Package 'CFilt'

October 12, 2022

Type Package

Title Collaborative Filtering by Reference Classes

Version 0.2.1

Author Thiago Lima, Jessica Kubrusly

Maintainer Thiago Lima <thiagoaugusto@id.uff.br>

Description

The collaborative Filtering methodology has been widely used in recommendation systems, which uses similarities between users or items to make recommendations. A class called CF was implemented,
where it is structured by matrices and composed of recommendation and database manipulation functions.
See Aggarwal (2016) <doi:10.1007/978-3-319-29659-3> for an overview.

Depends R (>= 3.5.0)

Imports methods, utils

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-14 13:10:02 UTC

R topics documented:

CF-class *A Reference Class to represent a object CF*

Description

A class of objects created structured with the following objects: the MU - Utility Matrix, the SU1 and SU2 - Matrices of Similarity between Users, the SI1 e SI2 - Matrices of Similarity between Items, and the vectors averages_u, averages_i, n_aval_u and n_aval_i. The class contains methods, general functions with the objectives of manipulating the data and making recommendations, from the structures present in the class. The data manipulation methods comprise addsimilarity, addnewuser, addnewemptyuser, deleteuser, addnewitem, addnewemptyitem, deleteitem, newrating and deleterating, while the recommendations methods recommend, kclosestitems, topkusers, topkitems are created through choices available in the Collaborative Filtering methodology. All objects and methods are accessed through the "$" character. A CF class object is created through the CF-builder function.

Fields

`MU` A utility matrix, matrix that contains all the users ratings. The rows comprise users and the columns, itens.

`SU1` A superior triangular user similarity matrix that contains the similarities between users, calculated using Cosine similarity

`SU2` A superior triangular user similarity matrix that contains the similarities between users, calculated using Pearson Correlation.

`SI1` A superior triangular item similarity matrix that contains the similarities between items, calculated using Cosine similarity.

`SI2` A superior triangular item similarity matrix that contains the similarities between items, calculated using Adjusted Cosine similarity.

`averages_u` A vector that contains the averages of users ratings.

`averages_i` A vector that contains the averages of item ratings.

`n_aval_u` A vector that contains the numbers of ratings performed by each user.

`n_aval_i` A vector that contains the numbers of ratings received for each item.

Methods

`addnewemptyitem(Id_i)` Adds a new item without ratings. The object CF matrices and vectors will be updated. Id_i : a character, an item ID. If you want to add more items, you can use lists where Id_i is a list of characters.

`addnewemptyuser(Id_u)` Adds a new user without ratings. The object CF matrices and vectors will be updated. Id_u : a character, a user ID. If you want to add more users, you can use lists, where Id_u is a list of characters.

`addnewitem(Id_i, Ids_u, r)` Adds a new item that has been rated by one or more users. The object CF matrices and vectors will be updated. Id_i : a character, an item ID; Ids_u : a character vector, a user IDs; r : a vector with its respective ratings. If you want to add more

items, you can use lists, where Id_i is a list of characters; Ids_u is a list of vectors of characters; r is a list of vectors of ratings.

`addnewuser(Id_u, Ids_i, r)` Adds a new user who rated one or more items. The object CF matrices and vectors will be updated. Id_u : a character, a user ID; Ids_i : a character vector, item IDs; r : a vector with its respective ratings. If you want to add more users, you can use lists, where Id_u: list of characters; Ids_i: list of vectors of characters; r: list of vectors of ratings.

`addsimilarity(sim_user = "none", sim_item = "none")` Adds new methodologies even after the construction and modification of the CF object used. The matrices of similarities representing each requested methodology will be added. sim_user: a methodology used to estimate the rating by users similarity. Can be 'cos','pearson','both' or 'none'. If it equals 'cos' (Cosine Similarity), the SU1 will be built. If it equals 'pearson' (Pearson Similarity), the SU2 will be built. If it equals 'both', the SU1 and SU2 will be built. If it equals 'none', nothing will be built. sim_item: A methodology used to estimate the rating by itens similarity. Can be 'cos','adjcos','both' or 'none'. If it equals 'cos' (Cosine Similarity), the SI1 will be built. If it equals 'adjcos' (Adjusted Cosine Similarity), the SI2 will be built. If it equals 'both', the SI1 and SI2 will be built. If it equals 'none', nothing will be built.

`changerating(Id_u, Id_i, r)` Changes the rating from user Id_u to item Id_i. The object CF matrices and vectors will be updated. Id_u : A character, a user ID; Id_i : A character, an item ID; r : The new rating. If you want to change more ratings, you can use lists where Id_u and Id_i are lists of characters and r is a list of ratings.

`deleteitem(Id_i)` Deletes an already registered item. The object CF matrices and vectors will be updated. Id_i : a character, a item ID that will be deleted. If you want to delete more items, you can use lists, where Id_i is a list of characters.

`deleterating(Id_u, Id_i)` Deletes the rating from user Id_u to item Id_i. The object CF matrices and vectors will be updated. Id_u : A character, a user ID; Id_i : A character, an item ID. If you want to delete more ratings, you can use lists, where Id_u and Id_i are lists of characters.

`deleteuser(Id_u)` Deletes an already registered user. The object CF matrices and vectors will be updated. Id_u : A character, a user ID that will be deleted. If you want to delete more users, you can use lists where Id_u is a list of characters.

`estimaterating( Id_u, Id_i, type, neighbors = 5, similarity = ifelse(type == "user", "pearson", "adjcos")` A function that returns the estimated rating for the evaluation of item Id_i by user Id_u. The recommendation can be made through similarity between users, when type = 'user', and also through the similarity between items, when type = 'item'. Id_u: A character, a user ID; Id_i: A character, an item ID; type: A character string, 'user' or 'item'; neighbors: Number of similarities used for the estimates.(default=5); similarity: the methodology used to estimate the rating. If type = 'user', must be one of 'cos', for cosine similarity, or 'pearson' (default), for pearson similarity. If type='item', must be one of 'cos', for cosine similarity, or 'adjcos' (default), for adjusted cosine similarity. This choice can alter the way the estimate is calculated.

`kclosestitems(Id_i, k = 5, similarity = "adjcos")` A function that returns the k items most similar to an item. Id_i : A Character, a Item ID; k : Number of items most similar to item Id_i (deafult = 5); similarity: the methodology used to estimate the rating. Must be one of 'cos', for cosine similarity, or 'adjcos' (default), for adjusted cosine similarity. This choice can alter the way the estimate is calculated.

`newrating(Id_u, Id_i, r)` Adds a new rating from user Id_u to item Id_i.The object CF matrices and vectors will be updated. Id_u : a character, a user ID; Id_i : a character, an item ID; r :

the rating. If you want to add more ratings, you can use lists, where Id_u and Id_i are lists of characters and r is a list of ratings.

`recommend( Id_u, Id_i, type, neighbors = 5, cuts = 3.5, similarity = ifelse(type == "user", "pearson", "adj`
A function that returns True if user Id_u will like item Id_i or returns FALSE, otherwise. The recommendation can be made through similarity between users, when type = 'user', as well as through the similarity between items, when type = 'item'. Id_u : a character, a User ID; Id_i : a character, an Item ID; type: a character string, 'user' or 'item'; neighbors: number of similarities used for to estimates (default = 5); cuts: cut score designated to determine if it is recommended (default=3.5); similarity: the methodology used to estimate the rating. If type = 'user', must be one of 'cos', for cosine similarity, or 'pearson' (default), for pearson similarity. If type='item', must be one of 'cos', for cosine similarity, or 'adjcos' (default), for adjusted cosine similarity. This choice can alter the way the estimate is calculated.

`topkitems( Id_u, k = 5, type, neighbors = 5, cuts = 3.5, similarity = ifelse(type == "user", "pearson", "adj`
A function that recommends k items for an Id_u user. The recommendation can be made through similarity between users, when type = 'user', as well as through similarity between items, when type = 'item'. Id_u : a character, a User ID; k : number of recommendations (default=5); type: a character string, 'user' or 'item'; neighbors: number of similarities used for the estimates(default=5); cuts: cut score designated to determine if it is recommended (default = 3.5); similarity: the methodology used to estimate the rating. If type = 'user', must be one of 'cos', for cosine similarity, or 'pearson' (default), for pearson similarity. If type='item', must be one of 'cos', for cosine similarity, or 'adjcos' (default), for adjusted cosine similarity. This choice can alter the way the estimate is calculated.

`topkusers( Id_i, k = 5, type, neighbors = 5, cuts = 3.5, similarity = ifelse(type == "user", "pearson", "adj`
A function that indicates the k users who will like the item Id_i.The recommendation can be made through similarity between users, when type = 'user', as well as through similarity between items, when type = 'item'. Id_i : A Character, a Item ID; k : Number of recommendations (default=5); type: A character string, 'user' or 'item'; neighbors: Number of similarities used for the estimates (default=5); cuts: Cut score designated to determine if it is recommended (default=3.5); similarity: the methodology used to estimate the rating. If type = 'user', must be one of 'cos', for cosine similarity, or 'pearson' (default), for pearson similarity. If type='item', must be one of 'cos', for cosine similarity, or 'adjcos' (default), for adjusted cosine similarity. This choice can alter the way the estimate is calculated.

Author(s)

Thiago Lima, Jessica Kubrusly.

References

- LINDEN, G.; SMITH, B.; YORK, J. Amazon. com recommendations: Item-toitem collaborative filtering. Internet Computing, IEEE, v. 7, n. 1, p. 76-80,2003
- Aggarwal, C. C. (2016). Recommender systems (Vol. 1). Cham: Springer International Publishing.
- Leskovec, J., Rajaraman, A., & Ullman, J. D. (2020). Mining of massive data sets. Cambridge university press.

See Also

CFbuilder

Examples

```
ratings<-movies[1:1000,]
objectCF<-CFbuilder(Data = ratings, sim_user="pearson", sim_item="adjcos")
objectCF$addsimilarity(sim_user="cos",sim_item="cos")
objectCF$MU
objectCF$SU1
objectCF$SU2
objectCF$SI1
objectCF$SI2
objectCF$averages_u
objectCF$averages_i
objectCF$n_aval_u
objectCF$n_aval_i
objectCF$addnewuser(Id_u = "Thiago",Ids_i = "The Hunger Games: Catching Fire",r = 5)
objectCF$addnewemptyuser(Id_u = "Jessica")
objectCF$deleteuser(Id_u = "Jessica")
objectCF$addnewitem(Id_i = "Avengers: Endgame",Ids_u = c("1","2"),r = c(5,3))
objectCF$addnewemptyitem(Id_i = "Star Wars")
objectCF$deleteitem(Id_i="Star Wars")
objectCF$newrating(Id_u = "1", Id_i = "Till Luck Do Us Part 2",r = 2)
objectCF$recommend(Id_u = "2", Id_i = "Iron Man 3", type = "user")
objectCF$kclosestitems(Id_i = "Iron Man 3", k = 3)
objectCF$topkitems(Id_u = "3",k = 3, type = "user")
objectCF$topkusers(Id_i = "Thor: The Dark World", k = 3,type = "item")
objectCF$estimaterating(Id_u = "2",Id_i = "Iron Man 3", type = "user")
objectCF$deleterating("1","Brazilian Western")
objectCF$changerating("1","Wreck-It Ralph",2)
```

CFbuilder *A function to create and build a CF class object*

Description

A CF class object constructor. This function can perform two procedures: If Data is a data frame, style: User Id - Item Id - Ratings, it creates and builds an FC class object from the data frame, containing a Utility Matrix, a User Similarity Matrix, an Item Similarity Matrix, a vector with the number of user ratings, a vector with the number of ratings received for the items, a vector with the average ratings of each user and another vector with the average ratings received from each item. If Data is the Utility Matrix, it also constructs all matrices and vectors. When building the object, the progress percentage is shown. Step 1: Building the MU and vectors. Step 2: Building the SU. Step 3: Building the SI.

Usage

```
CFbuilder(Data, sim_user, sim_item)
```

Arguments

Data	A data frame by style: User ID - Item ID - Ratings, or a Utility Matrix.
sim_user	A methodology used to estimate the rating by users similarity. Can be 'cos','pearson','both' or 'none'. If it equals 'cos', the SU1 will be built. If it equals 'pearson', the SU2 will be built. If it equals 'both', the SU1 and SU2 will be built. If it equals 'none', nothing will be built.
sim_item	A methodology used to estimate the rating by itens similarity. Can be 'cos','adjcos','both' or 'none'. If it equals 'cos', the SI1 will be built. If it equals 'adjcos', the SI2 will be built. If it equals 'both', the SI1 and SI2 will be built. If it equals 'none', nothing will be built.

Value

a CF class object.

Author(s)

Thiago Lima, Jessica Kubrusly.

References

LINDEN, G.; SMITH, B.; YORK, J. Amazon. com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, v. 7, n. 1, p. 76-80,2003

See Also

CF-class

Examples

```
ratings<-movies[1:1000,]
objectCF<-CFbuilder(Data = ratings, sim_user = "pearson", sim_item = "adjcos")
```

CFilt *CFilt: A package about Collaborative Filtering by RC in R.*

Description

The CFilt package provides one builder function CFbuilder and one class CF with methods that serve to change objects and recommend items or users.

Details

Two main goals:

- Structure the database so that changes can be made in a practical way through object-oriented programming.
- Make recommendations through choices by the Collaborative Filtering methodology in a practical, fast and efficient manner.

Author(s)

Authors:

- Jessica Quintanilha Kubrusly - jessicakubrusly@id.uff.br
- Thiago Augusto Santos Lima - thiagoaugusto@id.uff.br

`movies` *Movie ratings by users*

Description

A dataset containing 7276 ratings for 50 movies by 526 users. This database was created by Giglio (2014).

Usage

```
movies
```

Format

A data frame with 7276 rows and 3 variables:

Id Users Users identifier. Numbers 1 to 526.

Id Items Movies identifier. Movies list:

1. Iron Man 3
2. Despicable Me 2
3. My Mom Is a Character
4. Fast & Furious 6
5. The Wolverine
6. Thor: The Dark World
7. Hansel & Gretel: Witch Hunters
8. Wreck-It Ralph
9. Monsters University
10. The Hangover Part III
11. Vai Que Dá Certo
12. Meu Passado me Condena
13. We're So Young
14. Brazilian Western
15. O Concurso
16. Mato sem Cachorro
17. Cine Holliudy
18. Odeio o Dia dos Namorados
19. Argo

20. Django Unchained
21. Life of Pi
22. Lincoln
23. Zero Dark Thirty
24. Les Miserables
25. Silver Linings Playbook
26. Beasts of the Southern Wild
27. Amour
28. A Royal Affair
29. American Hustle
30. Capitain Phillips
31. 12 Years a Slave
32. Dallas Buyers Club
33. Gravity
34. Her
35. Philomena
36. The Wolf of Wall Street
37. The Hunt
38. Frozen
39. Till Luck Do Us Part 2
40. Muita Calma Nessa Hora 2
41. Paranormal Activity: The Marked Ones
42. I, Frankenstein,
43. The Legend of Tarzan
44. The Book Thief
45. The Lego Movie, , ,
46. Walking With Dinosaurs
47. The Hunger Games: Catching Fire
48. Blue Is The Warmest Color
49. Reaching for the Moon
50. The Hobbit: The Desolation of Smaug

Ratings Movie ratings by users. The ratings follows the Likert scale: 1 to 5.

References

Giglio , J. C. (2014). Recomendação de Filmes Utilizando Filtragem Colaborativa [Recommending Films Using Collaborative Filtering]. Undergraduate thesis - Universidade Federal Fluminense.

Index