fcaR, Spreading FCA to the Data Science World*

Pablo Cordero$^{1[0000-0002-5506-6467]}$, Manuel Enciso$^{2[0000-0002-0531-4055]}$, Domingo López-Rodríguez$^{1[0000-0002-0172-1585]}$, and Ángel Mora$^{1[0000-0003-4548-8030]}$

[1] Departamento de Matemática Aplicada, Universidad de Málaga
{pcordero,dominlopez,amora}@uma.es

[2] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga
enciso@uma.es

Abstract. Formal concept analysis (FCA) has become a mature tool for extracting helpful knowledge for real problems based on solid mathematical foundations rooted in logic and lattice theory. However, in areas such as machine learning, big data, artificial intelligence, database, etc. remains a stranger. The R language is one of the main languages used in data science, and this work describes an R package called fcaR that implements FCA's core notions and techniques. One of the main goals is to spread FCA to the rest of the world. The main facilities of the tool are shown with a running example.

Keywords: R programming language · Data science · Formal concept analysis

1 Introduction

We assume the main FCA works [1, 2] are known and show in this short introduction some features of the developed package and the main references of the mathematical methods developed in fcaR.

Classic FCA is devoted to the study of binary datasets (formal contexts) where variables are called attributes. Extensions of FCA (see [3, 4]) have been developed to model real-world problems for datasets containing imprecise, graded or vague information that is not adequately represented as binary values. This fuzzy extension is able to model problems with numerical and categorical attributes since these can be scaled to a truth value describing the degree of fulfilment of the attribute.

As it is well known, from a dataset (binary or fuzzy), FCA can compute maximal clusters, named concepts, between objects and attributes with a hierarchy between the concepts and relationships between the attributes (rules or implications) are computed with the same computational cost in FCA.

We emphasize the notion of if-then rules as a efficient way to compact knowledge and and enable automatic handling by using logic. In this direction, [5] introduced a logic, named simplification logic for functional dependencies (SL_{FD}),

* Supported by Grants TIN2017-89023-P, UMA2018-FEDERJA-001 and PGC2018-095869-B-I00 of the Junta de Andalucia, and European Social Fund.

firmly based on a simplification rule, which allows us to narrow the functional dependency set by removing redundant attributes. Although the semantic of implications or if-then rules in other areas are different, the logic can be used too. Using directly SL_{FD}, some automated deduction methods directly based on this inference system have been developed for classical systems and fuzzy systems [6–10].

Also, a generalization of SL_{FD} to the fuzzy framework [11] was developed. FASL, fuzzy attribute simplification logic, has become a helpful reasoning tool for the fuzzy extension.

As we have said, one of the main goals of the `fcaR` package is to provide a user-friendly computational interface to the principal operators and methods of binary-fuzzy FCA, including the mentioned logic tools. The use of R language can spread FCA to others communities.

As of today the package has 21 000 downloads, published in CRAN repositories (https://cran.rstudio.com/web/packages/fcaR/index.html) with a living live cycle https://github.com/Malaga-FCA-group/fcaR and with vignettes to spread the package https://neuroimaginador.github.io/fcaR/.

The work is organized as follows: Section 2 describes the internal classes implemented in the library. Section 3 shows how to use the package. In Section 4, a real application of the package is shown Finally, some conclusions and future works are presented in Section 5.

2 Structure of fcaR

The `fcaR` package provides data structures which allow the user to work seamlessly with formal contexts and sets of implications. More explicitly, the following main classes are implemented, using the `R6` object-oriented-programming paradigm in R:

- `FormalContext` encapsulates the definition of a formal context (G, M, I), being G the set of objects, M the set of attributes and I the (fuzzy) relationship matrix, and provides methods to operate on the context using FCA tools.
- `ImplicationSet` represents a set of implications over a specific formal context.
- `Set` encapsulates a class for storing variables (attributes or objects) in an efficient way.

As an advantage, object oriented programming style of R language and all the knowledge (concepts, implications, minimal generators, etc.) will be stored inside the *formal context object* `fc`.

The main and computationally hard methods of FCA have been developed in C and linked to `fcaR`.

3 fcaR

In this section, we present the very essential methods in the FCA framework using a well-known running example about planets. From a dataset, we build an *formal context object*, named `fc`, in R using the function `FormalContext`.

	small	medium	large	near	far	moon	no_moon
Mercury	×			×			×
Venus	×			×			×
Earth	×			×		×	
Mars	×			×		×	
Jupiter			×		×	×	
Saturn			×		×	×	
Uranus		×			×	×	
Neptune		×			×	×	
Pluto	×				×	×	

Table 1. Planets dataset.

Sets of attributes or objects will be stored in variables of type `Set`. For the variable `fc` containing the formal context, a list of some methods are available: `fc$clarify()`, `fc$attributes`, `fc$objects`, `fc$concepts`, `fc$implications`, etc.

As an example, with the planets dataset (Table 1), we compute the intent, extent and the closure of a set of attributes:

```
> set_objetcs <- Set$new(fc$objects)
> set_objetcs$assign(Mars = 1, Earth = 1)
> fc$intent(set_objetcs)
{small, near, moon}
> set_attributess1 <- Set$new(fc$attributes)
> set_attributess1$assign(medium = 1, far = 1)
> fc$extent(set_attributess1)
{Uranus, Neptune}
> set_attributess2 <- Set$new(fc$attributes)
> set_attributess2$assign(medium = 1)
> fc$closure(set_attributess2)
{medium, far, moon}
```

To extract knowledge, we will use some methods associated to the variable `fc`. Some concepts and implications are shown next:

```
> fc$find_concepts()
> fc$concepts[3:4]
```

```
A set of 2 concepts:
1: ({Jupiter, Saturn, Uranus, Neptune, Pluto}, {far, moon})
2: ({Jupiter, Saturn}, {large, far, moon})
> fc$find_implications()
> fc$implications[1:2]
Implication set with 2 implications.
Rule 1: {no_moon} -> {small, near}
Rule 2: {far} -> {moon}
```

And for the concepts and implications (inside the variable *fc*) we could use the main methods and algorithms developed:

– For concepts: `infimum()`, `supremum()`, `top()`, `bottom()`, `plot()`, `size()`, `join_irreducibles()`, `meet_irreducibles()`, `lower_neighbours()`, etc.
– For implications: `apply_rules()`, `cardinality()`, `to_basis()`, `filter()`, etc.

4 A case of study

In this section, a real case of study showing `fcaR` on real-world problems is presented. The goal is to extract knowledge about the features of tourist destinations given a user profile.

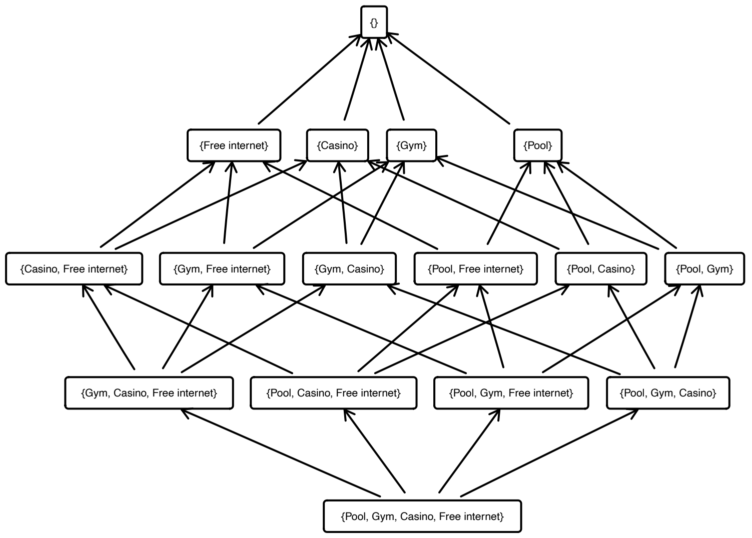
The dataset `vegas` (see more information in the package) stores more than 500 TripAdvisor reviews of hotels in Las Vegas Strip. The main attributes are:

– `Period of Stay`: 4 categories are present in the original data, which produces as many binary variables: `Dec-Feb`, `Mar-May`, `Jun-Aug` and `Sep-Nov`.
– `Traveler type`: five binary categories are created from the original data: `Business`, `Couples`, `Families`, `Friends` and `Solo`.
– `Pool`, `Gym`, `Tennis court`, `Spa`, `Casino`, `Free internet`: binary variables for the services offered by each destination hotel.
– `Stars`: five binary variables are created, according to the number of stars of the hotel, `3`, `3.5`, `4`, `4.5` and `5`.
– `Score`, the score assigned in the review, from `1` to `5`, five variables are created.

We can load the dataset, create a `FormalContext` object, and compute concepts and implications with:

```
> data(vegas)
> fc <- FormalContext$new(vegas)
> fc$find_implications()
```

In this case, it is complicated to visualize the lattice with 2082 concepts, thus we opt for plotting a sublattice where we impose a minimum support:



This exploration gives some hints about the most important attributes in the dataset. After that, the set of implications is manipulated to remove redundancies and remove those rules with zero support:

```
> fc$implications$apply_rules(c("simplification",
+                               "composition",
+                               "generalization"))
> fc$implications <- fc$implications[fc$implications$support() > 0]
```

We are now in position to pose the question that must be answered by means of the extracted knowledge: for a given couple, searching for a hotel in Las Vegas with Spa, which are the additional services that would make the highest score (5)?

In order to answer this question, let us begin with a subset of the implications, those related to couples travelling:

```
> base_implications <- fc$implications$filter("Traveler type=Couples")
```

Then, specify the minimum services (Spa) in a `Set`:

```
> Setattr1 <- Set$new(fc$attributes)
> Setattr1$assign("Traveler type=Couples" = 1, "Spa" = 1)
```

And compute the closure by using the simplification logic, since we are interested in the knowledge that can be inferred from the condition given by the set :

```
> cl <- base_implications$closure(Setattr1, reduce = TRUE)
> specific_implications <- cl$implications
```

There are 36 implications representing the knowledge in the formal context for the required case. Since the problem stated to extract the additional features needed to get a score of 5, let us filter the new `ImplicationSet` by this condition on the RHS removing redundancies previously:

```
> specific_implications$filter(rhs = c("Score=5"))
Implication set with 5 implications.
Rule 1: {Period of stay=Mar-May, Stars=4.5} -> {Score=5}
Rule 2: {Period of stay=Jun-Aug, Stars=4.5} -> {Score=5}
Rule 3: {Period of stay=Jun-Aug, Tennis court, Stars=3.5} -> {Score=5}
Rule 4: {Period of stay=Dec-Feb, Tennis court, Stars=3.5} -> {Score=5}
Rule 5: {Period of stay=Dec-Feb, Tennis court, Stars=3} -> {Score=5}
```

From these implications, we can infer the additional services that would make a perfect stay for the user.

5 Conclusions

The main objective in this work has been the development of an R package able to be useful not just for the FCA community but in general to perform knowledge retrieval from binary or fuzzy (graded) datasets. It is the first R package implementing the core methods in FCA.

To sum up, the `fcaR` package is designed to:

– Manage formal contexts (datasets), implementing the core notions of formal concept analysis: objects, attributes, derivation operators, concepts, closures, implications, etc.
– Extract the concepts and the concept lattice from a context.
– Find implications (exact association rules) that are true in the context.
– Provide tools to visualize the extracted knowledge.
– Implement the simplification logic for fuzzy and binary settings as the core of automated methods based on logic to remove redundancy in an easy way (only applying the rules of the logic), to compute closures and make recommendations.

Thus, `fcaR` implements a wide range of features, and with the help of the included documentation and vignettes, any user can start analysing datasets with FCA tools.

From the point of view of efficiency, the `fcaR` package uses the vectorial and parallelization capabilities of the R language, whereas algorithmic bottlenecks have been implemented in C. In addition, we have used sparse matrices as the main internal data structure of the package.

Currently, the package is under active development of several extensions or enhancements: improvement of the efficiency of fuzzy algorithms, adding other algorithms of the FCA community to compute the concept lattice or the implication basis, or the incorporation of advanced algorithms such as the calculation of direct bases of implications and minimal generators that have proved useful in practical applications.

References

1. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. In: Ordered sets. pp. 445–470. Springer (1982).
2. Ganter, B., Wille, R.: Formal concept analysis - mathematical foundations. Springer (1999).
3. Belohlávek, R., Vychodil, V.: Attribute dependencies for data with grades I. International Journal of General Systems. 45, 864–888 (2016).
4. Belohlávek, R., Vychodil, V.: Attribute dependencies for data with grades II. International Journal of General Systems. 46, 66–92 (2017). https://doi.org/10.1080/03081079.2016.1205712.
5. Cordero, P., Enciso, M., Mora, A., P. de Guzmán, I.: SL_{FD} logic: Elimination of data redundancy in knowledge representation. In: IBERAMIA. pp. 141–150. Springer (2002).
6. Mora, A., Enciso, M., Cordero, P., Guzmán, I.P. de: An efficient preprocessing transformation for functional dependencies sets based on the substitution paradigm. In: CAEPIA 2003. pp. 136–146. Springer (2003).
7. Cordero, P., Enciso, M., Mora, A., Ojeda-Aciego, M.: Computing minimal generators from implications: A logic-guided approach. In: CLA 2012. pp. 187–198. CEUR-WS.org (2012).
8. Mora, A., Cordero, P., Enciso, M., Fortes, I., Aguilera, G.: Closure via functional dependence simplification. International Journal of Computer Mathematics. 89, 510–526 (2012).
9. Rodríguez Lorenzo, E., Bertet, K., Cordero, P., Enciso, M., Mora, A.: The direct-optimal basis via reductions. In: CLA 2014. pp. 145–156. CEUR-WS.org (2014).
10. Rodríguez Lorenzo, E., Adaricheva, K.V., Cordero, P., Enciso, M., Mora, A.: From an implicational system to its corresponding d-basis. In: CLA 2015. pp. 217–228. CEUR-WS.org (2015).
11. Belohlávek, R., Cordero, P., Enciso, M., Mora, A., Vychodil, V.: Automated prover for attribute dependencies in data with grades. International Journal of Approximate Reasoning. 70, 51–67 (2016).