**IMPLEMENTANDO A AHP COM R**

Lyncoln Sousa de Oliveira[[1]](#footnote-1), Luciane Ferreira Alcoforado 2, Steven Dutt Ross 3 e

Rodolfo Hauret 4

# Resumo

Esse trabalho propõe a criação de funções e um pacote utilizando a linguagem R para realizar o método AHP proposto por Saaty. As funções foram criadas utilizando a filosofia *tidyverse* e podem ser acessadas no repositório do autor disponível no GitHub. Os códigos foram criados com intuito de serem de uso simples para que o usuário não perca tempo aprendendo usar as funções e assim possa ter seu resultado de maneira rápida e fácil, para isso não é preciso usar objetos do R, podem ser importadas bases de dados do Excel utilizando uma função criada. O pacote conta com uma vignette tutorial onde o usuário pode aprender a usar as funções e também existe base de dados exemplos para que seja possível visualizar o seu funcionamento mesmo sem um banco de dados. Foi desenvolvido também um aplicativo em shiny para facilitar ainda mais o uso do pacote.

.

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

# Abstract

This work proposes the creation of functions and a package using the R language to perform the AHP method proposed by Saaty. The functions were created using the tidyverse philosophy and can be accessed in the author's repository available on GitHub. The codes were created with the intention of being simple to use so that the user does not waste time learning to use the functions and thus can have its result quickly and easily, for this it is not necessary to use R objects, databases can be imported from Excel using a function created. The package has a tutorial vignette where the user can learn to use the functions and there is also a database of examples so that it is possible to visualize its operation even without a database. A shiny application was also developed to further facilitate the use of the package.

**Keywords:** AHP, Saaty, pacote, R.

# Objetivos

Objetivo Geral: Implementar o método AHP utilizando a linguagem computacional R.

Objetivos Específicos: Elaborar a estrutura do banco de dados para os valores de entrada à luz da compreensão do método AHP e suas etapas; implementar a estrutura de saída dos resultados; construir funções específicas para o desenvolvimento do método; desenvolver um sistema para o processamento da ahp através de um aplicativo shiny integrado ao pacote.

# 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.

# 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.

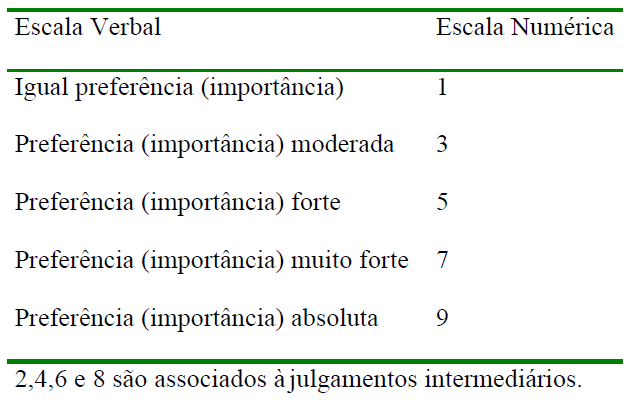
De acordo com o Saaty (1991), a aplicação do AHP pode ser dividia nas seguintes etapas: estruturação dos critérios e alternativas; coleta de julgamentos; cálculo de prioridades; verificação da consistência do julgamento; e, por último, o cálculo das prioridades globais das alternativas.

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



**Fonte:** Costa, 2002

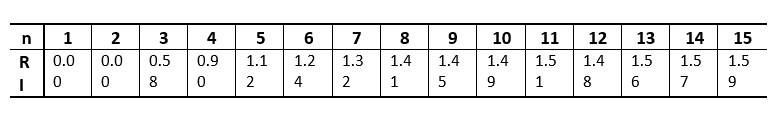
Na etapa de cálculo de prioridades são utilizadas ferramentas matemáticas como autovetores e autovalores nas matrizes de julgamento para quantificar os pesos de cada critério e também de cada alternativa em relação ao objetivo.

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 muitos critérios, Saaty (1991) propôs procedimentos para calcular a inconsistência derivadas dos julgamentos, estas são: IC (Índice de Consistência) e RC (Razão de consistência).

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

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



**Fonte:** Saaty, 1991

|  |  |  |
| --- | --- | --- |
|  |  | (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 um repositório no github para a hospedagem das funções e documentação do pacote criado, está disponível em: [www.github.com/NomedoAutor/AHP2](http://www.github.com/NomedoAutor/AHP2), nele 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 aplicação pelo usuário e que apresentem 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.

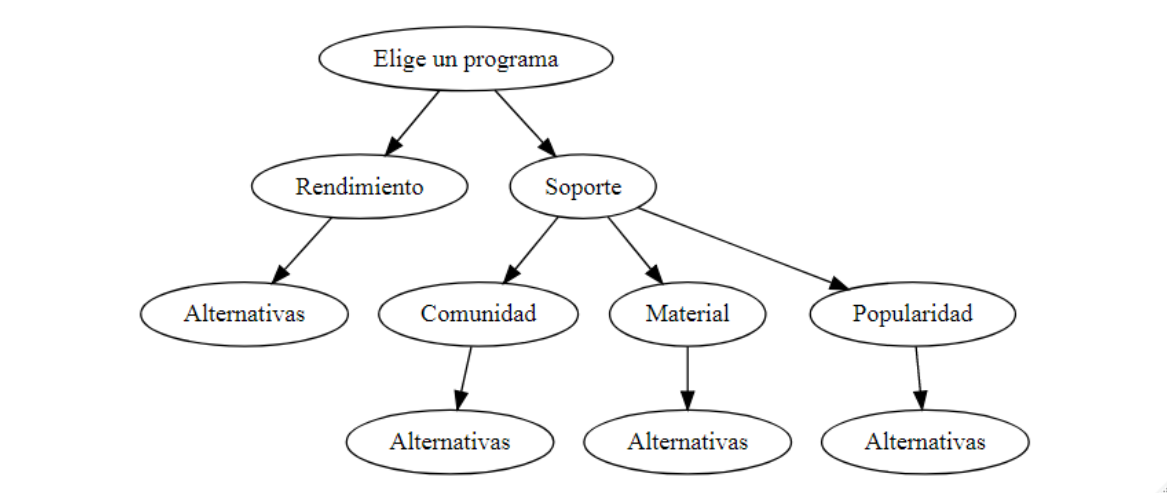
# Resultados e Discussão

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.

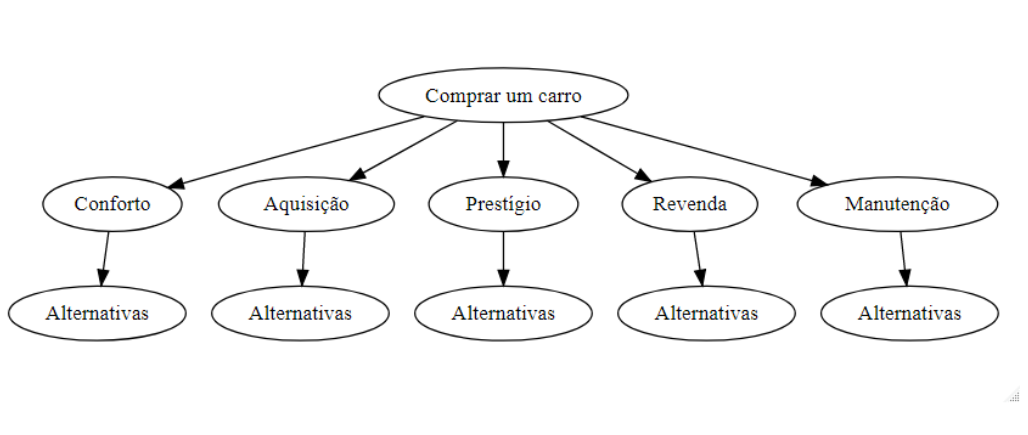
Caso o usuário escolha usar um arquivo de planilhas em formato xlsx, as planilhas devem ser organizadas de acordo com a figura 2.



**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 e também a 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

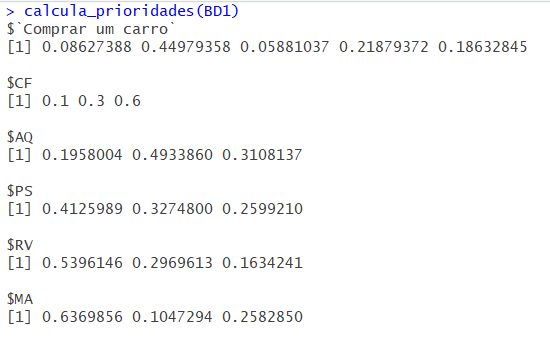
A função CR é da forma CR(matriz), espera-se como argumento uma matriz para que se possa retornar o *Consitency Ratio* (Razão de Consistência) dessa matriz. Exemplo para a matriz de comparação de critérios do BD1 está disponível na figura 4.



**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.



**Figura 5 –** Peso de critérios e alternativas para o BD1

Fonte: Pacote AHP, 2019

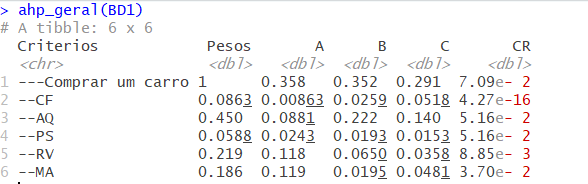
A função ahp\_geral é da forma ahp\_geral(base, mapeamento = "PADRÃO", nomes\_alternativas = "PADRÃO"). A base de entrada pode ser uma lista contendo matrizes de paridade no R ou um caminho para um arquivo de planilhas xlsx.

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).

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.

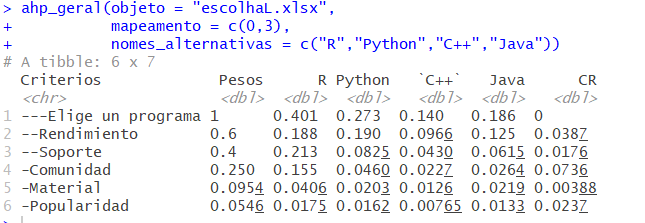
O resultado é uma tabela contendo as informações dos pesos de cada critério em relação ao objetivo e também os pesos de cada alternativa mediante aos critérios e também ao foco objetivo e pôr fim a coluna CR que apresenta a razão de consistência de cada matriz da base de dados.

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)



**Figura 7 –** Saída da função ahp\_geral para BD1

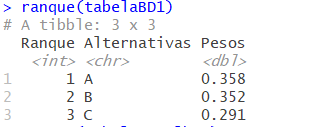
Fonte: Pacote AHP, 2019



**Figura 8 –** Saída da função ahp\_geral para o problema de escolha de linguagem

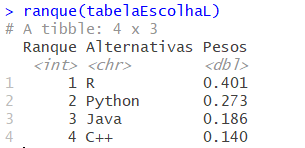
Fonte: Pacote AHP, 2019

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 9 e 10 para os problemas do BD1 e escolha de linguagem, respectivamente.



**Figura 9 –** Ranque das alternativas do BD1

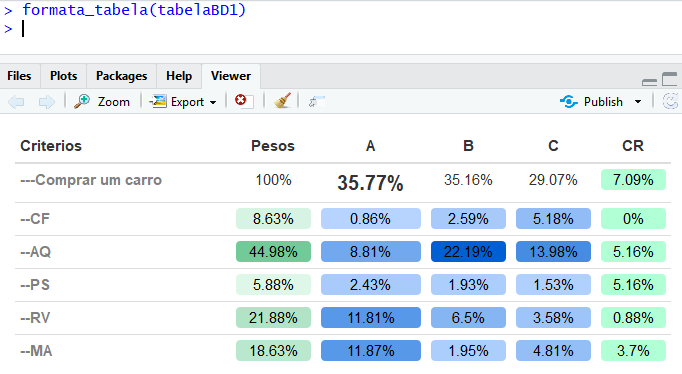
Fonte: Pacote AHP, 2019



# Figura 10 – Ranque das alternativas da escolha de uma linguagem

Fonte: Pacote AHP, 2019

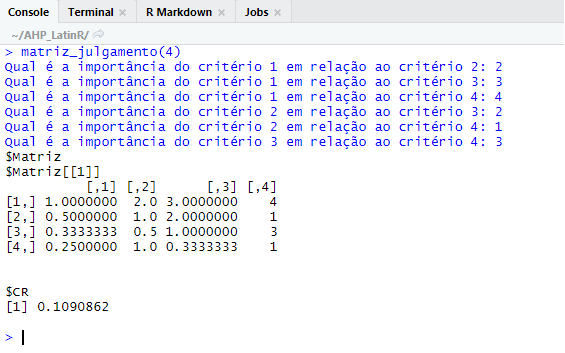
A função formata\_tabela que é da forma formata\_tabela(tabela, cores = “PADRAO”) foi criada para que o usuário possa obter uma tabela visualmente mais atrativa e intuitiva. Os argumentos de entrada são a tabela como um objeto R, como a função ranque() e a cor que o usuário queira para a tabela. Atualmente 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 11 apresenta a função sendo utilizada para o banco de dados BD1 com as cores “PADRAO”.



**Figura 11 –** Saída da formata\_tabela() para a base BD1

Fonte: Pacote AHP, 2019

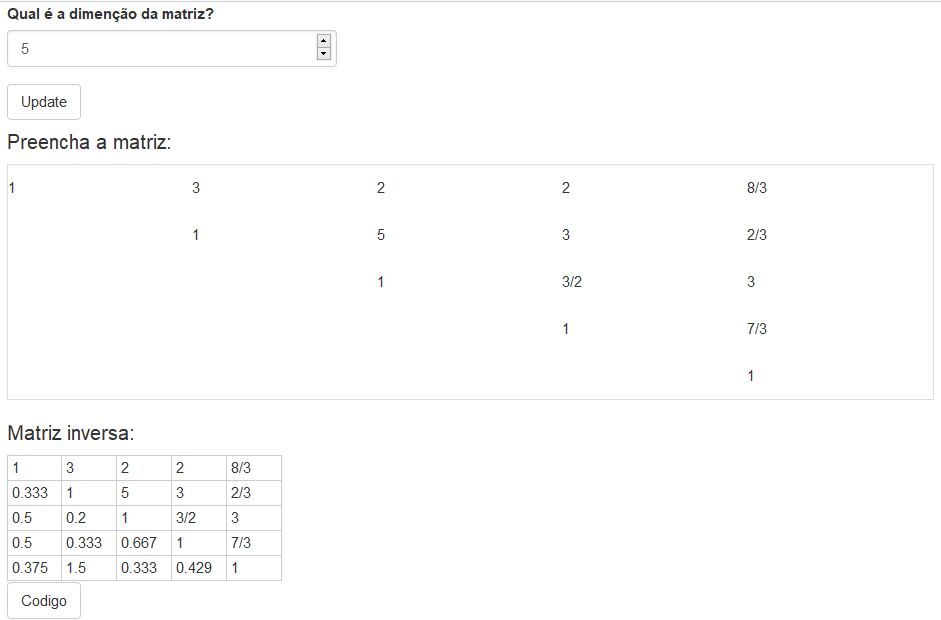
A função matriz\_julgamento() que é da forma matriz\_julgamento(qtd\_comparacoes, CR = TRUE, qtd\_matrizes = 1) orienta a criação de matrizes na interface do R. o argumento qtd\_comparações é para ser preenchido pela quantidade de comparações da matriz. CR = TRUE apresenta a razão de consistência junto com a matriz de julgamento, e o qtd\_matrizes informa a quantidade de matrizes a ser criada. Observe a criação de matrizes pela figura 12.



**Figura 12 –** 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 13.



**Figura 13 –** Criação de matrizes pelo app shiny em desenolvimento

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 vigenette tutorial onde é apresentado cada função e uma aplicação seguido de uma explicação de como deve ser feito a entrada dos dados.

**Conclusão**

A escolha da linguagem R para implementação do pacote se mostrou viável, consistente e segura. 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 e também há aquelas que realizam procedimentos intermediários ao método.

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 desenvolvida 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/,](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,Alternativas,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,function(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)

1. Nome Completo da Instituição (e sigla), e-mail do 1º autor 2 Nome Completo da Instituição (e sigla), e-mail do 2º autor 3 Nome Completo da Instituição (e sigla), e-mail do 3º autor 4 Nome Completo da Instituição (e sigla), e-mail do 4º autor [↑](#footnote-ref-1)