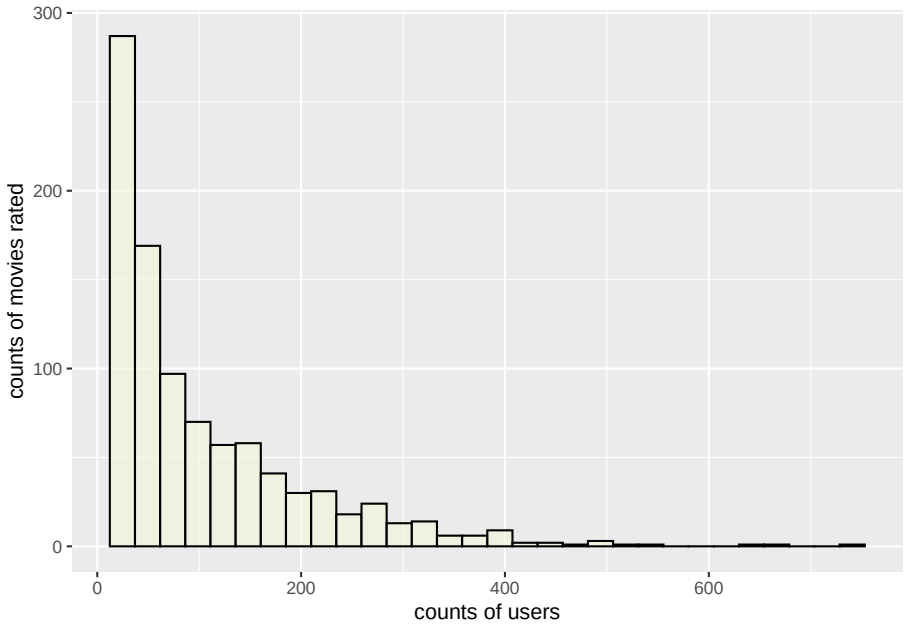
```
ggplot(ratings.movie1, aes(x = ratings)) +
  geom_histogram(fill = "beige", color = "black",
    alpha = 0.7) + xlab("rating") + ylab("count")
```



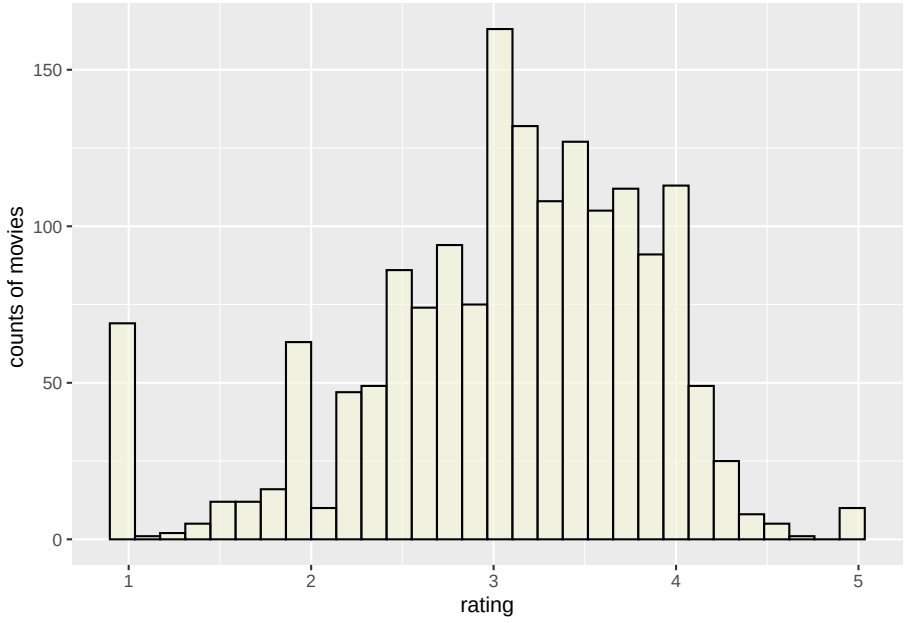
2.2 用户的电影点评数

我们还对用户的电影点评数进行描述性分析，具体结果如下所示。

```
movie.count <- data.frame(count = rowCounts(MovieLense))
ggplot(movie.count, aes(x = count)) +
  geom_histogram(fill = "beige", color = "black",
    alpha = 0.7) + xlab("counts of users") + ylab("counts of movies rated")
```




```
rating.mean <- data.frame(rating = colMeans(MovieLense))
ggplot(rating.mean, aes(x = rating)) +
  geom_histogram(fill = "beige", color = "black",
    alpha = 0.7) + xlab("rating") + ylab("counts of movies ")
```



3 建立推荐模型与模型评估

对于 realRatingMatrix 有六种方法：IBCF(基于物品的推荐)、UBCF（基于用户的推荐）、SVD（矩阵因子化）、PCA（主成分分析）、RANDOM（随机推荐）、POPULAR（基于流行度的推荐）。

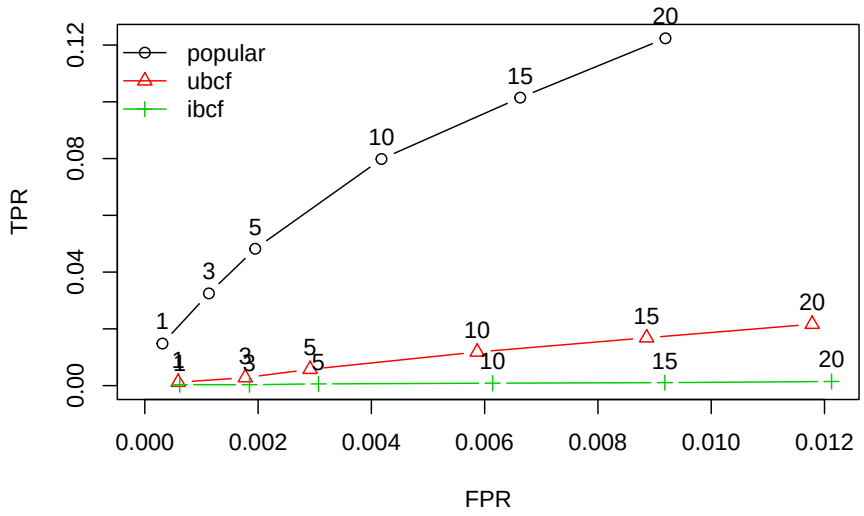
模型评估主要使用：recommenderlab 包中自带的评估方案，对应的函数是 evaluationScheme，能够设置采用 n-fold 交叉验证还是简单的 training/train 分开验证，本文采用后一种方法，即将数据集简单分为 training 和 test，在 training 训练模型，然后在 test 上评估。接下来我们使用三种不同技术进行构建推荐系统，并利用评估方案比较三种技术的好坏。

```
library(recommenderlab)
data(MovieLense)
scheme <- evaluationScheme(MovieLense, method = "split",
  train = 0.9, k = 1,  given = 10, goodRating = 4)
algorithms <- list(popular = list(name = "POPULAR",
  param = list(normalize = "Z-score")),
    ubcf = list(name = "UBCF", param = list(normalize = "Z-score",
```

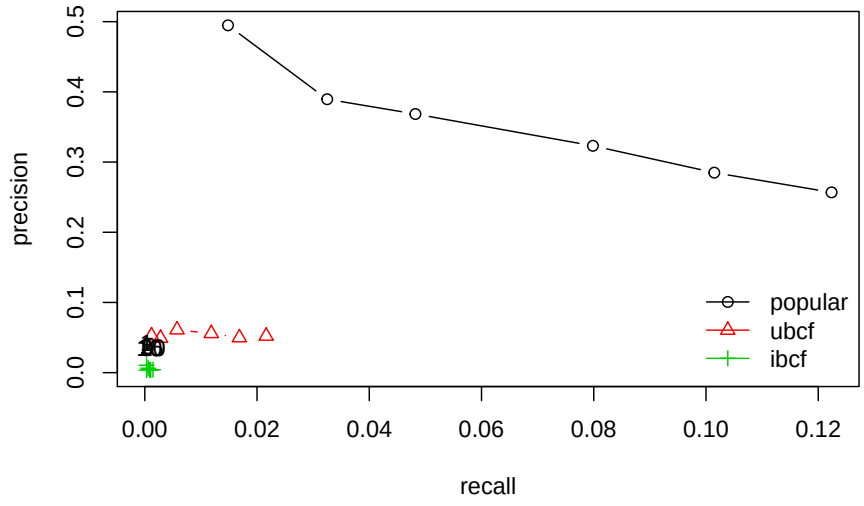
```
      method = "Cosine",nn = 25, minRating = 3)),
  ibcf = list(name = "IBCF", param = list(normalize = "Z-score")))
results <- evaluate(scheme, algorithms, n = c(1, 3, 5, 10, 15, 20))
```

```
## POPULAR run fold/sample [model time/prediction time]
##   1  [0.08sec/0.52sec]
## UBCF run fold/sample [model time/prediction time]
##   1  Available parameter (with default values):
## method    =  cosine
## nn    =  25
## sample    =  FALSE
## weighted  =  TRUE
## normalize     =  center
## min_matching_items    =  0
## min_predictive_items  =  0
## verbose   =  FALSE
## [0.05sec/1.03sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [51.32sec/0.11sec]
```

```
plot(results, annotate = 1:3, legend = "topleft") #ROC
```




```
plot(results, "prec/rec", annotate = 3)#precision-recall
```



```
# 按照评价方案建立推荐模型
model.popular <- Recommender(getData(scheme, "train"), method = "POPULAR")
model.ibcf <- Recommender(getData(scheme, "train"), method = "IBCF")
model.ubcf <- Recommender(getData(scheme, "train"), method = "UBCF")
# 对推荐模型进行预测
predict.popular <- predict(model.popular, getData(scheme, "known"), type = "ratings")
predict.ibcf <- predict(model.ibcf, getData(scheme, "known"), type = "ratings")
predict.ubcf <- predict(model.ubcf, getData(scheme, "known"), type = "ratings")
# 做误差的计算
predict.err <- rbind(calcPredictionAccuracy(predict.popular,
  getData(scheme, "unknown")),calcPredictionAccuracy(predict.ubcf, getData(scheme,
    "unknown")), calcPredictionAccuracy(predict.ibcf,getData(scheme, "unknown")))
rownames(predict.err) <- c("POPULAR", "UBCF", "IBCF")
predict.err
```

```
##           RMSE   MSE    MAE
## POPULAR 1.005 1.009 0.7954
## UBCF    1.196 1.431 0.9403
## IBCF    1.567 2.455 1.1818
```

通过结果我们可以看到：基于流行度推荐系统对于本案例数据的效果最好，RMSE，MSE，MAE 都是三者中的最小值。其次是基于用户的推荐，最后是基于项目协同过滤。

参考资料

1. Recommenderlab 包实现电影评分预测 (R 语言)
2. R 语言：recommenderlab 包的总结与应用案例