ELC1098 - Trabalho 1

Marcos Visentini, Henrique Fochesatto

Sistemas de Informação - CT - UFSM

{mvisentini,hfochesatto}@inf.ufsm.br

Índice

1. Observações

- Para a resolução do trabalho foi utilizada a linguagem de programação R e o ambiente RStudio, com o apoio das bibliotecas externas tools e arules.
- Os passos abaixo não necessariamente são realizados pelo script na sequência em que são descritos, mas estão organizados aqui de tal forma a fim de facilitar o entendimento do leitor.

2. Seleção e pré-processamento de dados

2.1. Importação

Primeiramente, é realizada a importação dos arquivos necessários (`_ASSOC_BGFriends_01.csv` e `_ASSOC_BGFriends_02.csv`) diretamente da página da disciplina, a fim de facilitar o manuseio destes evitando a necessidade de lidar com diretórios relativos. O segundo arquivo (`_ASSOC_BGFriends_02.csv`) contém problemas de codificação (e.g., símbolos latinos como "ç" não conseguem ser representados), então ele precisa ser importado com a codificação Latin-1 ao invés da codificação padrão (UTF-8).

Além disso, para facilitar a próxima etapa do processo, os dois arquivos foram combinados de forma a produzir apenas um data frame contendo todos os dados. Isso não foi um problema, já que eles possuem a mesma estrutura.

2.2. Limpeza

Após a importação dos arquivos, foi possível observar a presença de inconsistências nos dados. Na coluna `Jogadore(a)s`, há um conjunto finito de nomes. São eles: Alonso, Ana, Barbara, François, Jimmy, Rick, Shelda, Steeve e Yuriko. No entanto, eles não são representados sempre da mesma maneira, apesar de possuírem o mesmo significado. Por exemplo, por diversas vezes o nome "François" é escrito como "Francois", "Steeve" como "steeve", "Shelda" como "shelda", etc. São pequenas variações que não mudam o significado do nome efetivamente, mas interferem quando queremos representar uma variável específica.

Para solucionar esse problema, foi criada a função `achar_nomes_unicos`, que é responsável por encontrar os nomes únicos presentes na coluna `Jogadore(a)s`, retornando-os em um vetor. Essa função faz uso da função adist, utilizada para calcular a distância aproximada de um nome entre todos os nomes, ignorando aqueles que são muito semelhantes (ou seja, que potencialmente representam a mesma coisa). Como resultado, temos um vetor com apenas os nomes únicos do data frame.

Além de tratar os nomes, também é necessário remover a coluna `Partida`, já que ela contém apenas o número da partida, não agregando nenhum dado relevante.

Por fim, são removidas aquelas linhas onde a partida resultou em empate, já que elas não contribuem para a nossa análise e podem influenciar no resultado final de forma indesejada.

2.3. Transformação

Com os dados tratados, o próximo passo é prepará-los para a extração de regras de associação através da função apriori, que requer como um de seus argumentos um objeto que represente transações. Para isso, vamos utilizar o método One-Hot Encoding, que transforma cada categoria distinta em uma coluna, para então ser preenchida com 1 (Verdadeiro) ou 0 (Falso). No nosso caso, cada nome distinto de jogador(a) vai virar uma coluna e o valor dela vai ser 1 caso o(a) jogador(a) tenha participado da partida em questão ou 0 caso contrário.

Além das colunas com os nomes, é adicionada uma última coluna chamada `Vitoria`, que é preenchida com o valor 1 caso a partida tenha resultado em vitória ou 0 caso tenha resultado em derrota. Essa informação é obtida através da comparação entre as colunas `Oponentes` e `Amigos`: se `Oponentes[i] < Amigos[i]`, então é uma vitória; se `Oponentes[i] > Amigos[i]`, derrota. Com a adição da coluna `Vitoria`, não se faz mais necessário a presença das colunas `Oponentes` e `Amigos`, então elas são excluídas do data frame final.

Portanto, o data frame final utilizado para a extração de regras de associação contém 10 colunas: as 9 primeiras são os nomes de jogador(a) e a última contém o resultado da partida. Abaixo, um exemplo da estrutura final do data frame:

Alonso	Ana	Barbara	François	Jimmy	Rick	Shelda	Steeve	Yuriko	Vitoria
1	0	0	1	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0
0	0	0	0	1	1	0	0	1	1
0	0	1	0	1	0	0	0	1	1
0	0	1	1	0	0	0	1	0	0
0	0	0	0	0	1	1	0	1	1
0	0	0	1	1	0	0	0	1	1
0	0	0	1	1	0	0	1	0	1
0	0	1	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0

3. Regras de associação

Para extrair as regras de associação do data frame final, foi utilizada a função apriori da biblioteca arules. Os valores de suporte e confiança para as regras de simples foram, respectivamente, 0.1 e 0.5; para as de duplas, 0.1 e 0.7.

3.1. Melhor jogador

As regras para encontrar o melhor jogador foram encontradas após filtrar o conjunto de regras de simples pelos seguintes critérios:

- `size(lhs) == 1`: O lhs da regra deve conter exatamente 1 variável (um jogador).
- `lhs %pin% "=1"`: O lhs da regra deve conter apenas variáveis com o valor 1 (jogadores que participaram da partida).
- `rhs %in% "Vitoria=1"`: O rhs da regra deve conter apenas a variável `Vitoria` com o valor 1.

Ao inspecionar o subconjunto gerado e ordená-lo pela maior confiança, as seguintes regras foram geradas:

```
> inspect(sort(regras_melhor_jogador, decreasing = TRUE, by = "confidence"), linebreak = FALSE)
    lhs             rhs          support   confidence coverage  lift     count
[1] {Yuriko=1} => {Vitoria=1} 0.2500000 0.5937500  0.4210526 1.307971 38
[2] {Jimmy=1}  => {Vitoria=1} 0.2236842 0.5483871  0.4078947 1.208041 34
[3] {Rick=1}   => {Vitoria=1} 0.1578947 0.5333333  0.2960526 1.174879 24
>
```

Ou seja, o melhor jogador foi Yuriko, com suporte de 0.25 e confiança de 0.59.

3.2. Melhor dupla

As regras para encontrar a melhor dupla foram encontradas após filtrar o conjunto de regras de duplas pelos seguintes critérios:

- `size(lhs) == 2`: O lhs da regra deve conter exatamente 2 variáveis (dois jogadores).
- `!(lhs %pin% "=0")`: O lhs da regra não deve conter variáveis com o valor 0 (jogadores que não participaram da partida).

- `rhs %in% "Vitoria=1"`: O rhs da regra deve conter apenas a variável `Vitoria` com o valor 1 .

Ao inspecionar o subconjunto gerado e ordená-lo pela maior confiança, foi obtida a seguinte regra:

```
> inspect(sort(regras_melhor_dupla, decreasing = TRUE, by = "confidence"), linebreak = FALSE)
    lhs                     rhs          support confidence coverage  lift     count
[1] {Jimmy=1, Yuriko=1} => {Vitoria=1} 0.125   0.7037037  0.1776316 1.550188 19
> |
```

Com base nela, podemos dizer que a melhor dupla foi Jimmy e Yuriko, com suporte de 0.125 e confiança de 0.7.

3.3. Pior dupla

As regras para encontrar a pior dupla foram encontradas após filtrar o conjunto de regras de duplas pelos seguintes critérios:

- `size(lhs) == 2`: O lhs da regra deve conter exatamente 2 variáveis (dois jogadores).
- `!(lhs %pin% "=0")`: O lhs da regra não deve conter variáveis com o valor 0 (jogadores que não participaram da partida).
- `rhs %in% "Vitoria=0"`: O rhs da regra deve conter apenas a variável `Vitoria` com o valor 1 .

Ao inspecionar o subconjunto gerado e ordená-lo pela maior confiança, a seguinte regra foi obtida:

```
> inspect(sort(regras_pior_dupla, decreasing = TRUE, by = "confidence"), linebreak = FALSE)
    lhs                          rhs          support   confidence coverage  lift     count
[1] {Barbara=1, François=1} => {Vitoria=0} 0.1710526 0.7878788  0.2171053 1.442862 26
> |
```

Ou seja, a pior dupla foi Barbara e François, com suporte de 0.17 e confiança de 0.79.

4. Conclusão

- Melhor jogador: Yuriko (suporte = 0.25, confiança = 0.59).
- Melhor dupla: Jimmy e Yuriko (suporte = 0.125, confiança = 0.7).
- Pior dupla: Barbara e François (suporte = 0.17, confiança = 0.79).