REVISTA DO SEMINÁRIO INTERNACIONAL DE ESTATÍSTICA COM R | **ISSN: 2526-7299**
VOL 5, Nº 1, ABRIL DE 2024

NOVAS FUNCIONALIDADES DO PACOTE AHP

Lyncoln Sousa Oliveira - Universidade Federal Fluminense

Luciane Ferreira Alcoforado - Academia da Força Aérea

Steven Dutt Ross - Universidade Federal do Estado do Rio de Janeiro

Rodolfo Hauret - Universidade Federal Fluminense

Resumo.

Este trabalho é uma continuação de um projeto desenvolvido com apoio da bolsa do programa de Iniciação Científica (PIBIC) iniciado em 2018. O projeto é focado no uso da linguagem R, utilizando bibliotecas auxiliares como algumas que compõe o *tidyverse*, para criação de funções que realizam o método AHP (*Analytic Hierarchy Process*) para uma base de dados devidamente formatada, podendo ser usados objetos R ou dados auxiliares como planilhas do formato xlsx. Inicialmente o projeto foi desenvolvido para realizar o método em tipos de hierarquia simples, somente critérios sem subcritérios. Neste trabalho será apresentado as novas funções que aplicam o método para hierarquias compostas, isto é, com critérios e subcritérios, e também funções que realizam procedimentos auxiliares como a criação de uma base de dados utilizando um aplicativo desenvolvido com o pacote *shiny*.

Palavras-chave: AHP, Saaty, pacote, R.

Abstract

This work is a continuation of a project developed with support from the *Cientific Initiation Program (PIBIC)* grant initiated in 2018. The project is focused on the use of the R language, using auxiliary libraries such as some that make up tidyverse, to create functions that perform the AHP method for a properly formatted database, that database can be objects in R or auxiliary data such as spreadsheets of the xlsx format. Initially the project was developed to carry out the method in simple hierarchy types, only criteria without sub-criteria. In this work will be present the new functions that apply the method for composite hierarchies, with criteria and subcriteria, as well as functions that perform auxiliary procedures such as the creation of a database using an application developed with the shiny package.

Keywords: AHP, Saaty, package, R.

Introdução

O processo de tomada de decisão sob incerteza tem mostrado importância em qualquer situação da vida pessoal ou profissional de um indivíduo, uma vez que o ser humano é levado a tomar decisões em grande parte do seu tempo de vida. Decisões uma vez tomadas, podem se revelar boas ou ruins a curto, médio e a longo prazo. Ao se tomar uma decisão de forma intuitiva nem sempre é possível prever se a alternativa escolhida é a mais viável considerando-

se alguns critérios subjetivos. Tendo em vista esse pensamento, justifica-se estudar métodos matemáticos/estatísticos que possam auxiliar a tomada de decisão em situações práticas.

A tomada de decisão nas organizações tem sido objeto de constantes pesquisas e estudos comprovando a importância que este tema representa no desempenho dessas organizações. Segundo Gomes et al (2002), um sistema de apoio à decisão (SAD) é uma ferramenta computacional que envolve técnicas de sistemas de informação, inteligência artificial, métodos quantitativos, psicologia cognitiva e comportamental, sociologia das organizações, entre outros, e visam oferecer ao usuário condições favoráveis e acessíveis ao suporte, para de modo prático, melhor escolher uma entre diversas alternativas, minimizando assim a chance de erro na tomada de decisão.

Um SAD concilia os recursos intelectuais individuais com a capacidade do computador em melhorar a qualidade da decisão (MORTON E KEEN, 1978), assim, o apoio à decisão significa auxiliar a tomada de decisão na escolha de alternativas, gerando as estimativas dos pesos destas alternativas, a comparação e a escolha.

O processo *Analytic Hierarchy Process* (AHP), baseado em matemática e psicologia, foi desenvolvido na década de 1970 pelo professor Thomas Saaty. O AHP pode ser classificado como o mais conhecido e utilizado dos métodos de análise multicritério cuja modelagem se divide em três etapas: construção dos níveis hierárquicos, definição das prioridades através de julgamentos paritários dos critérios estabelecidos e avaliação da consistência lógica dos julgamentos paritários.

Neste trabalho apresenta-se a implementação do Método de AHP proposto por (SAATY, 1991), utilizando-se a linguagem computacional R para automatização do método e apresentação dos resultados de maneira intuitiva para melhorar a experiência do usuário.

Objetivos

Objetivo Geral: Aprimorar e desenvolver funções para o pacote AHP utilizando a linguagem computacional R.

Objetivos Específicos: Desenvolver a estrutura do banco de dados para os valores de entrada à luz da compreensão do método AHP e suas etapas; criar a estrutura de saída dos resultados para problemas com critério e subcritério; construir funções auxiliares para o desenvolvimento do método; desenvolver um sistema para o processamento da AHP através de um aplicativo *shiny* integrado ao pacote.

Material e Método

O método AHP

O AHP (*Analytic Hierarchy Process)* é um dos processos de SAD (Sistema de apoio a decisão) bastante conhecido e utilizado para problemas complexos e subjetivos que envolvam escolha de alternativas mediantes a múltiplos critérios. Criado por Thomas L. Saaty na década de 1970, o método tem como principal característica o uso de um modelo matemático que reflita o funcionamento da mente humana na avaliação de alternativas diante de um problema de decisão. Além disso, o método permite lidar com problemas que envolvem tanto os valores tangíveis como os intangíveis, graças a sua capacidade de criar medidas para as variáveis qualitativas com base em julgamentos subjetivos emitidos pelos decisores (Saaty, 1991). Como os decisores são humanos, além da escolha da melhor alternativa mediante aos critérios, um dos objetivos da AHP é utilizar ferramentas matemáticas para medir a consistência de cada julgamento.

De acordo com o Saaty (1991), a aplicação do AHP pode ser dividia nas seguintes etapas: estruturação dos critérios e alternativas; coleta de julgamentos; cálculo de prioridades; verificação da consistência do julgamento; e, por último, o cálculo das prioridades globais das alternativas.

A estruturação dos critérios consiste em modelar o problema de decisão em uma estrutura do tipo árvore hierárquica, onde o objetivo principal é ramificado em critérios que por sua vez são ramificados em alternativas. As alternativas são avaliadas mediante a cada critério que por sua vez são avaliados de maneira geral em relação ao objetivo.

Para a fase de coleta de julgamentos Saaty sugere que o indivíduo que julga seja um conhecedor do objetivo avaliado, para que assim os riscos de inconsistência sejam reduzidos. Os julgamentos são organizados em uma matriz que é chamada de matriz de paridade ou de julgamento, onde se compara primeiramente a importância dos critérios e depois das alternativas mediante aos critérios. As comparações são feitas 2 a 2.

O julgamento deve ser baseado na escala de Saaty (1991) conforme a Tabela 1, buscando-se primeiro o julgamento conceitual e, em seguida, a conversão para a escala numérica a fim de registrá-lo na matriz, como também, o julgamento recíproco associado (Costa, HG 2002).

Tabela 1 – Escala de julgamento de Saaty

Escala Verbal	Escala Numérica
Igual preferência (importância)	1
Preferência (importância) moderada	3
Preferência (importância) forte	5
Preferência (importância) muito forte	7
Preferência (importância) absoluta	9

2,4,6 e 8 são associados à julgamentos intermediários.

Fonte: Costa, 2002

Na etapa de cálculo de prioridades são utilizadas ferramentas matemáticas como autovetores e autovalores nas matrizes de julgamento para quantificar os pesos de cada critério e também de cada alternativa em relação ao objetivo.

Na verificação da consistência do julgamento considera-se as dificuldades naturais do ser humano em tomar decisões diante de um problema com muitas informações que possuem diversos critérios, Saaty (1991) propôs procedimentos para calcular a inconsistência derivadas dos julgamentos, estas são: IC (Índice de Consistência) e RC (Razão de consistência).

$$IC = \frac{\lambda}{n-1} \tag{1}$$

- λ refere − se ao autovetor da matriz
- n refere − se a ordem da matriz

Conforme Saaty (1991), o índice randômico, *Random Index* (RI), é o índice de consistência de uma matriz recíproca gerada, randomicamente, pelo laboratório Oak Ridge. A Tabela 2 mostra a tabela RI contendo os índices randômicos calculados pelo laboratório Oak Ridge para matrizes recíprocas quadradas de ordem n.

Tabela 2 – Índices de consistência aleatória

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Fonte: Saaty, 1991

$$RC = \frac{CI}{RI} \tag{2}$$

Assim é possível calcular o RC para cada matriz de julgamento. Caso o RC calculado para uma matriz seja superior a 10%, Saaty sugere que o indivíduo refaça os julgamentos com mais atenção.

Por fim é realizado o cálculo das prioridades globais das alternativas. De acordo com Saaty (1991), e tomando como base a estrutura hierárquica do AHP, as prioridades globais calculadas para cada critério correspondem à importância de cada critério em relação ao objetivo. Entretanto, no nível das alternativas, a prioridade é encontrada ao se multiplicar a prioridade local da alternativa em relação a um determinado critério pela prioridade global deste. Portanto, para se obter a prioridade global das alternativas, deve-se calcular o somatório das prioridades globais das alternativas calculadas em cada critério. Essa prioridade determinará a contribuição da alternativa para o objetivo principal.

Criação do pacote

Foram utilizadas as plataformas Git e Github para a hospedagem e criação do pacote. O Git é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, enquanto o GitHub é uma plataforma de hospedagem de código-fonte com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou de código aberto de qualquer lugar do mundo.

Foi necessário a criação de repositórios no github para a hospedagem das funções e documentação do pacote criado. Para a primeira fase do projeto os códigos e documentação estão disponíveis em: https://github.com/Lyncoln/AHP . Já os da nova etapa estão disponíveis em: https://github.com/Lyncoln/AHP2 , neste está visível a seguinte arrumação de pastas:

- R: Códigos do pacote.

- Banco_de_dados: Dados em formato xlsx (objeto Excel) para testes e criação de base de dados.
- Data: Dados em formato rda (objeto R).
- Man: Documentação das funções e dados do pacote.
- Vignettes: Arquivos de ajuda à utilização do pacote em formato markdown.
- Documentacao: Documentos escritos do pacote

A estrutura citada foi criada utilizando o pacote *devtools*, que é uma biblioteca criada pelo Hadley Wickham que está disponível no github e também no CRAN. Esta biblioteca é bastante conhecida e utilizada pelos desenvolvedores de pacotes em R. Nela existem ferramentas que tornam o processo de criação de bibliotecas mais simples.

Pelo motivo deste pacote ter sido criado em um computador que possui sistema operacional Windows, foi necessário a instalação do *Rtools,* que é uma coleção de recursos para a criação de pacotes para *o* R no *Microsoft Windows* ou para a criação do próprio R, criado pelo Prof. Brian Ripley e Ducan Murdoch; atualmente mantido pela Jeroen Ooms.

Para a criação das funções do pacote AHP foram utilizadas as seguintes bibliotecas auxiliares*: formattable, dplyr, tibble, readxl e tidyr*. As funções foram criadas com o intuito de serem simples para a aplicação e apresentar uma saída visualmente intuitiva para o usuário

Implementou-se funções para facilitar ao usuário a aplicação do método. Para os cálculos envolvidos no método foram implementadas as funções com as seguintes etapas:

- Ler os dados (matrizes de julgamentos fornecida pelo usuário);
- Calcular os pesos e a consistência;
- Retornar tabela com os pesos finais de cada alternativa, informando o índice de consistência dos julgamentos de cada critério considerado no problema.

Foi utilizado a biblioteca *shiny* para a criação de um aplicativo que oriente o usuário a criar as matrizes paritárias de forma interativa no próprio ambiente R. O aplicativo ainda está com um visual e aplicações simples pois ainda está na fase piloto.

Resultados e Discussão

Na primeira fase do projeto cujo o pacote está disponível em https://github.com/Lyncoln/AHP foram criado códigos para a utilização do método AHP para problemas somente com um critério. Já a nova etapa que será apresentada neste trabalho, que está disponível https://github.com/Lyncoln/AHP2 , foram desenvolvidas funções que

aplicam o método AHP para problemas com critérios e subcritérios, além de procedimentos auxiliares para facilitar a utilização do usuário.

Para realização do método, o programa espera que o usuário possua as matrizes de paridade em planilhas no formato xlsx, ou que o usuário utilize as funções auxiliares do pacote para criar as matrizes como objeto R.

Foram desenvolvidas algumas bases de dados exemplo que podem ser acessadas ao carregar o pacote, usando a função help do R em cada base é possível visualizar um cenário fictício em que um especialista criou as matrizes de julgamento e como estão estruturadas. Essas bases foram chamadas de BD1, BD2 e BD3. Os dados de BD1 e BD2 são provenientes de um problema somente com um nível de critério, isto é, sem subcritérios, já BD3 possui critérios e subcritérios.

Os códigos foram criados para aceitarem tanto objetos lista com matrizes de paridade no R e também planilhas em formato xlsx. As arrumações das matrizes devem respeitar a seguinte posição: Primeiro a matriz de comparação dos critérios em relação ao objetivo, em seguida, se houver subcritério a matriz de comparação dos subcritérios em relação ao seu critério, e pôr fim a matriz de comparação das alternativas mediante a cada subcritério (se houver) ou a cada critério.

Para exemplificação observe a figura 1, que apresenta uma hierarquia para um problema cujo o objetivo é escolha de uma linguagem de computação

Figura 1 – Hierarquia para escolha de uma linguagem de computação
Fonte: Autor, 2019

É necessário que as matrizes de paridade estejam organizadas do seguinte modo:

1. Matriz de comparação dos critérios a luz do objetivo que é a escolha de uma linguagem.

2. Matriz de comparação das alternativas mediante ao critério rendimento (não possui subcritérios).
3. Matriz de comparação dos subcritérios Suporte.
4. Matriz de comparação das alternativas mediante ao subcritério Comunidade.
5. Matriz de comparação das alternativas mediante ao subcritério Material.
6. Matriz de comparação das alternativas mediante ao subcritério Popularidade.

Caso o usuário escolha usar um arquivo de planilhas em formato xlsx, as planilhas devem ser organizadas de acordo com a figura 2.

Figura 2 – Arrumação de planilhas xlsx

Fonte: Autor, 2019

Atualmente o pacote conta com 6 funções e essas são: *ahp_geral(), ranque(), CR(), matriz_julgamento(), calcula_prioridade(), formata_tabela()*. Para exemplificação serão utilizadas a base de dados BD1 já disponível no pacote que apresente um cenário fictício de escolha de um carro entre 3 alternativas como também um segundo exemplo representado pela hierarquia apresentada na figura 1. O cenário do BD1 é um problema de apenas um nível de critérios sem subcritérios, estes critérios são: Conforto (CF); Aquisição (AQ); Prestígio (PS); Revenda (RV) e Manutenção (MA). A hierarquia pode ser observada na figura 3.

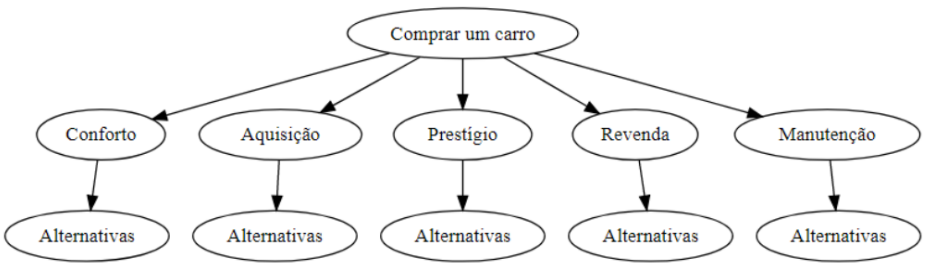


Figura 3 – Hierarquia do problema do BD1

Fonte: Pacote AHP, 2019

A função *CR* é da forma *CR(matriz),* espera-se como argumento uma matriz para que se possa retornar o *Consitency Ratio* (Razão de Consistência) dessa matriz. Exemplo para a matriz de comparação de critérios do BD1 está disponível na figura 4. Como o CR é menor que 10%, logo a matriz de julgamento é considera consistente.

```
> CR(BD1[[1]])
[1] 0.07091014
```

Figura 4 – Razão de Consistência dos critérios do BD1

Fonte: Pacote AHP, 2019

A função *calcula_prioridades* é da forma *calcula_prioriades(lista),* espera-se como entrada a lista contendo as matrizes de paridade para que possa retornar os pesos de cada critério e de cada alternativa mediante a cada critério, observe na figura 5 a aplicação da função no BD1.

A saída é uma lista de vetores. O primeiro vetor possui os pesos de cada alternativa em ordem, isto é, 0.08627388 é o peso para o critério CF, 0.44979358 é referente a AQ, assim por diante. Do segundo ao último vetor são os pesos das alternativas mediantes a cada critério, por exemplo, 0.1 é o peso da primeira alternativa a luz do critério CF.

```
> calcula_prioridades(BD1)
$`Comprar um carro`
[1] 0.08627388 0.44979358 0.05881037 0.21879372 0.18632845

$CF
[1] 0.1 0.3 0.6

$AQ
[1] 0.1958004 0.4933860 0.3108137

$PS
[1] 0.4125989 0.3274800 0.2599210

$RV
[1] 0.5396146 0.2969613 0.1634241

$MA
[1] 0.6369856 0.1047294 0.2582850
```

Figura 5 – Peso de critérios e alternativas para o BD1

Fonte: Pacote AHP, 2019

A função *ahp_geral* é da forma *ahp_geral(base, mapeamento = "PADRÃO", nomes_alternativas = "PADRÃO").* A base de entrada pode ser uma lista contendo matrizes de paridade no R ou um caminho para um arquivo de planilhas xlsx.

O argumento mapeamento deve ser preenchido somente em problemas que existem subcritérios e deve ser preenchido da forma *c(quantidade de subcritérios do primeiro critério, quantidade de critérios do segundo subcritério, ..., quantidade de subcritérios do último critério)*.

Em *nomes_alternativas* deve ser preenchido caso o usuário queira alterar o nome das alternativas na tabela de saída, se não preenchido será atribuído A, B, …, Z conforme a quantidade de alternativas.

O resultado é uma tabela contendo as informações dos pesos de cada critério em relação ao objetivo e também os pesos de cada alternativa mediante aos critérios e também ao foco objetivo e pôr fim a coluna CR que apresenta a razão de consistência de cada matriz da base de dados.

Na figura 6 é possível visualizar a entrada e saída da função para o problema do BD1 (sem subcritérios). Na figura 7 é mostrado como deve ser preenchido o argumento mapeamento para o problema com subcritérios de escolha de uma linguagem tratado anteriormente. O mapeamento deve ser escrito da forma *c(0,3).*

A saída pode ser interpretada da seguinte forma: A coluna Peso representa as proporções de preferências para cada critério e subcritério. Na primeira linha é possível identificar os pesos das alternativas. Prefere-se a alternativa que possui maior peso, tendo em vista que não é recomendável possuir um valor maior que 10% na coluna CR, se possuir, é necessário refazer o julgamento para aquela matriz.

```
> ahp_geral(BD1)
# A tibble: 6 x 6
  Criterios            Pesos       A       B       C       CR
  <chr>                <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
1 ---Comprar um carro 1      0.358   0.352   0.291   7.09e- 2
2 --CF                0.0863 0.00863 0.0259 0.0518 4.27e-16
3 --AQ                0.450  0.0881  0.222   0.140   5.16e- 2
4 --PS                0.0588 0.0243  0.0193 0.0153 5.16e- 2
5 --RV                0.219  0.118   0.0650 0.0358 8.85e- 3
6 --MA                0.186  0.119   0.0195 0.0481 3.70e- 2
```

Figura 6 – Saída da função ahp_geral para BD1

Fonte: Pacote AHP, 2019

```
> ahp_geral(objeto = "escolhaL.xlsx",
+           mapeamento = c(0,3),
+           nomes_alternativas = c("R","Python","C++","Java"))
# A tibble: 6 x 7
  Criterios             Pesos      R Python   `C++`    Java      CR
  <chr>                 <dbl>  <dbl>  <dbl>   <dbl>   <dbl>   <dbl>
1 ---Elige un programa 1      0.401  0.273  0.140   0.186  0
2 --Rendimiento        0.6    0.188  0.190  0.0966  0.125  0.0387
3 --Soporte            0.4    0.213  0.0825 0.0430  0.0615 0.0176
4 -Comunidad           0.250  0.155  0.0460 0.0227  0.0264 0.0736
5 -Material            0.0954 0.0406 0.0203 0.0126  0.0219 0.00388
6 -Popularidad         0.0546 0.0175 0.0162 0.00765 0.0133 0.0237
```

Figura 7 – Saída da função ahp_geral para o problema de escolha de linguagem

Fonte: Pacote AHP, 2019

Para utilizar a função *ranque*, que é da forma *ranque(tabela),* é necessário salvar a tabela resultado da *ahp_geral* em um objeto e passar como argumento. A função retorna somente os pesos das alternativas em relação ao objetivo. Observe as figuras 8 e 9 para os problemas do BD1 e escolha de linguagem, respectivamente. A coluna pesos apresenta de forma crescente as melhores alternativas para o problema. No exemplo do BD1 o melhor carro a ser comprado é o A, enquanto no da escolha de uma linguagem, a melhor é o R.

```
> ranque(tabelaBD1)
# A tibble: 3 x 3
  Ranque Alternativas Pesos
   <int> <chr>         <dbl>
1      1 A             0.358
2      2 B             0.352
3      3 C             0.291
```

Figura 8 – Ranque das alternativas do BD1

Fonte: Pacote AHP, 2019

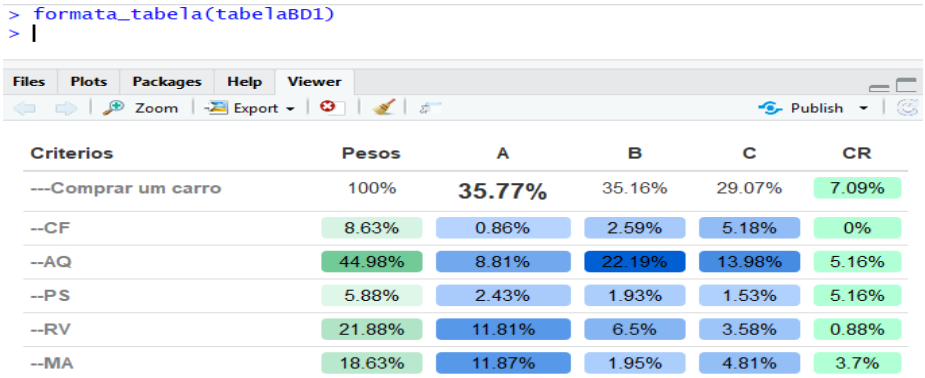
```
> ranque(tabelaEscolhaL)
# A tibble: 4 x 3
  Ranque Alternativas Pesos
   <int> <chr>         <dbl>
1      1 R             0.401
2      2 Python        0.273
3      3 Java          0.186
4      4 C++           0.140
```

Figura 9 – Ranque das alternativas da escolha de uma linguagem

Fonte: Pacote AHP, 2019

A função *formata_tabela* que é da forma *formata_tabela(tabela, cores = "PADRAO")* foi criada para que o usuário possa obter uma tabela visualmente mais atrativa e intuitiva. Os argumentos de entrada são a tabela como um objeto R, como a função *ranque()* e a cor que o usuário queira para a tabela. Atualmente a função conta com as cores "PADRAO" que se escolhida irá ser aplicado um degrade que vai do verde ao azul, quanto maior o peso da alternativa ou do critério, mais próximo do azul irá ser apresentado; "CINZA" o degrade irá ser

de um cinza mais claro para um cinza mais escuro; ou “BRANCO” caso o usuário não queira o efeito de cores. A Figura 10 apresenta a função sendo utilizada para o banco de dados BD1 com as cores “PADRAO”.



Criterios	Pesos	A	B	C	CR
---Comprar um carro	100%	**35.77%**	35.16%	29.07%	7.09%
--CF	8.63%	0.86%	2.59%	5.18%	0%
--AQ	44.98%	8.81%	22.19%	13.98%	5.16%
--PS	5.88%	2.43%	1.93%	1.53%	5.16%
--RV	21.88%	11.81%	6.5%	3.58%	0.88%
--MA	18.63%	11.87%	1.95%	4.81%	3.7%

Figura 10 – Saída da formata_tabela() para a base BD1

Fonte: Pacote AHP, 2019

A função *matriz_julgamento()* que é da forma *matriz_julgamento(qtd_comparacoes, CR = TRUE, qtd_matrizes = 1)* orienta a criação de matrizes na interface do R. o argumento *qtd_comparações* é para ser preenchido pela quantidade de comparações da matriz. *CR = TRUE* apresenta a razão de consistência junto com a matriz de julgamento, e o *qtd_matrizes* informa a quantidade de matrizes a ser criada. Observe a criação de matrizes pela figura 11.

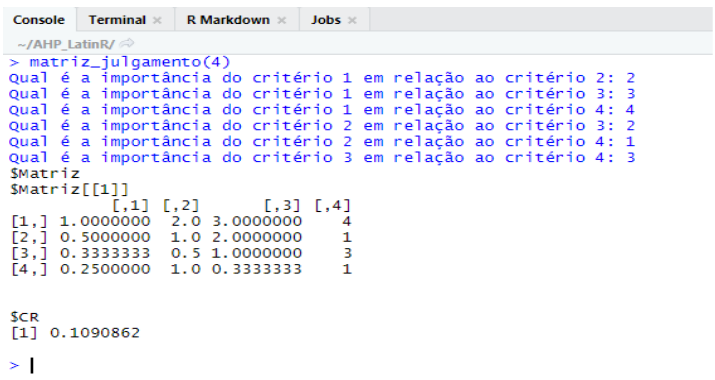


Figura 11 – Criação de Matriz de julgamento pela função matriz_julgamento()

Fonte: Pacote AHP, 2019

Está sendo desenvolvido um aplicativo utilizando o pacote *shiny* que orienta a criação de matrizes de julgamento de forma intuitiva na própria interface do R e disponibiliza o código

de criação da matriz para o usuário. O aplicativo ainda está na fase piloto e pode ser visto na figura 12.

Figura 12 – Criação de matrizes pelo app shiny em desenvolvimento

Fonte: Pacote AHP, 2020

Caso o usuário tenha alguma dúvida em utilizar alguma das funções do pacote, pode ser usado a função *help()* do R seguido do nome da função, ou então também é possível visualizar a *vignette* tutorial onde é apresentado cada função e uma aplicação seguido de uma explicação de como deve ser realizada a entrada dos dados.

Conclusão

A escolha da linguagem R para implementação e manutenção do pacote tem se mostrado viável, consistente e segura. Foram implementadas novas funções, ampliando o alcance do método para problemas mais gerais, contemplando níveis de critérios e subcritérios. O pacote foi elaborado para facilitar a entrada dos dados, assim dando ao usuário possibilidades de escolha, podendo ser utilizado o próprio R, ou planilhas externas, com os dados estruturados de acordo com os critérios considerados na hierarquização do problema. As funções foram implementadas na linguagem R de forma a produzir uma tabela final com um resumo das probabilidades de cada critério, subcritérios e alternativas. Foram desenvolvidas opções de formatação da tabela final de resultados, ampliando sua aplicabilidade para uma rápida interpretação por parte do tomador de decisão, podendo ser publicada em apresentações ou artigos.

A principal vantagem deste pacote é a facilidade que o usuário tem para realizar a entrada de dados que também foi desenhada para ser feita por arquivo do tipo xlsx. As

tabelas de saída são apresentadas de forma compacta para que o tomador de decisão possa atingir seu objetivo com rapidez e eficiência.

O pacote apresenta tutorial para que o usuário consiga utilizá-lo baseando-se num exemplo prático, tornando-se útil para gestores com conhecimentos básicos de linguagem de programação.

Como ações futuras será aprimorada a aplicação *shiny* com foco na utilização das funções do pacote AHP para usuários que não dominam o R, para isso o aplicativo será transformado em um *addin* para o RStúdio.

Este projeto está sendo desenvolvido na Universidade Federal Fluminense com o apoio da bolsa do programa de Iniciação Científica (PIBIC).

Referências

COSTA, H. G. Introdução ao método de análise hierárquica: análise multicritério no auxílio à decisão. Niterói, RJ, 2002.

COSTA, J.F.S., GONÇALVES, G.C., VAZ, L.M.M et al. Uma abordagem multicritério da telefonia móvel no Estado do Rio de Janeiro através do Método de Análise Hierárquica (AHP). Cadernos do IME – Série Estatística, RJ, 2007.

GOMES, L. F., GOMES, C. F. S., ALMEIDA, A. T. Tomada de Decisão Gerencial: Enfoque Multicritério. Ed Atlas, SP, 2002.

KEEN, P.G.W, & MORTON, S. Decision support systems: an organization perspective. AddisonWesley. Reading, Mass, 1978.

R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/, 2018.

WICKHAM, H.; BRYAN, J. *readxl: Read Excel Files. R package version 1.0.0.* https://CRAN.R-project.org/package=readxl, 2017.

MÜLLER, K. & WICKHAM, H. *tibble: Simple Data Frames. R package version 1.4.2.* https://CRAN.R-project.org/package=tibble, 2018.

WHICKHAM, H. & HENRY, L. *tidyr: Tidy Messy Data. R package version* 1.0.0. https://CRAN.R-project.org/package=tidyr, 2019.

WHICKHAM, H.; HENRY, L.; FRANÇOIS, L. dplyr: *A Grammar of Data Manipulation. R package version* 0.8.3. https://CRAN.R-project.org/package=dplyr, 2019.

REN, K. & RUSSELL, K. *formattable: Create "Formattable" Data Strucutres. R package version 0.2.0.1.* https://CRAN.R-project.org/package=formattable, 2016.

SAATY, T. L. Método de Análise Hierárquica. Rio de Janeiro: Makrom Books, 2Ed, 1991.

GOMEDE, E. & MIRANDA, R. Utilizando o Método Analytic Hierarchy Process (AHP) para Priorização de Serviços de TI: Um Estudo de Caso, 2012. URL: http://www.lbd.dcc.ufmg.br/colecoes/sbsi/2012/0041.pdf

RIBEIRO, M. C. C. & ALVES, A. S. Aplicação do método AHP com a mensuração absoluta num problema de seleção qualitativa. Revistasg, 2016. URL: http://www.revistasg.uff.br/index.php/sg/article/view/988/493

Anexo 1: Códigos auxiliares para aplicação do AHP.

```
autoVetor = function(matriz){

  #Achando o autovetor associado ao maior autovalor
  autoValores = Re(eigen(matriz)$values)
  autoVetores = Re(eigen(matriz)$vectors)
  autoValorMax = which.max(autoValores)
  autoVetorAssociado = autoVetores[,autoValorMax]
  #

  autoVetorNormalizado = autoVetorAssociado/sum(autoVetorAssociado)

  return(autoVetorNormalizado)


}
#'CR
#'
#'Calcula a taxa de consistencia de saaty para uma matriz pareada
#'
#'@param Matriz pareada
#'
#'@return Taxa de consistencia de saaty
#'
#'@export
CR = function(matriz){
  autoValores = Re(eigen(matriz)$values)
  autoVetores = Re(eigen(matriz)$vectors)
  autoValorMax = max(autoValores)
  tamanhoMatriz = length(matriz[1,])
  IndiceConsistencia = abs(autoValorMax - tamanhoMatriz) / (tamanhoMatriz - 1)
  consistenciaAleatoria =
c(0,0,0.52,0.89,1.11,1.25,1.35,1.40,1.45,1.49,1.51,1.54,1.56,1.57,1.58)
  if(consistenciaAleatoria[tamanhoMatriz]==0) return(0)
  return(IndiceConsistencia/consistenciaAleatoria[tamanhoMatriz])
}

#'matriz_julgamento
#'
#'Cria matrizes pareadas e pode testar taxa de consistencia de saaty
#'
#'@param qtd_comparacoes Quantida de elementos para serem avaliados
#'@param CR Se TRUE retorna tambem a taxa de consistencia de saaty, se FALSE
retorna apenas matriz
```

```
#'@param qtd_matrizes Quantidade de matrizes a serem criadas
#'
#'@return Retorna uma lista com 2 posicoes. Primeira posicao contem as
matrizes pareadas e a segunda posicao as suas taxas de consistencia
#'
#'@export

matriz_julgamento = function(qtd_comparacoes,CR = TRUE, qtd_matrizes = 1){
  matrizes = list()
  erros = c()
  conjunto = list()
  for(k in 1:qtd_matrizes){
  if(qtd_matrizes != 1) print(paste0("Preencha a matriz ",as.character(k)))
  matriz = diag(1, qtd_comparacoes ,qtd_comparacoes)
    for (i in 1:(qtd_comparacoes-1)){
      for(j in (i+1):(qtd_comparacoes)){
      valor = eval(parse(text = (readline(paste0("Qual é a importância do
critério ",as.character(i)," em relação ao critério ",as.character(j),":
")))))
      matriz[i,j] = valor
      matriz[j,i] = 1/valor
      }
    }
    matrizes[[k]] = matriz
    if(CR == TRUE) erros[k] = CR(matriz)
  }

  conjunto[[1]] = matrizes; names(conjunto) = "Matriz"
  if(CR == TRUE) {conjunto[[2]] = erros; names(conjunto) = c("Matriz","CR")}
  return(conjunto)
}



#'ler
#'
#'Le um arquivo do excel contendo as matrizes pareadas e transforma todas suas
planilhas em uma lista de matrizes no R
#'
#'@param caminho Endereco para um arquivo excel que contem as planilhas
#'
#'@return Retorna uma lista contendo as matrizes pareadas do arquivo excel
#'
#'@import readxl
#'
```

```
#'@export

ler = function(caminho){
  planilhas = readxl::excel_sheets(caminho)
  matrizes = suppressMessages(lapply(planilhas ,function(x)
readxl::read_excel(path = caminho, col_names = FALSE, sheet = x)))
  names(matrizes) = planilhas
  return(matrizes)
}

#'calcula_prioridades
#'
#'Calcula o vetor de prioridades de uma matriz pareada
#'
#'@param lista Lista de matrizes pareadas
#'
#'@return Retorna uma lista contendo vetores de prioridades para cada matriz
na lista lida
#'
#'@export

calcula_prioridades = function(lista){
  prioridades = lapply(lista, function(x) autoVetor(x))
  return(prioridades)
}
```

Anexo 2: Código de aplicação do AHP.

```
ahp = function(base,mapeamento,nomes_alternativas){

  preferencias = calcula_prioridades(base); preferencias
  objetivo = preferencias[1]; objetivo
  criterios = preferencias[2:(length(mapeamento)+1)]; criterios
  alternativas = preferencias[(length(mapeamento) + 2):length(preferencias)];
alternativas
  matriz_criterios = base[2:(length(mapeamento)+1)]
  matriz_alternativas = base[(length(mapeamento) + 2):length(base)]

  #normalizando

  criterios_normalizados = c()
  for(i in 1:length(mapeamento)){
    criterios_normalizados[[i]] = criterios[[i]] * objetivo[[1]][i]
  }
```

```
  names(criterios_normalizados) = names(criterios); criterios_normalizados

  #Gerando nomes para a tabela

  nomes = c(paste0("---",names(objetivo)))
  CR_saaty = c(CR(base[[1]]))
  aux = 1 # Me ajuda a me guiar pelas matrizes de subcriterios
  for(i in 1:length(mapeamento)){
    nomes = append(nomes, paste0("--",names(criterios[i])))
    CR_saaty = c(CR_saaty, CR(matriz_criterios[[i]]))
    for(j in 1:mapeamento[i]){
      if(mapeamento[i] == 0) break
      nomes = append(nomes, paste0("-",names(alternativas[aux])))
      CR_saaty = c(CR_saaty, CR(matriz_alternativas[[aux]]))
      aux = aux + 1
    }
  }
  nomes

  #Gerando coluna de pesos de critérios e subcritérios

  pesos = c(sum(objetivo[[1]]))
  for(i in 1:length(mapeamento)){
    pesos = append(pesos, objetivo[[1]][i])
    aux = 1
    for(j in 1:mapeamento[i]){
      if(mapeamento[i] == 0)break
      pesos = append(pesos,criterios_normalizados[[i]][aux])
      aux = aux+1
    }
  }
  pesos

  #testando primeira parte
  tabela = tibble(Criterios = nomes, Pesos = pesos)
  #tabela


  #Aqui estou organizando a proporção de cada criterio por alternativas
  qtd_alternativas = length(alternativas[length(alternativas)][[1]]);
qtd_alternativas
  pesos_alternativas = list()
  #Criando a lista que serão preenchidas
  for(i in 1:qtd_alternativas){
    pesos_alternativas[[i]] = 0
```

```
  }
  #names(pesos_alternativas) = LETTERS[1:qtd_alternativas]
  names(pesos_alternativas) = nomes_alternativas
  pesos_alternativas


  aux = 1 #Navega entre a posição das matrizes de alternativas
  aux2 = 1 #Navega entra a posição do preenchimento das alternativas na nova
lista ordenada por linha
  for(i in 1:length(mapeamento)){
    #Se não existir subcriterios:
    if(mapeamento[[i]] == 0 ){
      for(j in 1:length(criterios_normalizados[[i]])){
        pesos_alternativas[[j]][aux2] = criterios_normalizados[[i]][j]
      }
      aux2 = aux2 + 1
    }
    #Se existir subcritérios:
    else{
      for(j in 1:length(criterios_normalizados[[i]])){
        #print("----")
        #print(criterios_normalizados[[i]][j])
        #print(names(alternativas[aux]))
        for(k in 1:length(alternativas[aux])){
          #print(alternativas[[aux]])
          for(p in 1:qtd_alternativas) {
            #print(alternativas[[aux]][p])
            pesos_alternativas[[p]][aux2] =
alternativas[[aux]][p]*criterios_normalizados[[i]][j]
          }
          aux2 = aux2 + 1
        }
        aux = aux +1
      }
    }

  }
  pesos_alternativas


  ##


  #Agora irei aplicar a soma de proporções dos critérios para sub critérios
```

```
  pesos_alternativas_organizados = pesos_alternativas

  for( i in 1:qtd_alternativas){
    inferior = 2
    superior = 0
    pesos_alternativas_organizados[[i]] =
c(sum(pesos_alternativas_organizados[[i]]),
pesos_alternativas_organizados[[i]])
    #print(names(pesos_alternativas[i]))
    vetor = c(pesos_alternativas_organizados[[i]][1])
    #print(pesos_alternativas_organizados[[i]])
    for(j in 1:length(mapeamento)){
      if(mapeamento[j] == 0 ) {
        vetor = c(vetor, pesos_alternativas_organizados[[i]][inferior])
        inferior = inferior + 1
      }
      else{
        superior = inferior + mapeamento[j] - 1
        #print("----")
        #print(pesos_alternativas_organizados[[i]][inferior:superior])
        valor = sum(pesos_alternativas_organizados[[i]][inferior:superior])
        vetor = c(vetor, valor,
pesos_alternativas_organizados[[i]][inferior:superior])
        #print(valor)
        #print("----")
        inferior = superior + 1
      }

    }
    pesos_alternativas_organizados[[i]] = vetor
  }

  pesos_alternativas_organizados

  #CR_saaty = unlist(lapply(base, function(x) return(CR(x))),use.names = F);
CR_saaty
  tabela = append(tabela,pesos_alternativas_organizados)
  tabela = append(tabela, list("CR"= CR_saaty))
  return(dplyr::as_tibble(tabela))
}

#'ahp_geral
#'
#'Realiza o calculo da AHP em uma lista de matrizes pareadas ou em um
#'endereco de planilhas do excel devidamente formatados.
```

```
#'
#'@param objeto Lista de matrizes pareadas ou caminho do excel contendo as
#'matrizes pareadas devidamente formatadas.
#'@param mapeamento Vetor contendo a quantidade de subcriterios de cada
criterio
#'da sua hierarquia, da esquerda para direita. Se nao preenchido o padrao
#'e preencher com 0. Em caso de duvida, veja a vignette tutorial.
#'@param nome_alternativas Vetor contendo os nomes das alternativas da sua
hierarquia,
#'se nao preenchido retorna um vetor de LETTERS[1:qtdAlternativas]
#'
#'@return Tabela contendo as relacoes dos criterios, subcriterios (Se houver)
e Alternativas
#'Utilizando o metodo AHP.
#'
#'@export
#'
#'

ahp_geral = function(objeto, mapeamento = "PADRAO", nomes_alternativas =
"PADRAO"){
  if(class(objeto) == "character") base = ler(objeto)
  else base = objeto
  if(mapeamento[1] == "PADRAO") mapeamento = rep(0,length(base[[1]]))
  if(nomes_alternativas[[1]] == "PADRAO") nomes_alternativas =
LETTERS[1:length(base[[length(base)]][[1]])]
  tabela = ahp(base, mapeamento,nomes_alternativas)

  return(tabela)
}

#'Ranque
#'
#'Cria o ranque de alternativas em ordem crescente de uma tabela AHP
#'criada pela funcao ahp_geral()
#'
#'@param tabela tabela AHP criada pela cuncao ahp_geral()
#'
#'@return Ranque das alternativas em ordem crescente
#'
#'@import dplyr
#'@import tidyr
#'
#'@export
ranque = function(tabela){
```

```
  num_colunas = length(tabela[1,])
  nun_linhas = length(tabela[,1])
  alternativas = tabela[1,3:(num_colunas-1)]

return(dplyr::select(dplyr::mutate(dplyr::arrange(tidyr::gather(alternativas,A
lternativas,Pesos),desc(Pesos)),Ranque =
c(1:length(alternativas[1,]))),Ranque,dplyr::everything()))
}
```

Anexo 3: Código de formatação da tabela com formattable.

```
transforma_tabela = function(tabela){
  numero_linhas = dim(tabela)[1]
  numero_colunas = dim(tabela)[2]

  tabela_porcento = dplyr::mutate_if(tabela, is.numeric, function(x)
paste0(round(100*x,2),"%"))
  #tabela_porcento = dplyr::slice(tabela_porcento, numero_linhas,
1:(numero_linhas - 1))
  #nomes_criterios = c(tabela_porcento$Criterios[1],
unlist(lapply(tabela_porcento$Criterios[2:numero_linhas],function(x) paste0("-
",x))))
  #tabela_porcento = dplyr::mutate(tabela_porcento)

  return(tabela_porcento)

}
#'formata_tabela
#'
#'Formata uma tabela AHP criada pela funcao ahp_geral()
#'
#'@param tabela Tabela AHP criada pela funcao ahp_geral()
#'@param cores Padrao de cores para formatar a tabela. Se "PADRAO" retorna o
padrao de cores (verde, azul, verde ou azul); se "CINZA" retorna o padrao de
cores de cinza;
#'se "BRANCO" retorna a tabela sem cores
#'
#'@return Retorna uma tabela formatada com cores defundo responsivas as
quantidades de prioridade dos elementos
#'
#'@import formattable
#'
#'@export
#'
```

```
formata_tabela = function(tabela, cores = "PADRAO"){
  if(cores[1] == "PADRAO"){ #Cores escolhidas utilizando a regra de harmonia
de cores triade
    limiteInferiorCriterios = "#DeF7E9"
    limiteSuperiorCriterios = "#71CA97"
    limiteInferiorAlternativas = "#B6D4FF"
    limiteSuperiorAlternativas = "#0060D3"
    limiteInferiorCR = "#ff7f7f"
    limiteSuperiorCR = "#B0FFD5"
    cor_letra = "black"
  }
  if(cores[1] == "CINZA"){
    limiteInferiorCriterios = "#9e9e9e"
    limiteSuperiorCriterios = "#4f4f4f"
    limiteInferiorAlternativas = "#9e9e9e"
    limiteSuperiorAlternativas = "#4f4f4f"
    limiteInferiorCR = "#4f4f4f"
    limiteSuperiorCR = "#9e9e9e"
    cor_letra = "white"
  }
  if(cores[1] == "BRANCO"){
    limiteInferiorCriterios = "#ffffff"
    limiteSuperiorCriterios = "#ffffff"
    limiteInferiorAlternativas = "#ffffff"
    limiteSuperiorAlternativas = "#ffffff"
    limiteInferiorCR = "#ffffff"
    limiteSuperiorCR = "#ffffff"
    cor_letra = "black"

  }

  numero_linhas = dim(tabela)[1]
  numero_colunas = dim(tabela)[2]


  tabela_porcento = transforma_tabela(tabela)
  maior_alternativa =
round(max(100*as.numeric(unlist(lapply(tabela[1,3:(numero_colunas-
1)],function(x) gsub("%","",x))))),2)

  formato = function(cor1,cor2){formattable::formatter(.tag = "span",
                                        style
=function(x)formattable::style("background-color"
=formattable::csscolor(formattable::gradient(as.numeric(unlist(lapply(x,functi
on(x) gsub("%","",x)))), cor1, cor2)),
```

```
                                                                    "border-
radius" = "4px",
                                                                    "color" =
cor_letra,
                                                                    display =
"block"))
  }

  formata_maior_alternativa = formattable::formatter("span",
                                          style = x ~ formattable::style("font-
weight" = ifelse(as.numeric(unlist(lapply(x,function(x) gsub("%","",x)))) ==
maior_alternativa, "bold", NA),
                                                                    "font-size" =
ifelse(as.numeric(unlist(lapply(x,function(x) gsub("%","",x)))) ==
maior_alternativa, "130%", NA)))

  formato_CR = formattable::formatter(.tag = "span",
                          style =function(x)formattable::style("background-
color" =ifelse(as.numeric(unlist(lapply(x,function(x) gsub("%","",x)))) >=
10,limiteInferiorCR,limiteSuperiorCR),
                                                "border-radius" = "4px",
                                                "color" = cor_letra,
                                                display = "block"))

  tabela_formatada = formattable::formattable(tabela_porcento,
                                                align = c("l",rep("c",
numero_colunas - 1)),
                                                list(
                                                  "Criterios" =
formattable::formatter("span", style = ~ formattable::style(color =
"grey",font.weight = "bold")),
                                                  formattable::area(row =
2:(numero_linhas), col = 2) ~
formato(limiteInferiorCriterios,limiteSuperiorCriterios),
                                                  formattable::area(row =
2:(numero_linhas), col = 3:(numero_colunas-1)) ~
formato(limiteInferiorAlternativas,limiteSuperiorAlternativas),
                                                  formattable::area(col =
numero_colunas) ~ formato_CR,
                                                  formattable::area(row = 1,
col = (3:numero_colunas-1)) ~ formata_maior_alternativa

                                                  )

                                                )
```

```
  return(tabela_formatada)
}
```

Anexo 4: Código do aplicativo shiny.

```
library(shiny)
library(shinyMatrix)
library(rhandsontable)
library(rlang)


reseta_matriz = function(dim){
  matriz = diag(dim)
  for(i in 1:(dim-1)){
    matriz[(i+1):dim,i] = c("")
  }
  return(matriz)
}

verifica = function(num){
  num = eval(parse(text=num))
  if(num == 0 ) return("")
  return(round(1/num,3))
}

inverso = function(matriz){
  dim = dim(matriz)[1]
  for(i in 1:(dim-1)){
    matriz[(i+1):dim,i] = mapply(function(x) verifica(x), matriz[i,(i+1):dim])
  }
  return(matriz)
}



m = reseta_matriz(3)

ui = fluidPage(

numericInput("dim", "Qual é a dimenção da matriz?",3),
actionButton("button", "Update"),
h4("Preencha a matriz: "),
matrixInput(inputId = "matrix",value = m),
```

```
h4("Matriz inversa:"),
rHandsontableOutput("table"),
actionButton("button2", "Codigo"),
verbatimTextOutput("texto")

)



server = function(input, output, session){
  output$table <- renderRHandsontable({

    matriz = input$matrix
    rhandsontable(inverso(matriz))

  })

  observeEvent(input$button,{
    nrow = input$dim
    m = reseta_matriz(nrow)
    updateMatrixInput(session, "matrix", m)

  })

  observeEvent(input$button2,{
    output$texto = renderPrint({

      codigo = paste0("matrix(c(",paste(inverso(input$matrix),collapse=",
"),"),ncol = 3)")
      codigo


    })

  })


}

shinyApp(ui = ui, server = server)
```

REVISTA DO SEMINÁRIO INTERNACIONAL DE ESTATÍSTICA COM R | **ISSN: 2526-7299**
VOL 5, Nº 1, ABRIL DE 2024