[p. 46 : 위에서 5번째 줄 다음에 역자 주석 추가]

해당 패키지를 사용하기 위해서는 설치한 후 다음과 같이 library()를 실행해서 패키지 로드 및 추가를 해야 한다.

```
library(reshape2)
library(data.table)
library(dplyr)
```

[p. 60 : 아래에서 4번째 줄]

[수정 전]

```
library(reshape2)
library(data.table)
library(dplyr)
# 데이터 로드
ratings = read.csv("C:/Users/Suresh/Desktop/movie_rating.csv")
# 데이터 처리 및 형식 변환
movie_ratings = as.data.frame(acast(ratings, title~critic,
value.var="rating"))
# 유사도 계산
sim_users = cor(movie_ratings[,1:6],use="complete.obs")
#sim_users[colnames(sim_users) == 'Toby']
sim_users[,6]
# 미지의 값 계산하기
# Toby가 등급을 매기지 않은 영화 선정
rating_critic =
setDT(movie_ratings[colnames(movie_ratings)[6]],keep.rownames =
TRUE)[]
names(rating_critic) = c('title','rating')
titles_na_critic = rating_critic$title[is.na(rating_critic$rating)]
ratings_t =ratings[ratings$title %in% titles_na_critic,]
# 각 사용자의 유사도 값을 새로운 변수로 추가
x = (setDT(data.frame(sim_users[,6]),keep.rownames = TRUE)[])
names(x) = c('critic','similarity')
ratings_t = merge(x = ratings_t, y = x, by = "critic", all.x = TRUE)
# 등급과 유사도 값 곱하기
```

```
ratings_t$sim_rating = ratings_t$rating*ratings_t$similarity
# 평가하지 않은 영화 추측
result = ratings_t %>% group_by(title) %>% summarise(sum(sim_
rating)/sum(similarity))
# 추천 생성 함수
generateRecommendations <- function(userId){
rating_critic =
setDT(movie_ratings[colnames(movie_ratings)[userId]],keep.rownames =
TRUE)[]
names(rating_critic) = c('title','rating')
titles_na_critic = rating_critic$title[is.na(rating_critic$rating)]
ratings_t =ratings[ratings$title %in% titles_na_critic,]
# 새로운 변수로 각 사용자들의 유사도 값 추가
x = (setDT(data.frame(sim_users[,userId]),keep.rownames = TRUE)[])
names(x) = c('critic','similarity')
ratings_t = merge(x = ratings_t, y = x, by = "critic", all.x = TRUE)
# 등급과 유사도 값 곱하기
ratings_t$sim_rating = ratings_t$rating*ratings_t$similarity
# 아직 평가하지 않은 영화들의 등급 예측
result = ratings_t %>% group_by(title) %>%
summarise(sum(sim_rating)/sum(similarity))
return(result)
}
```

->

[수정 후]

```
library(reshape2)
library(data.table)
library(dplyr)
# 데이터 로드
ratings = read.csv("C:/Users/Suresh/Desktop/movie_rating.csv")
# 데이터 처리 및 형식 변환
movie_ratings = as.data.frame(acast(ratings, title~critic,
                                    value.var="rating"))
# 유사도 계산
sim_users = cor(movie_ratings[,1:6], use="complete.obs")
# 미지 값 예측하기
```

```
# Tobby가 등급을 매기지 않은 영화 선정
rating_critic =
  setDT(movie_ratings[colnames(movie_ratings)[6]], keep.rownames=TRUE)[]
names(rating_critic) = c('title','rating')
titles_na_critic = rating_critic$title[is.na(rating_critic$rating)]
ratings_t = ratings[ratings$title %in% titles_na_critic,]
# 각 사용자의 유사도 값을 새로운 변수로 추가
x = (setDT(data.frame(sim_users[,6]), keep.rownames=TRUE)[])
names(x) = c('critic','similarity')
ratings_t = merge(x=ratings_t, y=x, by="critic", all.x=TRUE)
# 등급과 유사도 값 곱하기
ratings_t$sim_rating = ratings_t$rating*ratings_t$similarity
# 평가하지 않은 영화 추측하기
result = ratings_t %>% group_by(title) %>%
  summarise(sum(sim_rating)/sum(similarity))
# 추천 생성 함수
generateRecommendations <- function(userId) {
  rating_critic =
    setDT(movie_ratings[colnames(movie_ratings)[userId]],
          keep.rownames=TRUE)[]
  names(rating_critic)=c('title','rating')
  titles_na_critic = rating_critic$title[is.na(rating_critic$rating)]
  ratings_t = ratings[ratings$title %in% titles_na_critic,]
  # 새로운 변수로 각 사용자들의 유사도 값 추가
  x=(setDT(data.frame(sim_users[, userId]), keep.rownames=TRUE)[])
  names(x) = c('critic','similarity')
  ratings_t = merge(x=ratings_t, y=x, by="critic", all.x=TRUE)
  # 등급과 유사도 값 곱하기
  ratings_t$sim_rating = ratings_t$rating*ratings_t$similarity
  # 아직 평가하지 않은 영화들의 등급 예측
  result=ratings_t %>% group_by(title) %>%
    summarise(sum(sim_rating)/sum(similarity))
  return(result)
}
```

[p. 108 : 아래에서 7번째 줄]

[수정 전]

```
# MF
library(recommenderlab) data("MovieLense") dim(MovieLense)
# NMF를 사용해 MF에 적용
mat = as(MovieLense,"matrix")
mat[is.na(mat)] = 0 res = nmf(mat,10) res
# 적합한 값
r.hat <- fitted(res) dim(r.hat)
p <- basis(res)
dim(p)
q <- coef(res)
dim(q)
```

->

[수정 후]

```
# MF
library(recommenderlab)
data("MovieLense")
dim(MovieLense)

# NMF를 사용해 MF에 적용
mat = as(MovieLense,"matrix")
mat[is.na(mat)] = 0
res = nmf(mat,10)
res

# 적합한 값
r.hat <- fitted(res)
dim(r.hat)

p <- basis(res)
dim(p)
q <- coef(res)
dim(q)
```

[p. 117 : 위에서 8번째 줄]

[수정 전]

# glm에 주입
df = data.frame(y=y,x1=x1,x2=x2) glm(y~x1+x2,data=df,family="binomial")

->

[수정 후]

# glm에 주입
df = data.frame(y=y,x1=x1,x2=x2)
glm(y~x1+x2,data=df,family="binomial")

[p. 118 : 아래에서 7번째 줄]

[수정 전]

cl = train$Species library(caret)

->
[수정 후]

cl = train$Species
library(caret)

[p. 122 : 위에서 3번째 줄]

[수정 전]

cost =10 is chosen from summary result of tune variable

->

[수정 후]

(삭제, 실제 출력에는 안 나옴)

[p. 124 : 아래에서 3번째 줄]

[수정 전]

plot(model) # plot 트리

->

[수정 후]

해당 코드가 동작하지 않으면, par(mar=c(1,1,1,1))를 먼저 실행 필요

[p. 128 : 위에서 4번째 줄]

[수정 전]

회기

->

[수정 후]

회귀

[p. 132 : 아래에서 4번째 줄]

[수정 전]

```
for(i in 1:100){
kmeans<- kmeans(x=iris, centers=i, iter.max=50)
cost_df<- rbind(cost_df, cbind(i, kmeans$tot.withinss))
}
```

->

[수정 후]

```
for(i in 1:100){
   kmeans<- kmeans(x=iris, centers=i, iter.max=50)
   cost_df<- rbind(cost_df, cbind(i, kmeans$tot.withinss))
}
```

[p. 135 : 아래에서 10번째 줄]

[수정 전]

```
data(USArrests) head(states)
```

->

[수정 후]

```
data(USArrests)
states=rownames(USArrests)
head(states)
```

[p. 157 : 아래에서 1번째 줄]

[수정 전]

```
names(recommender_models)
```

->

[수정 후]

```
recommender_models <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
names(recommender_models)
```

[p. 156 : 아래에서 2번째 줄]

[수정 전]

object.size(as(Jester5k, "matrix"))/object.size(Jester5k) 0.925001079083901
bytes

->
[수정 후]

object.size(as(Jester5k, "matrix"))/object.size(Jester5k)
0.925001079083901 bytes

[p. 161 : 아래에서 3번째 줄]

[수정 전]

which_train <- sample(x = c(TRUE, FALSE), size = nrow(Jester5k), replace =
TRUE, prob = c(0.8, 0.2)) head(which_train)

->
[수정 후]

which_train <- sample(x = c(TRUE, FALSE), size = nrow(Jester5k), replace = TRUE, prob = c(0.8, 0.2))
head(which_train)

[p. 167 : 위에서 8번째 줄]

[수정 전]

boxplot(model_data)

->

[수정 후]

boxplot(rowMeans(model_data))

[p. 168 : 아래에서 5번째 줄]

[수정 전]

model_data = model_data [rowMeans(model_data)>=-5 & rowMeans(model_data)<=
7] dim(model_data)

->

[수정 후]

model_data = model_data [rowMeans(model_data)>=-5 & rowMeans(model_data)<=7]
dim(model_data)

[p. 171 : 아래에서 2번째 줄]

[수정 전]

method = model_to_evaluate, parameter = model_parameters)
Recommender of type 'UBCF' for 'realRatingMatrix' learned using 2528 users

->
[수정 후]

method = model_to_evaluate, parameter = model_parameters)
eval_recommender

Recommender of type 'UBCF' for 'realRatingMatrix' learned using 2528 users

[p. 175 : 위에서 4번째 줄]

[수정 전]

columns_to_sum] head(indices_summed)

->

[수정 후]

columns_to_sum]
head(indices_summed)

[p. 177 : 아래에서 2번째 줄]

[수정 전]

model_data

->

[수정 후]

dim(model_data)

[p. 180 : 위에서 3번째 줄]

[수정 전]

model_recommender Recommender of type 'IBCF' for 'realRatingMatrix' learned
using 2506 users.

->

[수정 후]

model_recommender

Recommender of type 'IBCF' for 'realRatingMatrix' learned using 2506 users.

[p. 189 : 아래에서 2번째 줄]

[수정 전]

names(model1) [1] "IBCF_cos_k_5" "IBCF_cos_k_10" "IBCF_cos_k_20"

->

[수정 후]

names(model1)

[1] "IBCF_cos_k_5" "IBCF_cos_k_10" "IBCF_cos_k_20"

[p. 190 : 위에서 4번째 줄]

[수정 전]

names(model2) [1] "IBCF_pea_k_5" "IBCF_pea_k_10" "IBCF_pea_k_20"

->

[수정 후]

names(model2)
[1] "IBCF_pea_k_5" "IBCF_pea_k_10" "IBCF_pea_k_20"

[p. 190 : 아래에서 8번째 줄 (가독성 위해 인덴트 정리)]

[수정 전]

IBCF run fold/sample [model time/prediction time] 1 [0.139sec/0.311sec] 2
[0.143sec/0.309sec] 3 [0.141sec/0.306sec] 4 [0.153sec/0.312sec]
IBCF run fold/sample [model time/prediction time] 1 [0.141sec/0.326sec] 2
[0.145sec/0.445sec] 3 [0.147sec/0.387sec] 4 [0.133sec/0.439sec]
IBCF run fold/sample [model time/prediction time] 1 [0.14sec/0.332sec] 2
[0.16sec/0.327sec] 3 [0.139sec/0.331sec] 4 [0.138sec/0.339sec] IBCF
run fold/sample [model time/prediction time] 1 [0.139sec/0.341sec] 2
[0.157sec/0.324sec] 3 [0.144sec/0.327sec] 4 [0.133sec/0.326sec]

->

[수정 후]

IBCF run fold/sample [model time/prediction time]
1 [0.139sec/0.311sec]
2 [0.143sec/0.309sec]
3 [0.141sec/0.306sec]
4 [0.153sec/0.312sec]
IBCF run fold/sample [model time/prediction time]
1 [0.141sec/0.326sec]
2 [0.145sec/0.445sec]
3 [0.147sec/0.387sec]
4 [0.133sec/0.439sec]
IBCF run fold/sample [model time/prediction time]
1 [0.14sec/0.332sec]
2 [0.16sec/0.327sec]
3 [0.139sec/0.331sec]
4 [0.138sec/0.339sec]
IBCF run fold/sample [model time/prediction time]
1 [0.139sec/0.341sec]
2 [0.157sec/0.324sec]
3 [0.144sec/0.327sec]
4 [0.133sec/0.326sec]

[p. 323 : 아래에서 3번째 줄]

[수정 전]

basic usage :
getting help on Neo4J in the browser:
:help

->

[수정 후]

Neo4J 브라우저에서 도움말 관련 기본 사용법 :

:help

[p. 355 : 아래에서 8번째 줄]

[수정 전]

Export JAVA_HOME = path/to/java7
경로 설정
export MAHOUT_HOME = /home/softwares/ apache-mahout-distribution-0.12.2
export MAHOUT_LOCAL = true
독립 모드일 경우
export PATH = $MAHOUT_HOME/bin
CLASSPATH = $MAHOUT_HOME/lib:$CLASSPATH

->

[수정 후]

export JAVA_HOME = path/to/java7
경로 설정
export MAHOUT_HOME = /home/softwares/ apache-mahout-distribution-0.12.2
export MAHOUT_LOCAL = true
독립 모드일 경우
export PATH = $MAHOUT_HOME/bin
export CLASSPATH = $MAHOUT_HOME/lib:$CLASSPATH

[p. 376 : 아래에서 15번째 줄]

[수정 전]

public Recommender buildRecommender(DataModel model) throws
TasteException {

->

[수정 후]

```
public Recommender buildRecommender(DataModel model)
  throws TasteException {
```

[p. 378 : 아래에서 4번째 줄]

[수정 전]

```
public Recommender buildRecommender(DataModel model)
throws TasteException {
  UserSimilarity similarity = new
    EuclideanDistanceSimilarity(model);
  UserNeighborhood neighborhood =
    new NearestNUserNeighborhood (2, similarity, model);
  return
  new GenericUserBasedRecommender (model, neighborhood,
    similarity);
}
};
// 아이템 기반 추천인
public Recommender buildRecommender(DataModel model)
throws TasteException {
ItemSimilarity similarity = new LogLikelihoodSimilarity(model);
return
new GenericItemBasedRecommender(model, similarity);
}
};
```

->

[수정 후]

```
public Recommender buildRecommender(DataModel model)
throws TasteException {
  UserSimilarity similarity = new
    EuclideanDistanceSimilarity(model);
  UserNeighborhood neighborhood =
    new NearestNUserNeighborhood (2, similarity, model);
  return
```

```
        new GenericUserBasedRecommender (model, neighborhood,
            similarity);
}

// 아이템 기반 추천인
public Recommender buildRecommender(DataModel model)
throws TasteException {
    ItemSimilarity similarity = new LogLikelihoodSimilarity(model);
    return
        new GenericItemBasedRecommender(model, similarity);
}
```